COMLI

System Description

COMLI

System Description

Release: 9903 Document version: 3-2

Document number: 493-0192-11

Printed in Sweden

Foreword

Since the first edition of the COMLI system description was published, significant changes have taken place both on ABB Automation's control systems and the COMLI protocol. New systems have been launched and old systems have become obsolete. This has produced new demands on COMLI resulting in new messages emerging and old messages disappearing.

In this new version, the chapters related to individual control systems have been omitted to facilitate future updating. Instead, the reader is referred to the COMLI description for the relevant system. Major sections of the description have also been restructured.

We hope that these changes will provide a description that is easier to use and which is updated with all the message types currently in use. ABB Automation reserves the right to make changes to the documentation without prior notice.

If readers discover any errors, unclear information or have other suggestions related to the contents, ABB Automation will be grateful to receive their comments.

Contents

1 COMLI

1.1 Introduction

COMLI (COMmunication LInk) is a ABB Automation data transmission system for communication between control systems and between control systems and computers. COMLI is a conventional communication link using serial, asynchronous data transmission in half duplex mode, i.e. one direction at a time, and according to the master/slave principle.

This document provides detailed information on COMLI for personnel requiring a thorough understanding of the system in order to implement COMLI in a particular application or locate faults on a system.

The description assumes a knowledge of ABB Automation's control systems and the terminology used in the control system descriptions.

COMLI is the most common data communication system used with the ABB Automation family of control systems. Latest generation installations providing operator presentation as well as PLC facilities may use new data transmission systems such as MMS. When COMLI is also used in these new systems, implementation is slightly different to when COMLI is used for data communication between SattCon systems only. Throughout this description it is assumed that only SattCon systems are used.

Definitions

Master and Slave

The terms master and slave are used frequently throughout this handbook. Master in a communications context means the controlling station which always initiates the message transmission sequence. Slave means a reply station which cannot start a communication sequence but simply responds to calls from the Master station.

Hexadecimal notation and ASCII format

When communication via COMLI is used 2-digit hexadecimal notation is used to described a byte. This is indicated by the suffix H after the hexadecimal characters. The data in the last byte of the message is coded in hexadecimal format, other bytes are coded in ASCII-hexadecimal format.

I/O-bit

ABB Automation's control systems include a memory area called I/O RAM containing the I/O rack's digital input and output signal statuses. In addition to I/O RAM, storage for memory cells, timer indications, limiter modules etc. is also included. An advantage with I/O RAM is that unused channels on the input and output cards can be used as working memory cells for temporary storage of intermediate results. Throughout the remainder of this description the term I/O bit refers to a location in this I/O RAM.

Letter codes

The following letters are used to identify the various types of address:

P = an address of 16 contiguous I/O-bits in I/O-RAM.

AI = analogue input signal address.

A = analogue output signal address.

R = 16-bit register address.

1.2 COMLI - principle of operation

ABB Automation’s systems communicate with each other and with other computers using COMLI (COMmunication LInk). COMLI is one of ABB Automation’s communication networks using serial, asynchronous data transmission in half duplex mode (one direction at a time) and in accordance with the Master/Slave principle (the Slave does not reply if there is an error in the message received).

Master and Slave can be linked together in different ways to achieve the desired function, i.e. Point-to-Point, Multipoint (Multidrop) or radial configuration (see the section on Network configuration). Up to 32 Slave systems can be connected to a Master system in a multipoint configuration. Master and Slave are linked via the serial channels on the different systems which are to communicate with each other. Master and Slave need not use the same physical channel numbers in both the systems. They must however have the same character format, transmission speed etc.

Communication is controlled by the PLC programs in the different control systems. By activating a predetermined memory cell, the PLC program in the control system acting as Master transmits a message. When the Slave receives the message, it responds either by sending the requested information or by acknowledging the information received. The memory cell is reset when the Master receives the information or an acknowledgement from the Slave.

The information in ABB Automation's control systems is stored in the different memory modules. About 50 different message types are defined for the various types of information to be transferred.

When the master starts a communication sequence, a communication area determines the type of data to be transferred. The data is transmitted and received via a serial channel defined using a channel definition area.

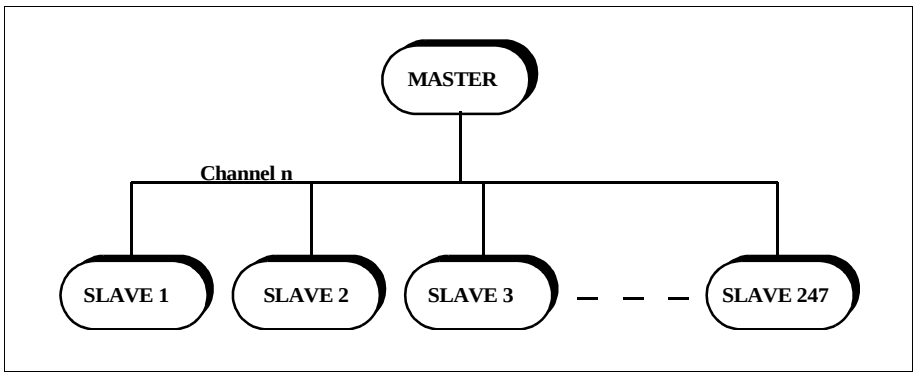
Network configurations

There are two network configurations for a COMLI serial communication network:

- Multipoint communication
- Point to point communication

Multipoint communication

Several Slave systems are connected to a master in multipoint communication. Communication takes place between Master and one Slave at a time. Direct communication between Slave systems is not possible. A particular message from the Master is applied to all Slaves but only the slave whose unique identity corresponds to the identity contained in the message accepts the data. The number of Slaves which can be connected to each Master is limited by the communication interface. The RS485 interface must be used in multipoint communication. The Master transmit line is connected to all Slave receive lines and all Slave transmit lines are commoned and connected to the Master receive line. RS485 permits 32 Slave systems to be connected to a Master. The COMLI communications software can handle a maximum of 247 Slaves.

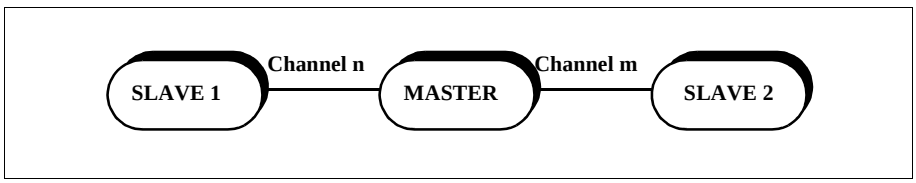


Example of multipoint communication

Point to point - communication

In point to point communication, only one Slave system is connected to the Master via one communication interface. Several Slaves can be connected to the Master but this must take place via different communication interfaces. This form of point to point configuration is called a radial (star) network and can provide a higher capacity than multipoint communication.

The electrical interface can be either RS232/V24 or RS485. A current loop can also be used in certain special cases. This is stated in the relevant system description.



Example of point-to-point communication

Master

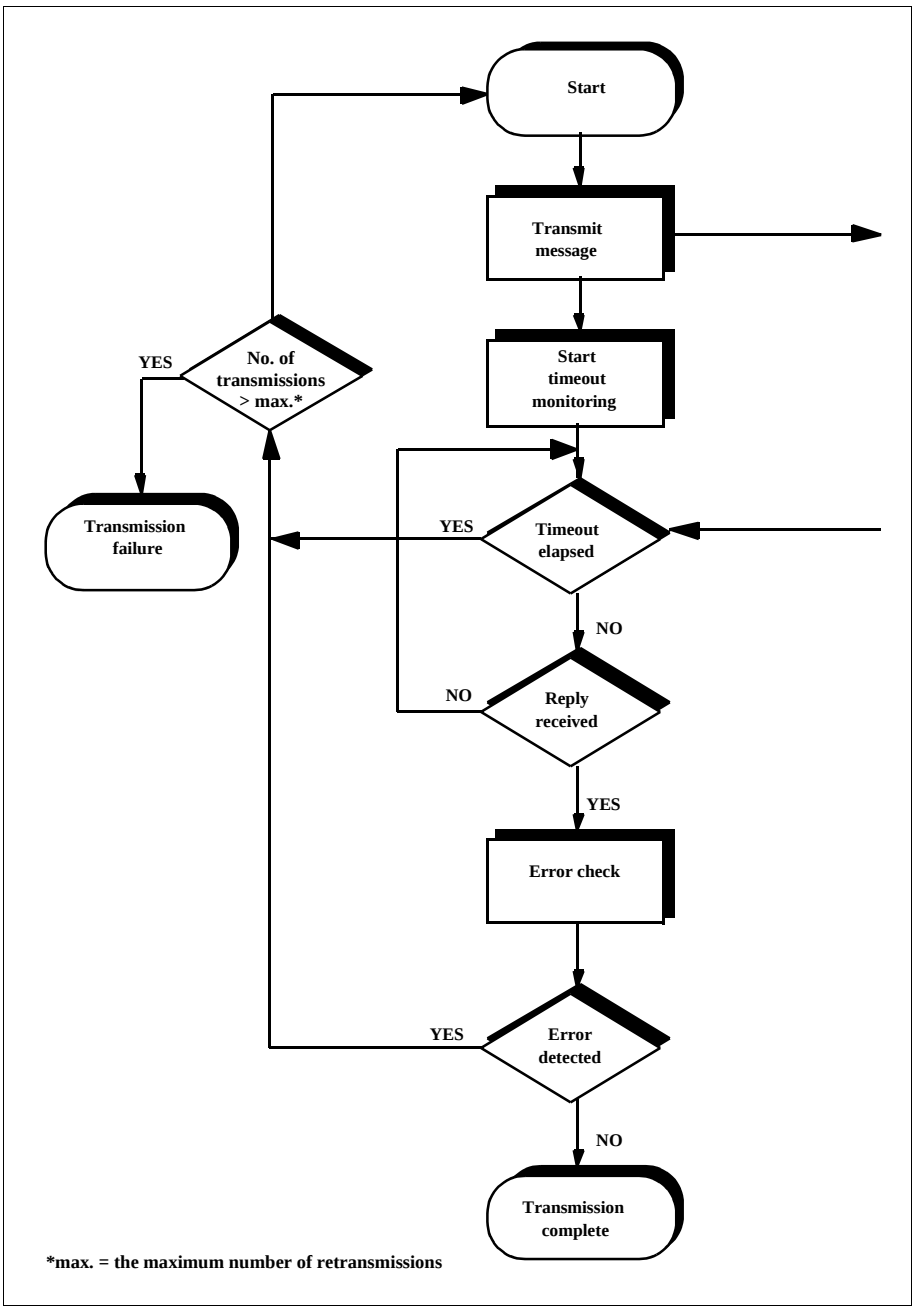
The Master always initiates the message transmission sequence, whether transmitting data to a slave or requesting data from a slave. The main functions of the Master are:

- To transmit data to the Slaves and receive the data transmitted by the Slaves.
- To process the messages received from the Slave.
- To repeat a message transmission if errors occur.

The processing of messages consists of checking that they are valid, determining the message type and taking the action appropriate to that message type (e.g. entering the status of an I/O bit when a message requesting the status of an I/O bit is transmitted and a valid reply is received).

Since the Slave never replies with an “error message”, the Master always knows how many characters it should receive in a message. If the Master requests a certain quantity of data, the Slave replies with the stated number of bytes or not at all.

Up to 10 retransmissions can be attempted if any errors are detected (the actual number permitted is system dependent). This is followed by automatic error message printouts in earlier SattCon systems. In other systems, alarm supervision must be connected to a memory cell in the I/O RAM for an error message to be printed out.



Master transmit/receive sequence

Master timeout

Master timeout is the period the master waits to receive a reply before assuming an error and retransmitting the message.

To avoid timing problems during COMLI communication between systems, the recommendation is that the following timeout periods are used for Masters operating at the various transmission speeds:

Transmission speed	Master timeout
50 baud	25 s
110 baud	13 s
150 baud	10 s
300 baud	7 s
600 baud	5 s
1200 baud	4 s
2400 baud	3 s
4800 baud	3 s
9600 baud	3 s
19200 baud	3 s
38400 baud	3 s

Slave

The Slave is the passive unit in the communication. A Slave unit waits continuously, listening to its receive line. The functions of the Slave are as follows:

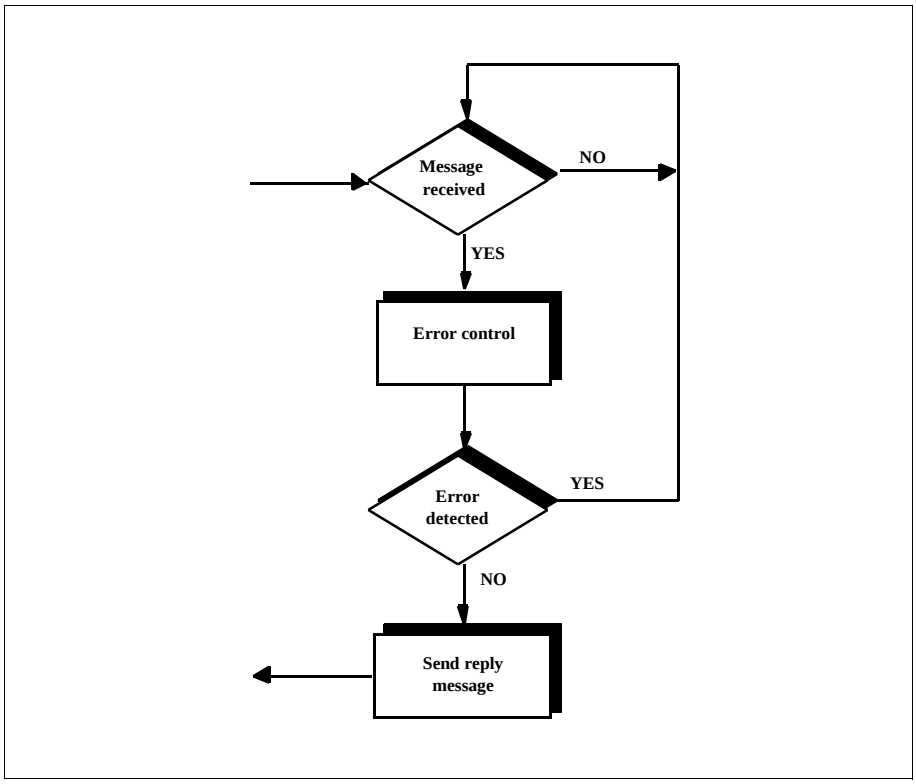
- To receive messages from the Master.
- To process the message received.
- To reply to the Master when the processing is complete.

The processing of the message consists of checking the contents, and taking the necessary action according to the message type (e.g. store the status of an I/O bit when a message containing the status is received and sending a reply indicating the request has been implemented).

The following are examples of errors which prevent the Slave from replying:

- Characters in the message received have parity errors.
- The BCC (Block Check Count, see COMLI message format later in this chapter) in the message does not agree with the BCC calculated by the Slave.
- The incoming data cannot be processed, e.g. if there is no program for this type of message.

If any of the above errors occur the Master automatically retransmits the message after the timeout period has elapsed.



Slave receive/transmit sequence

Slave timeout

The Slave timeout period is the maximum time the Slave waits for a complete message after STX (Start of TeXt) is received by the Slave.

To avoid timing problems during COMLI communication between systems, it is recommended that the following timeout periods are used for the different transmission speeds:

Transmission speed	Slave timeout
50 baud	24 s
110 baud	12 s
150 baud	9 s
300 baud	6 s
600 baud	4 s
1200 baud	3 s
2400 baud	2 s
4800 baud	2 s
9600 baud	2 s
19200 baud	2 s
38400 baud	2 s

1.3 Communication interfaces

One of the following standard communication interfaces is used for serial communication with COMLI:

- RS485.
- RS232C/V24.
- Current loop.

RS-485 interface

This is the most usual interface used for communications with COMLI since it is less sensitive to interference than RS232C. This communication interface should be used where possible.

The RS485 is a standard communications interface from EIA (Electronic Industries Association, USA) and is a voltage driven interface which operates with voltages between 0 and +5V. The interface is differential, i.e. for every logic state it sends two signals where one signal is inverted in relation to the other. By detecting differences between these two signals, a more interference tolerant transfer is achieved compared with RS232C. RS485 is a later version of RS422 which can handle a greater load but is otherwise identical to the earlier standard. RS485 can handle a maximum of 32 units.

Unlike RS232C, RS-485 defines only the electrical parameters. RS485 permits a maximum transmission speed of 10 Mbits/s and can operate over a longer transmission distance than RS232C. The maximum transmission distance is 1200 metres at 9600 baud.

RS-232C/V24 interface

RS232C/V24 is used primarily for connections to terminals but can also be used for COMLI communication in point to point configuration. RS232 cannot be used for multidrop configurations.

The RS232C communications interface is a standard from EIA (Electronics Industries Association, USA). The standard specifies serial, asynchronous communication up to 20 kbit/s with a maximum distance between communicating units of 15 metres. Under various circumstances it is however possible to communicate over much longer distances via RS232C. The standard specifies signal levels, driver circuits and mechanical structure, i.e. the connector device and the pin configuration.

The communication interface is an international standard which merely specifies signal levels. RS232C and V24 have the same signal levels and the voltage levels +15V/+5V mean logic zero whereas voltage levels -5V/-15V mean logic 1. Handshaking takes place with XON-XOFF or with the signals RTS-CTS.

Current loop

A current loop employs switched current to represent data on the line. Usually, a closed circuit, current flow, means a logic one and an open circuit (no current flow) a logic zero. A current loop is used in less demanding applications or where electrical isolation of linked units is required. A current loop usually operates with a current of 20 mA.

The transmission distance depends on the transmission speed and varies between 75 m/9600 baud and 1000 m/110 baud. Most ABB Automation control systems do not permit a speed higher than 600 baud in a current loop. Check the relevant system description. This type of communications link is seldom used nowadays.

Interface signals

The signals in the communication interfaces are listed in the table below; a particular application may employ all or only some of the signals.

Reference	Name	Remarks
SG (GNDE)	Signal Ground	
TD (EIAT)	Transmitted Data	
RD (EIAR)	Received Data	
RTS	Request To Send	Indicates that the channel transmitter is active.
CTS	Clear To Send	Indicates that a channel is ready to receive data.
DCD	Data Carrier Detect	Output signal indicating that the connection is established.
DSR	Data Set Ready	Output signal indicating that the station is ready to transmit or receive.

Distance – speed

The maximum transmission distance depends on the interface used and the transmission speed. The table below provides guidelines for the different interfaces and speeds.

Speed[1]	Distance[2] RS-232C/V24	Distance[2] RS-485	Distance[2] Current loop
110	600	1 200	1 000
150	600	1 200	1 000
300	600	1 200	1 000
600	600	1 200	1 000
1200	300	1 200	600
2400	150	1 200	300
4800	75	1 200	150
9600	40	1 200	75

The values stipulated are for screened twisted-pair cable, type Belden 8723 or Belden 9502.
Longer distances require a short range modem.

1. Bits per second
2. Metres

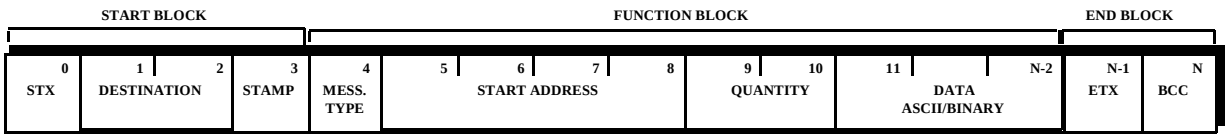
1.4 COMLI message format

General

COMLI is a protocol which defines the format used to transmit the data between two computers. Examples of the characteristics defined are the handshaking procedure, message start and stop characters used, if the message has a header, contents of the acknowledgement message etc.

The formats for different types of messages are similar. Each message starts with a start character (STX) followed by characters for destination, stamp, message type and data. The message ends with an end character (ETX) and a check character (BCC). The number of characters in a message can vary between 8 to 77. A maximum of 64 bytes of data can be transferred in one message. E.g. I/O-status for 512 bits or register values for 32 16-bit registers. All the characters in a message are transferred in an uninterrupted sequence.

COMLI message layout:



Messages can be divided into three categories:

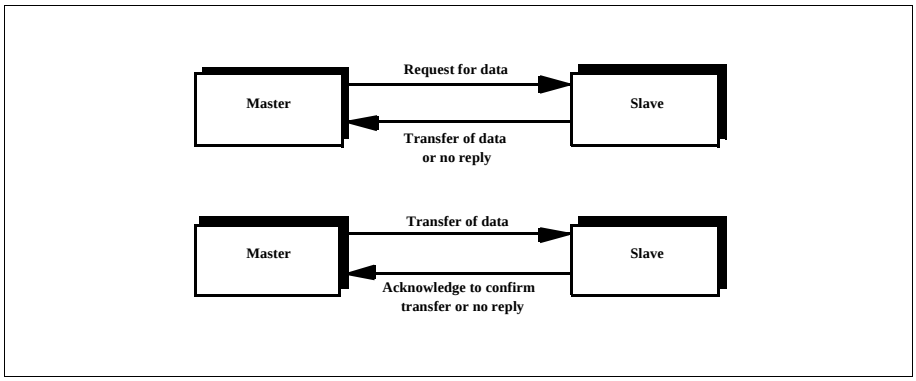
- Requests for data from Master to Slave. 13 characters.
- Transfer of data from Master to Slave or Slave to Master. 13–77 characters.
- Acknowledge message (transfer OK) from Slave to Master. 8 characters.

