# Contents

- Safety Considerations ................................................................................................................. 4
- Introduction BIS C .................................................................................................................. 5/6
- Application BIS C-600 .......................................................................................................... 7
- Configuration ..................................................................................................................... 8-22
- User Configuration ........................................................................................................... 23-26
- Protocol ................................................................................................................................ 27/28
- Read/Write Times ................................................................................................................. 29/30
- Programming .................................................................................................................... 31-48
- Error Numbers ...................................................................................................................... 49/50
- Interface ................................................................................................................................... 51-54
- Assembly of Read/Write Head/Processor ........................................................................... 55-57
- Technical Data ...................................................................................................................... 58/59
- Ordering Information ................................................................................................................. 60
- ASCII Table .......................................................................................................................... 61

---

## Safety Considerations

BIS C-600 processors together with the other BIS C system components comprise the Identification System and may be used only for this purpose.

### Installation and Operation

Installation and operation should be carried out by technical trained personnel only. Unauthorized access and improper use will lead to loss of warranty and liability claims.

When installing the processor, consult the section on wiring diagrams carefully. Special caution must be used when wiring the processor to external controllers, particularly with respect to selection and polarity of the signals and power supply.

Only approved power supplies may be used with the processor. See the section on Technical Data for details.

### Use and Checking

The relevant safety procedures must be followed when using the Identification System. In particular, steps must be taken to ensure that no danger to persons or equipment can arise should a fault occur in the Identification System.

This includes maintaining the published ambient operating conditions and regular checking of the functionality of the Identification System with all its associated components.

### Fault Conditions

As soon as there is evidence that the Identification System is not functioning properly, it should be taken out of service and protected against unauthorized use.

### Scope

This description is valid for processors in the series BIS C-600-024-...-04-KL1.
This manual is designed to assist the user in setting up the control program and installing and starting up the components of the BIS C-600 Identification System, and to assure rapid, trouble-free operation.

The BIS C-600 Identification System belongs in the category of non-contact systems for reading and writing. This dual function permits applications not only for transporting information in fixed-programmed code tags, but also for gathering and passing along up-to-date information as well.

Some of the notable areas of application include:

- for controlling material flow in production processes (e.g. in model-specific processes),
- for workpiece conveying in transfer lines,
- in data gathering for quality assurance,
- for gathering safety-related data,
- in tool coding and monitoring;
- in equipment organization;
- in storage systems for monitoring inventory movement;
- in transporting and conveying systems;
- in waste management for quantity-based fee assessment.

The main components of the BIS C-600 Identification System are:

- Processor,
- Read/Write Heads, and
- Code Tags.

Baluff Bus RS 485 (2-wire)

Processor BIS C-600

with Adapter BIS C-650

Read/Write Heads

Schematic representation of an Identification System (example)
Selecting System Components

A BIS C-600 processor may be used with a single series BIS C-65_ read/write head directly attached, comprising a compact unit. A second, series BIS C-3_ _ read/write head with 5m of cable (except BIS C-350 and -352) may be cable connected to the BIS C-600.

If the BIS C-600 processor is fitted with the BIS C-650 adapter instead of a BIS C-65_ read/write head, two (2) series BIS C-3_ _ read/write heads may be cable attached (except BIS C-350 and -352).

For additional information concerning series BIS C-65_ and BIS C3_ _ read/write heads together with suitable combinations of code tags, consult the manuals for the respective read/write heads.

Which of the above described arrangements of read/write heads is the most logical will be determined mainly by the spatial arrangement of the components. There are no functional limitations. All read/write heads are suitable for both static and dynamic reading and writing. Sensing distance and traverse velocity will depend on the individual code tags(s).

The system components are electrically powered by the processor. The code tag represents a stand-alone unit, and does not require an external source of power. It receives its energy from the read/write head. The latter sends a constant carrier signal which supplies the code tag as soon as the required sensing distance is reached. The read/write process takes place during this phase. Reading and writing may be static or dynamic.

Application
BIS C-600 Processor

Before programming, the processor configuration must be carried out, in case the factory settings will not be used.

Configuration is done using a computer and the Balluff BISC600B.EXE software, and it is stored in the processor. It may be overwritten at any time. The configuration can be stored in a file, making it accessible when required.

Configuration

Important. Please note the selected settings on the stick-on label supplied (to be pasted on the inside of the processor cover) as well as on page 23 and 25 in the customer configuration section, so that in case of repair of the processor the settings can be saved or otherwise can also be used to set other processor units.
Configuration

Interface BIS C-600

The first screen shows the parameters baud rate, number of data and stop bits, and parity type for the serial interface selected. The graphic shows the factory settings. The other settings are carried out in the corresponding masks which are illustrated in the following pages.

19200 baud can only be used with the RS 232/V.24 interface, not with the TTY interface.

If the initializing data are available in short form (e.g. on the processor cover after a replacement of the unit) the data can be entered directly into the "Shortform of initialization BIS C-600" mask (see also Customer configuration on page 23).

Shortform of initialization

If the initializing data are available in short form (e.g. on the processor cover after a replacement of the unit) the data can be entered directly into the "Shortform of initialization BIS C-600" mask (see also Customer configuration on page 23).

Protocol Type

Factory setting is for BCC blockcheck with error number.

Example for a telegram with BCC blockcheck:

Telgram with command, address, and no. of bytes: 'R 000 0001 BCC <--- as termination
Acknowledgement: <ACK> '0'
Configuration

Paging

In the memory organization of the code tag, a distinction is made between page size 32 bytes or 64 bytes (sometimes also referred to as block size).

Code tags < 2 kByte use 32 Byte block size,
Code tags ≥ 2 kByte use 64 Byte block size.

Parameters

- RS 485 / 2-wire < RS 422 / 4-wire>
  This function selects the type of the interface.
  - When it is marked the RS 485 / 2-wire interface is selected.
    - With this interface type the output is switched to high impedance as long as there are no data transmitted by the BIS C-600 processor.
  - When it is not marked the RS 422 / 4-wire interface is selected.

- Dynamic Mode
  This function switches off the error-message "No code tag present", i.e.:
  - In dynamic mode, a read or write telegram is stored until a code tag enters the working range of the corresponding read/write head.
  - Without dynamic mode, a read or write telegram is acknowledged with an error message (NAK 1) if there is no code tag present in front of a read/write head; the processor goes into the ground state.

- CT Present on Output 1
  If CT Present is selected for Output 1, the LED message Codetag Present is also output on Output 1. In this way the presence of a code tag can be directly verified as a digital hardware signal.

- Process Outputs if CT Present
  The output functions are normally processed only after a read command. But since code tag recognition is also an automatic tag read (reads first page, either 32 or 64 bytes depending on type), the output processing can occur simultaneously with Codetag Present. If the addresses for output processing are located on the first page, then the processor can itself carry out short control commands without a separate command.
  - For very fast transactions, see next section.

Fast Code Tag Recognition

For very fast transactions, the number of code tag addresses used for code tag recognition can be reduced from 32 or 64 bytes to 4 bytes. The code tag recognition time is thereby reduced to ca. 50 ms (instead of ca. 150 ms for tags with < 2 kByte or ca. 250 ms for code tags with ≥ 2 kByte of memory).

- Please note this when using the parameter "Process outputs with CT-present".

Addresses

Here the device address and the predecessor address are entered.

Continuous Read

Here the start address and the number of bytes to be read are entered. This is required for the function "Continuous read".

This function will be implemented at a later date. The factory setting must not be changed.

Input

The function of the digital control input of the BIS C-600 can be selected. The factory setting is "Reset".

- Reset
  - If Reset is selected, a High signal on this input causes a reset of the BIS C-600 processor. Commands already started will be canceled.

- Head Select
  - If Head Select is selected, this input is used to select read/write heads.
    - Input Low: Head 1 selected.
    - Input High: Head 2 selected.
  - For this function has always priority. The function "Both Read/Write Heads Active", which is selected using command "HT", is then deactivated.
Configuration

Input (continued)

- **Data bit on Code tag**
  On recognition of a new code tag a freely definable bit of a given address will be written, direct or inverted, on the code tag. After a successful write operation, an output which can also be freely defined is set until the code tag leaves the active read/write range.
  - The parameter "Dynamic operation" will be automatically reset.
- **Not used**
  The input has no function.

