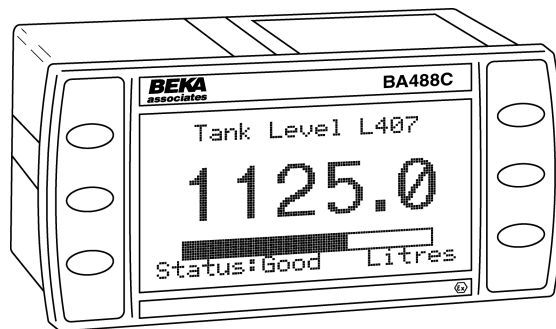
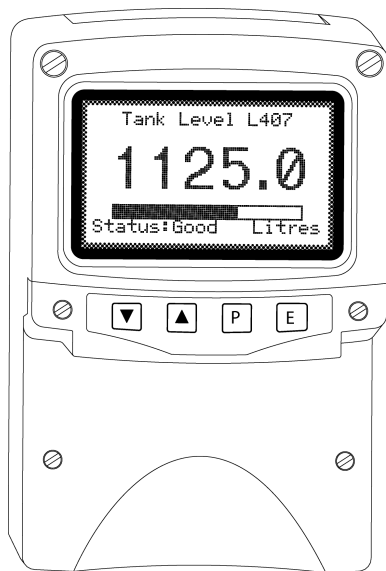




**Serial Text Display**  
**Modbus Interface Guide**  
*[Version 3.4 Firmware]*



**This guide applies to the following models:**

**BA488C** - *Panel mounted, Intrinsically Safe*

**BA484D** - *Field mounted, Intrinsically Safe*

**BA688C** - *Panel mounted, Safe Area*

**BA684D** - *Field mounted, Safe Area*

# Contents

<b>Introduction .....</b>	<b>4</b>
<i>What's in this Modbus Interface Guide .....</i>	<i>4</i>
<i>What's in the Programming Guide.....</i>	<i>4</i>
<i>What's in the Instruction Manuals.....</i>	<i>4</i>
<i>What's new in this version .....</i>	<i>4</i>
<i>Other sources of information.....</i>	<i>4</i>
<b>Product Overview .....</b>	<b>5</b>
<i>Display .....</i>	<i>5</i>
<i>Analogue Input Display .....</i>	<i>5</i>
<i>Switch Inputs.....</i>	<i>5</i>
<i>Switch Outputs .....</i>	<i>5</i>
<b>Standard Screens .....</b>	<b>6</b>
<b>General Operation .....</b>	<b>7</b>
<i>Setting the Address.....</i>	<i>7</i>
<i>Setting the Screen Option .....</i>	<i>7</i>
<i>Configuring the values to be displayed.....</i>	<i>7</i>
<i>Data Timeout.....</i>	<i>8</i>
<i>Data Status .....</i>	<i>8</i>
<i>Configuring the decimal places to be displayed.....</i>	<i>8</i>
<i>Configuring Units display and Tag description .....</i>	<i>9</i>
<i>Configuring Bargraph Displays .....</i>	<i>9</i>
<i>Reading the keypress status .....</i>	<i>9</i>
<i>Controlling the alarm outputs.....</i>	<i>9</i>
<b>Advanced Operation.....</b>	<b>10</b>
<i>Transmitting Text Display Commands.....</i>	<i>10</i>
<i>Transmitting and Receiving Bulk Data needed by some commands.....</i>	<i>11</i>
<i>Transmitting Scripts.....</i>	<i>11</i>
<b>Scripting .....</b>	<b>12</b>
<i>Uploading Scripts.....</i>	<i>12</i>
<i>Controlling Scripts .....</i>	<i>12</i>
<i>Debugging Scripts.....</i>	<i>12</i>
<i>Don't Panic !.....</i>	<i>12</i>
<b>Modbus Register Address Map .....</b>	<b>13</b>
<i>Supported Modbus Functions.....</i>	<i>15</i>

# **Introduction**

This guide gives all the necessary information to use our Serial Text Displays in a Modbus installation.

Several other protocols are commonly used in industry which we may choose to support as market demand rises. This guide and others will periodically be updated, so please come back to our website regularly for the latest information.

For hardware installation information, please refer to the separate instruction manuals available for each model.

For information on customising the screen displays of the Serial Text Display, please refer to the “Serial Text Display Programming Guide”.

## **What’s in this Modbus Interface Guide**

- An overview of each instrument
- A description of the parameters that are applicable to each instrument.
- Instructions on how to use the instrument in its standard non-programmed modes

## **What’s in the Programming Guide**

- A description of the Serial Text Display
- An overview of the protocol
- Specific information on more advanced features
- A command summary, where the commands are grouped together by function and presented in a series of tables
- A command reference, where each command is listed in alphabetical order and covered in detail. The information is presented in a consistent layout and examples given to demonstrate the use of the command in context.

## **What’s in the Instruction Manuals**

- An overview of the instrument
- Intrinsic Safety Certification information
- System Design and Installation
- Configuration
- Maintenance

## **What’s new in this version**

- Numeric data can now be entered into 16 and 32 bit integer registers in addition to floating point registers
- Two additional standard screens have been added to show all 8 variables at once
- Unused variables can now be hidden from standard screens
- String variables have been added to scripting commands

## **Other sources of information**

Our website at [www.beka.co.uk](http://www.beka.co.uk) is kept up to date with the latest literature and information

After reading through this guide, if you still have a problem getting the results you need then email us at [support@beka.co.uk](mailto:support@beka.co.uk) and we will do our best to help you

# **Product Overview**

A detailed overview of the instrument is given in the instruction manual for each product. This should be read before implementing any system using these instruments. However it is useful to summarise the main features of the product before attempting to design any controlling software application.