A request is only transmitted by a Master when the Master requires data from a Slave. If the Slave can’t carry out the request or an error occurs in the request message, no reply is sent by the Slave.

A transfer message can be transmitted by a Master or a Slave. When a Master needs to change data in a Slave, a message is transmitted by the Master which results in an acknowledgement by the Slave. The Slave does not respond if the message is not received correctly or if the data transmitted cannot be processed. A transfer can also be transmitted from a Slave to a Master as a reply to a request for data. In this case, the reply containing the data is also an acknowledgement that the request was received and processed correctly.

An acknowledgement can be sent from a Slave only when data from the master is received correctly.

The following figures illustrate the interchange between host and Slave systems in the above cases.



Message format

Irrespective of whether a message is a request, a data transfer or an acknowledgement, a message is made up of three blocks - start block, information block and end block. The start block and the end block have the same format in all types of messages but the information block varies for the different types of message. The characters in the start block and the end block are always in ASCII format. The contents of the information block can be in binary format or ASCII format.

Start block

The start block comprises three parts:

- STX
- Identity
- Stamp

The functions of the three parts are as follows:

STX

Start of TeXt start character which indicates the start of the block. The content of this byte is always 02H.

Identity

Indicates which station the message is intended for. Comprises two characters which can assume values between 30H–39H, 41H–46H. The Master always has the identity 0. A Slave can have an identity between 1–247. Always coded in ASCII.

STAMP

The transmission mark, STAMP, indicates if the message is being sent for the first time or if it is a retransmission. The stamp value changes for each new message. A retry is not treated as a new message. A reply message from Slave n (n = 1–247) is given the same value as the control message from Master. Refer to Section 3.2 for a more detailed description of the stamp function. Always coded in ASCII.

Function block

The function block contains the functional information and data as follows:

- Message type
- Start address
- Quantity
- Data block

Message type

This indicates the type of message being transmitted or received. The characters can be considered as a function code indicating how the Master or Slave interprets the data received. The message type is included in all message categories, i.e. request, data transfer, acknowledge. The character can assume a value between 30H and 7FH. Always coded in ASCII.

Start address, quantity and data block

The start address, quantity and data block have different meanings depending on whether the message is a request, a data transfer or an acknowledgement. The meanings are as follows:

Request

Start address indicates the first address from which the requested data is to be read. In the case of data that does not have an address, e.g a controller or limiter module, the number of the requested device is given, i.e. the controller number or limiter module number. Always represented in ASCII code.

Quantity indicates the number of items requested, e.g. the number of I/Os, number of registers, number of limiter modules etc. Always coded in ASCII.

The data block is not used in a request.

Data transfer

Start address indicates the first destination address for the data being transferred. In the case of data that does not have an address, e.g a controller or limiter module, the number of the requested device is given, i.e. the controller number or limiter module number. Always coded in ASCII.

Quantity indicates the number of bytes of data being transferred. Always coded in ASCII.

The data block contains the data being transferred. The arrangement of the data depends on the type of message. The size of the data block can be between 1 and 64 bytes. The data can be coded in binary or coded in ASCII depending on the chosen communication method.

Note. The data block must contain as many byte as requested. If there isn´t enough data, then fill the block with 00H.

Acknowledgement

During acknowledgement there is only one byte in addition to the character for the message type and this character has the value 06H. This is the character for Acknowledge according to the ASCII table. This means that an acknowledge message comprises a total of 8 characters.

End block

The end block comprises two parts:

- ETX
- BCC

The functions of these two parts are as follows:

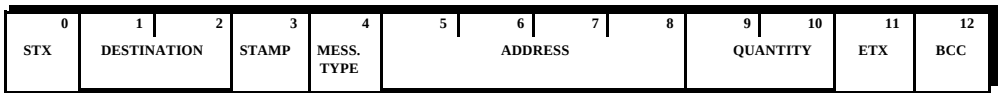
ETX

End of TeXt character, 03H which indicates the end of the block. Always coded in ASCII.

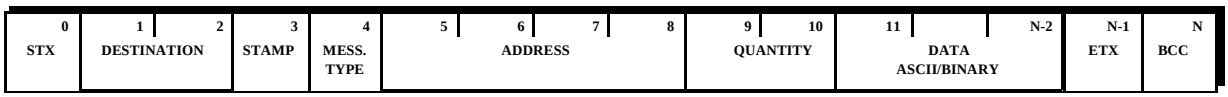
BCC

Block Check Count is a check sum for the complete message. The corresponding check sum is calculated in the receiver and compared with the message BCC to check if the message has been fully and correctly received. If the two BCCs do not agree, the Slave does not send an acknowledgement to the Master. Refer to Section 3.1 for a more detailed description of the check sum calculation. Always coded in ASCII.

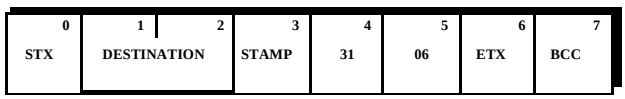
Examples of the 3 message categories are shown below:



Request message



Data transfer message



Acknowledgement message (Destination always 30/30)

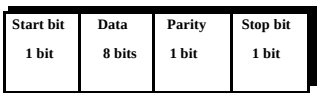
Character format

Asynchronous data transfer

The simplest way to transmit data (characters) between two computers is in parallel with 8 conductors – one for each bit plus a special conductor to indicate when data on the data conductors is valid. This is called parallel transfer.

For serial asynchronous transmission, which is the most common, one bit at a time is transmitted on a conductor. Synchronization is achieved by enclosing each character between a start bit and a stop bit and transmitting the bits at a given speed (baud rate). This is the method used by COMLI.

A character consists of the following:



Each character starts with a start bit. This is followed by the data bits, 7 or 8 depending on the format selected, and a parity bit which is inserted to ensure the number of 1s is either always even or odd (even or odd parity). A character bit changed during a transmission is thus detected as a parity error. The Block Check Character (BCC) is a further check that no characters have changed. It is always sent at the end of the message.

The standard format for COMLI is asynchronous transmission with 1 start bit, 8 data bits, 1 parity bit (odd parity) and one stop bit.

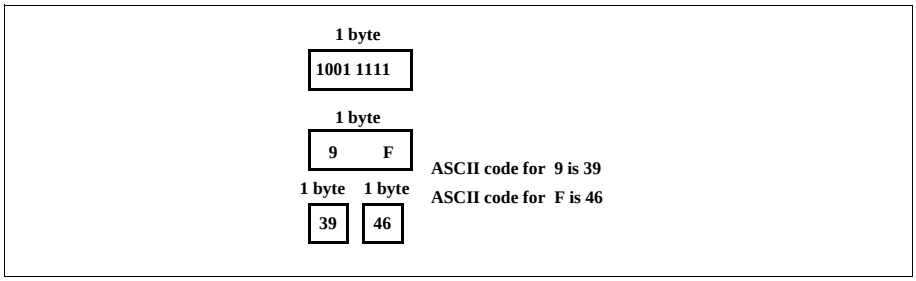
ASCII and binary communication

The character format during communication is either ASCII or binary. The choice of type, either ASCII or binary, only affects encoding of the data for transmission. Binary communication is the most common form of communication between control systems. Only half as many binary characters are required compared with ASCII characters when sending the same information. Communication with ASCII characters is used primarily when linking computers that cannot communicate using binary characters (8 bits).

In ASCII communication, the characters used are those in the character set for 7 bits (Swedish Standard 85 02 00). In binary communication on the other hand, any combination of 8 bits can be used. Thus the 8 bits representing a character can assume any value between 00–FFH. The complete set of ASCII characters is given in a table at the end of this handbook.

ASCII communication

In ASCII communication, each character has 7 or 8 data bits. The start character (STX) and the stop character (ETX) have unique codes which can be used as data characters in the message. Each byte in binary is represented by two bytes in ASCII. A binary byte is divided into two 4-bit parts which are represented as a 2-digit hexadecimal value and an ASCII byte is used to represent each hexadecimal digits using the ASCII codes 0–9, A-F. See the following example:



Binary represented using ASCII characters

Binary communication

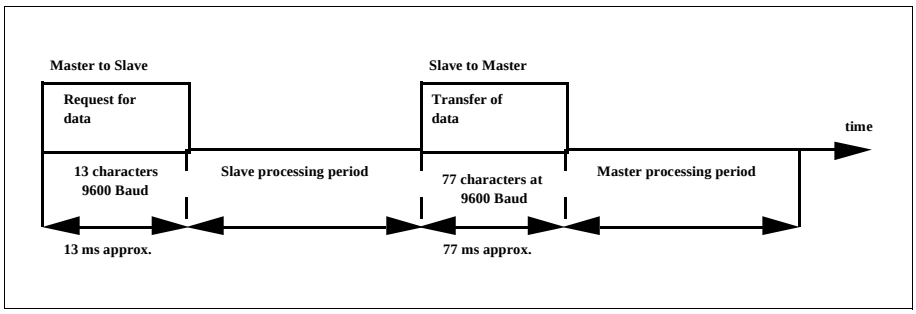
In binary communication each byte occupies one byte, thereby doubling the packing density of the data block. As a result, fewer bytes are required to transfer a given quantity of data compared with ASCII communication and data transfer times are considerably reduced.

1.5 Capacity

The following must be considered in order to calculate how many messages can be transmitted and received during a given period:

- Communication speed.
- Length of the messages.
- Processing times in the systems served.

This can be represented as shown below:



Example

Assume that a Master requests the transmission of the status of 512 I/O bits from a Slave. Communication is in binary format and takes place at 9600 baud. The actual request, which is 13 characters long, takes approximately 13 ms to transmit over the channel. The data transferred is 77 characters and takes approximately 77 ms to transmit over the channel.

This means that, if the processing time in the Master and Slave can be neglected, it will take approximately 90 ms to request and transfer 512 I/O bits over the channel. Thus 5120 I/O bits can be transmitted in one second. However, processing times in the different systems cannot be neglected. These times must also be added to the time it takes to transmit the characters over the channel. The processing time depends on the system being used and the load on the system.

1.6 Procedure for linking control systems with COMLI

The procedure for connecting different control systems to a common COMLI communication network can be summarised as follows:

- Select the message types by establishing the type of information to be transmitted between systems.
- Select the network configuration. Choose either multipoint or point to point and at the same time which communication interface is to be used, i.e. RS232, RS485 or current loop.
- Select the channel(channels) to be used. The selection depends on which channels are available and whether the channel can transfer the required information at the required speed.
- Decide which system is to be Master and which is to be Slave. The Master controls the data transmission operations since only the master can initiate a message transmission sequence.
- Connect the data transmission cables between the various systems on the network.

When the above operations are completed, the various communication parameters can be defined in a number of data fields.

Communication area

Each message is defined in a series of data fields which together are called a communication area.

There may be many such definitions programmed to produce a collection of communication areas which are stored in a data array.

Each individual message is controlled from a PLC bit which, when set, causes the master to transmit the message.

Within the data array a special area for messages containing mixed signal types must be included.

Channel definition

Each channel to be used for communication must then be defined so that the channels used in the different systems are set up in the same manner. This setting up is performed in each system’s channel definition program.

Channel definition also includes selecting which system is to control the network, i.e. the Master station. Each network contains only one Master but several Slaves can be included. When more than one Slave is included, the channel definition for each slave must stipulate the identity of the Slave. This identity must be unique in each communication network.

With all systems connected to the network and all data fields defined, the Master commences communications when the communication area memory cell is activated by the PLC program.

2 Message types

2.1 General

This chapter describes the various COMLI messages currently used in the different control systems. The chapter is arranged so that messages related logically are described together. E.g. all messages associated with controllers are described in one section.

Only message parts which are unique for the message are described. Parts common to all messages, i.e. start and end blocks, are described in Section 1.4, “Message format”. Binary communication is used unless it is stated otherwise.

2.2 List of COMLI messages

A complete list[1] of the COMLI messages described is given below:

0	30H	Transfer I/O bits or a register
1	31H	Acknowledge
2	32H	Request several I/O bits or registers
3	33H	Transfer individual I/O-bits
4	34H	Request individual I/O-bits
5	35H	Request controller static data
6	36H	Transfer controller static data
7	37H	Request limiter modules
8	38H	Transfer limiter modules
9	39H	Request analogue signals
:	3AH	Request alarm text (new alarm text function)
;	3BH	Transfer alarm text (new alarm text function)
<	3CH	Request high registers
=	3DH	Transfer high registers
A	41H	Transfer analogue signals
B	42H	Request controller dynamic data
C	43H	Transfer controller dynamic data
D	44H	Change to terminal mode
E	45H	Request alarm text (old alarm function)
F	46H	Transfer alarm text (old alarm function)
G	47H	Request measuring range and type
H	48H	Transfer measuring range and type
I	49H	Request date and time
J	4AH	Transfer date and time
K	4BH	Request RAM memory in address range 0–64 Kbyte (see Section 3.4)
L	4CH	Request RAM memory in address range 64–128 Kbyte (see Section 3.4)
M	4DH	Transfer RAM memory in address range 0–64 Kbyte (see Section 3.4)
N	4EH	Transfer RAM memory in address range 64–128 Kbyte (see Section 3.4)
O	4FH	Request floating point register
P	50H	Transfer floating point register
Q	51H	Request trend curve data
R	52H	Transfer trend curve data
S	53H	Request timer
T	54H	Transfer timer
U	55H	Request PLC-RAM (see Section 3.4)
V	56H	Transfer PLC-RAM (see Section 3.4)
W	57H	Request memory area
X	58H	Transfer memory area
Y	59H	Transfer memory area after verifying
Z	5AH	Request mixed data types
Ä	5BH	Transfer time marked events (English keyboard key [)
Ö	5CH	Transfer mixed data types (English keyboard key \)
Å	5DH	Request time marked events (English keyboard key])
Ü	5EH	Request system information (English keyboard key ^)
_	5FH	Transfer system information

1. Note that this is a complete list containing all kinds of COMLI messages. Not all of ABB Automations systems has all of these message types implemented.

2.3 Status of I/O-bits

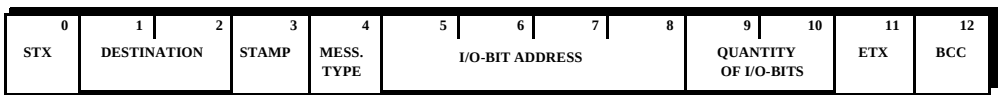
Request the status of individual I/O-bits (Message type 4)

Application

Used to request the status of a single I/O bit. Note that there is a special message to request several I/O bits. Any I/O bit in the address range 0000 (octal) to 37777 (octal) can be requested. The upper limit in different systems can be larger or smaller than the message allows. For systems containing addresses above 37777 (octal), bits in the out of range addresses must first be moved to addresses under 40000 (octal) before they can be transferred via COMLI.

Message structure

The message structure is as follows:



Message type (byte 4)

The message type is “4”, i.e. ASCII code 34H.

Address (bytes 5–8)

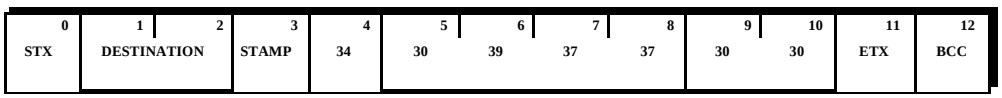
Address of the I/O bit whose status is to be transferred.

Quantity of bytes requested (bytes 9–10)

Not used since only one I/O bit is requested. Both bytes are set to zero, i.e. ASCII code 30H.

Example

To request the status of the digital input or output with address 4567(octal) the message contents are as follows:



4567 (octal) = 0977H

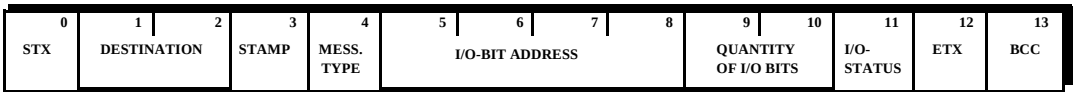
Transfer the status of individual I/O-bits (Message type 3)

Application

Used to transfer the status of a single I/O bit. Note that there is a special message to transfer several I/O bits. Any I/O bit in the address range 0000 (octal) to 37777 (octal) can be transferred. The upper limit in different systems can be larger or smaller than the message allows. For systems containing addresses above 37777 (octal), bits in the out of range addresses must first be moved to addresses under 40000 (octal) before they can be transferred via COMLI.

Message structure

The message structure is as follows:



Message type (byte 4)

The message type is “3”, i.e. ASCII code 33H.

Address (bytes 5–8)

Address of the I/O bit whose status is to be transferred.

Number of I/O bits (bytes 9–10)

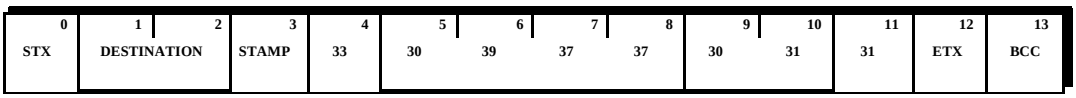
Since the same quantity of data is always transferred with this message, i.e. one byte, these bytes are always set to 01.

Data block (byte 11)

The data block consists of one byte indicating the status of the I/O bit, i.e. ’0’ ASCII code 30H, ’1’ ASCII code 31H.

Example

To request the status of the digital input or output with address 4567 (octal) the message contents are as follows:



4567 (octal) = 0977H

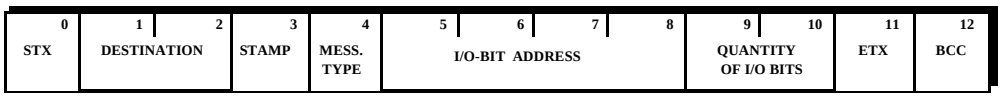
Request the status of several I/O bits (Message type 2)

Application

Used to request the status of up to 512 I/O bits using binary communication (256 using ASCII communication). Note that there is a special message to request the status of a single I/O bit. I/O bits starting at any address divisible by 8 in the address range 0000 (octal) to 37770 (octal) can be requested. The upper limit in different systems can be larger or smaller than the message allows. For systems containing addresses above 37770 (octal), bits in the out of range addresses must first be moved to addresses under 40000 (octal) before they can be transferred via COMLI.

Message structure

The message structure is as follows:



Message type (byte 4)

The message type is “2”, i.e. ASCII code 32H. Note that a request for a register and a request for I/O bits share the same message type. The address determines if the request is for I/O bits or for a register. An address less than 4000H indicates that the request is for I/O bits, an address 4000H or above indicates registers.

Start address (bytes 5–8)

Indicates the first address of the group of input or output I/O bits being requested. The address must always be exactly divisible by 8, i.e. the octal address must always end with a zero. Requests for bits in the address range 00000 (octal) to 37770 (octal) are permitted.

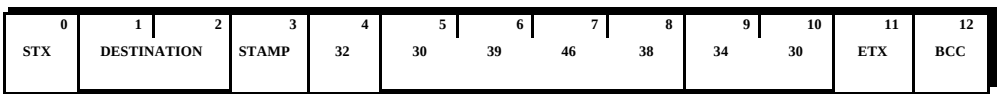
Number of I/O bits requested (bytes 9–10)

The number of bits which can be requested depends on whether binary or ASCII communication is being used. Every byte can contain the status of 8 I/O bits in binary communication (4 I/O bits in ASCII communication). A data block can contain a maximum of 64 bytes. Thus the minimum and maximum quantities of data which can be requested are as follows:

Min. 1 byte = 8 I/O bits using binary communication.
Min. 1 byte = 4 I/O bits using ASCII communication.
Max. 64 byte (64x8) = 512 I/O bits using binary communication.
Max. 64 byte (64x4) = 256 I/O bits using ASCII communication.

Example

A request for 512 I/O bits beginning at address 4770(octal). Binary communication is assumed.The message is as follows:



4770(octal) = 09F8H

64(dec)= 40H

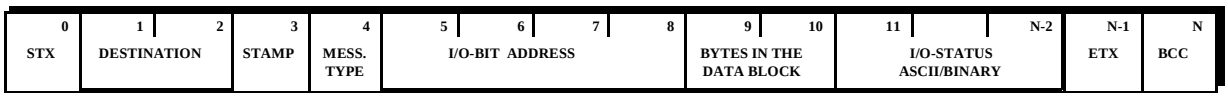
Transfer the status of several I/O-bits (Message type 0)

Application

Used to transfer the status of up to 512 I/O bits in one message. Note that there is a special message to request the status of a single I/O bit. I/O bits starting at any address divisible by 8 in the address range 0000 (octal) to 37770 (octal) can be transferred. The upper limit in different systems can be larger or smaller than the message allows. For systems containing addresses above 37770 (octal), bits in the out of range addresses must first be moved to addresses under 40000 (octal) before they can be transferred via COMLI.

Message structure

The message structure is as follows:



Message type (byte) 4)

The message type is “0”, i.e. ASCII code 30H. Note that transfer of registers and transfer of I/O bits share the same message type. The address determines if I/O bits or registers are transferred. An address less than 4000H indicates that I/O bits are to be transferred, an address 4000H or above indicates registers.

Start address (bytes 5–8)

Indicates the first address of the group of input or output I/O bits to be transferred. The address must be exactly divisible by 8, i.e. the octal address must end with a zero. Requests for bits in the address range 00000 (octal) to 37770 (octal) are permitted.