Input/Output Configuration

The outputs can be configured for various functions. The output functions are always processed during a read. Prerequisite for the execution is that the corresponding address was read during the previous read operation.

**IN/OUTPUT CONFIGURATION**

- ( ) Outputs not used.
- ( ) Output halfbyte of data contents of an address.
- ( ) Compare contents of multiple addresses with a fixed value.
- ( ) Compare contents of an address with various fixed values.
- ( ) Compare contents of multiple addresses with content of one address.
- ( ) Output data bits of variable addresses.
- ( ) Program input as data bit to code tag.

**Outputs not used**

"Outputs not used" means processing of the outputs is deactivated.

**OUTPUTS NOT USED**

- ( ) Data to BIOS ( ) Store ( ) ESC - Exit ( ) F1 - Help

"Data to BIOS" transmits the data to the processor. "Store" stores the data in the configuration file on your computer.
Configuration

Output Halfbyte of the Data Contents of an Address

- Output not used.
- Output halfbyte of data contents of an address.
- Compare contents of multiple addresses with a fixed value.
- Compare contents of an address with various fixed values.
- Compare contents of multiple addresses with a content of one address.
- Output data bits of variable addresses.
- Program input as data bit to code tag.

< OK > < Shortform > < ESC = Exit > < Print . . . . > < F1 = Help >

( ) Output not used.
( ) Output halfbyte of data contents of an address.
( ) Compare contents of multiple addresses with a fixed value.
( ) Compare contents of an address with various fixed values.
( ) Compare contents of multiple addresses with a content of one address.
( ) Output data bits of variable addresses.

Either the upper or lower 4 bits (Nibble) of the 8 bits data contents of an address is output (Bit 0 on Output 1, Bit 1 on Output 2, etc.). The address is given in decimal.

Configuration

Compare Contents of multiple Addresses with a fixed Value

The data contents of up to 4 addresses are compared with a fixed decimal value. To each address can be assigned which of the Outputs 1 to 4 is set or cancelled by a positive result of the comparison and whether in case of a negative result of the comparison the output shall not be changed or shall be set in contrary to the definition with the positive result (inverted response).

All addresses found within the read command will be processed.

<table>
<thead>
<tr>
<th>Address</th>
<th>Output</th>
<th>Positive compar.</th>
<th>Negative compar.</th>
</tr>
</thead>
<tbody>
<tr>
<td>9999</td>
<td>1</td>
<td>*</td>
<td>Set</td>
</tr>
<tr>
<td>9999</td>
<td>2</td>
<td>*</td>
<td>Set</td>
</tr>
<tr>
<td>9999</td>
<td>3</td>
<td>*</td>
<td>Set</td>
</tr>
<tr>
<td>9999</td>
<td>4</td>
<td>*</td>
<td>Set</td>
</tr>
</tbody>
</table>

< Data to BIS > < Store > < ESC = Exit > < F1 = Help >

If the parameter "Process outputs with CT-Present" is included in the initialization, then this function also will be carried out on recognition of a new code tag (one or more of the addresses given should be on the first page of the codetag).
Configuration

Compare an Address with various fixed Values

The data contents of an address is compared with 4 fixed decimal values. For each fixed value is indicated which of the Outputs 1 to 4 is set or cancelled by a positive result of the comparison and whether in case of a negative result of the comparison the output shall not be changed or shall be set in contrary to the definition with the positive result (inverted response).

<table>
<thead>
<tr>
<th>Address: [9999]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed value</td>
</tr>
<tr>
<td>[000]</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>[000]</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>[000]</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>[000]</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

If the parameter "Process outputs with CT-Present" is included in the initialization, then this function also will be carried out on recognition of a new code tag (one or more of the addresses given should be on the first page of the codetag).
Configuration

Output Data Bits of variable Addresses

1 data bit of an address or 1 bit each from up to 4 addresses can be output on one of the 4 outputs and then inverted or not inverted.

Address | Bit-Number | Output | Invert
-------- | ---------- | ------ | -----
[9999]   | [1]        | [1]    | [ ]  yes
[9999]   | [1]        | [2]    | [ ]  yes
[9999]   | [1]        | [3]    | [ ]  yes
[9999]   | [1]        | [4]    | [ ]  yes

If the parameter "Process outputs with CT-Present" is included in the initialization, then this function also will be carried out on recognition of a new code tag (one or more of the addresses given should be on the first page of the codetag).

Configuration

Programming a Data Bit on the Code Tag depending on the Input

On recognition of a new code tag the state of the digital input will be written as a direct or inverted bit on the codetag. The address range is 0...31! Bit number of the address is 1...8. The outputs to be used for the ready and Enable signals should also be given. If Enable output is given as "0" then the Enable function will not be used. The procedure is described below.

Program Input as data bit to code tag

address | bit number | Enable output | Input inverse ?
---------|-----------|---------------|------------------
[00]     | [1]       | [0]           | [ ]

Procedure without Enable Signal

On recognition of a new code tag the defined bit of the given address will be written, direct or inverted. After a successful write operation, the given ready output is set until the codetag leaves the active read/write range. The input must hold its state until the ready output is set.

The input state that is to be written as information on the code tag must be present already before the presence of the new code tag is recognized.
In this example the bit is set to 1.

After the recognition of a new code tag, the input state is continuously sampled until it is set (Enable given). The processor sets the Enable output and waits 50 ms. The input is then sampled and its state is taken over as bit-value. This value is written on the codetag directly or inverted depending on the selected configuration. After the write operation, the given ready output is set until the code tag leaves the active read/write range. The Enable output then reverts to low.

**Procedure with Enable Signal**

If the initializing data are available in short form (e.g. on the processor cover after a replacement of the unit) the data can be entered directly into the mask “Shortform of I/O configuration BIS C-600” (see also User Configuration on page 24).
User Configuration

Initialization

Please note the settings in the label fields on the inside of the processor cover so that in case of repair of the processor the settings can be reset in the factory. Note the settings also in the following fields so that you can set, e.g. other units, to an identical configuration.

On the following page you will find an example which shows how you can print-out after initializing. Enter the settings in the appropriate fields so that you have them handy and can reproduce the settings at any time. You can then enter the data in short form into the mask. (see also page 9).

SHORTFORM OF INITIALIZATION BIS C-600

<table>
<thead>
<tr>
<th>Baudrate</th>
<th>Data bits</th>
<th>Stop bits</th>
<th>Parity</th>
<th>Protocol type</th>
<th>Addr</th>
<th>Predecessor addr</th>
<th>Continuous read</th>
<th>Start address</th>
<th>Number of bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>19200</td>
<td>8</td>
<td>1</td>
<td>None</td>
<td>BCC with error number</td>
<td>15</td>
<td>14</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
</tbody>
</table>

The entries in the field are either in clear text (as with Interface settings) or the number of the line marked. In the case of 'Parameter' the marked line is indicated by a 1.

User Configuration

Initialization

Example of a print-out after initialization which you can print with the software BISC600B.EXE.

Interface settings:
- Baudrate: 19200 baud
- Data bits: 8
- Stop bits: 1
- Parity: None

Protocol:
- BCC with error number

Addresses:
- Device address: 15
- Predecessor address: 14

Continuous read:
- Start address: 00
- Number of bytes: 00

Parameter:
- RS485/2-wire
- Dynamic mode
- Codetag Present signal on Output 1
- Process outputs with Codetag Present
- Fast codetag recognition

Block size:
- 32 Byte page size
- 64 Byte page size

Input:
- Input = Reset
- Input = Head select
- Input = data bit on code tag
- Input = not used

As long as the function "Continuous read" is not realized, these fields must not be changed.
Input/Output Configuration

Please note the settings in the label fields on the inside of the processor cover so that in case of repair of the processor the settings can be reset in the factory. Note the settings also in the following fields so that you can set, e.g., other units, to an identical configuration.