## **Display**

The instrument display is organised as 120 pixels horizontally by 64 pixels vertically. Each pixel is approximately 0.7mm square which makes it ideal for displaying text and simple graphics. The size of the pixels improves the contrast and hence the readability at greater distances.

The display is also backlit by an ultra-efficient green LED module which enables the screen to be viewed in all conditions, from bright sunlight to total darkness.

## **Analogue Input Display**

The primary purpose of this instrument is to display process variables that exist in a system. Eleven pre-programmed screen layouts are available to display one, two, three, four or eight variables simultaneously. A total of eight (8) variables can be accessed by using the front panel push buttons.

For applications that require a customised display, the unit can be programmed by following the instructions in the “Serial Text Display - Programming Guide” (available from [www.beka.co.uk](http://www.beka.co.uk)). It is possible to map a number of Modbus registers to a corresponding set of formatted text strings such that they are automatically updated without any further intervention. As this guide concentrates on the non-programmed modes of operation, such advanced use is outside the scope of this document.

## **Switch Inputs**

There are six switches on the front of the panel mounted instrument, and four on the field mounted instrument. Both models have the option of overriding these with up to six external switches which can be sized and labelled to suit the application. The status of these switches can be read via Modbus registers.

## **Switch Outputs**

There are two switch outputs available. These are totally isolated and can be energised or de-energised independently of each other. They are controlled by Modbus registers and are intended to be used for indication only.

# Standard Screens

There are eleven standard screens available which require no application programming. They are capable of displaying a selection of up to eight process variables, together with their units of measure and tag description. Once a screen format has been chosen, each input variable can be brought into view by pressing the up and down arrow keys.

These standard screens are ideal for many simple applications and can be implemented very quickly. However, where a unique display format is required these can be built up using the commands that can be found in the “Serial Text Display – Programming Guide”

The screen format is selected by either using the local menu (as described in the Instruction Manual) or by using the “**Screen Option**” Modbus register. One of eleven standard display formats can be selected as shown in the following table:

Screen Option 1	Inst1 Tag <b>21.835</b> Status:Good    Units	Screen Option 2	Inst1 Tag    Units <b>21.8350</b> ----- Inst2 Tag    Units <b>529.3300</b>																																												
Screen Option 3	<table style="border-collapse: collapse; margin: auto;"> <tr> <td style="border-right: 1px solid black; padding: 2px;">Inst1 Tag Units 21.835</td> <td style="padding: 2px;">Inst3 Tag Units -3.105</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 2px;">Inst2 Tag Units 529.33</td> <td style="padding: 2px;">Inst4 Tag Units -5600.</td> </tr> </table>	Inst1 Tag Units 21.835	Inst3 Tag Units -3.105	Inst2 Tag Units 529.33	Inst4 Tag Units -5600.	Screen Option 4	Inst1 Tag <b>21.835</b>  Status:Good    Units																																								
Inst1 Tag Units 21.835	Inst3 Tag Units -3.105																																														
Inst2 Tag Units 529.33	Inst4 Tag Units -5600.																																														
Screen Option 5	<table style="border-collapse: collapse; margin: auto;"> <tr> <td style="text-align: right;">Temperature</td> <td style="text-align: left;">°C</td> </tr> <tr> <td></td> <td></td> </tr> <tr> <td style="text-align: right;">21.46</td> <td></td> </tr> <tr> <td style="text-align: right;">Pressure</td> <td style="text-align: left;">Pa</td> </tr> <tr> <td></td> <td></td> </tr> <tr> <td style="text-align: right;">1.7500</td> <td></td> </tr> </table>	Temperature	°C			21.46		Pressure	Pa			1.7500		Screen Option 6	Temperature  <b>25.25</b>  °C																																
Temperature	°C																																														
21.46																																															
Pressure	Pa																																														
1.7500																																															
Screen Option 7	<table style="border-collapse: collapse; margin: auto;"> <tr> <td style="text-align: right;">Temp</td> <td style="text-align: left;">°C</td> <td style="text-align: right;">Pressure</td> <td style="text-align: left;">Pa</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td style="text-align: right;">25.22</td> <td></td> <td style="text-align: right;">1.750</td> <td></td> </tr> </table>	Temp	°C	Pressure	Pa					25.22		1.750		Screen Option 8	<table style="border-collapse: collapse; margin: auto;"> <tr> <td style="text-align: right;">Temp</td> <td style="text-align: right;">Press</td> <td style="text-align: right;">Flow</td> </tr> <tr> <td></td> <td></td> <td></td> </tr> <tr> <td style="text-align: right;">24.46 °C</td> <td style="text-align: right;">1.7500 Pa</td> <td style="text-align: right;">48.9 l/min</td> </tr> </table>	Temp	Press	Flow				24.46 °C	1.7500 Pa	48.9 l/min																							
Temp	°C	Pressure	Pa																																												
25.22		1.750																																													
Temp	Press	Flow																																													
24.46 °C	1.7500 Pa	48.9 l/min																																													
Screen Option 9	<table style="border-collapse: collapse; margin: auto;"> <tr> <td style="text-align: right;">Temp</td> <td style="text-align: right;">Pres</td> <td style="text-align: right;">Flow</td> <td style="text-align: right;">Fill</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td style="text-align: right;">22.73</td> <td style="text-align: right;">1.750</td> <td style="text-align: right;">45.4</td> <td style="text-align: right;">11.36</td> </tr> </table>	Temp	Pres	Flow	Fill					22.73	1.750	45.4	11.36	Screen Option 10	<table style="border-collapse: collapse; margin: auto;"> <tr> <td>In_1</td> <td>Tag</td> <td>10.000</td> <td>Units</td> </tr> <tr> <td>In_2</td> <td>Tag</td> <td>20.000</td> <td>Units</td> </tr> <tr> <td>In_3</td> <td>Tag</td> <td>30.000</td> <td>Units</td> </tr> <tr> <td>In_4</td> <td>Tag</td> <td>40.000</td> <td>Units</td> </tr> <tr> <td>In_5</td> <td>Tag</td> <td>50.000</td> <td>Units</td> </tr> <tr> <td>In_6</td> <td>Tag</td> <td>60.000</td> <td>Units</td> </tr> <tr> <td>In_7</td> <td>Tag</td> <td>70.000</td> <td>Units</td> </tr> <tr> <td>In_8</td> <td>Tag</td> <td>80.000</td> <td>Units</td> </tr> </table>	In_1	Tag	10.000	Units	In_2	Tag	20.000	Units	In_3	Tag	30.000	Units	In_4	Tag	40.000	Units	In_5	Tag	50.000	Units	In_6	Tag	60.000	Units	In_7	Tag	70.000	Units	In_8	Tag	80.000	Units
Temp	Pres	Flow	Fill																																												
22.73	1.750	45.4	11.36																																												
In_1	Tag	10.000	Units																																												
In_2	Tag	20.000	Units																																												
In_3	Tag	30.000	Units																																												
In_4	Tag	40.000	Units																																												
In_5	Tag	50.000	Units																																												
In_6	Tag	60.000	Units																																												
In_7	Tag	70.000	Units																																												
In_8	Tag	80.000	Units																																												
Screen Option 11	<table style="border-collapse: collapse; margin: auto;"> <tr> <td style="text-align: right;">In_1</td> <td></td> <td>10.000Unit</td> </tr> <tr> <td style="text-align: right;">In_2</td> <td></td> <td>20.000Unit</td> </tr> <tr> <td style="text-align: right;">In_3</td> <td></td> <td>30.000Unit</td> </tr> <tr> <td style="text-align: right;">In_4</td> <td></td> <td>40.000Unit</td> </tr> <tr> <td style="text-align: right;">In_5</td> <td></td> <td>50.000Unit</td> </tr> <tr> <td style="text-align: right;">In_6</td> <td></td> <td>60.000Unit</td> </tr> <tr> <td style="text-align: right;">In_7</td> <td></td> <td>70.000Unit</td> </tr> <tr> <td style="text-align: right;">In_8</td> <td></td> <td>80.000Unit</td> </tr> </table>	In_1		10.000Unit	In_2		20.000Unit	In_3		30.000Unit	In_4		40.000Unit	In_5		50.000Unit	In_6		60.000Unit	In_7		70.000Unit	In_8		80.000Unit	Screen Option 0	Custom screens																				
In_1		10.000Unit																																													
In_2		20.000Unit																																													
In_3		30.000Unit																																													
In_4		40.000Unit																																													
In_5		50.000Unit																																													
In_6		60.000Unit																																													
In_7		70.000Unit																																													
In_8		80.000Unit																																													

Setting the “**Screen Option**” Modbus register to a value of 0 will allow custom screens to be displayed by using BEKA protocol commands.

# General Operation

The primary purpose of a BEKA Serial Text Display when using the Modbus protocol is to enable local indication of up to 8 process variables. This is normally achieved by writing to the Modbus cyclic data registers which are mapped out in tables starting on page 13.

It is important to note that some input registers are capable of accepting numbers that the Display cannot handle. To indicate when this has happened, the **"Write Error"** register is set to a value of "1". The host can therefore check to see if the display has been able to process the data satisfactorily.

The Text Display will return an Exception Response "Slave Busy" to any modbus command if it is in the local configuration menus.

## Setting the Address

Each Serial Text Display should be given a unique Modbus address via the user menu. Any address between 1 and 247 is acceptable. Note that the broadcast address "0" is supported, allowing multiple displays to show identical data in different locations, but should only be used if no other instrument types are connected at the same time; doing so will yield unpredictable results.

## Setting the Screen Option

A Serial Text Display is normally configured to respond to ASCII commands. In order to use the standard screens the unit has to be configured by setting the **"Screen Option"** register to a value between 1 and 11, as shown in the preceding table. Setting it to a value above 11 will result in the **"Write Error"** register being set to "1".

Setting the **"Screen Option"** register to a value of 0 will allow custom screens to be created and displayed by using BEKA protocol commands.

## Configuring the values to be displayed

Up to eight process variables can be displayed. These are written to the **"IN<sub>x</sub> Value"** in the Modbus Register Address Map, where 'x' is a value between 1 and 8. The data format can now be either 16 bit integer, 32 bit integer or big endian IEEE floating point. 32 bit integers and floats must be written as a pair of registers, with the lowest register address first. Multiple values can be written with one command.