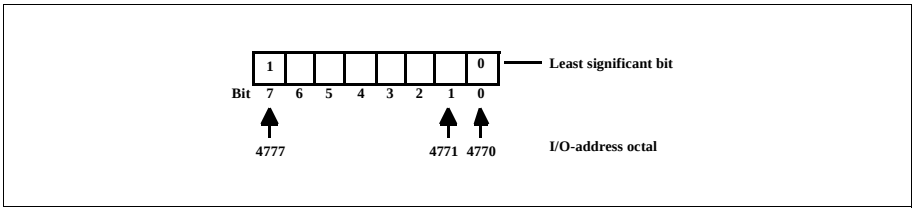
Number of bytes in the data block (bytes 9–10)

A minimum of 1 byte and a maximum of 64 bytes can be transferred in a data block. The number of I/O bits which can be transferred depends on whether binary or ASCII communication is being used. Every byte can contain the status of 8 I/O bits in binary communication (4 I/O bits in ASCII communication) Thus the minimum and maximum quantities of data that can be requested are as follows:

Min. 1 byte = 8 I/O bits using binary communication.
Min. 1 byte = 4 I/O bits using ASCII communication.
Max. 64 byte (64x8) = 512 I/O bits using binary communication.
Max. 64 byte (64x4) = 256 I/O bits using ASCII communication.

Data block using binary communication (byte 11 onwards)

Contains the status of the I/O bits to be transferred. Each byte has space for 8 I/O bits. The least significant bit position in the data block contains the I/O bit with the lowest address. Thus, transfer of I/O bits from addresses 4770–4777 (octal) occupies byte 11 of the data block as shown below:



Data block using ASCII communication (byte 11 onwards)

Contains the status of the I/O bits to be transferred. Each byte has space for one I/O bit. The least significant bit position in the data block contains the I/O bit with the lowest address.

Note that ASCII communication requires 8 times as many message bits as binary communication to transfer the same quantity of data.

Example

Assume that addresses 4770 to 5067 octally are to be transferred, the I/O bit in address 4770 has the status zero, all remaining I/O bits have the status ’1’ and ASCII communication is being used. The message is as follows:

0	1	2	3	4	5	6	7	8	9	10
STX	DESTINATION		STAMP	30	30	39	46	38	34	30

11	12	13	14	15	16	17	18	19	20
30	31	31	31	31	31	31	31	31	31

21	22	23	24	25	26	27	28	29	30
31	31	31	31	31	31	31	31	31	31

31	32	33	34	35	36	37	38	39	40
31	31	31	31	31	31	31	31	31	31

41	42	43	44	45	46	47	48	49	50
31	31	31	31	31	31	31	31	31	31

51	52	53	54	55	56	57	58	59	60
31	31	31	31	31	31	31	31	31	31

61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76
31	31	31	31	31	31	31	31	31	31	31	31	31	31	ETX	BCC

2.4 Register

Request register (Message type 2)

See also “Request the status of several I/O bits”.

Application

Used to request transfer of up to 32 16-bit registers. Registers numbered from 0 to 3071 (dec) can be requested. The upper limit in different systems can be larger or smaller than the message allows. For registers with numbers greater than 3071(dec) message type ”<” can be used.

Message structure

The message structure is as follows:

Message type (byte 4)

The message type is “2”, i.e. ASCII code 32H. Note that a request for a register and a request for I/O bits share the same message type. The address determines if the request is for I/O bits or for a register. An address less than 4000H indicates that the request is for I/O bits, an address 4000H or above indicates registers.

Start address (bytes 5–8)

Contains the address for the register or the first of the registers requested. An address is calculated by multiplying the register number by 16 and adding the offset 4000H. The lowest permitted address is 4000H, i.e. register number 0. The highest address permitted is FFF0H, i.e. register number 3071.

Address = 4000H + register number x 16.

Quantity of bytes (bytes 9–10)

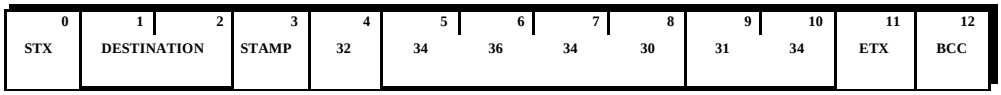
A minimum of 2 bytes and a maximum of 64 bytes can be requested. The number of 16-bit registers that can be transferred depends on whether binary or ASCII communication is being used. Thus the minimum and maximum number of registers that can be requested are as follows:

Min. 2 byte = 1 register using binary communication.
Min. 4 byte = 1 register using ASCII communication.
Max. 64 byte = 32 registers using binary communication.
Min. 64 byte = 16 registers using ASCII communication.

Note that only an even number of bytes can be stated when register values are requested.

Example l

Assume 10 registers are requested beginning at register number 100. The message is as follows:



Address= 4000H + 16 x 100(dec) = 4000H + 640H
= 4640H

Bytes requested = 10(dec) x 2 = 20(dec) = 14H

Request high register (Message type <)

Application

Used to request transfer of up to 32 16-bit registers. Registers numbered from 0 to 65535 can be requested. The upper limit in different systems can be larger or smaller than the message allows.

Message structure

The message structure is as follows:

0	1 2	3	4	5 6 7 8	9 10	11	12
STX	DESTINATION	STAMP	MESS. TYPE	REGISTER ADDRESS	QUANTITY OF BYTES	ETX	BCC

Message type (byte 4)

The message type is ”<”, i.e. ASCII code 3CH.

Start address (byte 5–8)

Contains the address for the first register or the first of the registers requested. An address is specified as the number of the register in hex notation, e.g. register 4569 (decimal) gives a start address of 11D9 .

Quantity of bytes (byte 9–10)

A minimum of 2 bytes and a maximum of 64 bytes can be requested. This telegram is specified for binary communication only.

Example 1

Assume 10 registers are requested beginning at register 4569. The message is as follows:

0	1 2	3	4	5	6	7	8	9	10	11	12
STX	DESTINATION	STAMP	3C	31	31	43	39	31	34	ETX	BCC

Transfer register (Message type 0)

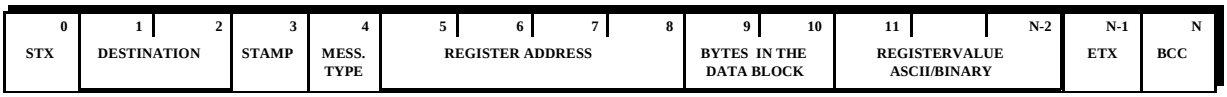
See also “Transfer the status of several I/O bits”.

Application

Used to transfer of up to 32 16-bit registers. A maximum of 32 registers can be transferred using binary communication or 16 using ASCII communication. Registers numbered from 0 to 3071 (dec) can be transferred. The upper limit in different systems can be larger or smaller than the message allows. For registers with numbers greater than 3071 (dec) message type ”=” can be used.

Message structure

The message structure is as follows:



Message type (byte 4)

The message type is “0”, i.e. ASCII code 30H. Note: transfer of registers and transfer of I/O bits share the same message type. The address determines if I/O bits or registers are transferred. An address less than 4000H indicates that I/O bits are to be transferred, an address 4000H or above indicates registers.

Start address (bytes 5–8)

Contains the address for the register or the first of the registers to be transferred. An address is calculated by multiplying the register number by 16 and adding the offset 4000H. The lowest permitted address is 4000H, i.e. register number 0. The highest address permitted is FFF0H, i.e. register number 3071.

Address = 4000H + register number x 16.

Number of bytes (bytes 9–10)

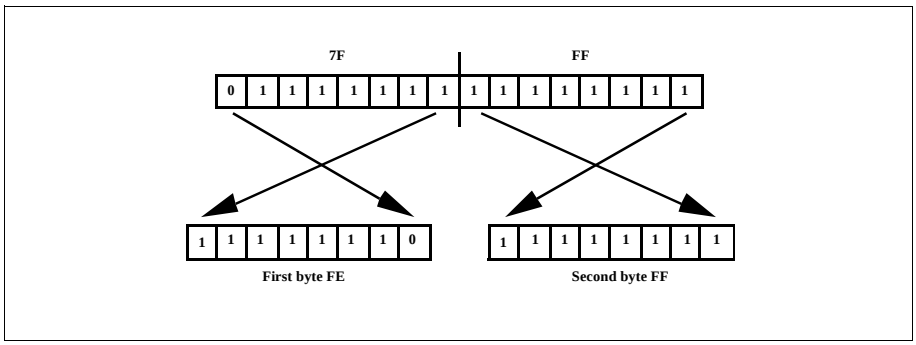
A minimum of 2 bytes and a maximum of 64 bytes can be transferred. The number of 16-bit registers that can be transferred depends on whether binary or ASCII communication is being used. Thus the minimum and maximum number of registers that can be transferred are as follows:

Min. 2 byte = 1 register using binary communication.
Min. 4 byte = 1 register using ASCII communication.
Max. 64 byte = 32 registers using binary communication.
Min. 64 byte = 16 registers using ASCII communication.

Note: only an even number of bytes can be stated when register values are transferred.

Data block (byte 11 onwards)

Transfer of a 16-bit register requires 2 bytes (binary communication). During transfer the register's most significant bit is located in the least significant bit of the first byte and the least significant bit is in the most significant bit position of the second byte.



Example

Register number 100 contains the value 7FFFH = 50% and register number 101 contains the value 1000H = 6.2%. Only these two registers are to be transferred using binary communication. The message is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
STX	DESTINATION		STAMP	30	34	36	34	30	30	34	FE	FF	08	00	ETX	BCC

The same message using ASCII communication is as follows:

0	1	2	3	4	5	6	7	8	9	10
STX	DESTINATION		STAMP	30	34	36	34	30	30	38

11	12	13	14	15	16	17	18	19	20
46	45	46	46	30	38	30	30	ETX	BCC

Transfer high register (Message type =)

Application

Used to transfer of up to 32 16-bit registers. A maximum of 32 registers can be transferred using binary communication. This telegram is specified for binary communication only. Registers numbered from 0 to 65535 can be transferred. The upper limit in different systems can be larger or smaller than the message allows.

Message structure

The message structure is as follows:

Message type (byte 4)

The message type is ”=”, i.e. ASCII code 3DH.

Start address (byte 5–8)

Contains the address for the first register or the first of the registers to be transferred. An address is specified as the number of the register in hex notation, e.g. register 4569 (decimal) gives a start address of 11D9 .

Quantity of bytes (byte 9–10)

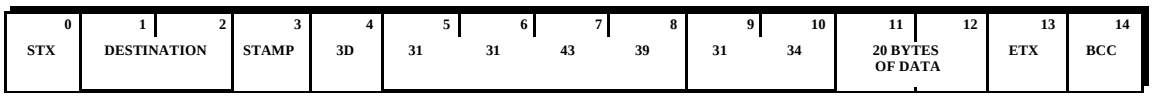
A minimum of 2 bytes and a maximum of 64 bytes can be requested. This telegram is specified for binary communication only.

Data block (byte 11 and onwards)

Transfer of a 16-bit register requires 2 bytes (binary communication). During transfer the registers most significant bit is located in the least significant bit of the first byte and the least significant bit is in the most significant bit position of the second byte. (See also message type 0).

Example 1

Assume 10 registers are transferred beginning at register 4569. The message is as follows:



2.5 Controller data

Request static data for controllers (Message type 5)

Application

Used to request static data for a controller.

Message structure

The message structure is as follows:

0	1 2	3	4	5	6 7 8	9 10	11	12
STX	DESTINATION	STAMP	MESS. TYPE	ASCII/ BINARY	CONTROLLER NUMBER	QUANTITY OF CONTROLLERS	ETX	BCC

Message type (byte 4)

The message type is “5”, i.e. ASCII code 35H.

Type of communication (byte 5)

Indicates if binary or ASCII communication is being used. When ASCII communication is used there is insufficient space in one message consequently two separate messages are sent and each message contains information indicating if it is the first or second message.The values of byte 5 for these indications are as follows:

30H binary communication.
31H ASCII communication and first half of controller data.
32H ASCII communication and second half of controller data.

Controller number (bytes 6–8)

Indicates the controller number. Numbers between 0 and 62 are permitted.

Number of controllers requested (bytes 9–10)

Only one controller can be requested at a time consequently bytes 9 and 10 always contain the value 1, i.e. ASCII code 30H, 31H.

Example

Request static data for controller number 3. The message is as follows:

0	1 2	3	4	5	6 7 8	9 10	11	12
STX	DESTINATION	STAMP	35	30	30 30 33	30 31	ETX	BCC

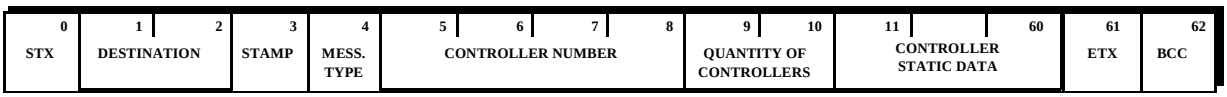
Transfer of static data for controllers (Message type 6)

Application

Used to transfer controller static data. This message has undergone changes and the version described here is for SattCon31, Revision 4. The changes include two new types of signal for registers type 8 and 9 which enable register numbers in the range 0 to 8446 (dec) to be specified. Section 4.4 contains a description of the message before SattCon31, Revision 4.

Message structure

The message structure is as follows:



Message type (byte 4)

The message type is “6”, i.e. ASCII code 36H.

Type of communication (byte 5)

Indicates if binary or ASCII communication is being used. When ASCII communication is used there is insufficient space in one message consequently two separate messages are sent and each message contains information indicating if it is the first or second message.The values of byte 5 for these indications are as follows:

30H binary communication.
31H ASCII communication and first half of controller data.
32H ASCII communication and second half of controller data.

Controller number (bytes 6–8)

Indicates the controller number. Numbers between 0 and 62 are permitted.

Number of bytes transferred (bytes 9–10)

During transfer of static data for controllers 50 bytes are always transferred irrespective of whether binary or ASCII communication is being used. Thus, bytes 9 and 10 always have the value 50, i.e. ASCII code 35H, 30H.

Data block for binary communication (byte 11–60)

The numbering of each byte below refers to inside the data block only. To obtain the correct number for a byte in the message an 11 byte offset is added.

Bytes 0 to 3 inclusive

32 bits containing the controller’s present state. In SattCon 31 this is the PI part of the controllers output signal.

Bytes 4 and 5

16 bits which include the process value. The functions of the four most significant bits B12, B13, B14 and B15 are as follows:

B12 Internal use.
B13 Indicates that the controller output is set in manual mode.
B14 Internal use.
B15 Controller not yet tuned in to the correct process state.
The 12 least significant bits indicate the value of the analogue input, i.e. process value. Min. is 0 = 0%, max is FFFH = 100%.

Bytes 6 and 7

16 bits containing X (K-1) for D-part. Min. is 0 = 0%, max. is FFFH = 100%.

Bytes 8 and 9

16 bits containing E (K-1) D-part. Min. is 0 = 0% and max. is FFFH = 100%.

Bytes 10 and 11

16 bits for the demanded signal when the controller is in the Manual mode. Min. is 0 = 0% and max. is FFFFH = 100%. Set to 0 if controller in auto mode.

Byte 12

8 bits describing the type of controller together with the controller number.The two most significant bits describe the type of controller as follows:

0 = P controller
1 = PD controller
2 = PI controller
3 = PID controller

The six least significant bits indicate the number of the controller for which data is being transferred.

Min value = 0, max value = 62 (3EH)

If the value of byte 12 is FFH, there are no more controllers to be sent.

Byte 13

8 bits B7, B6, B5, B4, B3, B2, B1 and B0 indicating the following:

B0 Type of control, 0 inverse control, 1 direct control.
B1 0 Manual mode, 1 Auto mode.
B2 Indicates if a feedback signal is used with a digital controller.
1 = Yes, 0 = No.
B3 Type of derivative influence:
1 if only for positive process value changes
0 if positive and/ or negative changes.
B4 Type of derivative influence:
1 if both positive and negative changes
0 for either negative or positive changes.
B5 1 if parameter control is used otherwise 0.
B6 Type of controller output:
0 = digital controller output, 1 = analogue controller output.
B7 Internal use.

Bytes 14 and 15

16 bits indicating the I/O address used to select Auto mode. Lowest address 0, highest address up to 3FFFH = 37777 (octal). Note that the highest address is also system dependent.

Bytes 16 and 17

16 bits indicating address and type of the analogue input signal. The four most significant bits indicate the source of the controller input signal as follows:

4 most sign bits	12 least sign bits
6	Register no, (range 0–254).
8	Register no-255, (range 255–4350).
9	Register no-4351, (range 4351–8446).
AH	P-word (I/O address, 16 bits).
EH	Analogue input address.

Bytes 18 and 19

16 bits indicating the address and type of the set point input signal. The four most significant bits indicate the source of the set point signal as follows:

4 most sign bits	12 least sign bits
0	Set point as stipulated from operator’s keyboard, min. 0 = 0%, max. FFFH = 100.0.%
6	Register no, (range 0–254).
8	Register no-255, (range 255–4350).
9	Register no-4351, (range 435–8446).
AH	P-word (I/O address, 16 bits).
EH	Analogue input address.

Byte 20

Indicates the set point dead zone. Values between 0 and FFH are permitted. FFH corresponding to 6.2%.

Byte 21

Indicates the gain. Values between 0 and FFH are permitted. This corresponds to a range of 0.0–25.5 if bit 0 in byte 25 has the value zero. If bit 0 in byte 25 has the value’1’ the range is 0.00–0.99.

Bytes 22 and 23

Indicate the integration time. Values between 0 and FFFFH. FFFFH corresponding to up to 1 hour 49 minutes and 13.5 seconds are permitted.

Byte 24

Indicates the sample time in tenths of a second.

Byte 25

Bit 0 in this byte indicates the gain range applicable to byte 21. See Byte 21 explanation.

Byte 26

Not used.

Byte 27

Internal use.

Bytes 28 and 29

Indicate derivation time. Values between 0 and 1770H are permitted. 1770 H corresponds to 10 minutes.

Bytes 30 and 31

Indicate the type of signal and its address which determines the D-part. The four most significant bits indicate the following types of signal for determining the D-part:

4 most sign bits	12 least sign bits
6	Register no, (range 0–254).
8	Register no-255, (range 255–4350).
9	Register no-4351, (range 4351–8446).
AH	P-word (I/O address, 16 bits).
EH	Analogue input address.

Bytes 32 and 33

Indicate the Hand Interlock Speed which is the time it takes for the output signal to increase from 0 to 100%. The value can vary between 5H = 0.5 seconds and 13FFH = 8 minutes and 31.9 seconds.

Bytes 34 and 35

Indicate the calculated ramp period for the output signal. This value can vary between 20H if the Hand Interlock Speed is 8 minutes and 31.9 seconds and FFFFH if Hand Interlock Speed is 0.5 seconds.

Bytes 36 and 37

Indicate the upper limit of the output signal. The value can vary between 0 and FFFFH. FFFFH corresponds to 100.0%.

Bytes 38 and 39

Indicates the lower limit of the output signal. The value can vary between 0 and FFFFH. FFFFH corresponds to 100.0%.

Bytes 40 and 41

Describe the controller output signal. If the output is an analogue signal the following types of signal apply:

4 most sign bits	12 least sign bits
6	Register no, (range 0–254).
8	Register no-255, (range 255–4350).
9	Register no-4351, (range 4351–8446).
AH	P-word (I/O address, 16 bits).
EH	Analogue output address.

For a digital controller all 16 bits indicate the address of the controller’s “OPEN” output. The address can vary between 0 and 3FFFH, 37777 (octal). Note that the upper limit is system dependent.

Bytes 42 and 43

The address of the digital controller's "CLOSE". The address can vary between 0 and 3FFFH, 37777 (octal). Note: upper limit is system dependent.

Bytes 44 and 45

Indicate the address and type for the external connection signal. The four most significant bits (11–15) indicate the source of the external connection signal and the 12 least significant bits (0–11) as follows:

4 most sign bits	12 least sign bits
6	Register no, (range 0–254).
8	Register no-255, (range 255–4350).
9	Register no-4351, (range 4351– 8446).
AH	P-word (I/O address, 16 bits).
EH	Analogue input address.

Byte 46

Indicates the dead zone of the output signal. The value can vary between 0 and FFH. FFH corresponds to 6.2%.

Byte 47

Indicates the positive deviation limit. The value can be between 0 and FFH. FFH corresponds to 25.5%.

Byte 48

Indicates the negative deviation limit. The value can be between 0 and FFH. FFH corresponds to 25.5%.