On the following page you will find an example which shows how you can print-out after initializing. Enter the settings in the appropriate fields so that you have them handy and can reproduce the settings at any time. You can then enter the data in short form into the mask.

Example:
Shortform I/O Configuration BIS C-600

Example of a print-out after initialization which you can print with the software BIS600B.EXE.

Input/output configuration
(    ) Output not used.
(    ) Output halfbyte of data contents of an address.
(    ) Compare contents of multiple addresses with a fixed value.
(    ) Compare contents of an address with various fixed values.
(    ) Compare contents of multiple addresses with content of one address.
(    ) Output data bits of variable addresses.
(    ) Program input as data bit to code tag.

Definition
Fixed value: 123
Address: 0010
Output: 1
Positive comparison: (+) Set (    ) No change (+) Invert
Address: 0072
Negative comparison: (    ) Clear (+) Invert
Address: 0072
Output: 2
Positive comparison: (+) Set (    ) No change (+) Invert
Address: 0114
Negative comparison: (    ) Clear (+) Invert
Address: 0010
Output: 3
Positive comparison: (+) Set (    ) No change (+) Invert
Address: 0054
Negative comparison: (    ) Clear (+) Invert
Address: 0010
Output: 4
Positive comparison: (+) Set (    ) No change (+) Invert
Address: 0054
Negative comparison: (    ) Clear (+) Invert
Multiple BIS C-600 processors are connected over the Balluff-Bus. They are controlled by the host device over a bus master:

- by a controlling computer (e.g., industrial PC) or
- by an external programmable logic controller (PLC).

Up to 31 processors can be connected on the Balluff-Bus. Each processor has a unique 2-character address.

Each processor controls and manages the data communications between codetags and read/write heads. Data are exchanged between the BIS C-600 Identification System and the external controller over the RS 485 (2-wire) serial interface.

Data communication between the processors and the host system is done using defined telegrams. The individual processors respond to their address number. The Bus Master is always Address 01.

The protocol execution is described in this section schematically in the form of function blocks. The "Command" block contains the address for the corresponding processor.

The telegram contents is described in detail, including the blockcheck algorithm, in the section on Programming Information.

The following descriptions are valid only if:

- The processor is in the ground state.
- A code tag is present in front of a read/write head.

**Without Head Select**

**Read:**

<table>
<thead>
<tr>
<th>PLC</th>
<th>BIS C-600</th>
<th>Time</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

a) If no error:

- t1 depending on no. of bytes to read (see page 29/30)
- t2 ≥ 0 (is not monitored by the processor)
- t3 = max. 50 ms

b) With error:

- t1 depending on no. of bytes to read (see page 29/30)
- Error type (recommended monitor time: 15 s)

**Write:**

<table>
<thead>
<tr>
<th>PLC</th>
<th>BIS C-600</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

a) If no error:

- t1 = max. 50 ms
- t2 ≥ 0 (is not monitored by the processor)
- t3 Depending on no. of bytes to write (see page 29/30)

b) With error in command:

- t1 = max. 50 ms
- t2 ≥ 0 (is not monitored by the processor)
- t3 Depending on no. of bytes to write (see page 29/30)

- Error type (recommended monitor time: 30 s for code tags with 32 byte block size, 60 s for code tags with 64 byte block size)

**With Head Select**

**Read:**

<table>
<thead>
<tr>
<th>PLC</th>
<th>BIS C-600</th>
<th>Time</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

a) If no error:

- t1 depending on no. of bytes to read (see page 29/30)
- t2 ≥ 0 (is not monitored by the processor)
- t3 = max. 50 ms

b) With error:

- t1 depending on no. of bytes to read (see page 29/30)
- Error type (recommended monitor time: 15 s)

**Write:**

<table>
<thead>
<tr>
<th>PLC</th>
<th>BIS C-600</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

a) If no error:

- t1 = max. 50 ms
- t2 ≥ 0 (is not monitored by the processor)
- t3 Depending on no. of bytes to write (see page 29/30)

b) With error in command:

- t1 = max. 50 ms
- t2 ≥ 0 (is not monitored by the processor)
- t3 Depending on no. of bytes to write (see page 29/30)

- Error type (recommended monitor time: 30 s for code tags with 32 byte block size, 60 s for code tags with 64 byte block size)
Read/Write Times

Read Times from Code tag to Processor in Static Mode (Configuration: without dynamic mode)

- For double read and compare:
  - Code tag with 32 byte blocks
    - No. of bytes | Read time [ms]
    - from 0 to 31 | 110
    - for each additional 32 bytes add | 120
    - from 0 to 255 | 950
  - Code tag with 64 byte blocks
    - No. of bytes | Read time [ms]
    - from 0 to 63 | 220
    - for each additional 64 bytes add | 230
    - from 0 to 2047 | 7350

Write Times from Processor to Code tag in Static Mode (Configuration: without dynamic mode)

- Including readback and compare:
  - Code tag with 32 byte blocks
    - No. of bytes | Write time [ms]
    - from 0 to 31 | 110 + n * 10
    - for 32 bytes or more | y * 120 + n * 10
  - Code tag with 64 byte blocks
    - No. of bytes | Write time [ms]
    - from 0 to 63 | 220 + n * 10
    - for 64 bytes or more | y * 230 + n * 10

Write Times from Processor to Code tag in Dynamic Mode (Configuration: with dynamic mode)

- Example: Write 17 bytes starting with Address 187. Code tag has 32 byte block size. Blocks 5 and 6 are processed, since start address 187 is in block 5 and last address 203 is in block 6.
  - t = 2 * 120 + 17 * 10 = 410 ms

Read Times from Code tag to Processor in Dynamic Mode (Configuration: with dynamic mode)

- Read Times for 1 Block with double read and compare:
  - The indicated times apply after the code tag has been recognized. If the code tag is not yet recognized, an additional 30 ms must be added to allow for the energy field to be established until the code tag is recognized.
  - Code tag with 32 byte blocks
    - No. of bytes | Read time [ms]
    - from 0 to 3 | 14
    - for each additional byte add | 3.5
    - from 0 to 31 | 112
  - Code tag with 64 byte blocks
    - No. of bytes | Read time [ms]
    - from 0 to 3 | 14
    - for each additional byte add | 3.5
    - from 0 to 63 | 224

- Example: Read 11 bytes starting at address 9. This means the highest address to be read is 20 (substituted for "m" in the formula).
  - t = 14 ms + (m - 3) * 3.5 ms
  - This results in 73.5 ms.

Including readback and compare:

- Code tag with 32 byte blocks
  - No. of bytes | Write time [ms]
  - from 0 to 3 | 14 + n * 10
  - for each additional byte add | 3.5
- Code tag with 64 byte blocks
  - No. of bytes | Write time [ms]
  - from 0 to 3 | 14 + n * 10
  - for each additional byte add | 3.5
### Programming

Now that the basic telegram sequence and the configuration have been shown in the preceding sections, what follows is information concerning addressing of the processors on the Balluff-Bus as well as the correct structure of a telegram.

There are specific telegrams for the various operations of the BIS C Identification System. Each begins with the control character `<ENQ>` (05 Hex) and the device address. Then follows the command, which is associated with the telegram type:

<table>
<thead>
<tr>
<th>Telegram Type</th>
<th>Command</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>Assign</td>
<td>Assign an address to a processor</td>
</tr>
<tr>
<td>U</td>
<td>Read</td>
<td>Read out the processor address</td>
</tr>
<tr>
<td>K</td>
<td>Delete</td>
<td>Delete the processor address</td>
</tr>
<tr>
<td>L</td>
<td>Read</td>
<td>Read the code tag with read/write select and block size</td>
</tr>
<tr>
<td>P</td>
<td>Write</td>
<td>Write to the code tag with read/write select and block size</td>
</tr>
<tr>
<td>W</td>
<td>Write</td>
<td>Write to the code tag</td>
</tr>
<tr>
<td>H</td>
<td>Select</td>
<td>Select the read/write head and block size with the variations</td>
</tr>
<tr>
<td>?</td>
<td>Find</td>
<td>Find the next code tag (one time)</td>
</tr>
<tr>
<td>!</td>
<td>Find</td>
<td>Find the next code tag (continuously)</td>
</tr>
<tr>
<td>B</td>
<td>Process Outputs</td>
<td></td>
</tr>
<tr>
<td>Q</td>
<td>Restart</td>
<td>Restart the processor (acknowledge)</td>
</tr>
</tbody>
</table>

Please note:
- The minimum wait time between two commands is 300 ms!