When an integer value is written to the display it is internally converted to a float prior to being displayed or saved.

The **"Divisor"** register defines the number of times an integer value is divided by ten before being processed by the display. For example, if the **"Divisor"** register is set to 2 and a cyclic value of 12345 is written using a 32-bit integer, the value 123.45 will be displayed.

If the display over-ranges all the digits display the question mark character: '?'.

The operator can select which input variable is displayed on the screen by pressing the up and down arrow keys, or the host can take control by setting the **"Screen Variable"** register to a value between 1 and 8. This forces the unit to display that input variable using the current screen format i.e. if 2 variables per screen are being shown, then setting the **"Screen Variable"** register to a value of 4 will show input variables 3 and 4. Note that the **"Screen Variable"** register will be updated to the first variable being shown i.e. 3 in this example.

A menu option **"Last Input"** and a new register **"Variable Last"** have been added to limit the number of variables displayed in standard screens. This allows scrolling via the up/down buttons of only the used variables rather than cycle through the complete set of 8. In multivariable screens, variable entries above the 'last variable shown' are just blank

## Data Timeout

The "*Cyclic Data Timeout*" register is used to set the longest allowable time between cyclic data updates. The number in this register is multiplied by 10 Seconds to give the actual time period. The default value is 1, giving a timeout period of 10 Seconds.

As each input variable can be updated independently, there are eight "*IN<sub>x</sub> Timeout*" registers to indicate which "*IN<sub>x</sub> Value*" has not been updated recently. In addition, if the input value is being displayed on the screen, its status is shown as **BAD**.

## Data Status

The "*IN<sub>x</sub> Status*" is used to mark the data **GOOD** or **BAD**. When marked **BAD** the appearance of the value will be in inverse video i.e. clear pixels on a dark background.

The value in the status register must be set to a "1" to mark it as **GOOD**. Setting the value to "0" will mark it as **BAD**, and the values will be displayed in inverse video. Note that data will always be marked as **BAD** if the timeout period has been exceeded.

## Configuring the decimal places to be displayed

The number of decimal places that are displayed is selected by writing to the "*IN<sub>x</sub> Number Format*" register.

This function enables the position of the decimal point on each of the eight displayed variables to be individually defined. Six options are available:

- 0 - No decimal point
- 1 - 1 digit on right of decimal point
- 2 - 2 digits on right of decimal point
- 3 - 3 digits on right of decimal point
- 4 - 4 digits on right of decimal point
- 5 - Maximum resolution with the selected display format.

The total number of display digits available depends upon the display screen selected:

Screen Option 1	1 Variable	5,7,11 or 17 digits available (Font automatically resizes)
Screen Option 2	2 Variables	7 digits available
Screen Option 3	4 Variables	5 digits available
Screen Option 4	1 Variable + Bar	5,7,11 or 17 digits available (Font automatically resizes)
Screen Option 5	2 Variable + Bars	7 digits available
Screen Option 6	1 Variable + Bar	6 digits available
Screen Option 7	2 Variable + Bars	4 digits available
Screen Option 8	3 Variable + Bars	6 digits available
Screen Option 9	4 Variable + Bars	4 digits available
Screen Option 10	8 Variables	5 digits available
Screen Option 11	8 Variable + Bars	5 digits available

If a negative number is likely to be displayed, a digit must be allocated for the negative sign. If the display over-ranges all the digits display the question mark character: '?'.  
'?'

For all options, leading zeros are automatically suppressed.

The default value of "*IN<sub>x</sub> Number Format*" is "5", but once set is stored in non-volatile memory.



## Configuring Units display and Tag description

Each of the eight process variables may be displayed with a ‘Tag description’ that can consist of up to sixteen characters. This is simply achieved by writing to the "**IN\_x Description**" register. Each variable can also have an associated eight character ‘Units’ value displayed alongside by writing to the "**IN\_x Units**" register. Note that for some Screen Options the display space available for the Tag and Units is smaller than the register size, so it is important to check that the resultant display is acceptable.

In each case, the character strings can include upper and lower case, numbers and punctuation. To simplify temperature display, the ‘°’ character (alt+096) is mapped to the degrees symbol.

For example, the string **Temp °C** is displayed as **Temp °C**

Information written to these two registers is saved to non-volatile memory and is retained if power to the display is lost.

## Configuring Bargraph Displays

Some standard screens incorporate horizontal and vertical bargraphs. These can be scaled by writing to the "**IN\_x Minimum Bargraph Limit**" and "**IN\_x Maximum Bargraph Limit**" registers in either IEEE floating point format or 16/32 bit integers with a divisor. Floats must be written as a pair of registers, with the lowest register address first.

If these registers are not configured, the defaults of 0 and 100 will be used, which may (or may not) be suitable for the intended application. If the input variable is higher or lower than both bargraph limits then the bargraph is drawn as an unfilled dotted outline only, indicating that a bargraph should be present but cannot be meaningfully drawn.

## Reading the keypress status

The "**KEY x**" registers returns information on the keys pressed since the parameter was last read. A “1” in the register indicates that the key has been pressed. Each time the parameter is read it will be reset to “0”. Care must be taken in the configuration of the host application such that keypresses are not missed by polling at inappropriate times.

The unit has the facility to connect external switches in addition to the front panel buttons. By selecting the appropriate “Keys” configuration in the local menu these external switches can be simple normally open or closed contacts that can be used for a variety of basic signalling tasks.