Byte 49

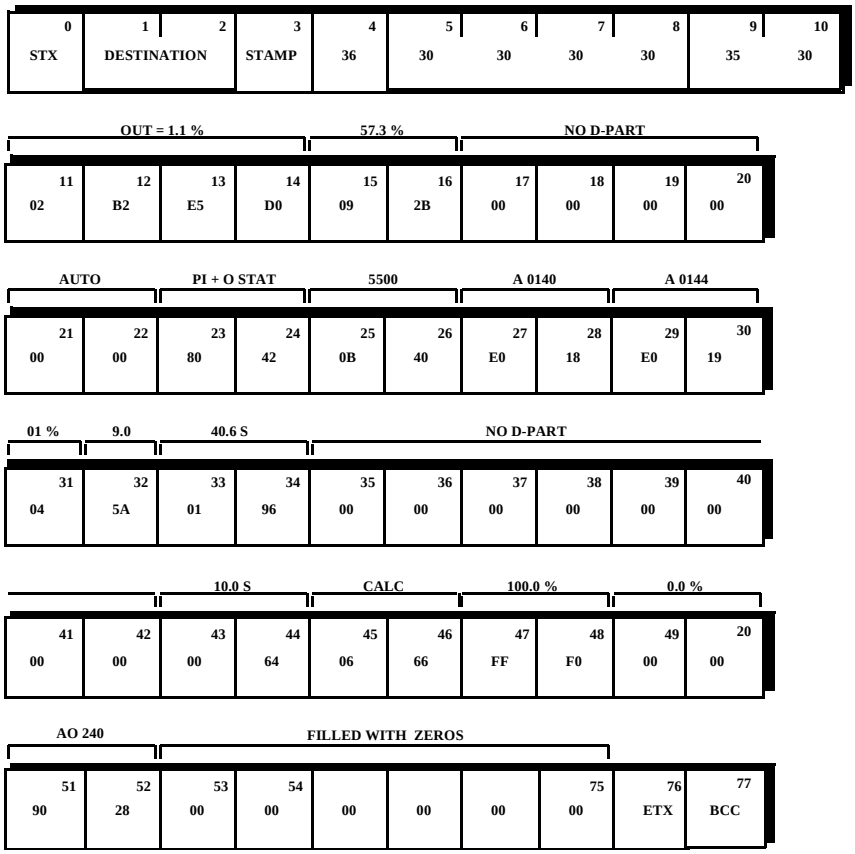
Not used.

Example

Assuming controller 0 in a system is programmed as follows:

Controller number	0
Controller type	PI
I/O for Auto mode	5500
Controller ON	5501
Interlock and open	5502
Interlock and close	5503
Analogue input	AI 0140
Setpoint	AI 0144
Dead zone, process value-setpoint (%)	0.1
Gain	9.0
Integration time	40S6
Hand Interlock Speed	10S0
Maximum output signal(%)	100.0
Minimum output signal(%)	0.0
Direct control	No
Analogue controller	Yes
Address of analogue output	A 0240

The message is as follows using binary communication:



Request dynamic data for controllers (Message type B)

Application

Used to request dynamic data for up to 4 controllers.

Message structure

The message structure is as follows:

0	1 2	3	4	5 6 7 8	9 10	11	12
STX	DESTINATION	STAMP	MESS. TYPE	CONTROLLER NUMBER	QUANTITY OF CONTROLLERS	ETX	BCC

Message type (byte 4)

The message code is “B”, i.e. ASCII code 42H.

Controller number (bytes 5–8)

Controller number which can be any value between 0 and 3FH, 62 (dec).

Number of controllers for which data is requested (bytes 9 and 10)

Indicates the quantity of controllers for which dynamic data is required. Minimum number is 1 and the maximum number is 4 using binary communication (2 using ASCII communication).

Example:

Request for dynamic data for 3 controllers beginning at controller 13. The message is as follows:

0	1 2	3	4	5	6	7	8	9	10	11	12
STX	DESTINATION	STAMP	42	30	30	30	44	30	33	ETX	BCC

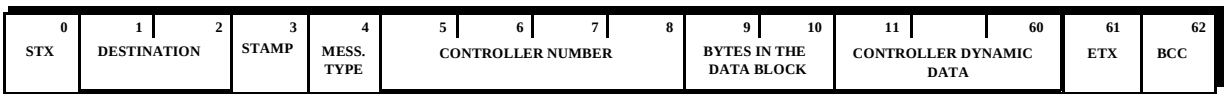
Transfer dynamic data for controllers (Message type C)

Application

Used to transfer dynamic data for up to 4 controllers.

Message structure

The message structure is as follows:



Message type (byte 4)

The message code is “C”, i.e. ASCII code 43H.

Controller number (bytes 5–8)

Indicates the controller number which can be any value between 0 and 3FH, 62 (dec).

Number of bytes in the data block (bytes 9–10)

The dynamic data for each controller transferred occupies 14 bytes using binary communication (28 bytes using ASCII communication). Since a data block can contain up to 64 bytes a single message can transfer dynamic data for up to 4 controllers.

Data block (byte 11–)

The numbering of each byte below refers to the first controller inside the data block only. To obtain the correct number for a byte in the message an 11 byte offset is added for the first controller and an additional 14 for each subsequent controller. It is assumed binary communication is being used.

Byte 0

8 bits describing the type of controller together with the controller number.The two most significant bits describe the type of controller as follows:

0 = P controller
1 = PD controller
2 = PI controller
3 = PID controller

The six most significant bits indicate the number of the controller being transferred. The value can be between 0 and 62.

If the value of byte 0 is FFH, there are no more controllers programmed in the system.

Byte 1

8 bits B7, B6, B5, B4, B3, B2, B1 and B0 indicating the following:

B0 Status for ramp and close.
B1 Status for ramp and open.
B2 Status for controller on.
B3 Status for Auto mode on.
B4 Status for digital output signal closed.
B5 Status for digital output signal open.
B6 Deviation alarm negative.
B7 Deviation alarm positive.

Bytes 2 and 3

Indicate the process value. The value can vary between 0 and FFFFH which corresponds to 0% to 100.0%.

Bytes 4 and 5

Indicate the set point value. The value can vary between 0 and FFFFH which corresponds to 0% to 100.0%.

Bytes 6 to 9 inclusive

Not used.

Bytes 10 and 11

16 bits indicating the current value of the signal which is the input to the D-part.

Bytes 12 and 13

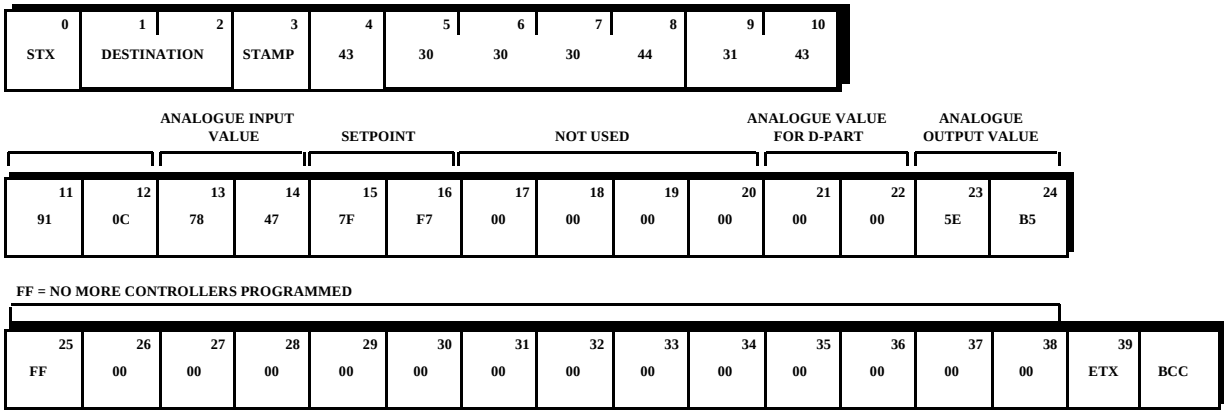
16 bits indicating the current status of the output signal from the controller. If it is an analogue controller these bits contain the analogue output value. If it is a digital controller with feedback the value of the feedback signal is indicated. For a controller without feedback the calculated value of this signal is indicated.

Example

Assuming a request is received for dynamic data for controllers 17 and 18. Controller 17 is the only controller programmed in the system and binary communication is being used. The dynamic data is as follows:

Controller type	PI with analogue output
Controller on	Yes
Auto mode	Yes
Process value	47.0%
Setpoint	50.0%
Output signal	37.0%

The message is as follows using binary communication:



2.6 Limiter modules

Request limiter modules (Message type 7)

Application

Used to request data for limiter modules. It is possible to request data for a maximum of 6 limiter modules using binary communication (3 using ASCII communication). The following data is received for each requested limiter module:

- Limiter module number.
- The type of signal to which the limiter module is connected (register, I/O-bit or analogue output).
- Register number, address of the digital input or output, address of the analogue input signal.
- Type of limiter module, i.e. module of the maximum or minimum type.
- Upper and lower limiter values.

Message structure

The message structure is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12
STX	DESTINATION		STAMP	MESS. TYPE	LIMIT VALUE MODULE NUMBER				QUANTITY OF MODULES		ETX	BCC

Message type (byte 4)

The message type is “7”, i.e. ASCII code 37H.

Limiter module number (bytes 5–8)

Indicates the number of the first limiter module requested; modules can be numbered in the range 0 to 254.

Quantity of requested limiter modules (bytes 9–10)

Data can be requested for 1–6 limiter modules using binary communication (1–3 using ASCII communication).

Example

Assuming 5 limiter modules are being requested and the number of the first module is 3. The message contents are as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12
STX	DESTINATION		STAMP	37	30	30	30	33	30	35	ETX	BCC

Transfer data for limiter modules (Message type 8)

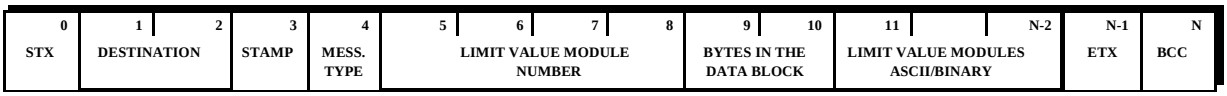
Application

Used to transfer data for a maximum of 6 limiter modules using binary communication or a maximum of 3 using ASCII communication. The following data is transferred for each module:

- limiter module number.
- Type of signal to which the limiter module is connected (register, I/O-bit or analogue output).
- Register number, address of the digital input or output, address of the analogue input signal.
- Type of limiter module, i.e. module of the maximum or minimum type.
- Upper and lower limiters.

Message structure

The message structure is as follows:



Message type (byte 4)

The message type is “8”, i.e. ASCII code 38H.

Limiter module number (bytes 5–8)

Indicates the number of the first limiter module to be transferred; modules can be numbered in the range 0 to 254.

Bytes in the data block (bytes 9–10)

Indicates the quantity of bytes contained in the data block. Each limiter module occupies 10 bytes when binary communication is used and 20 bytes for ASCII communication.

Limiter module data (bytes 11–)

Binary communication is assumed consequently 10 bytes are required to transfer the data. The functions of the bytes are as follows (byte number refers to inside the data block):

Byte 0

Limiter module’s number. Lowest permitted number is 0H and highest number is FEH, 254 (dec). If this byte contains the value FFH there are no other limiter modules programmed in the system. All following bytes in the data block have the value zero.

Bytes 1 and 2

16 bits indicating address and type of signal connected to the module. The remaining 12 bits indicate as follows:

4 most sign bits	12 least sign bits
6H	Register number.
AH	I/O bit (I/O address).
EH	Address of analogue input divided with 4.

Byte 3

Type of limiter module as follows:

< ASCII code 3CH, limiter module is min. value type.
> ASCII code 3EH, limiter module is max. value type.

Bytes 4 and 5

The 12 least significant bits indicate the upper limit. Lowest permitted value is 0. The highest permitted value is FFFH which corresponds to 100%. The four most significant bits are not used and are set to 0.

Bytes 6 and 7

The 12 least significant bits indicate the lower limit. Lowest permitted value is 0. The highest permitted value is the upper limiter. The four most significant bits are not used and are set to 0.

Bytes 8 and 9

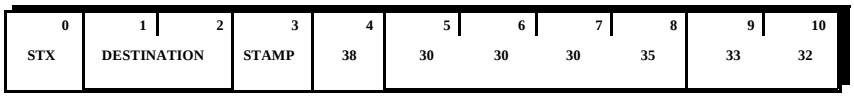
Not used. Set to zero.

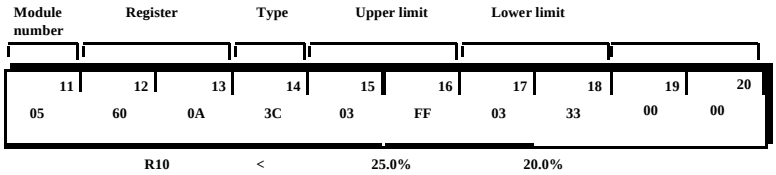
Example

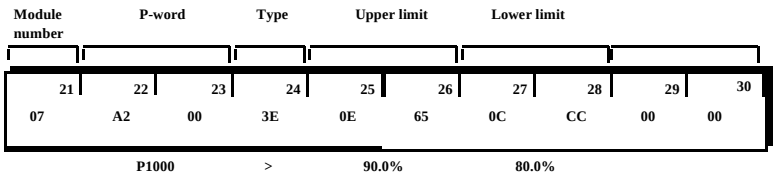
A request for 5 limiter modules starting at module number 5. The system includes only three modules 5, 7 and 8 whose data is as follows:

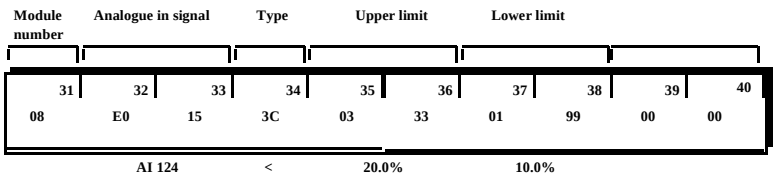
Limiter module	Connection	Type	Low limit	High limit
5	Register 10	<	20.0%	25.0%
7	P 1000	>	80.0%	90.0%
8	A 124	<	10.0%	20.0%

The message is as follows using binary communication:

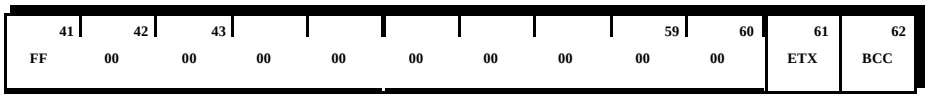








No other limit value modules programmed in the system



2.7 Analogue input signals

Request analogue input signals (Message type 9)

Application

Used to request the value of an analogue input signal. The number of values requested can vary between 1 and 32 using binary communication (1–16 using ASCII communication). The address range permitted is system dependent.

Message structure

The message structure is as follows:

0	1 2	3	4	5 6 7 8	9 10	11	12
STX	DESTINATION	STAMP	MESS. TYPE	ANALOGUE INPUT SIGNAL ADDRESS	NUMBER OF ANALOGUE INPUTS	ETX	BCC

Message type (byte 4)

The message type is “9”, i.e. ASCII code 39H.

First analogue signal address (bytes 5–8)

Analogue signals always have an even address exactly divisible by 4. The lowest address which can be requested is 0. The highest address is system dependent.

Number of requested values (bytes 9–10)

The number of requested values can vary between 1–32 using binary communication (1–16 using ASCII communication).

Example

Request the value of 5 analogue input signals starting at the signal with address 640 (octal), i.e. 01A0H. The message is as follows:

0	1 2	3	4	5 6 7 8	9 10	11	12
STX	DESTINATION	STAMP	39	30 31 41 30	30 35	ETX	BCC

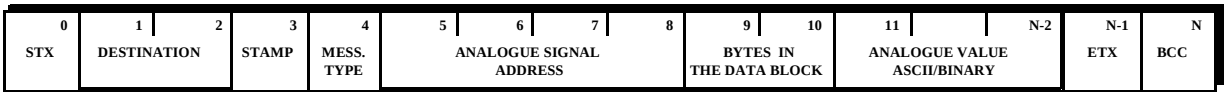
Transfer of analogue input signals (Message type A)

Application

Used to transfer the value of one or more analogue signals. The number of values transferred can vary between 1 and 32 using binary communication (1–16 using ASCII communication). The address range permitted is system dependent.

Message structure

The message structure is as follows:



Message type (byte 4)

The message code is “A”, i.e. ASCII code 41H.

First analogue signal address (byte 5–8)

Analogue signals always have an even address exactly divisible by 4. The lowest address which can be requested is 0. The highest address is system dependent.

Bytes in the data block (bytes 9–10)

Indicates the quantity of bytes being transferred in the data block. Each analogue signal requires 2 bytes using binary communication and 4 bytes using ASCII communication.

Data block (bytes 11–n)

This example assumes binary communication is being used. Each analogue input occupies 2 bytes, i.e. 16 bits without a sign bit. Older control systems obtain only 12 bits from analogue to digital converters. These 12 bits are expanded to 16 bits by copying the 4 most significant bits of the 12-bit number to the 4 least significant bits of the 16-bit number, see following example:

000 (hex 12 bits)	= 0000 (hex 16 bits)
123	= 1231
FFF	= FFFF

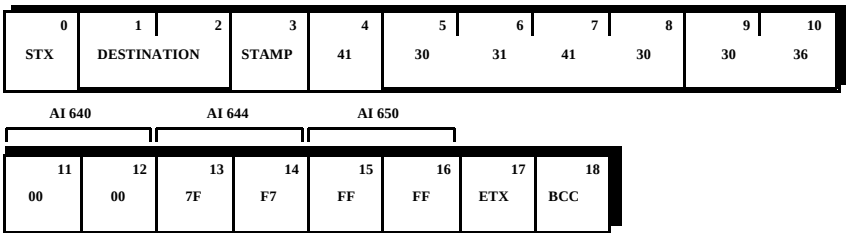
Later ABB Automation systems receive 16 bits direct from analogue to digital converters.

Example

3 analogue signals are to be transferred starting at address 640 (octal), 01A0H. The analogue signals have the following values:

A 640 = 0.0%
A 644 = 50.0%
A 650 = 100.0%

The message is as follows using binary communication:



The value for input signal AI 644 is obtained as follows:

50% of full signal after A/D conversion is 7FF (12 bits).This is expanded to 16 bits by taking the 4 most significant bits of the 12-bit value and copying them to the 4 least significant bits of the 16-bit value, i.e. 7FF7.

2.8 Alarm text

Request alarm text (old alarm text function) (Message type E)

Application

Used to request alarm text from the alarm buffer.

Message structure

The message structure is as follows:

0	1 2	3	4	5	6	7	8	9 10	11	12
STX	DESTINATION	STAMP	MESS. TYPE	PART1/ PART2	NOT USED	ALARM TEXT NO.	FIX/ VAR.	QUANTITY OF BYTES REQUESTED	ETX	BCC

Message type (byte 4)

The message type is “E”, i.e ASCII code 45H.

Part of alarm text (byte 5)

The maximum number of characters that can be requested at one time is 64. Byte 5 is used therefore to indicate whether the first or last part of the text is being requested. The value 30H indicates the first half containing 64 characters is requested. If the value is 31H the second half containing 64 characters is requested.

Byte 6

Byte 6 is not used and is always set to 30H.

Last alarm acknowledged (byte 7)

The value can vary between 0 and 9, i.e. ASCII codes vary between 30H and 39H. When a Master is started, after system reset for example, the value is 0, and subsequently varies between 1 and 9 to indicate the last alarm acknowledged.

Fixed/variable alarm text length (byte 8)

Enables the Master to determine if the reply alarm text is to contain a fixed or variable number of characters, i.e. 30H variable, 31H fixed.

Quantity of bytes expected in the reply data block (bytes 9–10)

If byte 8 has the value 30H indicating the reply can contain a variable number of bytes, bytes 9 and 10 are not used and are set to the value 30H. If on the other hand, the number of bytes is fixed, byte 8 has the value 31H, and bytes 9 and 10 indicate the quantity of bytes contained in the reply alarm text.

Example

Request for alarm text. The Master has just started and transfer of 6 characters of alarm text is requested. The message is as follows:

0	1 2	3	4	5	6	7	8	9 10	11	12
STX	DESTINATION	STAMP	45	30	30	32	31	30 36	ETX	BCC

Transfer alarm text (old alarm text function) or acknowledge alarm text (Message type F)

Application

Used to transfer alarm text from the system's alarm buffer or acknowledge an alarm. The transfer removes the alarm text from the alarm buffer consequently the text cannot be transferred to several supervisory systems that require the information. Also, a supervisory system cannot detect if an alarm has been acknowledged by another supervisory system. To eliminate this problem a new alarm messages have been introduced. See message types ":" and ";".

Message structure

The message structure is as follows for alarm acknowledge:

0	1 2	3	4	5	6	7	8	9	10	11	12
STX	DESTINATION	STAMP	MESS. TYPE	NOT USED	NOT USED	ALARM TEXT NO.	NOT USED	NOT USED	NOT USED	ETX	BCC

...and as follows for transfer of the alarm text.

0	1 2	3	4	5	6	7	8	9 10	11 N-2	N-1	N
STX	DESTINATION	STAMP	MESS. TYPE	TEXT COMP-LETE	QUEUE STATUS	ALARM TEXT NO.	BUFFER STATUS	QUANTITY OF BYTES REQUESTED	ALARM TEXT	ETX	BCC

Message type (byte 4)

The message type is "F", i.e. ASCII code 46H.

Bytes 5–10

These bytes are used differently depending on whether the message is an acknowledgement of alarm text or a transfer of alarm text in reply to a request.

Acknowledge alarm text – Bytes 5 and 6

Not used. The value is set to zero, i.e. ASCII code 30H.

Acknowledge alarm text – Byte 7

Indicates the alarm text being acknowledged. The value can vary between 1 and 9, i.e. ASCII codes 30H–39H.

Acknowledge alarm text – Bytes 8, 9 and 10

Not used. The value is set to zero, i.e. ASCII code 30H.

Transfer alarm text – Byte 5

Indicates if the alarm text is complete or if the Master must perform a second request to obtain the complete text:

30H Alarm text is complete.
31H Alarm text is incomplete (longer than 64 characters). The master must send a second request to obtain the remainder.