### Intel-Hex Conversion

To avoid the situation where the control character `<ENQ>` appears when transmitting data and performing the blockcheck, the data and the BCC are Intel-Hex converted. Both Hex characters are thus sent as ASCII characters.

- `A` = 41 Hex -> `'4'` (34Hex), `'1'` (31Hex)
- `<ACK>` = 06 Hex -> `'0'` (30Hex), `'6'` (36Hex)
- `K` = 4B Hex -> `'4'` (34Hex), `'B'` (42Hex)

The processor looks for a character delay time of 65 ms. If this time is exceeded, the processor goes into the ground state.

The BCC blockcheck is performed as an EXOR operation based on the serially transmitted binary characters in the telegram block.

#### Example:

<table>
<thead>
<tr>
<th>Character</th>
<th>ASCII</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>65</td>
<td>101001</td>
</tr>
<tr>
<td>'4'</td>
<td>68</td>
<td>110100</td>
</tr>
<tr>
<td>'1'</td>
<td>49</td>
<td>011001</td>
</tr>
<tr>
<td>50</td>
<td>32</td>
<td>001100</td>
</tr>
<tr>
<td>35</td>
<td>23</td>
<td>001101</td>
</tr>
<tr>
<td>P</td>
<td>80</td>
<td>010100</td>
</tr>
<tr>
<td>50</td>
<td>32</td>
<td>001100</td>
</tr>
<tr>
<td>0</td>
<td>30</td>
<td>000000</td>
</tr>
<tr>
<td>30</td>
<td>48</td>
<td>001100</td>
</tr>
<tr>
<td>0</td>
<td>30</td>
<td>000000</td>
</tr>
<tr>
<td>1</td>
<td>49</td>
<td>011001</td>
</tr>
<tr>
<td>31</td>
<td>51</td>
<td>011001</td>
</tr>
<tr>
<td>P</td>
<td>80</td>
<td>010100</td>
</tr>
<tr>
<td>50</td>
<td>32</td>
<td>001100</td>
</tr>
<tr>
<td>0</td>
<td>30</td>
<td>000000</td>
</tr>
<tr>
<td>30</td>
<td>48</td>
<td>001100</td>
</tr>
<tr>
<td>0</td>
<td>30</td>
<td>000000</td>
</tr>
<tr>
<td>41</td>
<td>65</td>
<td>101001</td>
</tr>
</tbody>
</table>

The resulting BCC blockcheck is: 0101 0100 = `T`
Generating the blockcheck BCC (continued)

The Hex value of the blockcheck BCC = (54<sub>Hex</sub>) is Intel-Hex converted:
'T' = (54<sub>Hex</sub>) -> '5' (35<sub>Hex</sub>) -> '4' (34<sub>Hex</sub>)
The processor acknowledges with <ACK> '0' if the transmission was successful. Then the controller begins to send the actual data.
The data to be written to the codetag has the following contents: The ASCII data line without BCC is: <STX> '4 B 6 G'. The data are Intel-Hex converted.
The BCC is performed:

- `<STX>` -> (02<sub>Hex</sub>) = 0000 0010 EXOR
- '4' -> (34<sub>Hex</sub>) = 0011 0011 EXOR
- 'B' -> (42<sub>Hex</sub>) = 0011 0100 EXOR
- '6' -> (36<sub>Hex</sub>) = 0011 0110 EXOR
- 'G' -> (47<sub>Hex</sub>) = 0011 0111 EXOR

Resulting blockcheck: BCC 0000 0101 = (05<sub>Hex</sub>)
The Hex value of the blockcheck BCC = (05<sub>Hex</sub>) is Intel-Hex converted:
(05<sub>Hex</sub>) -> '0' (30<sub>Hex</sub>) -> '5' (35<sub>Hex</sub>)
The Hex value 05<sub>Hex</sub> represents the control character <ENQ>, which would be interpreted upon receipt by the processor as the start of a new telegram and would by necessity cause a fault condition or lock up the system.
The processor acknowledges with <ACK> '0', if the telegram was received correctly.

Programming Information

Addressing the Processors

Each processor is factory set to address '00' (30<sub>Hex</sub>, 30<sub>Hex</sub>). Upon start-up on the Balluff-Bus, each new processor must first be assigned a valid device address in the range of '02' to '98'. Each address may only be used once.
Address '01' must be used for the Bus Master.
Address '99' is used as a general address to be used for communicating with all the processors on the Bus.

Sequence of Address Assigning

The first processor is connected on the Bus, and the address is assigned. Then the second unit is connected and its address assigned, etc.
When assigning addresses, only one (1) device on the Bus may have address '00' (factory setting), since the command for assigning an address is only accepted by this unit. If the command is executed successfully, the processor acknowledges with <ACK> '0', otherwise it sends the error message <NAK> + error number. See pages 31/32 for BCC information.
The telegram from the controller begins with the control character <ENQ> and the 2-digit device address (in ASCII) which is to be assigned to the processor. The address is given in ASCII, with a permissible range of '02' to '98'. When addressing, only one processor on the Bus is permitted to have address '00'. Only units having address '00' can be newly addressed using the 'G' command.
To re-address an already addressed unit, the current address must first be deleted using the 'K' command.
The address following the 'G' command represents the address of the preceding unit on the Bus. It is only needed for the 'D' and 'F' commands (for later implementation).
Programming Information

Example:
Addressing the Slave Units

The first processor connected on the Bus should receive address 02.
To do this, the following telegram is constructed in the controller and sent to the BIS processor:

<ENQ> '02 G 01 S0000 A0000 53' BCC (53 hex -> 35 hex, 33 hex)

Intel-Hex conversion:
- <ENQ> 0 5 Hex control characters (start of telegram)
- '0 3 0 Hex device address
- G 4 2 Hex command for assigning a new address
- 0 3 0 0 0 0 start address for continuous read
- 0 3 0 0 0 0 number of bytes for continuous read
- 0 3 0 0 0 0 53 3 Hex BCC 3

The processor responds with <ACK> '0' if the transaction was successful.
Otherwise <NAK> + error number is sent.

Start address and number of bytes for continuous read are transmitted with 4 digits, however, the sum of start address + number of bytes should not exceed the page size of the code tag:
- Code tag: Start address + number of bytes
  - < 2 kBytes < 32
  - > 2 kBytes < 64

As long as the function "Continuous read" is not realized, only 0 should be used here.

Assign address 03 to the 2nd unit. The predecessor address is 02:
The controller sends <ENQ> '03 G 02 S0000 A0000 51' BCC
The processor responds <ACK> '0'

The assigning of addresses does not need to be done in sequential or even increasing order. The 3rd unit could have address 20 and the 4th address 16. The 3rd unit would receive the following address:
The controller sends <ENQ> '20 G 03 S0000 A0000 51' BCC
The processor responds <ACK> '0'
The 4th unit would receive the following address:
The host controller sends <ENQ> '16 G 20 S0000 A0000 55' BCC
The processor responds <ACK> '0'

To find out which address has been assigned to any given processor use the 'U' command together with address 99. However, only the unit in question is permitted to be live on the Bus, since address 99 finds all slave units. To determine the address of the 3rd unit on the Bus, for example, all the other slave units on the Bus must first be powered down or disconnected from the Bus. The following telegram is sent:
The host controller sends <ENQ> '99 U 02' BCC
The processor responds <ACK> '0'

Reconnect or power up the other processors after finishing this operation.

Example:
Addressing the Slave Units (continued)
Delete (change) an Address

A processor address can be deleted. This is required before a processor with an existing address can be re-addressed. All the units on the Bus can remain intact as long as the assignment of the new address is done directly after the deletion. Otherwise, the unit with the deleted address must be disconnected from the Bus.