## Controlling the alarm outputs

The "**OUTPUT x**" registers are used to directly control the local alarm output circuits. Writing a “1” in the register will turn the output On (short circuit), and writing a “0” will turn the output Off (open circuit).

# Advanced Operation

Some applications may require a custom screen to be displayed, in which case the full range of BEKA protocol commands can be used. The possibilities are endless, but typical examples may be:

- Display of text and simple graphics. This permits the use of the display as a basic operator terminal. Keypad button presses are latched and can be read over Modbus.
- Design of custom screens that contain text, graphics and embedded process variables that are automatically updated from received cyclic data.

Full details of how to use these more advanced text and graphics capabilities of the display are contained in the “Serial Text Display - Programming Guide”.

The “Command String” and “Graphic Data” registers are used to send the required commands and graphics to the display.

## Transmitting Text Display Commands

Please refer to the “Serial Text Display - Programming Guide” which describes the commands in detail and gives practical examples of their use. A summary of the basic procedures are shown below:

The command format is: <AB[param1],[param2]...,[paramN]>

where:

AB is the command.

[ ] indicates optional parameters separated by commas

example:

<CS><CM4,90><CI>

where: CS - Clear Screen  
CM4,90 - Cursor Move to Row 4 Column 90  
CI - Command Implement

The commands are written to the “**COMMAND STRING**” register starting at address 300. They may be written either one at a time, or several may be grouped together into one long string. **NOTE:** Every command (or group of combined commands) must be terminated with the <CI> Command Implement command. The reception of this command causes the unit to process the contents of its input buffer. **No action will be taken if the <CI> is omitted.**

The maximum length of a command string is 32 bytes.

The result is obtained by reading the “**RESULT**” register, which is an unsigned 16 word with the following valid responses:

- 0x00 - indicates that the previous command/command set has been accepted.
- 0x01 - indicates a parameter or communications error has been detected in the previous command string.
- 0x02 - indicates the command is unrecognised.
- 0x04 - indicates that a message has been received but NOT actioned because the unit is in programming mode.
- 0x08 - indicates that no BEKA command has yet been actioned.
- 0x10 - indicates that a script is running.
- 0x20 - indicates that the script is busy. ( See the \*BB command )

As many commands may be passed and actioned during a screen update, a mechanism has been provided to ensure the host knows which command the result refers to. Two registers have been added to provide a method of matching commands to their results. The sequence of events should be as follows

1. Write a numeric value 'n' into the "**COMMAND\_ID**" register
2. Write the command string (including the terminating <CI> command) to the "**COMMAND\_STRING**" register
3. Continually read the "**RESULT\_ID**" parameter until it equals the value 'n' set in the "**COMMAND\_ID**" register
4. Read the "**RESULT**" parameter: This is the result given by the command string.

### **Transmitting and Receiving Bulk Data needed by some commands**

Some commands require a large amount of data to be supplied. Examples of this are the <DF>Download Font, <DG>Download Graphic and <DS>Download Screen commands.

When communicating via ACSII the command is issued first, and then data is transmitted. With Modbus, it is done slightly differently: The data is first transferred via the "**GRAPHIC\_DATA**" register, and then the command is issued. The main reason for this is that the data download can take place at any time, so there is no long delay between the command being issued and the screen changing appearance.

The graphics data to be download must first be loaded into the "**GRAPHIC\_DATA**" register. The size of this block is only 64 bytes, therefore, files must be split and loaded in segments.

The <GBn> command is used to specify the segment that a subsequent write to the "**GRAPHIC\_DATA**" register goes into. The value *n* can be in the range of 0 to 9. The file to be downloaded must start at the beginning of segment 0 and fill as many segments as necessary to download all of the data. Once the desired number of segments are filled with data, the <DF>, <DG> or <DS> command is then issued; The downloaded object is then processed and displayed.

Note that any data at the end of the file and in higher numbered segments is ignored.

The <US> command works in an identical way, but graphics data is made available by the display in 64 byte segments.

### **Transmitting Scripts**

The process described above is slightly changed when uploading a script to the display. The <GBn> command is not required, but the <DPn>Download Program command is instead used in a similar way.

The script to be download is loaded into the "**GRAPHIC\_DATA**" register, but as the size of this block is only 64 bytes the script must be split and loaded in segments.

The <DPn> command is used to transfer the 64 bytes from the "**GRAPHIC\_DATA**" register into memory. The transfer process **MUST** start with <DP0> and be done in ascending order. The value of *n* can be from 0 to 31, meaning that a 2048 byte script can be loaded. The end of the script is marked by a zero following the command zero terminator, i.e. two consecutive 0x00s.

For further information please refer to the following "Scripting" section, and the appropriate chapter in the "Serial Text Display – Programming Guide"

# Scripting

The Serial Text Display has a powerful scripting language built into the firmware. Full details on writing and controlling scripts are documented in the appropriate chapter in the “Serial Text Display – Programming Guide”. It is essential that a full understanding of the scripting facilities is gained before attempting to customise the display.

Scripts are conveniently developed and uploaded using ASCII commands directly from a workstation. However, the facilities have been duplicated over Modbus to give application developers the ability to control the display while connected to a live system. In such cases, the additional registers that appear shaded in the Modbus map (Page 13) should be used. The functions of these registers are briefly described below, but these should be read in conjunction with the information contained in the Programming Guide.