Transfer alarm text – Byte 6

Indicates the state of the alarm queue in the system that is replying as follows:

30H Alarm queue in the system replying is not full.
31H Alarm queue in the system replying is full.

Byte 7

Indicates the alarm text number which can be considered as a form of alarm stamp which enables the master to determine when new alarm text is being transferred. The value is normally between 1 and 8, ASCII 30H–38H, but if the Slave has just started the value is 9, ASCII 39H.

Byte 8

Indicates if the alarm buffer in the Slave is empty or contains more alarm text:

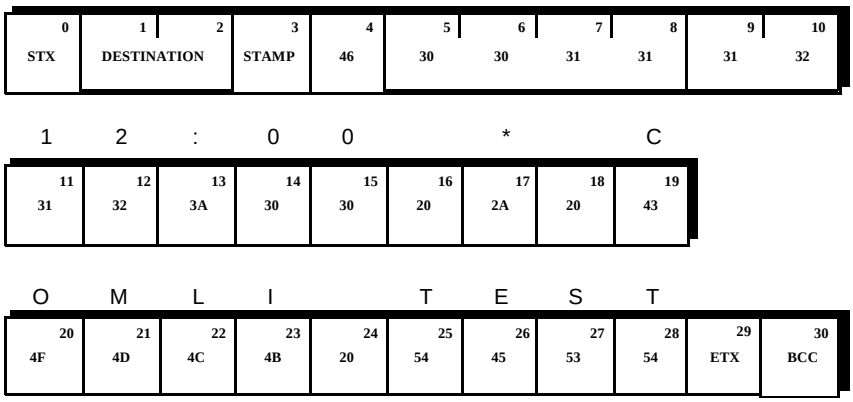
30H	Slave alarm buffer is empty.
31H	Slave alarm buffer contains alarm text.

Bytes 9 and 10

Indicate the number of bytes in the data block. The number can vary when the request from the Master contains the value 30H in byte 8. If this is not the case the Slave replies with the number of bytes requested by the Master. If there are no alarms in the queue, or the alarm text is shorter than the requested number of characters, the data block is filled or topped up with zeros, i.e. ASCII code 30H. If there is no alarm text in the queue bytes 9 and 10 have the value 30H.

Example

The alarm text “12:00 * COMLI TEST” is to be transferred. Thus the number of bytes (and characters) is 18. It is assumed that the request has not specified a fixed number of data block bytes (characters). The message is as follows:



New alarm text function

General

Two new message types have been introduced to request and transfer alarm text to supervisory systems such as SattGraph 1200. One difference in the new messages is that the alarm text is not removed from the alarm buffer when it is transferred as is the case with the “E” and “F” messages. This means that the text can be read and displayed several times by different supervisory systems. Another difference is that an alarm that is acknowledged in one of the supervisory systems results in the alarm being transferred to the subsystems’ alarm buffer. Thus other supervisory systems can detect that the alarm has been acknowledged and display this in their alarm lists. The new message types also enable predetermined alarms to be blocked.

Acknowledge

If, for example, an alarm refers to a valve, it may be necessary for the alarm condition to be indicated at the valve symbol in the application picture and not only in the alarm list, perhaps as follows:

Active unacknowledged alarm	flashing valve symbol
Active acknowledged alarm	non flashing alarm symbol
Normal condition	different colour from the above

This facility can be provided using memory cells in the subsystem, part for the alarm and part for alarm acknowledgement.

When the alarm text message is transferred to the supervisory system an acknowledge address may be included. This is the address of a memory cell that is to be set when the alarm acknowledge message is received. As soon as the alarm condition ceases this cell must be reset. This occurs automatically in some system but must be performed by the PLC program in other systems. Thus the supervisory system can determine the condition of the valve from the status of the alarm address and the status of the alarm acknowledge cell.
In SattCon systems the acknowledge address is defined using the command $ACKxxxx. The command $ACKxxxx is included in the alarm text and translates into the acknowledge address when the alarm text message is transferred.

Blocking

Alarms can be blocked by the supervisory system. This results in a message from the supervisory system containing a memory cell address which is in turn set to ’1’. The subsystem includes a PLC program which resets selected alarm monitoring memory cells.

Alarm sequence

A typical alarm sequence between a SattCon system and one or more SattGraph 1200 systems is as follows:

- The alarm text is defined “1000 Valve not operating $ACK1001”
- An alarm occurs, i.e. address 1000 is set to ’1’.
- The alarm is stored in the alarm buffer.
- SattGraph 1200 requests the alarm and receives the alarm text reference number, address 1000, address 1001, the time and the alarm text.
- The alarm is entered in the SattGraph alarm list.
- SattGraph 1200 uses the status of the alarm and acknowledge memory cells to display the condition of the valve in the application picture.
- The operator acknowledges the alarm.
- SattGraph 1200 transmits an acknowledgement message containing addresses 1000 and 1001 to the SattCon system. Acknowledgement is entered in the alarm buffer. If address 1000 is set to ’1’ address 1001 is also set to ’1’, otherwise 1001 is set to ’0’,
- If several SattGraph 1200 systems are connected to the SattCon system, routine interrogation of the alarm buffer eventually results in all systems detecting acknowledgement for entering in their alarm lists. Also, by monitoring the status of the alarm and acknowledgement memory cells, the correct condition of the valve can be displayed in all application pictures.
- The acknowledge is reset when the alarm cell is reset by the PLC program or automatically by the system.

Request alarm text (Message type :)

Application

Used by the supervisory system to request alarm text from the alarm buffer in the subsystem.

Message structure

The message structure is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12
STX	DESTINATION		STAMP	MESS. TYPE	REFERENCE NUMBER				PART1/ PART2	FIX/ VAR.	ETX	BCC

Message type (byte 4)

The message type is ":", i.e. ASCII code 3AH.

Reference number (bytes 5–8)

Indicates the reference number for the alarm text requested. The number can vary between 0 and 65535.

Part 1/Part 2 (byte 9)

Indicates if the first or second half of the alarm text is being requested; the value 30H indicates the first half and 31H the second half.

Fixed /Variable length alarm text (byte 10)

Indicates if the length of the alarm text requested is fixed or variable:

30H	alarm text can contain a variable number of characters.
31H	alarm text must always contain 64 characters.
32H	the alarm text must always contain 64 characters and the request is only valid for still active or not acknowledged alarms.

To ensure backward compatibility, systems not supporting this addition, will not answer to it.

Example

A request for alarm text reference number 35385. The alarm text can contain a variable number of characters and it is the first part of the text which is being requested. The message is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12
STX	DESTINATION		STAMP	3A	38	41	33	39	30	30	ETX	BCC

Transfer alarm text (Message type ;)

Application

Used to transfer alarm text from the alarm buffer. The transfer does not result in the alarm text being removed from the alarm buffer.

Message structure

The message structure is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	N-2	N-1	N
STX	DESTINATION		STAMP	MESS. TYPE	TEXT COMP-LETE	TEXT AS REQU-ESTED	NOT USED	NOT USED	BYTES IN THE DATA BLOCK		ALARM TEXT		ETX	BCC

Message type (byte 4)

The message type is ";", i.e. ASCII code 3BH.

Alarm text transferred complete or in 2 parts (byte 5)

Indicates if the alarm text is complete or if the Master must perform a second request to obtain the complete text:

30H Alarm text is complete.
31H Alarm text is incomplete (longer than 64 characters). The master must send a second request to obtain the remainder.

Alarm text is the requested text (byte 6)

Indicates to the system making the request if the alarm text is the requested text or a different alarm text is being transferred:

30H The requested alarm text is being transferred.
31H The requested alarm text is not being transferred, the next alarm in the alarm buffer is being transferred instead.
32H The requested alarm text is not being transferred. All alarm text in the alarm buffer has a lower reference number. The next alarm text in the buffer is being transferred instead.

Bytes 7–8

Not used. Set to value 30H.

Bytes in the data block (bytes 9–10)

Indicates the quantity of bytes in the data block. Min. is 0 and max. is 64 bytes.

Data block (bytes 11–)

The data block is always transferred using binary communication. Alarm text is transferred provided the alarm buffer is not empty. The functions of the first 14 bytes in the data block are described below. The numbering applies to inside the data block only. To determine the correct numbering in the message the offset 11 bytes is added.

Note. If the alarm buffer is empty byte 6 should be set to 32H. Furthermore the data area should be omitted, i.e. byte 9 and 10 is set to 00H.

Byte 0 and 1

Indicates the reference number. Value between 0–65535 (dec).

Byte 2

Indicates the status of the digital I/O bit monitored by the alarm text program. The indications are as follows for the various conditions:

30	Digital I/O monitored for all changes has changed from 1 -> 0.
31	Digital I/O monitored for all changes has changed from 0 -> 1.
32	Digital I/O monitored for changes to ON only has changed from 0 -> 1.
33	Digital I/O monitored for changes to OFF only has changed from 1 -> 0.
34	Acknowledgement I/O-bit.

Bytes 3 and 4

Indicates the address of the I/O bit for which the alarm text is transferred.

Bytes 5 and 6

Indicate the address of the I/O-bit used to acknowledge the alarm.

Byte 7

Year.

Byte 8

Month.

Byte 9

Day

Byte 10

Hour.

Byte 11

Minute.

Byte 12

Second.

Byte 13

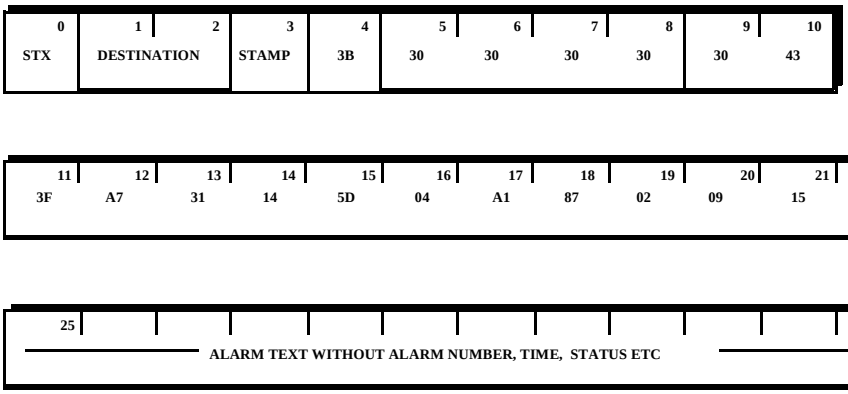
Tenths of a second.

Byte 14–N

Alarm text without the alarm number, status, etc.

Example

Transfer alarm text for address 12135 (octal), 145DH with acknowledge bit from address 2241 (octal), 04A1H. The alarm cell has changed from ’0’ to ’1’. The alarm text has the reference number 16295 (dec), 3FA7H. The date is 9/2/87 and the time 15:34:29:06. The message is as follows



Transfer acknowledge and blocking conditions (Message type ;)

Application

Used to acknowledge and block an alarm.

Message structure

The message structure is as follows:

0	1 2	3	4	5	6 7 8	9 10	11 N-2	N-1	N
STX	DESTINATION	STAMP	MESS. TYPE	ACKN./ BLOCK	NOT USED	QUANTITY OF BYTES	ADDRESS OF ACKNOWLEDGED OR BLOCKED ALARM	ETX	BCC

Message type (byte 4)

The message type is ";", i.e. ASCII code 3BH.

Acknowledge/blocking (byte 5)

Indicates if alarm acknowledge information or alarm blocking information that is being transferred:

30H Alarm acknowledge
31H Alarm blocking

Bytes 6–8

Not used. Set to value 30H.

Quantity of bytes (bytes 9–10)

Indicates the quantity of bytes in the data block. The minimum is 0 and the maximum is 64.

Data block (bytes 11–)

Binary communication is always used for transferring the data block.

Acknowledge information

The acknowledge information transferred comprises the address of the alarm being acknowledged together with, if specified, the acknowledge address. If no acknowledge address is specified value FFH is inserted. Each address requires 2 bytes, i.e. the acknowledge information requires a total of 4 bytes.

Alarm blocking information

The alarm blocking information comprises only the address of the alarm to be blocked. This requires 2 bytes.

Example

Transfer acknowledge information for alarms with the addresses 15511 (octal) and 15512 (octal), 1B49H and 1B4AH respectively. Acknowledge addresses are also specified, i.e. 7473 (octal) and 7474 (octal), 0F3BH and 0F3CH respectively. The message is as follows:

0	1	2	3	4	5	6	7	8	9	10
STX	DESTINATION		STAMP	3B	30	30	30	30	30	38

11	12	13	14	15	16	17	18	19	20
1B	49	0F	3B	1B	4A	0F	3C	ETX	BCC

Example

Transfer blocking information for the alarm with the address 12300 (octal) and 12301 (octal), 14C0H and 14C1H. The message is as follows:

0	1	2	3	4	5	6	7	8	9	10
STX	DESTINATION		STAMP	3B	31	30	30	30	30	34

11	12	13	14	15	16
14	C0	14	C1	ETX	BCC

2.9 Measuring ranges and units

Request measuring ranges and units (Message type G)

Application

Used to request up to 4 measuring ranges and the associated units from a sub-system. The measuring range can be requested for analogue input signals, groups of 16 I/O bits (P-words) and for registers.

Message structure

The message structure is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12
STX	DESTINATION		STAMP	MESS. TYPE	TYPE OF SIGNAL	SIGNAL ADDRESS			QUANTITY OF MEASUREMENT RANGES		ETX	BCC

Message type (byte 4)

The message type is “G”, i.e. ASCII code 47H.

Type of signal (byte 5)

Indicates the type of measuring signal:

36H	Register.
41H	P-word (16 I/O-bits).
45H	Analogue input.

Signal address (byte 6–8)

Indicates the address of the signal for which the measuring range is requested. The significance of these bytes depends on the type of signal defined in byte 5 as follows:

36H	Register number
41H	I/O-address
45H	Address of analogue input signal.

Quantity of requested measuring ranges (bytes 9–10).

Indicates the quantity of measuring ranges and types of signal requested. The lowest permitted number is 1.The highest number when binary communication is being used is 4 (2 when using ASCII communication).

Example

A request for 4 measuring ranges for registers starting at register number 20. The message is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12
STX	DESTINATION		STAMP	47	36	30	31	34	30	34	ETX	BCC

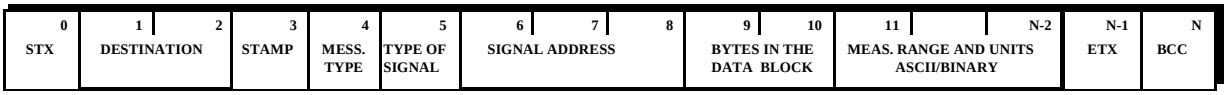
Transfer measuring ranges and units (Message type H)

Application

Used to transfer up to 4 measuring ranges and units from a subsystem. Measuring ranges can be transferred for analogue input signals, groups of 16 I/O bits (P-words) and for registers.

Message structure

The message structure is as follows:



Message type (byte 4)

The message type is “H”, i.e. ASCII code 48H.

Type of signal (byte 5)

Indicates the type of measuring signal:

36H	Register.
41H	P-word (16 I/O-bits).
45H	Analogue input.

Signal address (bytes 6–8)

Indicates the address of the signal for which the measuring range is requested. The significance of these bytes depends on the type of signal defined in byte 5 as follows:

36H	Register number
41H	I/O-address
45H	Address of analogue input signal.

Bytes in the data block (bytes 9–10)

Indicates the quantity of bytes in the data block. The measuring range and type of signal occupy 13 bytes using binary communication and 26 bytes using ASCII communication.

Data block (bytes 11–)

The functions of the 13 bytes transferred (assuming binary communication is being used) are as follows:

Bytes 0 and 1

16 bits stipulating the address and type of signal to which the measuring range is connected. The four most significant bits indicate the following:

4 most sign bits	12 least sign bits
6	Measuring range connected to a register (number).
AH	Measuring range connected to a P-word (I/O address, 16 bits).
EH	Measuring range connected to an analogue input address.

Bytes 2 and 3

Total of 16 bits including a sign bit indicating the lowest value of the measuring range.

The lowest permitted measured value is 8000H, –32768 (dec) and the highest is 7FFFH, 32767 (dec).

Bytes 4 and 5

Total of 16 bits indicating the difference between the lower and upper limits, i.e the measuring range. Only positive values are permitted and the value can vary between 0H and FFFFH. 0–65535 (dec).

Byte 6

Indicates the number of decimals in the lower and upper limits. Values between 0–5 are permitted.

Bytes 7 to 12

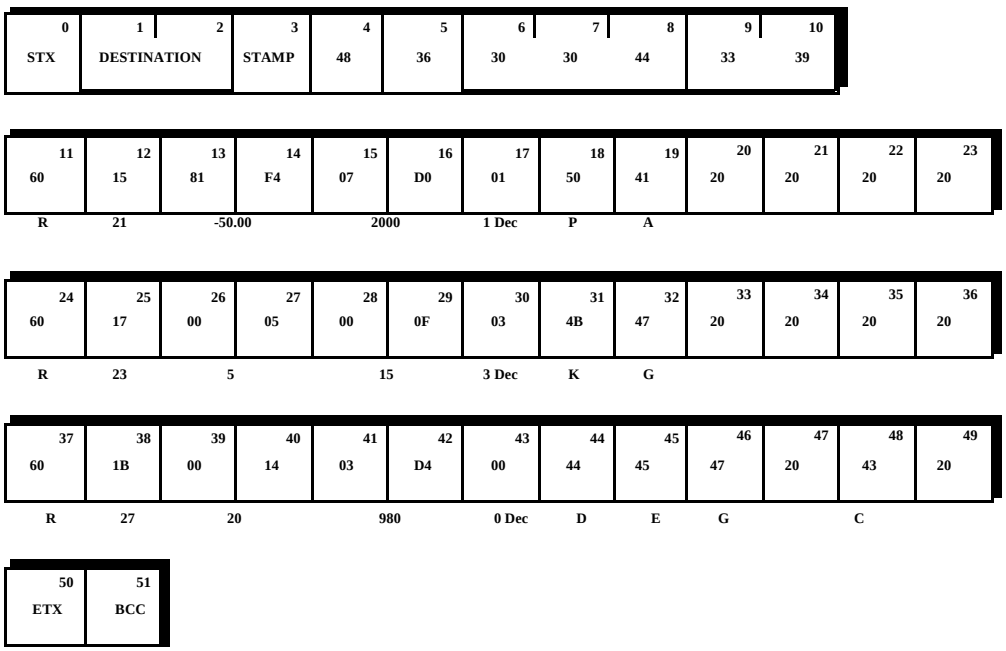
Space for 6 ASCII characters indicating the units.

Example

Transfer the measuring ranges and measuring units for the 3 registers shown below:

R 21	–50.0	150.0	PA
R 23	0.005	0.020	KG
R 27	20	1000	DEG C

The message is as follows:



2.10 Date and time

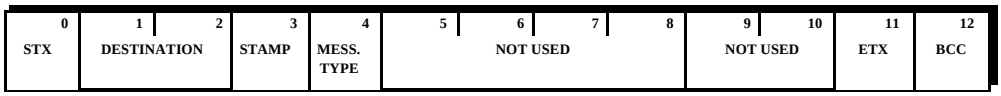
Request date and time (Message type I)

Application

This message is used to request transfer of the real time clock information from a subsystem to the supervisory system. The date and time requested contains the year, month, hours, minutes and seconds. Note that if this message is used to synchronise systems, time differences can occur due to the time required to transfer and process the data. Synchronising of the real time clocks is best performed by connecting an electrically isolated output from one of the systems to electrically isolated inputs in the other systems. When the output is activated PLC program routines in the respective systems can be used to transfer the same predetermined values from a group of registers to the real time clock in each system.

Message structure

The message structure is as follows:



Message type (byte 4)

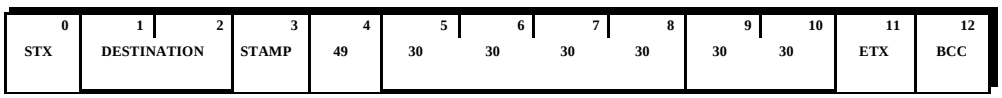
The message type is “I”, i.e. ASCII code 49H.

Byte 5–10

Not used. The values are always “0”, i.e. ACSII code 30H.

Example

A message requesting the date and time is as follows:



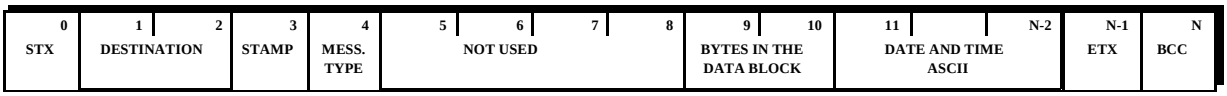
Transfer of date and time (Message type J)

Application

This message is used to transfer the real time clock information between systems. The date and time contains the year, month, hours, minutes and seconds. Note that if this message is used to synchronise systems time differences can occur due to the time required to transfer and process the data. Synchronising of the real time clocks is best performed by connecting an electrically isolated output from one of the systems to electrically isolated inputs in the other system. When the output is activated PLC program routines in the respective systems can be used to transfer the same predetermined values from a group of registers to the real time clock in each system.

Message structure

The message structure is as follows:



Message type (byte 4)

The message type is “J”, i.e. ASCII code 4AH.

Bytes 5–8

Not used. The values are always “0”, i.e. ASCII code 30H.