In the preceding example, the 4th processor shall be assigned address 25 instead of 16:

The host controller sends:  \textless ENQ \textgreater '16 K 4F' BCC
The processor responds: \textless ACK \textgreater '0'
Address 16 is deleted.

The host controller sends:  \textless ENQ \textgreater '25 G 20 S0000 A0000 55' BCC
The processor responds: \textless ACK \textgreater '0'
Address 25 is set.

Telegram Contents

<table>
<thead>
<tr>
<th>Start address and no. of bytes</th>
<th>The start address (A3, A2, A1, A0) and the number of bytes to send (L3, L2, L1, L0) are sent in decimal as ASCII characters. For the start address, the range 0000 to 8191 can be used, and for the number of bytes 0001 to 8192. A3 ... L0 represent one ASCII character each. Please note: Start address + number of bytes may not exceed the code tag capacity.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Head number and block size</td>
<td>In the 'L' command (read with head select and block size) and 'P' command (write with head select and block size), first the number of the read/write head K ('1' or '2') and then the code tag block size B ('0', '1') is sent. B = '0' means 64 Byte, B = '1' means 32 Byte.</td>
</tr>
<tr>
<td>Acknowledge</td>
<td>The acknowledgement \textless ACK \textgreater '0' is sent by the Identification System if the serially transmitted characters were recognized as correct and there is a code tag in the active zone of a read/write head. In the 'R' command, the \textless ACK \textgreater '0' is only sent if the data is ready for transmission. \textless NAK \textgreater + Error No. is sent if an error was recognized or if there is no code tag in the active zone of a read/write head.</td>
</tr>
<tr>
<td>Start</td>
<td>\textless STX \textgreater starts the data transmission.</td>
</tr>
<tr>
<td>Transmitted Bytes</td>
<td>The data are transmitted code transparent (no data conversion).</td>
</tr>
</tbody>
</table>
### Programming

<table>
<thead>
<tr>
<th>Task</th>
<th>Data Flow</th>
<th>Command</th>
<th>Start address of first byte to be sent</th>
<th>Number of bytes to be sent</th>
<th>Read/write Head No.</th>
<th>Block size</th>
<th>Acknowledge</th>
<th>Start transmission</th>
<th>Data from start address + no. of bytes</th>
<th>End Acknowledgement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read/Write from host system to BIS</td>
<td>ENQ xx P</td>
<td>A3 A2 A1 A0</td>
<td>'0   0   0   0' to '8   1   9   2'</td>
<td>'1' or '2'</td>
<td>1 or 0</td>
<td>64</td>
<td>ACK</td>
<td>D1 D2 D3 ... Dn</td>
<td>BCC</td>
<td>ACK or NAK + Error-No.</td>
</tr>
<tr>
<td>Read/Write from BIS to host system</td>
<td>ENQ xx P</td>
<td>A3 A2 A1 A0</td>
<td>'0   0   0   0' to '8   1   9   2'</td>
<td>'1' or '2'</td>
<td>1 or 0</td>
<td>64</td>
<td>ACK</td>
<td>D1 D2 D3 ... Dn</td>
<td>BCC</td>
<td>ACK or NAK + Error-No.</td>
</tr>
</tbody>
</table>

1) The command 'Quit' is not permitted at this point.
2) <ACK> '0' is returned as acknowledgement if there is no error, or <NAK> + Error-No. if an error occurs.

Values inside apostrophes represent the respective character(s) in ASCII code.
Programming

### Read from Code Tag, Write to Code Tag

<table>
<thead>
<tr>
<th>Task</th>
<th>Data Flow</th>
<th>Command</th>
<th>Start address of first byte to be sent</th>
<th>Number of bytes to be sent</th>
<th>Acknowledge</th>
<th>Start transmission</th>
<th>Data (from start address no. of bytes)</th>
<th>Acknowledge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>from host system to BIS</td>
<td>&lt;ENQ&gt;xx W</td>
<td>A3 A2 A1 A0 to '0 0 0 0'</td>
<td>L3 L2 L1 L0 to '0 0 0 1'</td>
<td>BCC</td>
<td>&lt;STX&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>from BIS to host system</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write</td>
<td>from host system to BIS</td>
<td>&lt;ENQ&gt;xx W</td>
<td>A3 A2 A1 A0 to '0 0 0 0'</td>
<td>L3 L2 L1 L0 to '0 0 0 1'</td>
<td>BCC</td>
<td>&lt;STX&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>from BIS to host system</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1) The command ‘Quit’ is not permitted at this point.
2) <ACK> ‘0’ is returned as acknowledgement if there is no error, or <NAK> + Error No. if an error occurs.

Values inside apostrophes represent the respective character(s) in ASCII code.

### Telegram example

**Read from Code Tag:**

Read 10 bytes starting at address 50.

The host sends: `<ENQ> '02 R 0 0 5 0 0 1 0 51' BCC -> (35 Hex, 31Hex)`

Address of first byte to read

The BIS processor acknowledges with `<ACK> '0'`

The host gives the start command `<STX>`

The BIS processor provides the data from the code tag `31 32 33 34 35 36 37 38 39 30 01' BCC`

**Write to Code Tag:**

Write 5 bytes starting at address 500.

The host system sends: `<ENQ> '02 W 0 5 0 0 0 0 5 50' BCC -> (35 Hex, 30 Hex)`

The BIS processor acknowledges with `<ACK> '0'`

The host sends the data `<STX> '31 32 33 34 35 05' BCC`

The BIS processor acknowledges with `<ACK> '0'`

The ‘R’ and ‘W’ commands represent a subtype of the ‘L’ and ‘P’ commands without selection of head or block size.

Values inside apostrophes represent the respective character(s) in ASCII code.
Programming

Selecting a Read/Write Head
(two heads have to be connected to the processor)

The 'H1' command selects Read/Write Head 1, 'H2' Read/Write Head 2, and 'HT' (Head Twin) both Read/Write Heads.

1. Only one code tag is allowed to be in the active zone of a read/write head at a time.
2. The read or write time increases by ca. 40 ms - regardless of the data amount to be read or written. (This does not apply to the code tag recognition).
3. The positive acknowledgement for a read or write action is no longer <ACK> '0' but rather <ACK> '1' or <ACK> '2', depending on at which read/write head there is a code tag to be read from or written to.

1) The commands 'Status' and/or 'Quit' are not permitted at this point.
2) <ACK> '0' is returned as acknowledgement if there is no error, or <NAK> + Error No. if an error occurs.

Values inside apostrophes represent the respective character(s) in ASCII code.

Telegram example: Selecting a Read/Write Head
Telegram with block check (BCC)

クト

Find Next Code Tag
(one time)

The following telegram is used to find the next code tag. The next following read/write head is selected and checked to see if a code tag is in front of this read/write head. If yes, the first 4 bytes of the code tag are read. The telegram reply then contains the corresponding number of the read/write head and the four bytes read. If no tag is found, the original read/write head is reselected and checked. If no code tag is found here, then the telegram reply is: 'H ? 0000 77'.

1) The commands 'Status' and/or 'Quit' are not permitted at this point.

Values inside apostrophes represent the respective character(s) in ASCII code.
Programming

Find Next Code Tag (continuous)

The following telegram is used to find the next code tag. The next following read/write head is selected and checked to see if a code tag is in front of this read/write head. If yes, the first four bytes of the code tag are read. The telegram reply then contains the corresponding number of the read/write head and the four bytes read. If no tag is found, the original read/write head is reselected and checked. This procedure is repeated until a code tag is found.

H “!” recognizes any code tag, regardless of the preset block size, assuming that read/write head and code tag are compatible.

1) The command "Quit" is not permitted at this point.

Telegram example: Find next code tag (continuous) at device address 33

1)  The command "Quit" is not permitted at this point.