## Uploading Scripts

Once a script has been developed, it can be uploaded using the procedures covered in the “Transmitting Scripts” section of this manual (Page 11).

## Controlling Scripts

If a script is loaded into the display it will automatically run when power is applied to the instrument.

The “*Execute Script*” coil register can be set to “0” to stop the script , or to “1” to start the script.

The “*Script Running*” status register can be interrogated to determine whether a script is running or not.

The “*Jump Control*” register provides a mechanism for the host to influence the running script. A value of 1 to 255 can be written by the host, and the script can use the \*GR*n* command to perform an action based on this value. As this is a polled command, it is typically used as a simple method of displaying a specific message in response to an alarm condition. The register is cleared as soon as the script engine has completed the jump.

The “*Event Control*” register is similar to “Jump Control” except that it does not require a register to be polled; the action is taken immediately. A value of 1 to 20 can be written to the register, and it is cleared as soon as the script engine has responded to the event.

A “*Slave Busy*” exception will be returned to a write to the jump register or event control register if a script is busy.

## Debugging Scripts

Errors that are encountered when running a script are displayed on the screen, giving the line number that the error occurred. The error also stops the script engine from executing any further commands and puts a value of “1” in the the “*Script Error*” status register.

The host can read the error message by interrogating the “*Error String*” register and find out the offending line number by reading the “*Error Line*” register.

## Don't Panic !

As it is very easy to stop the display from functioning correctly by uploading a defective script, we have provided the ability to control scripts from the user menu. Simply press the “P” and “E” buttons on the front panel and select the “Script Control” menu item. From here scripts can be stopped or erased, restoring the unit back to its unprogrammed state.

# Modbus Register Address Map

Note: Shaded rows denote advanced registers that can be disregarded when using Standard Screens.

Coils	Read / Write		
Address	Bits	Description	Functions Supported
1	1	IN 1 Status	1, 5, 15
2	1	IN 2 Status	1, 5, 15
3	1	IN 3 Status	1, 5, 15
4	1	IN 4 Status	1, 5, 15
5	1	IN 5 Status	1, 5, 15
6	1	IN 6 Status	1, 5, 15
7	1	IN 7 Status	1, 5, 15
8	1	IN 8 Status	1, 5, 15
9	1	Output 1	1, 5, 15
10	1	Output 2	1, 5, 15
11	1	Execute Script	1, 5, 15
12	1	Update Strings	1, 5, 15
13	1	Show Status	1, 5, 15

Notes:

Status:	0 = Bad	1 = Good
Outputs:	0 = Off	1 = On
Script:	0 = Stopped	1 = Running
Update Strings	0 = No Action	1 = Update strings
Status Indication	0 = Off	1 = On

Input Status	Read Only		
Address	Bits	Description	Functions Supported
1	1	Key 1	2
2	1	Key 2	2
3	1	Key 3	2
4	1	Key 4	2
5	1	Key 5	2
6	1	Key 6	2
7	1	IN 1 Timeout	2
8	1	IN 2 Timeout	2
9	1	IN 3 Timeout	2
10	1	IN 4 Timeout	2
11	1	IN 5 Timeout	2
12	1	IN 6 Timeout	2
13	1	IN 7 Timeout	2
14	1	IN 8 Timeout	2
15	1	Write Error	2
16	1	Script Error	2

Notes:

Keys:	0 = Not Pressed	1 = Pressed
Timeout:	0 = No Timeout	1 = Timeout
Errors:	0 = No Error	1 = Error

Input Registers	Read Only		
Address	Registers	Description	Functions Supported
1	9	Unit Type (18 ASCII bytes)	4
10	9	Software Version (18 ASCII bytes)	4
19	9	Error String (18 ASCII bytes)	4
28	1	Error Line (Unsigned Int)	4
29	1	Result ID (Unsigned 16)	4
30	1	Result (MSB = 0, LSB = Result)	4

Notes:

Result Enumeration:	0 = No Error
	1 = Parameter Error
	2 = Command Error
	4 = In Menus
	8 = No Command Yet Executed
	16 = Script Running
	32 = Script Busy