Bytes in the data block (byte 9–10)

Quantity of bytes transferred in the data block. The quantity of bytes is always the same irrespective of whether binary or ASCII communication is being used. The value of these bytes is therefore always 0CH, i.e. 12 (dec).

Year (bytes 11–12)

Year in ASCII code.

Month (bytes 13–14)

Month in ASCII code.

Day (bytes 15–16)

Day in ASCII code.

Hours (bytes 17–18)

Hours in ASCII code.

Minutes (bytes 19–20)

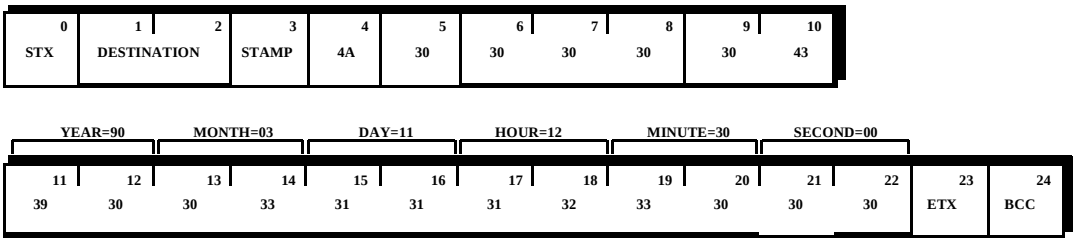
Minutes in ASCII code.

Seconds (bytes 21–22)

Seconds in ASCII code.

Example

Transfer date: 900311, year: 90, month: 03, date: 11, and time: 12:30:00.
The message is as follows:



2.11 Floating point register

Request floating point register (Message type O)

Application

Used to request up to 16 floating point registers. Only SattCon31using program module CALC31 can perform floating point arithmetic. Floating point registers numbered between 0 and 255 can be requested.

Message structure

The message structure is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12
STX	DESTINATION		STAMP	MESS. TYPE	FLOATING POINT REGISTER NUMBER				QUANTITY OF FLOATING POINT REGISTERS		ETX	BCC

Message type (byte 4)

The message type is “O”, i.e ASCII code 4FH.

First floating point register (bytes 5–8)

Indicates the number of the first floating point register to be transferred. Any register number in the range 0 to 255 can be used.

Quantity of floating point registers requested (bytes 9–10)

Indicates the quantity of floating point registers requested. A minimum of 1 and a maximum of 16 (8 using ASCII communication) can be requested.

Example

Request 16 floating point registers starting at register number 73. The message is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12
STX	DESTINATION		STAMP	4F	30	30	34	39	31	30	ETX	BCC

Transfer floating point registers (Message type P)

Application

Used to transfer up to 16 floating point registers. Only SattCon31using program module CALC31 can perform floating point arithmetic. Floating point registers numbered between 0 and 255 can be transferred.

Message structure

The message structure is as follows:

0	1 2	3	4	5 6 7 8	9 10	11 N-2	N-1	N
STX	DESTINATION	STAMP	MESS. TYPE	FLOATING POINT REGISTER NUMBER	BYTES IN THE DATA BLOCK	FLOATING POINT VALUE (IEEE STANDARD)	ETX	BCC

Message type (byte 4)

The message type is “P”, i.e. ASCII code 50H.

First floating point register (bytes 5–8)

Indicates the number of the first floating point register to be transferred. Any register number in the range 0 to 255 can be used.

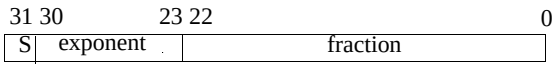
Bytes in the data block (bytes 9–10)

Indicates the quantity of bytes in the data block. Each register (32 bits) to be transferred occupies 4 bytes using binary communication (8 bytes using ASCII communication). This allows up to 16 registers (8 using ASCII communication) to be transferred in one message.

Data block (bytes 11–)

Each floating point register is transferred according to the IEEE-standard and occupies 4 bytes (8 bytes using ASCII communication).

IEEE-standard data block



S = sign bit

Example

Transfer 16 floating point registers starting at register number 73. The message is as follows:

0	1 2	3	4	5	6	7	8	9	10	11 N-2	N-1	N
STX	DESTINATION	STAMP	50	30	30	34	39	34	30	FLOATING POINT VALUE (IEEE STANDARD)	ETX	BCC

2.12 Trend curve data

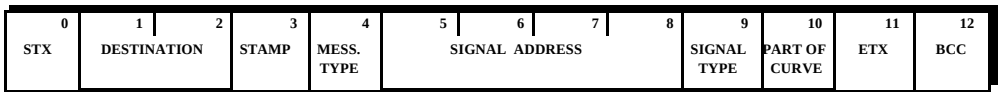
Request trend curve data (Message type Q)

Application

Used to request measured values. Note that this message has an extended function and it is now possible to stipulate a window start when requesting a trend curve.

Message structure

The message structure is as follows:



Message type (byte 4)

The message type is “Q”, i.e. ASCII code 51H.

Signal address (bytes 5–8)

Indicates the address of the signal for the trend curve being requested. The address can be a register number, a P-word address or the address of an analogue input signal. The lowest permitted address is 0 and the highest depends on the type of signal and the system.

Signal type (byte 9)

There are three signal type alternatives:

36H	Register.
41H	P-word.
45H	Analogue input signal.

Part of trend curve (byte 10)

Each signal can have 6 trend curves with different sampling times and 120 values in each curve. Since, when using ASCII communication, 4 messages are required to transfer 120 measured values, the trend curve is divided into four parts. When binary communication is being used only parts 1 and 3 of the trend curve are requested since 60 values can be transferred in each message.

The following table shows the value of byte 10 to request the various parts of the curves associated with a given signal address:

Assuming the value for trend curve 3 is being requested and binary communication is being used, the table indicates that byte 10 must be given the value 38H (ASCII code for 8) in the first message and the value 41H (ASCII code for A) in the second message. Hexadecimal A is the same as decimal 10.

Trend curve	1	2	3	4	5	6
Part 1	0	4	8	12	16	20
Part 2	1	5	9	13	17	21
Part 3	2	6	10	14	18	22
Part 4	3	7	11	15	19	23

Example

Request trend curve number 2 for the P-word with the address 80H, 200 (octal). Binary communication is assumed. To transfer all measured values requires the following two messages:

Message 1:

0	1	2	3	4	5	6	7	8	9	10	11	12
STX	DESTINATION		STAMP	51	30	30	38	30	41	34	ETX	BCC

Message 2:

0	1	2	3	4	5	6	7	8	9	10	11	12
STX	DESTINATION		STAMP	51	30	30	38	30	41	36	ETX	BCC

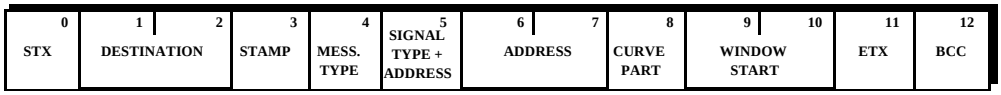
Request trend curve data with window start indication (Message type Q)

Application

Used to request measured values. The required window start can be included. The reply is similar to the Request for trend curve data without window start.

Message structure

The message structure is as follows:



Message type (byte 4)

The message type is “Q”, i.e. ASCII code 51H.

Address and signal type (bytes 5–7)

Previously the value of bit 6 of byte 5 was zero. This is now changed to use bit 6 to select either the old or new message definition; if bit 6 is zero the old definition is used but if bit 6 is “1” the definition described below is used. Bit 7 is always zero in all bytes.

Bit 7	6	5	4	3	2	1	0
0	1	x	Signal type			Address	

Byte 5

Bit 7	6	5	4	3	2	1	0
0	1	Address					

Byte 6

Bit 7	6	5	4	3	2	1	0
0	1	Address					

Byte 7

Signal type

The type of signal is indicated as follows:

111	Analogue input signal.
110	P-word.
011	Register.

Address

Indicates the address of the signal for the trend curve being requested. The address can be a register number, a P-word address or the address of an analogue input signal. The lowest permitted address is 0 and the highest depends on the type of signal and the system as follows:

0–9FC	Analogue input signal	A0–A4774
0–FF0	P-word	P0–P7760
0–20FE	Register	R0–R8446

Curve number and curve part (byte 8)

Bit 7	6	5	4	3	2	1	0
0	1	x	Curve number			Curve part	

Byte 8

Curve number

The curve number indicates one of six sampling times/methods as follows:

000 Fixed sampling time number 1.
001 Fixed sampling time number 2.
010 Fixed sampling time number 3.
011 Fixed sampling time number 4.
100 Freely selected sampling time.
101 Event controlled sampling.

Curve part

The 120 measured values transferred are divided into two parts when using binary communication and four parts when using ASCII communication. The indications are as follows:

00 Part 1.
01 Part 2 if ASCII communication.
10 Part 3 if ASCII communication, Part 2 if binary.
11 Part 4 if ASCII communication.

Window start (bytes 9–10)

Bit 7	6	5	4	3	2	1	0
0	1	x	x	Window start			

Byte 9

Bit 7	6	5	4	3	2	1	0
0	1	Window start					

Byte 10

Indicates the age, expressed as the number of measured values, of the most recent value to be transferred. The value can vary between 0–3E7H.

Example

Request the measured values for register 23 (dec). Curve number 3 is requested and the most recent value sampled is 28 measured values old. The message is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12
STX	DESTINATION		STAMP	51	4C	40	57	48	40	1C	ETX	BCC

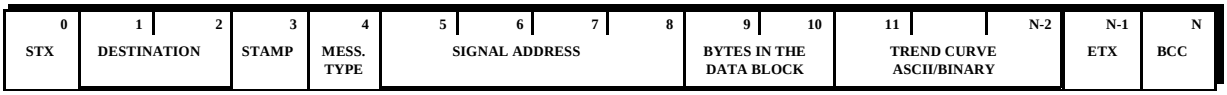
Transfer trend curve data (Message type R)

Application

Used to transfer trend curve data.

Message structure

The message structure is as follows:



Message type (byte 4)

The message type is “R”, i.e. ASCII code 52H.

Signal address (bytes 5–8)

Indicates the address of the signal for the trend curve being requested. The address can be a register number, a P-word address or the address of an analogue input signal. The lowest permitted address is 0 and the highest depends on the type of signal and the system.

Bytes in the data block (bytes 9–10)

Indicates the quantity of bytes in the data block. The quantity is always the same, 64 bytes, irrespective of whether binary or ASCII communication is being used. Bytes 9 and 10 always have the values 34H and 30H respectively.

Data block when using binary communication (bytes 11–74)

The numbering applies to inside the data block only. To determine the correct byte number in the message an offset of 11 bytes is added.

Byte 0

Indicates the signal type. There are three alternatives:

36H	Register.
41H	P-word.
45H	Analogue input signal.

Byte 1

Indicates the part of the trend curve to be transferred. Each signal can have 6 trend curves with different sampling times and 120 values in each curve. Since, when using ASCII communication, 4 messages are required to transfer 120 measured values, the trend curve is divided into four parts. When binary communication is being used only parts 1 and 3 of the trend curve are requested since 60 values can be transferred in each message. See also Request trend curve data for further information.

Bytes 2 and 3

Total of 16 bits indicating the sampling interval for collecting the trend curve data.

Lowest permitted value = 1H = (0.1 second)

Highest permitted value = FFFFH = 1 h, 49 m and 13.5 s

If trend curve number 6 (event controlled sampling) is transferred the sampling interval is always FFFFH.

A zero value indicates that the requested trend curve is not programmed.

Byte 4

Total of 8 bits containing the measured value for the trend curve.The first byte transferred contains the oldest value.

Lowest permitted value = 0H = 0.0%

Highest permitted value = FFH = 100%

Bytes 5–63

Contain other measured values as described for byte 4.

Data block when using ASCII communication (byte 11–74)

The numbering applies to inside the data block only. To determine the correct byte number in the message an offset of 11 bytes is added.

Byte 0

Indicates the signal type. There are three alternatives:

36H	Register.
41H	P-word.
45H	Analogue input signal.

Byte 1

Indicates the part of the trend curve to be transferred. Each signal can have 6 trend curves with different sampling intervals and 120 values in each curve. Since, when using ASCII communication, 4 messages are required to transfer 120 measured values, the trend curve is divided into four parts. See also Request trend curve data for further information.

Bytes 2 and 3

16 bits indicating the sampling interval for collecting trend curve data, when using ASCII communication. The 8 most significant bits of the sampling interval are sent when the even part of a trend curve is transferred. The 8 least significant bits are sent when the odd part of a trend curve is transferred.

Lowest permitted value = 1H = (0.1 second)

Highest permitted value = FFFFH = 1 h, 49 m and 13.5 s

If trend curve number 6 (event controlled sampling) is transferred the sampling interval is always FFFFH.

A zero value indicates that the requested trend curve is not programmed.

Thus, assuming trend curve 2 is transferred, transfer occurs in four parts numbered 4, 5, 6 and 7 (see Table on page 69). When parts 4 and 6 are transferred the 8 most significant bits are contained in bytes 2 and 3. Similarly, the 8 least significant bits are contained in bytes 2 and 3 when parts 5 and 7 are transferred.

Bytes 4 and 5

16 bits containing the measured value for the trend curve. Since ASCII communication is being used two bytes are required for each value. Contains 30 values which starts with the first value requested.

Lowest permitted value = 0H = 0.0%

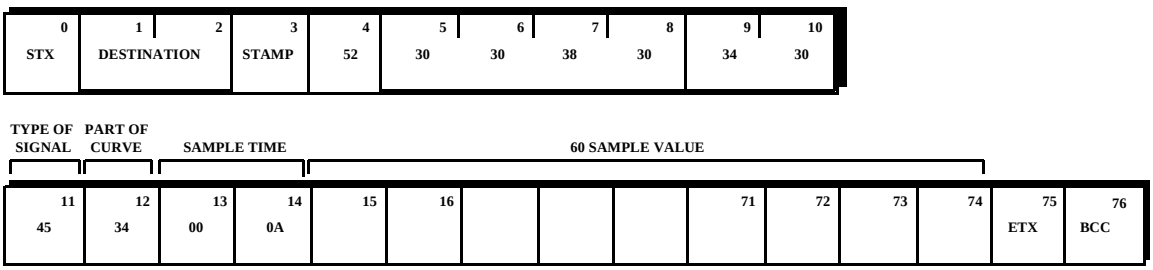
Highest permitted value = FFH = 100%

Bytes 6–7, ..., 62–63

Contains other measured values as described for bytes 4 and 5.

Example

To transfer the measured value of signal A 200 sampled at 1-second intervals with the signal value increasing. the message could be, for example, as follows:



2.13 Timers

Request timing periods (Message type S)

Application

Used to request the programmed timing periods from up to 16 timers in one message. The timing period is transferred as a 32-bit number without sign bit and a resolution of 0.1 second

Message structure

The message structure is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12
STX	DESTINATION		STAMP	MESS. TYPE	TIMER NUMBER				QUANTITY OF TIMERS		ETX	BCC

Message type (byte 4)

The message type is “S”, i.e. ASCII code 53H.

Timer number (bytes 5–8)

Indicates the number of the first timer requested. The lowest permitted number is 0H. The highest number permitted is system dependent.

Quantity of timers requested (bytes 9–10)

Indicates the quantity of timers requested starting at the timer selected by bytes 5 to 8. The lowest permitted number is 1. The highest number is 16 using binary communication (8 using ASCII communication).

Example

Request 16 timers starting at timer 42. The message is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12
STX	DESTINATION		STAMP	53	30	30	32	41	31	30	ETX	BCC

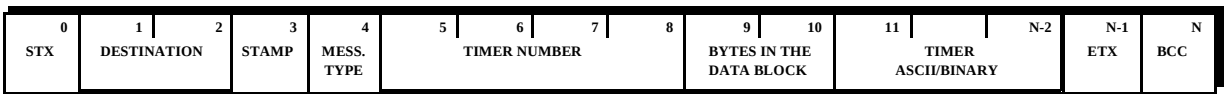
Transfer timing periods (Message type T)

Application

Used to transfer the programmed timing periods from up to 16 timers in one message. The timing period is transferred as a 32-bit number without sign bit and a resolution of 0.1 second

Message structure

The message structure is as follows:



Message type (byte 4)

The message type is “T”, i.e. ASCII code 54H.

Timer number (bytes 5–8)

Indicates the number of the first timer transferred. The lowest permitted number is 0H and the highest is system dependent.

Bytes in the data block (bytes 9–10)

Indicates the quantity of bytes transferred in the data block. 4 bytes are required to transfer one timing period using binary communication and 8 bytes are required using ASCII communication.

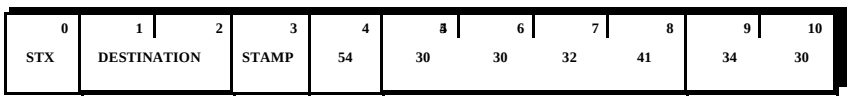
Data block (bytes 11–)

Binary communication is assumed. Each timing period transferred occupies 4 bytes, i.e 32 bits without a sign bit. If the requested timer is not programmed all bytes have the value 0.

The timing period is stipulated in tenths of a second. Thus, if a timer has a timing period programmed to expire after 500 hours, this corresponds to 18 000 000 tenths of a second which expressed in hexadecimal as 112A880H.

Example

Using binary communication, transfer the timing periods for 16 timers. This results in a data block containing 64 bytes. Timers numbered 42 and 57 are programmed for 500 hours and 0.1 seconds respectively. The other timers are not programmed. The message is as follows:



2.14 Data and program areas

General

Large installations often contain several systems which can communicate with each other. When security copies of the control systems' user memories are required a floppy or hard disk unit must be connected to each control system. The systems can be located quite remote from each other which means the process can be complicated and time consuming.

To simplify this process therefore COMLI messages have been introduced to enable security copying to be performed via the COMLI system. The operations are performed between a Master and a Slave system. The Master contains a storage unit where the security copies are stored.

Three types of operation are performed during security copying as follows:

- Dumping, i.e. user data from a Slave system is transferred to the Master.
- Loading, i.e. user data stored in the Master from a previous dumping operation is transferred to the user memory in a Slave.
- Verifying, i.e. user data stored in the Master from a previous dumping operation is transferred to the Slave user memory for comparison with the current user memory data.

To implement the operations messages are provided to request and transfer user data together with messages to request and transfer system information.

System information is a collective name for the data and commands which must be transferred between the Master and Slave to carry out the various functions. Messages for transferring system information can be sent both from Master to Slave and Slave to master. Transfers from Master to Slave contain commands to the Slave, while transfers from Slave to Master contain data and status information.

Dumping

When dumping user data, the Master first requests a memory disposition for the current module. To prevent editing of the user data while dumping is in progress, the Master sends a command which inhibits editing in the Slave.

The Master sends as many "requests for user data" as necessary to transfer all the user data. For each request received from the Master, the Slave responds by transferring user data.

To terminate the operation, a command is sent to the Slave to indicate that the module is once again available for editing.

Loading

During loading the Master transfers user data to the Slave. To ensure a fail free transfer the Master must indicate any faults which occur during loading and send commands to the Slave to stop execution of the module during loading and prevent editing of the modules while loading is in progress.

The following commands are sent from the Master to the Slave:

Reset load flags

Resets old fault indications.

Stop execution

Prevents the Slave from executing the module being loaded. This is implemented by setting a module flag to the “stopped” condition. If the module is included in the PBS execution sequence, PBS execution is also stopped. This is implemented by setting a system flag to the “stopped” condition.

Reading a module

Prevent editing of the module in the Slave. This is implemented by setting a module flag to the “locked” condition.

The Master checks the result of the above commands by sending a request for system information to the Slave. If a Slave fails to implement a command loading is stopped, the error message is read from the Slave and the message is displayed for the operator.

If no failures occur, the Master transfers the stored user data to the Slave by repeated sending of “transfer user data” messages. Each message received by the Slave is checked to ensure the address is within the defined range in the Slave.

When all the user data is transferred the Master requests the result of the transfer from the Slave. If errors are detected the “load-flag” is set to the “error” condition, the Master aborts the load operation and an error message is displayed for the operator.

If no errors are detected the Master sends a start command indicating to the Slave that execution of the module can recommence. If PBS execution is stopped using the “stop” command, the “start” command must be transmitted to the Slave to resume execution.

Verifying

The verify flag in the slave must be reset before verifying commences by sending the “reset verify” command. The user data to be verified is then transferred from the Master to the Slave.

It is not necessary for the Slave to verify each message received. A number of messages can be saved and verified together. The Slave checks that all messages contain addresses within a defined range.

When all the user data is transferred from the Master to the Slave, the Master sends a “set verify” command to the Slave. The Slave in turn indicates the result of the verification by setting the flag to “no error” or “error”. The Master reads these flags by transmitting “request system information” messages until one of the flags is set.

If an error is detected during verifying an error message is displayed for the operator.

The messages for loading, dumping and verifying are described on the following pages. The descriptions do not provide complete details however since implementation is system dependent and requires a thorough understanding of the system concerned. It is recommended that customers discuss their requirements with ABB Automation before attempting to implement the functions.

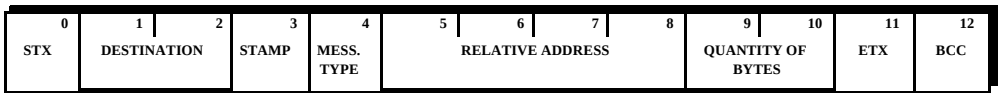
Request data and program areas (Message type W)

Application

Used to request user memory data for storing, i.e. dumping.