<table>
<thead>
<tr>
<th>Task</th>
<th>Data Flow</th>
<th>Start with address of processor</th>
<th>Command</th>
<th>Designator</th>
<th>Terminator</th>
<th>Acknowledge</th>
<th>Head number</th>
<th>data from code tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find next code tag (continuous)</td>
<td>from host system to BIS</td>
<td>xx</td>
<td>H</td>
<td>0 or 2</td>
<td>DT 02 0D 0A</td>
<td>BCC</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Values inside apostrophes represent the respective character(s) in ASCII code.

Configuring the Outputs

A telegram can be sent to configure or cancel the four outputs.

1) The command "Quit" is not permitted at this point.

Designator meaning:

- Output No.: 00 to 99
- Cancel output: 00 to 30
- Configure output: 01 to 31

Telegram example: Set output 2 at device address 22

1) The command "Quit" is not permitted at this point.

Values inside apostrophes represent the respective character(s) in ASCII code.
Show Output Condition

This telegram is used to check the condition (status) of all four outputs.

<table>
<thead>
<tr>
<th>Task</th>
<th>Data Flow</th>
<th>Start with address of processor</th>
<th>Command</th>
<th>Acknowledge</th>
<th>Condition of the 4 outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show output condition</td>
<td>from host system to BIS</td>
<td>ENQ 'xx'</td>
<td>0</td>
<td>AU</td>
<td>BCC</td>
</tr>
<tr>
<td>from BIS to host system</td>
<td></td>
<td>CAUK 'v'</td>
<td>XXXX</td>
<td>BCC</td>
<td></td>
</tr>
</tbody>
</table>

1) The command 'Quit' is not permitted at this point.

Output status is shown in sequential order 0 1 2 3

Telegram example:
Output all outputs at device address 02

-> Outputs 0 and 1 are set, outputs 2 and 3 are cancelled.

The host sends
<ENQ> '02 BAO 4B' BCC -> (34 Hex, 42Hex)

The BIS processor acknowledges with
<ACK> '0' '1100 36' BCC -> (33 Hex, 36Hex)

Values inside apostrophes represent the respective character(s) in ASCII code.

Programming

Restart the Processor (Quit)

Sending the Restart command causes a telegram in process to be aborted and puts the processor in the ground state. Use of address 99 is not allowed. After this telegram is acknowledged, a ca. 500 ms pause should be allowed before starting a new telegram.

Important! The Quit command is not permitted while the processor is waiting for the terminator (BCC). In this situation, the Quit would be incorrectly interpreted as a terminator or datum.

1) The command 'Quit' is not permitted at this point.

Telegram example:
Restart the Processor (Quit)

-> Put processor with device address 15 into the ground state.

The host sends
<ENQ> '15 Q 50' BCC (35 Hex, 30 Hex)

The BIS processor acknowledges with
'Q 51' BCC (35 Hex, 31Hex)

Values inside apostrophes represent the respective character(s) in ASCII code.
### Error Numbers

The BIS C-600 always outputs an error number. The meaning of these error numbers is indicated in the following table.

<table>
<thead>
<tr>
<th>Error No.</th>
<th>Description</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No code tag present</td>
<td>Telegram aborted, processor goes into ground state.</td>
</tr>
<tr>
<td>2</td>
<td>Read error</td>
<td>Read telegram aborted, processor goes into ground state.</td>
</tr>
<tr>
<td>3</td>
<td>Read aborted, since the code tag was removed</td>
<td>Processor goes into ground state.</td>
</tr>
<tr>
<td>4</td>
<td>Write error</td>
<td>Write telegram aborted, processor goes into ground state. CAUTION: Some new data may still have been written to the code tag!</td>
</tr>
<tr>
<td>5</td>
<td>Writing aborted, since the code tag was removed</td>
<td>Processor goes into ground state. CAUTION: Some new data may still have been written to the code tag!</td>
</tr>
<tr>
<td>6</td>
<td>Interface error</td>
<td>Processor goes into ground state. (parity or stop bit error)</td>
</tr>
<tr>
<td>7</td>
<td>Telegram format error</td>
<td>Processor goes into ground state. Possible format errors: - Command is not ‘R’, ‘W’, ‘L’, ‘P’, ‘H’, ‘Q’ or ‘S’. - Start address or number of bytes exceed permissible range</td>
</tr>
<tr>
<td>8</td>
<td>BCC error, the transmitted BCC is wrong</td>
<td>Telegram is aborted, processor goes into ground state.</td>
</tr>
<tr>
<td>9</td>
<td>Cable break, Codetag Present LED flashes</td>
<td>Telegram is aborted, processor goes into ground state. Cable break from read/write head or cable not connected. If both read/write heads were selected using ‘HT’, one head may not be connected. If both read/write heads are selected, the cable break message only comes if there is no code tag in front of the connected, functional head.</td>
</tr>
</tbody>
</table>

A New command not possible, since a read command is already in process. After error message the read command is stopped, internally, but not acknowledged. Processor goes into ground state.

B New command not possible, since a write command is already in process. After error message the write command is stopped, internally, but not acknowledged. Processor goes into ground state. CAUTION: If errors occur after new attempts to write to the code tag, no further error messages will be given.

C New command not possible, since a head select is already in process. After the error message, no positive acknowledge is given, even thought the head select was successful. Processor goes into ground state.
### Interface

<table>
<thead>
<tr>
<th>Condition</th>
<th>LED</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>System on</td>
<td>green</td>
<td>Supply voltage OK; no hardware error</td>
</tr>
<tr>
<td>Ready off</td>
<td></td>
<td>Supply voltage or hardware not OK</td>
</tr>
<tr>
<td>Codetag on</td>
<td>yellow</td>
<td>Code tag ready to read or write. If a read/write error occurs during a read/write operation, System Ready LED goes out, if the protocol variant &quot;without error number&quot; is used)</td>
</tr>
<tr>
<td>Codetag Present on (yellow) flashes</td>
<td>Read/write head cable break or not connected. If both read/write heads were selected using &quot;HT&quot;, one head may not be connected. If both read/write heads are selected, the cable break message only comes if there is no code tag in front of the connected, functional head.</td>
<td></td>
</tr>
<tr>
<td>Codetag Operating on (yellow) off</td>
<td>Command being processed</td>
<td>No code tag in active zone of read/write head</td>
</tr>
<tr>
<td>Operating off</td>
<td></td>
<td>No command in process</td>
</tr>
</tbody>
</table>

If all three LED’s flash on and off synchronously, the processor needs to be returned to the factory for repair.

### Opening the BIS C-600 Processor

In order to make the interface connections, the BIS C-600 processor must be opened. Take care that the unit has no power to it. Unthread the 4 screws and remove the cover. Feed the connector cable through the cable fittings. See the following pages for additional wiring details.

Mounting of the cover (4 screws), max. permissible tightening torque: 0.15 Nm

Write your configuration data on the sticker supplied and affix it to the inside of the cover.
### Connections to the Terminal Block

<table>
<thead>
<tr>
<th>Terminal Block Connections</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
</tr>
<tr>
<td>+ VS</td>
</tr>
<tr>
<td>POWER</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
</tr>
<tr>
<td>+ VS</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>INPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
</tr>
<tr>
<td>COM</td>
</tr>
</tbody>
</table>

### Interface Connections

#### RS 422 Interface (4-wire)

- **Terminal connections** for an RS 422 interface
- **Power, 2nd read/write head**
- **RS 422: Shunt-connector X 4/1 must be plugged in!**

#### RS 485 Interface (2-wire)

- **Terminal connections** for an RS 485 interface
- **Power, 2nd read/write head**
- **RS 485: Shunt-connector X 4/2 must be plugged in!**

#### Jumpers for termination resistor

<table>
<thead>
<tr>
<th>X4</th>
<th>X4/3</th>
</tr>
</thead>
<tbody>
<tr>
<td>X4/2</td>
<td>X4/4</td>
</tr>
</tbody>
</table>

- **Passive**
- **Active** plugged

---

**The BIS C-600-024...04 uses a serial interface for communication with the host system (PLC or PC). Depending on how it is configured, the interface can be used as RS 422 (4-wire) or RS 485 (2-wire) preferably.**
Assembly of the BIS C-600 Processor and Read/Write Head or Adapter

The processor is mounted using the 4 oval holes on the sides. Depending on model, the processor includes a read/write head or adapter for separate read/write head. Both the read/write head and the adapter can be rotated as desired by +/-90 deg. (see illustration).