Holding Registers	Read / Write				
Address	Registers	Description	Functions Supported	Default Value	Saved in E <sup>2</sup>
<b>Cyclic Data Registers</b>					
1	2	IN_1 Value (4 bytes IEEE format)	3, 16		-
3	2	IN_2 Value (4 bytes IEEE format)	3, 16		-
5	2	IN_3 Value (4 bytes IEEE format)	3, 16		-
7	2	IN_4 Value (4 bytes IEEE format)	3, 16		-
9	2	IN_5 Value (4 bytes IEEE format)	3, 16		-
11	2	IN_6 Value (4 bytes IEEE format)	3, 16		-
13	2	IN_7 Value (4 bytes IEEE format)	3, 16		-
15	2	IN_8 Value (4 bytes IEEE format)	3, 16		-
<b>Control Registers</b>					
25	1	Screen Option (MSB = 0, Screen Option = LSB)	3, 6, 16	0 (Text Display)	Saved
26	1	Cyclic Data Timeout (MSB = 0, Timeout Value = LSB)	3, 6, 16	1 (10S)	Saved
27	1	Screen Variable (MSB = 0, Screen Variable = LSB)	3, 6, 16		-
28	1	Jump Control (MSB = 0, Jump Control = LSB)	3, 6, 16		-
29	1	Event Control (MSB = 0, Event Control = LSB)	3, 6, 16		-
30	1	Variable Last	3, 6, 16		Saved
31	1	Divisor (n/10)	3, 6, 16	0	Saved
<b>Configuration Registers</b>					
100	8	IN_1 Description (16 ASCII bytes)	3, 6, 16	"Inst1 Tag"	Saved
108	8	IN_2 Description (16 ASCII bytes)	3, 6, 16	"Inst2 Tag"	Saved
116	8	IN_3 Description (16 ASCII bytes)	3, 6, 16	"Inst3 Tag"	Saved
124	8	IN_4 Description (16 ASCII bytes)	3, 6, 16	"Inst4 Tag"	Saved
132	8	IN_5 Description (16 ASCII bytes)	3, 6, 16	"Inst5 Tag"	Saved
140	8	IN_6 Description (16 ASCII bytes)	3, 6, 16	"Inst6 Tag"	Saved
148	8	IN_7 Description (16 ASCII bytes)	3, 6, 16	"Inst7 Tag"	Saved
156	8	IN_8 Description (16 ASCII bytes)	3, 6, 16	"Inst8 Tag"	Saved
164	4	IN_1 Units (8 ASCII bytes)	3, 6, 16	"Units"	Saved
168	4	IN_2 Units (8 ASCII bytes)	3, 6, 16	"Units"	Saved
172	4	IN_3 Units (8 ASCII bytes)	3, 6, 16	"Units"	Saved
176	4	IN_4 Units (8 ASCII bytes)	3, 6, 16	"Units"	Saved
180	4	IN_5 Units (8 ASCII bytes)	3, 6, 16	"Units"	Saved
184	4	IN_6 Units (8 ASCII bytes)	3, 6, 16	"Units"	Saved
188	4	IN_7 Units (8 ASCII bytes)	3, 6, 16	"Units"	Saved
192	4	IN_8 Units (8 ASCII bytes)	3, 6, 16	"Units"	Saved
196	1	IN_1 Number format (MSB = 0, LSB = Number Format)	3, 6, 16	5 (Auto)	Saved
197	1	IN_2 Number format (MSB = 0, LSB = Number Format)	3, 6, 16	5 (Auto)	Saved
198	1	IN_3 Number format (MSB = 0, LSB = Number Format)	3, 6, 16	5 (Auto)	Saved
199	1	IN_4 Number format (MSB = 0, LSB = Number Format)	3, 6, 16	5 (Auto)	Saved
200	1	IN_5 Number format (MSB = 0, LSB = Number Format)	3, 6, 16	5 (Auto)	Saved
201	1	IN_6 Number format (MSB = 0, LSB = Number Format)	3, 6, 16	5 (Auto)	Saved
202	1	IN_7 Number format (MSB = 0, LSB = Number Format)	3, 6, 16	5 (Auto)	Saved
203	1	IN_8 Number format (MSB = 0, LSB = Number Format)	3, 6, 16	5 (Auto)	Saved
250	2	IN_1 Minimum Bargraph Limit (4 bytes IEEE format)	3, 16	0	Saved
252	2	IN_1 Maximum Bargraph Limit (4 bytes IEEE format)	3, 16	100	Saved
254	2	IN_2 Minimum Bargraph Limit (4 bytes IEEE format)	3, 16	0	Saved
256	2	IN_2 Maximum Bargraph Limit (4 bytes IEEE format)	3, 16	100	Saved
258	2	IN_3 Minimum Bargraph Limit (4 bytes IEEE format)	3, 16	0	Saved
260	2	IN_3 Maximum Bargraph Limit (4 bytes IEEE format)	3, 16	100	Saved
262	2	IN_4 Minimum Bargraph Limit (4 bytes IEEE format)	3, 16	0	Saved
264	2	IN_4 Maximum Bargraph Limit (4 bytes IEEE format)	3, 16	100	Saved
266	2	IN_5 Minimum Bargraph Limit (4 bytes IEEE format)	3, 16	0	Saved
268	2	IN_5 Maximum Bargraph Limit (4 bytes IEEE format)	3, 16	100	Saved
270	2	IN_6 Minimum Bargraph Limit (4 bytes IEEE format)	3, 16	0	Saved
272	2	IN_6 Maximum Bargraph Limit (4 bytes IEEE format)	3, 16	100	Saved
274	2	IN_7 Minimum Bargraph Limit (4 bytes IEEE format)	3, 16	0	Saved
276	2	IN_7 Maximum Bargraph Limit (4 bytes IEEE format)	3, 16	100	Saved
278	2	IN_8 Minimum Bargraph Limit (4 bytes IEEE format)	3, 16	0	Saved
280	2	IN_8 Maximum Bargraph Limit (4 bytes IEEE format)	3, 16	100	Saved
300	16	Command String (32 ASCII bytes)	3, 16		-
316	32	Graphic Data (64 ASCII bytes)	3, 6, 16		-
348	1	Command ID (Unsigned 16)	3, 6, 16		-
<b>Cyclic Data Registers</b>					
401	1	IN_1 Value (16 Bit Integer format)	3, 6, 16		
402	1	IN_2 Value (16 Bit Integer format)	3, 6, 16		
403	1	IN_3 Value (16 Bit Integer format)	3, 6, 16		
404	1	IN_4 Value (16 Bit Integer format)	3, 6, 16		
405	1	IN_5 Value (16 Bit Integer format)	3, 6, 16		
406	1	IN_6 Value (16 Bit Integer format)	3, 6, 16		
407	1	IN_7 Value (16 Bit Integer format)	3, 6, 16		
408	1	IN_8 Value (16 Bit Integer format)	3, 6, 16		