Message structure

The message structure is as follows:



Message type (byte 4)

The message type is “W”, i.e. ASCII code 57H.

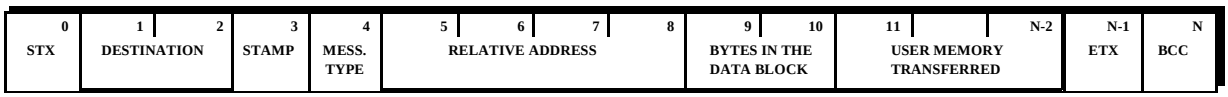
Transfer data and program areas (Message type X)

Application

Used to transfer user programs. If this message is sent from a Slave to a Master it corresponds to a back-up operation (dumping). If the message is sent from a Master to a Slave it corresponds to back-up retrieval (loading).

Message structure

The message structure is as follows:



Message type (byte 4)

The message type is “X”, i.e. ASCII code 58H.

Transfer of data and programs for verifying (Message type Y)

Application

Used by the Master to send user memory information to a Slave for verifying.

Message structure

The message structure is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	N-2	N-1	N
STX	DESTINATION		STAMP	MESS. TYPE	RELATIVE ADDRESS				BYTES IN THE DATA BLOCK		USER MEMORY FOR VERIFYING		ETX	BCC

Message type (byte 4)

The message type is “Y”, i.e. ASCII code 59H.

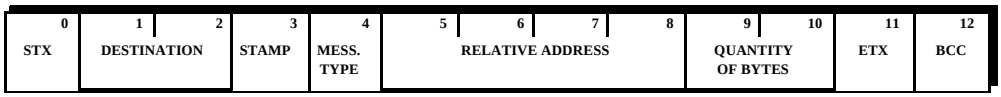
Request system information (Message type Ü)

Application

Used to request system information from a system.

Message structure

The message structure is as follows:



Message type (byte 4)

The message type is “Ü”, i.e. AXCII code 5EH.

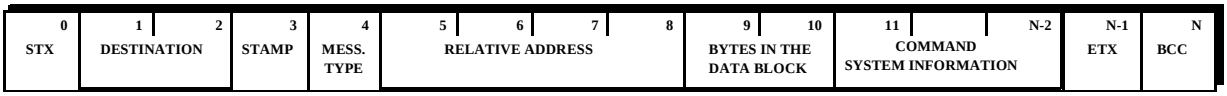
Transfer system information (Message type _)

Application

This message can be sent either from a Master to a Slave or vice versa. During transfer from a Master to a Slave the message contains commands to the Slave, whereas transfers from Slave to Master contain information on the status of the system for which security copying is being performed.

Message structure

The message structure is as follows:



Message type (byte 4)

The message type is “_”, i.e. ASCII code 54H.

2.15 Time marked events

Request time marked events (Message type Å)

Application

Requests 6 time marked events when using binary communication to transfer the data from a subsystem (3 events using ASCII communication). A knowledge of the type of alarm monitoring used in the system concerned is necessary to use this message correctly. See also the message “Transfer of time marked events”.

An increasing number of installations employ several control systems linked via a communications network. This type of installation often requires that events in different control systems are recorded by a supervisory system together with the times at which the events occur. The time recorded must be the actual time for the event excluding any delay involved in transferring the data from the subsystem to the supervisory system.

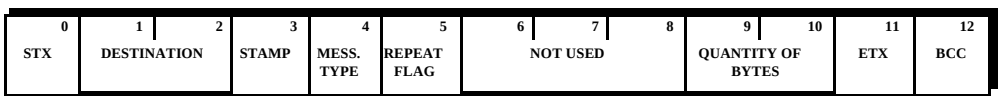
The messages which request and transfer time marked events have been introduced so that I/O-bits can be time marked in the system to which the signals are connected. Information supplied for each event requested is as follows:

- I/O address.
- Date and time the event occurred.
- Status of the I/O bit.

The load on the communications network can be reduced using time marked events. All subsystems that register events become Masters and send a message to the Slave supervisory system only when an event occurs.

Message structure

The message structure is as follows:



Message type (byte 4)

The message type is “Å”, i.e. ASCII code 5DH.

Repeat (retransmission) flag (byte 5)

Indicates if the message is the first transmission or a repeat transmission.

Bytes 6–8)

Not used. Always have the value 30H.

Quantity of bytes requested (bytes 9–10)

60 bytes are always requested. These can contain up to 6 events when binary communication is being used or up to 3 events when using ASCII communication.

Example

A first request for time marked events. The message is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12
STX	DESTINATION		STAMP	5D	30	30	30	30	33	43	ETX	BCC

Transfer time marked events (Message type Ä)

Application

Transfers 6 time marked events when using binary communication (3 events using ASCII communication). A knowledge of the type of alarm monitoring used in the system concerned is necessary to use this message correctly. See also “Request of time marked events”.

An increasing number of installations employ several control systems linked via a communication network. This type of installation often requires that events in different control systems are recorded by a supervisory system together with the times at which the events occur. The time recorded must be the actual time for the event excluding any delay involved in transferring the data from the subsystem to the supervisory system.

The messages which request and transfer time marked events have been introduced so that I/O-bits can be time marked in the system to which the signals are connected. Information supplied for each event requested is as follows:

- I/O address.
- Date and time the event occurred.
- Status of the I/O bit.

The load on the communication network can be reduced using time marked events. All subsystems that detect events become Masters and send a message to the Slave supervisory system only when an event occurs.

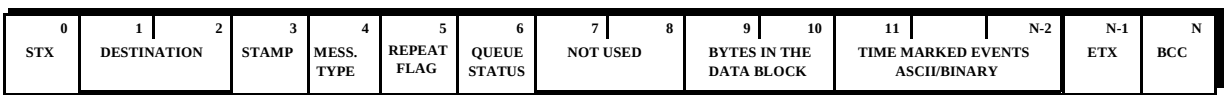
In the system detecting an event, alarm text must be linked to the I/O address being monitored by the alarm monitor program and subsequently transferred as a time marked event via COMLI. When a status change occurs it is stored in the usual alarm buffer as a time marked event and when the next transfer via COMLI occurs the event is read from the buffer. On receipt of the event at the destination system it can either be recorded there or sent further to the next system in the installation hierarchy. A text string can be added to the data transferred when it is displayed or printed. This is achieved by entering a suitable text in the destination system at the same address that is monitored in the system detecting the event.

The following points should be observed when using time marked events:

- Each time marked I/O address must be unique in the complete installation.
- Several Masters cannot request time marked events from the same Slave system. The Slave responds but a time marked event cannot be transferred more than once since an event is removed from the alarm buffer when it is transferred via COMLI.

Message structure

The message structure is as follows:



Message type (byte 4)

The message type is “Ä”, i.e. ASCII code 5BH.

Repeat (retransmission) flag (byte 5)

Indicates if the message is the first transmission or a repeat transmission.

Queue status (byte 6)

Indicates the buffer status when the event occurs. The values used are as follows:

30H = buffer empty.

31H = buffer occupied.

32H = buffer full.

Bytes 7–8

Not used. Always set to 0, i. e. 30H.

Bytes in the data block (bytes 9–10)

Indicates the quantity of bytes transferred in the data block. Each event occupies 10 bytes when binary communication is being used (20 bytes when using ASCII communication). Thus a maximum of 6 events (60 bytes) can be transferred when using binary communication (3 using ASCII communication).

Data block (bytes 11–)

Binary communication is assumed. The byte numbering refers to within the data block. To obtain the correct byte number for the message an offset of 11 must be added.

Byte 0

Indicates if the event is an alarm event (an event with both ON and OFF conditions) or a status change event (an event with either an ON or OFF condition). The indications are as follows:

30H = Alarm ON event 1 –> 0

31H = Alarm OFF event 0 –> 1

32H = Status event ON 1 –> 0

33H = Status event OFF 0 –> 1

Bytes 1 and 2

Contain the I/O address causing the event.The value can vary between 0 and FFFFH.

Byte 3

Year.

Byte 4

Month.

Byte 5

Day.

Byte 6

Hours.

Byte 7

Minutes.

Byte 8

Seconds.

Byte 9

Tenths and hundreths of seconds. Assuming the 8 bits are named D7, D6, D5, D4, D3, D2, D1, D0 where D0 is the least significant bit, their indications are as follows:

D3–D0 1/10 second.
D7–D4 1/100 second

D7–D4 can have values between 0 and A, where 0 indicates no 1/100s are included. 1–9 indicates the number of 1/100s and A means zero 1/100s.

Example

An alarm event is being transferred for I/O-address 1226 (octal) which has been set to ON at time and date 14:10:54:07, 890604. Events are present in the buffer. The message is as follows:

0	1	2	3	4	5	6	7	8	9	10
STX	DESTINATION		STAMP	5B	30	31	30	30	33	46

11	12	13	14	15	16	17	18	19	20
30	02	96	89	06	04	14	10	54	70

21	22	23	24	25	26	27	28	29	30
00H	00H	00H	00H	00H	00H	00H	00H	00H	00H

31	32	33	34	35	36	37	38	39	40
00H	00H	00H	00H	00H	00H	00H	00H	00H	00H

41	42	43	44	45	46	47	48	49	50
00H	00H	00H	00H	00H	00H	00H	00H	00H	00H

51	52	53	54	55	56	57	58	59	60
00H	00H	00H	00H	00H	00H	00H	00H	00H	00H

61	62	63	64	65	66	67	68	69	70	71	72
00H	00H	00H	00H	00H	00H	00H	00H	00H	00H	ETX	BCC

2.16 Mixed data types

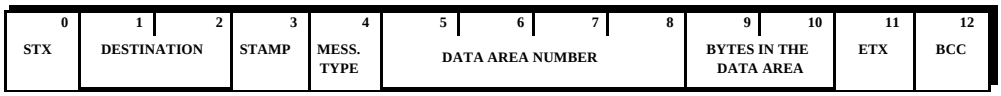
Request mixed data types (Message type Z)

Application

Requests transfer of values of all signal types (8 I/O-bits, P-words, analogue input signals, registers and floating point registers) in one message. As with other message types a communication area must be defined in the Master. A major difference between this message and other types is that a data area must be programmed in both the Master and the Slave. This defines a number of fields where the Master (Slave) writes the information and the Slave (Master) reads the information.

Message structure

The message structure is as follows:



Message type (byte 4)

The message type is “Z”, ASCII code 5AH.

Data area number (bytes 5–8)

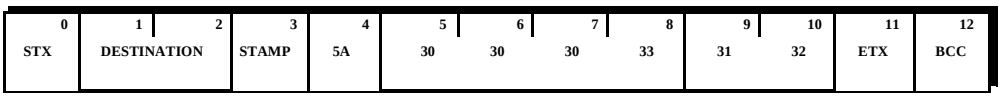
Indicates which data area containing mixed data types is to be transferred. Minimum is zero and the maximum is system dependent.

Bytes in the data area (bytes 9–10)

Indicates the quantity of bytes of mixed data types the requested data area contains. The value can vary between 0 and 64 bytes.

Example

Request data area number 3 containing 18 bytes of mixed data types. The message is as follows:



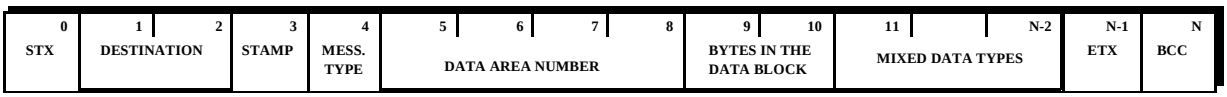
Transfer mixed data types (Message type Ö)

Application

Transfers values of all signal types (8 I/O-bits, P-words, analogue input signals, registers and floating point registers) in one message. As with other message types a communication area must be defined in the Master. A major difference between this message and other types is that a data area must be programmed in both the Master and the Slave. This defines a number of fields where the Master (Slave) writes the information and the Slave (Master) reads the information.

Message structure

The message structure is as follows:



Message type (byte 4)

The message type is “Ö”, i.e ASCII code 5CH.

Data area number (bytes 5–8)

Indicates which data area containing mixed data types is to be transferred. Minimum is zero and the maximum is system dependent.

Bytes in the data block (bytes 9–10)

Indicates the number of bytes of mixed data types the requested data area contains. The value can vary between 0 and 64 bytes.

Data block (byte 11–n)

The various signals are packaged in the data block as follows:

Individual I/O bits

The first I/O-bit is located in the least significant bit position of the first byte, the second in the next bit position etc.

Analogue inputs

See message type A.
Note. The value of the analogue input is a mirror image of the real value.

Register

See message type 0

P-words and groups of 8 I/O-bits

See message type 0

Floating point registers

See message type P.

2.17 Terminal mode

Changeover to terminal mode (Message type D)

Application

Changes a channel in a subsystem normally used for COMLI communication to terminal communication. Before changeover can occur the subsystem must acknowledge the changeover request, message type 1. When the subsystem receives an STX character COMLI communications are resumed. This occurs, among other conditions, when the next COMLI message is received. A COMLI message always begins with an STX character. A CTRL-B combination sent by a terminal can also be used to return to COMLI communication.

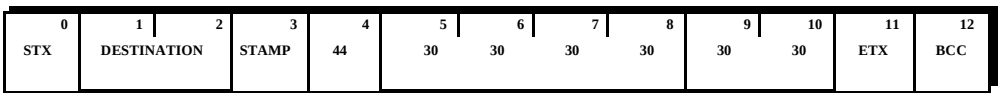
Most terminals enable a function key to be used to simulate a COMLI message requesting changeover to terminal mode. The function key must be programmed with the following sequence of ASCII characters:

CTRL-B 0 1 0 D 0 0 0 0 0 0 CTRL-C v

CTRL means key CTRL (= control) which must be held pressed while all the other character keys are pressed. The above message is for a Slave with the identity 1.

Message structure

The message structure is as follows:



Message type (byte 4)

The message type is “D”, i.e. ASCII code 44H.

Bytes 5–10

Not used consequently set to the value 0.

2.18 Acknowledge

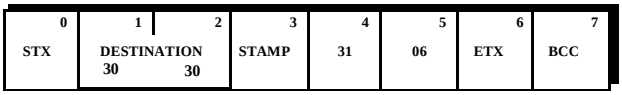
Acknowledge receipt of a message (Message type 1)

Application

Used by a subsystem to acknowledge transfer of a message from the controlling system. The message must be received correctly for an acknowledge message to be produced. Since the acknowledgement technique is used, the controlling system can alone decide if the transfer is approved or not (timeout).

Message structure

The message structure is as follows:



Message type (byte 4)

The message type is “1”, ASCII code 31H.

Acknowledge code (byte 5)

Contains the ASCII acknowledge code, i. e. 6.

3 Supplement

3.1 Calculation of checksum BCC

BCC (Block Check Count) is an error detection facility used to check that messages are transmitted correctly.

BCC is a check sum calculated according to the polynomial 1+x8 for all characters in the message excluding the STX character. The control sum is calculated as a modulo 2 addition character by character.

BCC = Character 1 + Character 2 + ... + ETX, where character 1 is the first character after STX.

+ means modulo 2 operator, i.e. (XOR function on each pair of bits).

If for example character 1 is a C (ASCII code 43H) having the binary representation 01000011 and character 2 has the value 3 (ASCII code 33H) having the binary representation 00110011 the modulo-2 addition is as follows:

```
01000011
00110011
--------
01110000
```

0 + 0 = 0, 0 + 1 = 1, 1 + 0 = 1 and 1 + 1 = 0

3.2 STAMP function

STAMP is a character in the message which is allotted the value 31H or 32H except at start, i.e. the first message transmitted, when the value is 30H. As a slave always answer on stamp 30H, this could be used for communication start-up after a transmission error.

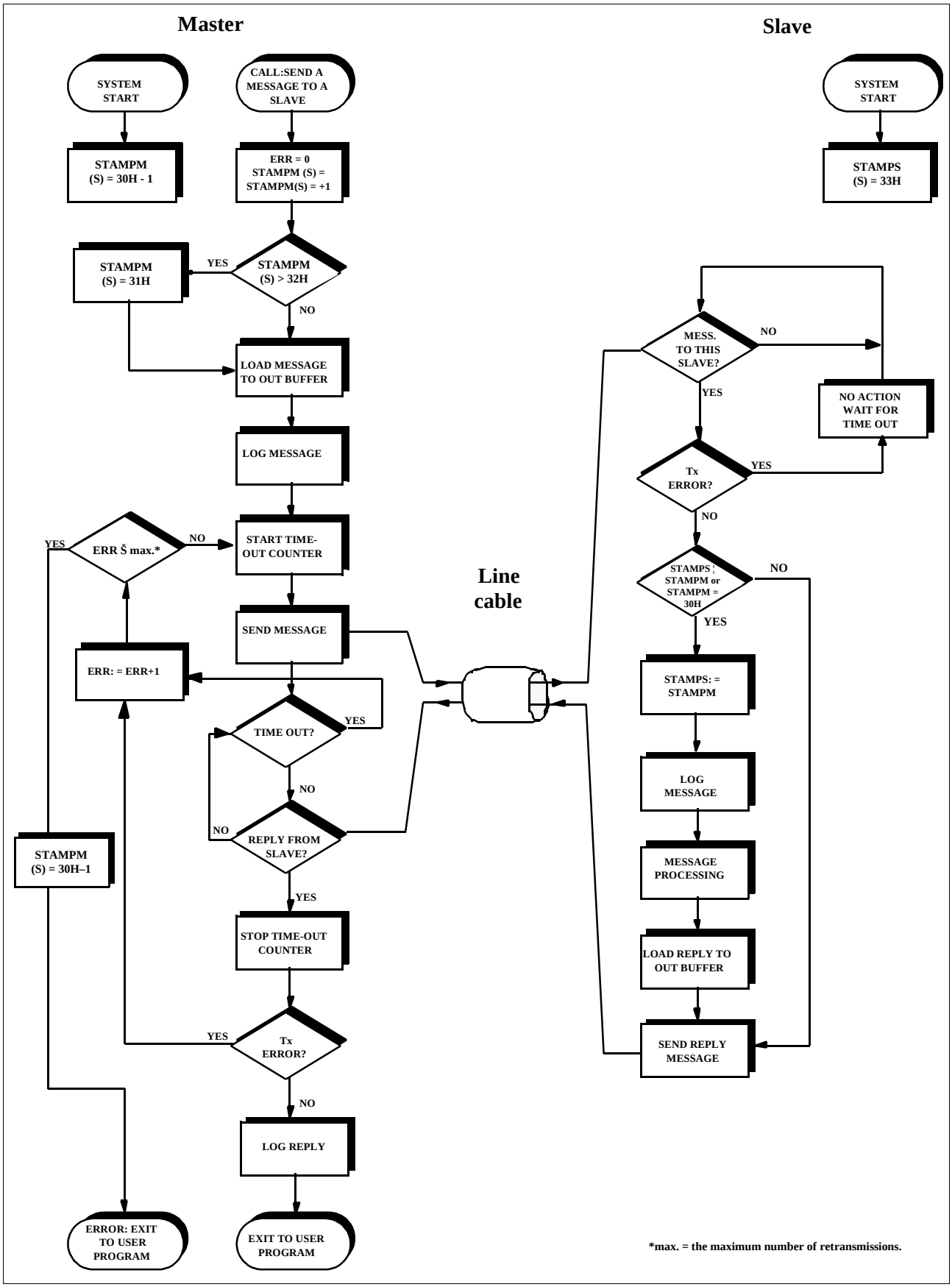
The Master changes the value of STAMP between 31H and 32H for each new message transmission but not for a retransmission. Thus, by comparing consecutive stamp values, the Slave can determine if the message is being received for the first time or if it is a retransmission, and avoid processing the same message twice.

The flow chart in Section 3.3 provides more detailed information on how the STAMP function is used during communications between the Master and Slave. It can be seen from the chart that the STAMP in the Master (STAMPM) alternates between 32H and 31H as each new message is transmitted.

STAMP in the Slave (STAMPS) always has the same value as STAMPM in the received message, consequently STAMPM and STAMPS are always different when a new message is received.

If on the other hand STAMPM and STAMPS are the same, a request for a retransmission is indicated and no message process operations occur. STAMP in the reply message sent by the Slave has the same value as in the received message. STAMP in the Master contains the index (STAMPM(S)) because several Slaves can be connected to a Master, whereas each Slave system must be administered using its own STAMP.

3.3 Flow chart - communication between Master and Slave



3.4 Old message types

Request memory contents in address areas 0–64 kilobytes

Application

Used in MIDI and MAXI systems to request RAM data in an address area 0–64 kilobytes.

Message structure

The message structure is as follows:

0	1 2	3	4	5 6 7 8	9 10	11	12
STX	DESTINATION	STAMP	MESS. TYPE	MEMORY START ADDRESS	QUANTITY OF BYTES	ETX	BCC

Message type (byte 4)

The message type is “K”, i.e. ASCII code 4BH.

Start address (bytes 5–8)

Start address of the area in RAM being requested. The address is stipulated in hexadecimal and each byte contains the ASCII code for the character represented. The address can vary between 0 and FFFFH.

Quantity of bytes requested (bytes 9–10)

Indicate in bytes the size of the memory area requested. The minimum size is 1 byte. The largest is 64 bytes when binary communication is being used (32 bytes using ASCII communication).