Be sure power is off. Remove both screws (indicated by arrows in diagram). Carefully remove the head or adapter in the direction of the arrow shown in right illustration.

Caution: cables inside! Assemble in the desired orientation and refasten the screws.

Assembling a Read/Write Head or Adapter

To change a read/write head: Turn power off and open processor. Remove mounting screws from read/write head (see page 55). Unthread the 4 screws and remove the cover (see page 52). Remove the read/write head connector from the board and pull the cable out through the cable passage. To attach the new head, follow these steps in reverse.

To attach a BIS C-650 Adapter, proceed as described above. Both connection cables must be plugged into the board.

Caution! When connecting the BIS C-650 Adapter, no read/write head is allowed to be connected to the terminal block.

Connection for integrated read/write head

Connections for Adapter BIS C-650

1 = Head 1
2 = Head 2

Mounting of the cover (4 screws), max. permissible tightening torque: 0.15 Nm
Before connecting a series BIS C-3_ read/write head (except BIS C-350 or -352) to the terminal block of the BIS C-600 processor, the connector at the end of the cable must first be removed.

Please note that the cable must be cut directly behind the connector as shown, since the cable length affects the function of the read/write head. If the cable is shortened, reliable operation is no longer assured.

The read/write head cable has a maximum length of 5 m.

**Preparation of the cable and connector**

The cable jacket should be stripped for a distance of ca. 5 cm (2”). The leads should be stripped for ca. 5 mm (1/4”) and fitted with crimp contacts with 0.25 to 0.34 mm² cross section.

<table>
<thead>
<tr>
<th>Lead color</th>
<th>Function</th>
<th>BIS C-600 terminal</th>
</tr>
</thead>
<tbody>
<tr>
<td>BU</td>
<td>AH</td>
<td>15</td>
</tr>
<tr>
<td>BN</td>
<td>EH</td>
<td>14</td>
</tr>
<tr>
<td>shield</td>
<td>GND</td>
<td>16</td>
</tr>
</tbody>
</table>

The wiring table for the 2nd read/write head to the BIS C-600 Processor:

Wiring table for the 2nd read/write head to the BIS C-600 Processor

**Technical Data**

**BIS C-600 Processor**

**Dimensions, Weight**

- **Housing**
  - Dimensions with read/write head BIS C-652: 167 x 90 x 35 mm
  - Dimensions with Adapter BIS C-650: 145 x 90 x 35 mm
- **Weight**: 500 g

**Temperature Range**

- **Operating temperature**: 0 °C to +60 °C

**Connections**

- **Terminal block**: 19-pin
- **Cable fittings**: 3 x PG 9
- **Conductor size**: 0.14 to 1 mm²
  - with crimp terminals: 0.25 to 0.34 mm²

**Protection Class**

- **IP Rating**: IP 65

**Electrical Connections**

- **Supply Voltage Vₛ, Input**: 24 V DC ± 20 %
- **Ripple**: ≤ 10 %
- **Current drain**: ≤ 400 mA
- **Serial Interface**: RS 485 (2-wire) / RS 422 (4-wire)
- **Read/Write Head**
  - **option with mounted adapter BIS C-650 1)**
    - integrated, BIS C-65_, and following 1);
    - 2 x connectors 4-pin (male)
    - for all read/write heads BIS C-3_ _ with 4-pin connector (female),
    - except BIS C-350 and BIS C-352

1) can be rotated by +/-90 deg.
### Technical Data
#### BIS C-600 Processor

<table>
<thead>
<tr>
<th><strong>Electrical Connections</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Digital Input (+IN, –IN)</strong></td>
<td>Optocoupler isolated</td>
</tr>
<tr>
<td>Control voltage active</td>
<td>4 V to 40 V</td>
</tr>
<tr>
<td>Control voltage inactive</td>
<td>1.5 V to –40 V</td>
</tr>
<tr>
<td>Input current at 24 V</td>
<td>11 mA</td>
</tr>
<tr>
<td>Typical delay time</td>
<td>5 ms</td>
</tr>
<tr>
<td><strong>Control Outputs (01 to 04)</strong></td>
<td>Optocoupler isolated</td>
</tr>
<tr>
<td>Output current</td>
<td>max. 20 mA</td>
</tr>
<tr>
<td>Voltage drop at 20 mA</td>
<td>ca. 2.5 V</td>
</tr>
<tr>
<td>Output resistance $R_o$</td>
<td>10 kΩ to $V_o$</td>
</tr>
<tr>
<td>Logic</td>
<td>PNP (emitter)</td>
</tr>
<tr>
<td>Supply voltage, Output $V_o$</td>
<td>24 V DC ± 5 %</td>
</tr>
<tr>
<td>Ripple</td>
<td>≤ 10 %</td>
</tr>
</tbody>
</table>

#### Function Displays
- **System Ready LED**: green
- **Codetag Present LED**: yellow
- **Codetag Operating LED**: yellow

---

**With the CE Mark we affirm that our products are in accordance with the requirements of the EU (European Union) Guideline 89/336/EEC (EMC Guideline) and the EMC Law. It has been verified in our EMC Laboratory, which is accredited by the DAfTec for Testing of Electromagnetic Compatibility, that Balluff products meet the EMC requirements of the Harmonized Standard EN 50081-2 (Emission), EN 50082-2 (Noise Immunity).**

---

### Ordering Information
#### BIS C-600 Processor

<table>
<thead>
<tr>
<th><strong>Ordering Code</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Balluff Identification System</strong></td>
<td></td>
</tr>
<tr>
<td>Series C</td>
<td></td>
</tr>
<tr>
<td>Hardware type</td>
<td></td>
</tr>
<tr>
<td>600 = Compact</td>
<td></td>
</tr>
<tr>
<td>Software type</td>
<td></td>
</tr>
<tr>
<td>024 = Balluff protocol for Balluff Bus</td>
<td></td>
</tr>
<tr>
<td><strong>Read/Write Head</strong></td>
<td></td>
</tr>
<tr>
<td>000 = no read/write head</td>
<td></td>
</tr>
<tr>
<td>650 = with two connections for external read/write heads BIS C-3_ _ (except BIS C-355 und -352)</td>
<td></td>
</tr>
<tr>
<td>651 = with read/write head BIS C-651 (circular antenna on top)</td>
<td></td>
</tr>
<tr>
<td>652 = with read/write head BIS C-652 (circular antenna on front)</td>
<td></td>
</tr>
<tr>
<td>653 = with read/write head BIS C-653 (bar antenna)</td>
<td></td>
</tr>
<tr>
<td><strong>Interface</strong></td>
<td></td>
</tr>
<tr>
<td>04 = RS 485 (2-wire)</td>
<td></td>
</tr>
<tr>
<td><strong>User Connections</strong></td>
<td></td>
</tr>
<tr>
<td>KL1 = 3x PG 9 cord seal fittings</td>
<td></td>
</tr>
</tbody>
</table>
### Appendix, ASCII Table

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>Ctrl@</td>
<td>NUL</td>
<td>22</td>
<td>16</td>
<td>CtrlV</td>
<td>SYN</td>
<td>44</td>
<td>2C</td>
<td></td>
<td></td>
<td>66</td>
<td>41</td>
<td>A</td>
<td></td>
<td>86</td>
<td>56</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>01</td>
<td>CtrlA</td>
<td>SOH</td>
<td>23</td>
<td>17</td>
<td>CtrlW</td>
<td>ETX</td>
<td>45</td>
<td>2D</td>
<td></td>
<td></td>
<td>67</td>
<td>43</td>
<td>C</td>
<td></td>
<td>87</td>
<td>57</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>02</td>
<td>CtrlB</td>
<td>STX</td>
<td>24</td>
<td>18</td>
<td>CtrlX</td>
<td>CAN</td>
<td>46</td>
<td>2E</td>
<td></td>
<td></td>
<td>68</td>
<td>44</td>
<td>D</td>
<td></td>
<td>88</td>
<td>58</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>03</td>
<td>CtrlC</td>
<td>ETX</td>
<td>25</td>
<td>19</td>
<td>CtrlY</td>
<td>EM</td>
<td>47</td>
<td>2F</td>
<td></td>
<td></td>
<td>69</td>
<td>45</td>
<td>E</td>
<td></td>
<td>89</td>
<td>59</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>04</td>
<td>CtrlD</td>
<td>EOT</td>
<td>26</td>
<td>1A</td>
<td>CtrlZ</td>
<td>SUB</td>
<td>48</td>
<td>30</td>
<td>0</td>
<td></td>
<td>105</td>
<td>6B</td>
<td>k</td>
<td></td>
<td>110</td>
<td>6E</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>05</td>
<td>CtrlE</td>
<td>ENQ</td>
<td>27</td>
<td>1B</td>
<td>Ctrl</td>
<td>ESC</td>
<td>49</td>
<td>31</td>
<td>1</td>
<td></td>
<td>112</td>
<td>70</td>
<td>p</td>
<td></td>
<td>113</td>
<td>71</td>
<td>q</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>06</td>
<td>CtrlF</td>
<td>ACK</td>
<td>28</td>
<td>1C</td>
<td>Ctrl\</td>
<td>PS</td>
<td>50</td>
<td>32</td>
<td>2</td>
<td></td>
<td>114</td>
<td>72</td>
<td>r</td>
<td></td>
<td>116</td>
<td>73</td>
<td>s</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>07</td>
<td>CtrlG</td>
<td>BEL</td>
<td>29</td>
<td>1D</td>
<td>Ctrl</td>
<td>GS</td>
<td>51</td>
<td>33</td>
<td>3</td>
<td></td>
<td>117</td>
<td>74</td>
<td>t</td>
<td></td>
<td>118</td>
<td>75</td>
<td>u</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>08</td>
<td>CtrlH</td>
<td>BS</td>
<td>30</td>
<td>1E</td>
<td>Ctrl</td>
<td>US</td>
<td>52</td>
<td>34</td>
<td>4</td>
<td></td>
<td>119</td>
<td>76</td>
<td>v</td>
<td></td>
<td>119</td>
<td>77</td>
<td>w</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>09</td>
<td>CtrlJ</td>
<td>FF</td>
<td>31</td>
<td>1F</td>
<td>Ctrl</td>
<td>US</td>
<td>53</td>
<td>35</td>
<td>5</td>
<td></td>
<td>120</td>
<td>78</td>
<td>x</td>
<td></td>
<td>121</td>
<td>79</td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0A</td>
<td>CtrlK</td>
<td>LF</td>
<td>32</td>
<td>20</td>
<td>Ctrl</td>
<td>SF</td>
<td>54</td>
<td>36</td>
<td>6</td>
<td></td>
<td>122</td>
<td>7A</td>
<td>z</td>
<td></td>
<td>122</td>
<td>7A</td>
<td>z</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>0B</td>
<td>CtrlL</td>
<td>VT</td>
<td>33</td>
<td>21</td>
<td>Ctrl</td>
<td>FS</td>
<td>55</td>
<td>37</td>
<td>7</td>
<td></td>
<td>123</td>
<td>7B</td>
<td>{}</td>
<td></td>
<td>123</td>
<td>7B</td>
<td>{}</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>0C</td>
<td>CtrlM</td>
<td>CR</td>
<td>34</td>
<td>22</td>
<td>Ctrl</td>
<td>RS</td>
<td>56</td>
<td>38</td>
<td>8</td>
<td></td>
<td>124</td>
<td>7C</td>
<td></td>
<td></td>
<td></td>
<td>124</td>
<td>7C</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>0D</td>
<td>CtrlN</td>
<td>SI</td>
<td>35</td>
<td>23</td>
<td>Ctrl</td>
<td>US</td>
<td>57</td>
<td>39</td>
<td>9</td>
<td></td>
<td>125</td>
<td>7D</td>
<td>\</td>
<td></td>
<td>125</td>
<td>7D</td>
<td>\</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>0E</td>
<td>CtrlO</td>
<td>SO</td>
<td>36</td>
<td>24</td>
<td>Ctrl</td>
<td>US</td>
<td>58</td>
<td>3A</td>
<td></td>
<td></td>
<td>126</td>
<td>7E</td>
<td>&quot;</td>
<td></td>
<td>126</td>
<td>7E</td>
<td>&quot;</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>0F</td>
<td>CtrlP</td>
<td>ST</td>
<td>37</td>
<td>25</td>
<td>Ctrl</td>
<td>US</td>
<td>59</td>
<td>3B</td>
<td></td>
<td></td>
<td>127</td>
<td>7F</td>
<td>DEL</td>
<td></td>
<td>127</td>
<td>7F</td>
<td>DEL</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>CtrlQ</td>
<td>DC1</td>
<td>38</td>
<td>26</td>
<td>Ctrl</td>
<td>US</td>
<td>60</td>
<td>3C</td>
<td></td>
<td></td>
<td>128</td>
<td>7G</td>
<td></td>
<td></td>
<td>128</td>
<td>7G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>CtrlR</td>
<td>DC2</td>
<td>39</td>
<td>27</td>
<td>Ctrl</td>
<td>US</td>
<td>61</td>
<td>3D</td>
<td></td>
<td></td>
<td>129</td>
<td>7H</td>
<td></td>
<td></td>
<td>129</td>
<td>7H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>CtrlS</td>
<td>DC3</td>
<td>40</td>
<td>28</td>
<td>Ctrl</td>
<td>US</td>
<td>62</td>
<td>3E</td>
<td></td>
<td></td>
<td>130</td>
<td>7I</td>
<td></td>
<td></td>
<td>130</td>
<td>7I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>13</td>
<td>CtrlT</td>
<td>DC4</td>
<td>41</td>
<td>29</td>
<td>Ctrl</td>
<td>US</td>
<td>63</td>
<td>3F</td>
<td></td>
<td></td>
<td>131</td>
<td>7J</td>
<td></td>
<td></td>
<td>131</td>
<td>7J</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>CtrlU</td>
<td>NAK</td>
<td>42</td>
<td>2A</td>
<td>Ctrl</td>
<td>US</td>
<td>64</td>
<td>40</td>
<td></td>
<td></td>
<td>132</td>
<td>7K</td>
<td></td>
<td></td>
<td>132</td>
<td>7K</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>15</td>
<td>CtrlV</td>
<td>KAN</td>
<td>43</td>
<td>2B</td>
<td>Ctrl</td>
<td>US</td>
<td>65</td>
<td>41</td>
<td></td>
<td></td>
<td>133</td>
<td>7L</td>
<td></td>
<td></td>
<td>133</td>
<td>7L</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Decimal Codes:**
- 0-31: Control codes
- 32-126: ASCII characters

**Hexadecimal Codes:**
- 20-7F: ASCII characters
- 80-FF: Non-ASCII control codes

**ASCII Decimals:**
- 0-9: Digits
- A-F: Letters
- A0-AF: Reserved

**Hex Codes:**
- 00-0F: ASCII characters
- 10-1F: Control codes

**Control Codes:**
- SOH: Start of Header
- STX: Start of Text
- ETX: End of Text
- EOT: End of Transmission
- ENQ: Enquiry
- ACK: Acknowledge
- BEL: Bell
- BS: Backspace
- HT: Horizontal Tab
- NL: Linefeed
- CR: Carriage Return
- FF: Form Feed
- DLE: Data Link Escape
- DC1: Device Control 1
- DC2: Device Control 2
- DC3: Device Control 3
- DC4: Device Control 4

**ASCII Characters:**
- @: @
- #: #
- $: $
- %: %
- &: &
- ' : '
- ( : (}
- ) : )
- * : *
- + : +
- , : ,
- . : .
- / : /
- : : :
- ; : ;
- < : <
- = : =
- > : >
- ? : ?
- @ : @
- ^ : ^
- _ : _
- ` : `