Example

A request for 64 bytes starting at address ABCDH. The message is as follows:

0	1 2	3	4	5	6	7	8	9	10	11	12
STX	DESTINATION	STAMP	4B	41	42	43	44	34	30	ETX	BCC

Request memory contents in address areas 64–128 kilobytes

Application

Used in MIDI and MAXI systems to request RAM data in an address area 64–128 kilobytes.

Message structure

The message structure is as follows:

0	1 2	3	4	5 6 7 8	9 10	11	12
STX	DESTINATION	STAMP	MESS. TYPE	MEMORY START ADDRESS	QUANTITY OF BYTES	ETX	BCC

Message type (byte 4)

The message type is “L”, i.e ASCII code 4CH.

Start address (bytes 5–8)

Start address of the area in RAM being requested. The address can vary between 0 and FFFFH. To obtain an address in the range 0 to FFFFH, an offset of 10000H must be subtracted from the required address. If, for example, the correct address is 12345H, the address stipulated in bytes 5–8 is 2345H since 12345H–10000H = 2345H. The address is stipulated in hexadecimal and each byte contains the ASCII code for the character represented.

Quantity of bytes requested (bytes 9–10)

Indicate in bytes the size of the memory area requested. The minimum size is 1 byte. The largest is 64 bytes when binary communication is being used (32 bytes using ASCII communication).

Example

A request for 64 bytes starting at address 12345H. The message is as follows:

0	1 2	3	4	5	6	7	8	9	10	11	12
STX	DESTINATION	STAMP	4C	32	33	34	35	34	30	ETX	BCC

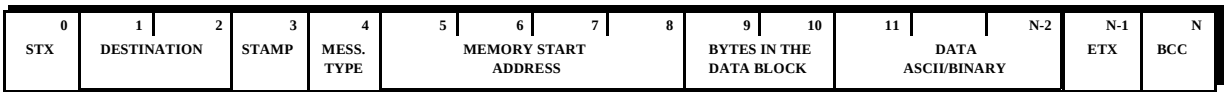
Transfer RAM contents in address area 0–64 kilobytes

Application

Used in MIDI and MAXI systems to transfer RAM data in an address area 0–64 kilobytes.

Message structure

The message structure is as follows:



Message type (byte 4)

The message type is “M”, i.e. ASCII code 4DH.

Start address (bytes 5–8)

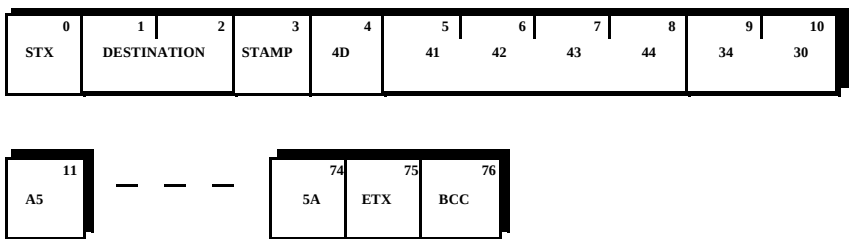
Start address of the area in RAM being requested. The address is stipulated in hexadecimal and each byte contains the ASCII code for the character represented. The address can vary between 0 and FFFFH.

Bytes in the data block (bytes 9–10)

Indicate in bytes the size of the memory area contained in the data block. The minimum size is 1 byte. The largest is 64 bytes when binary communication is being used (32 bytes using ASCII communication).

Example

A request for 64 bytes starting at address ABCDH. Address ABCDH has the value A5H and address ACOCH has the value 5AH. The message is as follows using binary communication:



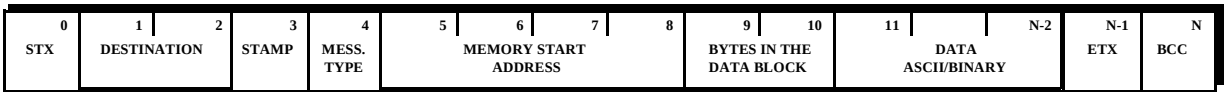
Transfer RAM contents in address areas 64–128 kilobytes

Application

Used in MIDI and MAXI systems to request RAM data in an address area 64–128 kilobytes.

Message structure

The message structure is as follows:



Message type (byte 4)

The message type is “N”, i.e. ASCII code 4EH.

Start address (bytes 5–8)

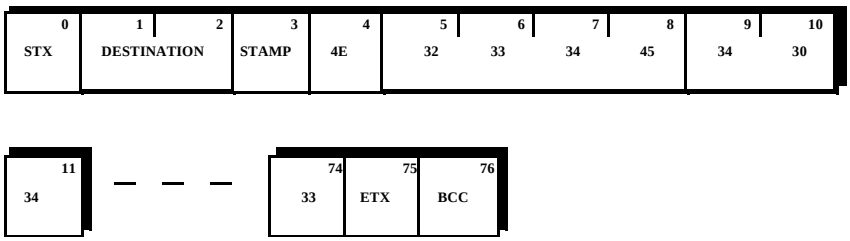
Start address of the area in RAM being requested. The address can vary between 0 and FFFFH. To obtain an address in the range 0 to FFFFH, an offset of 10000H must be subtracted from the required address. If, for example, the correct address is 12345H, the address stipulated in bytes 5–8 is 2345H since 12345H–10000H = 2345H. The address is stipulated in hexadecimal and each byte contains the ASCII code for the character represented.

Bytes in the data block (bytes 9–10)

Indicate in bytes the size of the memory area requested. The minimum size is 1 byte. The largest is 64 bytes when binary communication is being used (32 bytes using ASCII communication).

Example

A request for 64 bytes starting at address 12345H. Address 12345H has the value 84H and address 12384H has the value 33H. The message is as follows using binary communication:



Request contents of PLC-RAM

Application

Requests the contents of PLC-RAM in a MAXI system.

Message structure

The message structure is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12
STX	DESTINATION		STAMP	MESS. TYPE	MEMORY START ADDRESS				QUANTITY OF BYTES		ETX	BCC

Message type (byte 4)

The message type is “U”, i.e. ASCII code 55H.

Start address (bytes 5–8)

Start address of the requested memory area in PLC-RAM.The address is stipulated in hexadecimal and each byte contains the ASCII code for the character represented. The address can vary between 0 and FFFFH.

Quantity of bytes requested (bytes 9–10)

Indicate in bytes the size of the memory area requested. The minimum size is 1 byte. The largest is 64 bytes when binary communication is being used (32 bytes using ASCII communication).

Example

A request for 64 bytes starting at address ABCDH. The message is as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12
STX	DESTINATION		STAMP	55	41	42	43	44	34	30	ETX	BCC

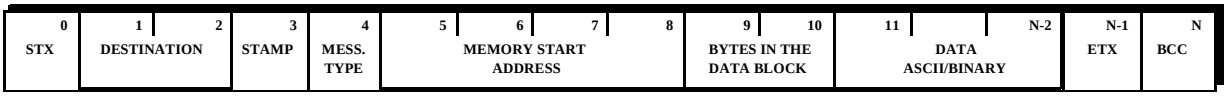
Transfer contents of PLC-RAM

Application

Transfers the contents of PLC-RAM in a MAXI system. Prior to transfer a message is sent to stop the PLC program. This is followed by transfer of the data and when all data has been transferred a message is sent to restart execution of the PLC program.

Message structure

The message structure is as follows:



Message type (byte 4)

The message type is “V”, i.e ASCII code 56H.

Stop PLC execution

Before transfer of the contents of PLC-RAM, the PLC program must be stopped. This is achieved by sending 30H in bytes 5–10.

Start PLC execution

After transfer of the contents of PLC-RAM in a control system, execution of the PLC program must be restarted. This is achieved by sending 31H in byte 5 and 30H in bytes 6–10.

Transfer of data

Start address (bytes 5–8)

Start address of the memory area in PLC-RAM to be transferred.The address is stipulated in hexadecimal and each byte contains the ASCII code for the character represented. The address can vary between 0 and FFFFH.

Bytes in the data block (bytes 9–10)

Indicate in bytes the size of the memory area in the data block. The minimum size is 1 byte. The largest is 64 bytes when binary communication is being used (32 bytes using ASCII communication).

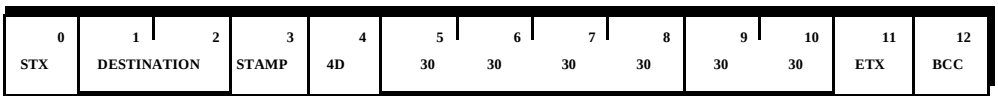
Data block (bytes 11–)

Contain the quantity of data indicated by the “Bytes in data block” field.

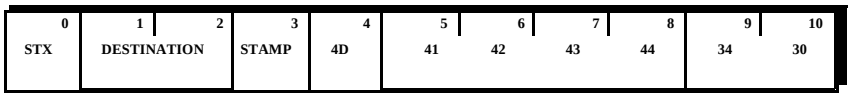
Example

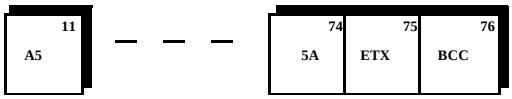
Transfer 64 bytes of memory data starting at address ABCDH. Address ABCDH has the value A5H and address ACOCH the value 5AH. The PLC program is stopped before the transfer, and restarted again when the transfer is completed. The three messages are as follows:

Stop PLC execution:

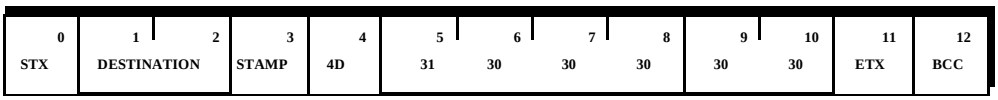


Transfer the memory data:





Start PLC execution:



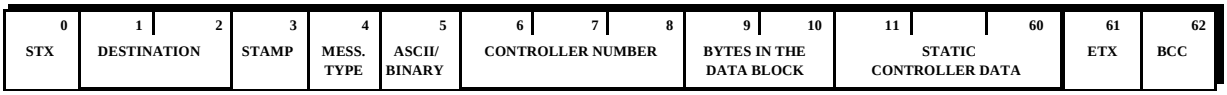
Transfer controller static data

Application

Used to transfer controller static data. This message has undergone changes and the version described here is for versions of SattCon31 earlier than Revision 4. The version used for SattCon31, Revision 4.0 and subsequent revisions is described in Chapter 2.

Message structure

The message structure is as follows:



Message type (byte 4)

The message type is “6”, i.e. ASCII code 36H.

Type of communication (byte 5)

Indicates if binary or ASCII communication is being used. When ASCII communication is used there is insufficient space in one message consequently two separate messages are sent and each message contains information indicating if it is the first or second message.The values of byte 5 for these indications are as follows:

30H binary communication.
31H ASCII communication and first half of controller data.
32H ASCII communication and second half of controller data.

Controller number (bytes 6–8)

Indicates the controller number. Numbers between 0 and 62 are permitted.

Bytes in the data block (bytes 9–10)

During transfer of static data for controllers 50 bytes are always transferred irrespective of whether binary or ASCII communication is being used. Thus, bytes 9 and 10 always have the value 50, i.e. ASCII code 35H,30H.

Data block for binary communication (byte 11–60)

The numbering of each byte refers to inside the data block only. To obtain the correct number for a byte in the message an 11 byte offset is added.

Bytes 0 to 3 inclusive

32 bits containing the controller’s accumulator value at the output.

Bytes 4 and 5

16 bits containing the process value. The functions of the four most significant bits B12, B13, B14 and B15 are as follows:

B12 Internal use.
B13 Manual percentage setting active.
B14 Internal use.
B15 Controller not in phase.

The 12 least significant bits indicate the process value at the analogue input. Min. is 0 = 0% and max is FFFH = 100%.

Bytes 6 and 7

16 bits containing X (K-1) for D-part. Min. is 0 = 0% and max. is FFFH = 100%.

Bytes 8 and 9

16 bits containing E (K-1) D-part. Min. is 0 = 0% and max. is FFFH = 100%.

Bytes 10 and 11

16 bits for the demanded signal when the controller is in the Manual mode. Min. is 0 = 0% and max. is FFFFH = 100%.

Byte 12

8 bits describing the type of controller together with the controller number.The two most significant bits describe the type of controller as follows:

0 = P controller
1 = PD controller
2 = PI controller
3 = PID controller

The six least significant bits indicate the number of the controller being transferred. The value can be between 0 and 62.

A value of FFH in byte 12 indicates that there are no further controllers programmed in the system.

Byte 13

8 bits B7, B6, B5, B4, B3, B2, B1 and B0 indicating the following:

B0 Type of control, 0 inverse control, 1 direct control.
B1 0 Manual mode, 1 Auto mode.
B2 Indicates if a feedback signal is used with a digital controller. 1 = Yes, 0 = No.
B3 Indicates how the derivative facility operates, 1 if only for positive process value changes, 0 if either for negative or positive process value changes or negative and positive process value changes.
B4 Indicates how the derivative facility operates, 1 if for both positive and negative process value changes, 0 for either negative or positive process value changes.
B5 Internal use.
B6 0 if controller output is digital, 1 if controller output is analogue.
B7 Internal use.

Bytes 14 and 15

16 bits indicating the I/O address used to select Auto mode. Lowest address 0, highest address up to 3FFFH = 37777 (octal). Note that the highest address is also system dependent.

Bytes 16 and 17

16 bits indicating the address and type of the analogue input signal. The four most significant bits indicate the type of the controller input signal as follows:

6 Register.
AH P-word (16 I/O-bits).
EH Analogue input.

The significance of the remaining 12 bits depends on the type of signal as follows:

6 Register number.
AH I/O-address.
EH Analogue signal address.

Bytes 18 and 19

16 bits indicating the address and type of the set point signal. The four most significant bits indicate the type of signal as follows:

0 Operator’s keyboard.
6 Register.
AH P-word (16 I/O-bits).
EH Analogue input.

The significance of the remaining 12 bits depends on the type of signal as follows:

0H Set point as stipulated from the terminal, min. 0 = 0%, max. FFFH = 100%
6H Register number.
AH P-word address.
EH Analogue input signal address.

Byte 20

Indicates the set point dead zone. Values between 0 and FFH are permitted. FFH corresponding to 6.2%.

Byte 21

Indicates the gain. Values between 0 and FFH are permitted. This corresponds to a range of 0–25.5.

Bytes 22 and 23

Indicate the integration time. Values between 0 and FFFFH corresponding to 0 and 1 hour 49 minutes and 13.5 seconds are permitted.

Bytes 24 t o 27

Not used.

Bytes 28 and 29

Indicate derivation time. Values between 0 and 1770H are permitted. 1770 H corresponds to 10 minutes.

Bytes 30 and 31

Indicate the type of signal, and its address, that determines the D-part. The four most significant bits indicate the following types of signal:

6 Register.
AH P-word (16 I/O-bits).
EH Analogue input.

The significance of the remaining 12 most significant bits depends on the type of signal as follows:

6 Register number.
AH I/O address.
EH Analogue input address.

Bytes 32 and 33

Indicate the Hand Interlock Speed which is the time it takes for the output signal to increase from 0 to 100%. The value can vary between 5H = 0.5 seconds and 13FFH = 8 minutes and 31.9 seconds.

Bytes 34 and 35

Indicates the calculated ramp period for the output signal. This value can vary between 20H if the Hand Interlock Speed is 8 minutes and 31.9 seconds and FFFFH if the Hand Interlock Speed is 0.5 seconds.

Bytes 36 and 37

Indicates the upper limit of the output signal. The value can vary between 0 and FFFFH. FFFFH corresponds to 100%.

Bytes 38 and 39

Indicates the lower limit of the output signal. The value can vary between 0 and FFFFH. FFFFH corresponds to 100%.

Bytes 40 and 41

Describe the controller output signal. If the output signal is analogue the following types apply:

6H Register.
AH P-word (16 I/O-bits).
EH Analogue output.

The significance of the remaining 12 bits depends on the type of signal as follows:

6H Register number.
9H Analogue output address.
AH I/O address.
EH Analogue input address.

If the output signal is digital all 16 bits indicate the address of the controller’s digital output when it is open. The address can vary between 0 and 3FFFH, 37777 (octal). Note that the upper limit is system dependent.

Bytes 42 and 43

Indicates the address of the controller's digital output when it is closed. The address can vary between 0 and 3FFFH, 37777 (octal). Note that the upper limit is system dependent.

Bytes 44 and 45

Indicate the address and type of the measured value used for the external connection signal. The four most significant bits indicate the type of signal as follows:

6H Register.
AH P-word (16 I/O bits).
EH Analogue input.

The significance of the remaining 12 bits depends on the type of signal as follows:

6H Register number.
AH I/O address.
EH Analogue signal address.

Byte 46

Indicates the dead zone of the output signal. The value can vary between 0 and FFH. FFH corresponds to 6.2%.

Bytes 47 till 49

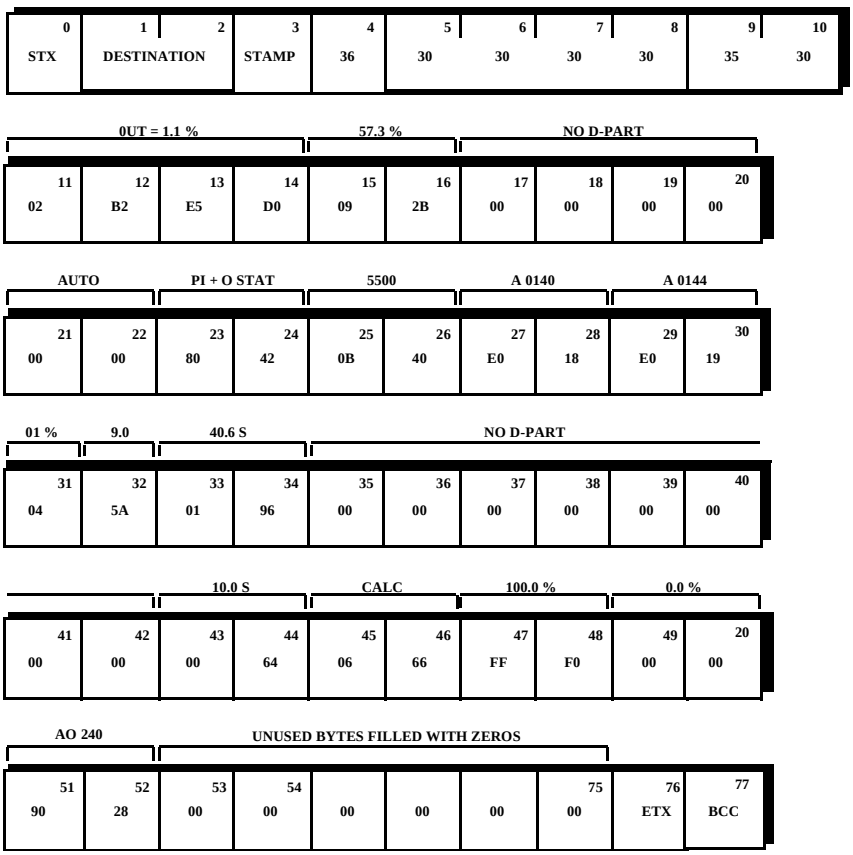
Not used.

Example

Assuming controller 0 in a system is programmed as follows:

Controller number	0
Controller type	PI
I/O for Auto mode	5500
Controller ON	5501
Ramp period at start	5502
Ramp period at stop	5503
Analogue input	AI 0140
Stipend	AI 0144
Dead zone, process van–set point (%)	0.1
Gain	9.0
Integration time	40S6
Hand Interlock Speed	10S0
Maximum output signal %)	100.0
Minimum output signal (%)	0.0
Direct control	No
Analogue controller	Yes
Address of analogue output	A 0240

The message is as follows using binary communication:



3.5 ASCII table

The binary codes (shown using hexadecimal numbering) for the characters in the ASCII set are as follows:

0	NUL	20	SP	40	@ (É)	60	` (é)
1	SOH	21	!	41	A	61	a
2	STX	22	"	42	B	62	b
3	ETX	23	#	43	C	63	c
4	EOT	24	$ (¤)	44	D	64	d
5	ENQ	25	%	45	E	65	e
6	ACK	26	&	46	F	66	f
7	BEL	27	’	47	G	67	g
8	BS	28	(	48	H	68	h
9	HT	29	)	49	I	69	i
A	LF	2A	*	4A	J	6A	j
B	VT	2B	+	4B	K	6B	k
C	FF	2C	,	4C	L	6C	l
D	CR	2D	-	4D	M	6D	m
E	SO	2E	.	4E	N	6E	n
F	SI	2F	/	4F	O	6F	o
10	DLE	30	0	50	P	70	p
11	DC1	31	1	51	Q	71	q
12	DC2	32	2	52	R	72	r
13	DC3	33	3	53	S	73	s
14	DC4	34	4	54	T	74	t
15	NAK	35	5	55	U	75	u
16	SYN	36	6	56	V	76	v
17	ETB	37	7	57	W	77	w
18	CAN	38	8	58	X	78	x
19	EM	39	9	59	Y	79	y
1A	SUB	3A	:	5A	Z	7A	z
1B	ESC	3B	;	5B	[(Ä)	7B	{ (ä)
1C	FS	3C	<	5C	\ (Ö)	7C	(ö)
1D	GS	3D	=	5D	] (Å)	7D	} (å)
1E	RS	3E	>	5E	^ (Ü)	7E	~ (ü)
1F	US	3F	?	5F	_	7F	DEL

493-0192-11
9903 Ver. 3-2