Holding Registers	Read / Write				
Address	Registers	Description	Functions Supported	Default Value	Saved in E <sup>2</sup>
<b>Configuration Registers</b>					
450	1	IN_1 Minimum Bargraph Limit (16 Bit Integer format)	3, 6, 16	0	Saved
451	1	IN_1 Maximum Bargraph Limit (16 Bit Integer format)	3, 6, 16	100	Saved
452	1	IN_2 Minimum Bargraph Limit (16 Bit Integer format)	3, 6, 16	0	Saved
453	1	IN_2 Maximum Bargraph Limit (16 Bit Integer format)	3, 6, 16	100	Saved
454	1	IN_3 Minimum Bargraph Limit (16 Bit Integer format)	3, 6, 16	0	Saved
455	1	IN_3 Maximum Bargraph Limit (16 Bit Integer format)	3, 6, 16	100	Saved
456	1	IN_4 Minimum Bargraph Limit (16 Bit Integer format)	3, 6, 16	0	Saved
457	1	IN_4 Maximum Bargraph Limit (16 Bit Integer format)	3, 6, 16	100	Saved
458	1	IN_5 Minimum Bargraph Limit (16 Bit Integer format)	3, 6, 16	0	Saved
459	1	IN_5 Maximum Bargraph Limit (16 Bit Integer format)	3, 6, 16	100	Saved
460	1	IN_6 Minimum Bargraph Limit (16 Bit Integer format)	3, 6, 16	0	Saved
461	1	IN_6 Maximum Bargraph Limit (16 Bit Integer format)	3, 6, 16	100	Saved
462	1	IN_7 Minimum Bargraph Limit (16 Bit Integer format)	3, 6, 16	0	Saved
463	1	IN_7 Maximum Bargraph Limit (16 Bit Integer format)	3, 6, 16	100	Saved
464	1	IN_8 Minimum Bargraph Limit (16 Bit Integer format)	3, 6, 16	0	Saved
465	1	IN_8 Maximum Bargraph Limit (16 Bit Integer format)	3, 6, 16	100	Saved
<b>Cyclic Data Registers</b>					
501	2	IN_1 Value (32 Bit Integer format)	3, 16		
503	2	IN_2 Value (32 Bit Integer format)	3, 16		
505	2	IN_3 Value (32 Bit Integer format)	3, 16		
507	2	IN_4 Value (32 Bit Integer format)	3, 16		
509	2	IN_5 Value (32 Bit Integer format)	3, 16		
511	2	IN_6 Value (32 Bit Integer format)	3, 16		
513	2	IN_7 Value (32 Bit Integer format)	3, 16		
515	2	IN_8 Value (32 Bit Integer format)	3, 16		
<b>Configuration Registers</b>					
550	2	IN_1 Minimum Bargraph Limit (32 Bit Integer format)	3, 16	0	Saved
552	2	IN_1 Maximum Bargraph Limit (32 Bit Integer format)	3, 16	100	Saved
554	2	IN_2 Minimum Bargraph Limit (32 Bit Integer format)	3, 16	0	Saved
556	2	IN_2 Maximum Bargraph Limit (32 Bit Integer format)	3, 16	100	Saved
558	2	IN_3 Minimum Bargraph Limit (32 Bit Integer format)	3, 16	0	Saved
560	2	IN_3 Maximum Bargraph Limit (32 Bit Integer format)	3, 16	100	Saved
562	2	IN_4 Minimum Bargraph Limit (32 Bit Integer format)	3, 16	0	Saved
564	2	IN_4 Maximum Bargraph Limit (32 Bit Integer format)	3, 16	100	Saved
566	2	IN_5 Minimum Bargraph Limit (32 Bit Integer format)	3, 16	0	Saved
568	2	IN_5 Maximum Bargraph Limit (32 Bit Integer format)	3, 16	100	Saved
570	2	IN_6 Minimum Bargraph Limit (32 Bit Integer format)	3, 16	0	Saved
572	2	IN_6 Maximum Bargraph Limit (32 Bit Integer format)	3, 16	100	Saved
574	2	IN_7 Minimum Bargraph Limit (32 Bit Integer format)	3, 16	0	Saved
576	2	IN_7 Maximum Bargraph Limit (32 Bit Integer format)	3, 16	100	Saved
578	2	IN_8 Minimum Bargraph Limit (32 Bit Integer format)	3, 16	0	Saved
580	2	IN_8 Maximum Bargraph Limit (32 Bit Integer format)	3, 16	100	Saved
600	8	Script String Variable 1	3, 16		
608	8	Script String Variable 2	3, 16		
616	8	Script String Variable 3	3, 16		
624	8	Script String Variable 4	3, 16		
632	8	Script String Variable 5	3, 16		
640	8	Script String Variable 6	3, 16		
640	8	Script String Variable 7	3, 16		
648	8	Script String Variable 8	3, 16		
656	8	Script String Variable 9	3, 16		
664	8	Script String Variable 10	3, 16		
672	8	Script String Variable 11	3, 16		
680	8	Script String Variable 12	3, 16		
688	8	Script String Variable 13	3, 16		
696	8	Script String Variable 14	3, 16		
704	8	Script String Variable 15	3, 16		
712	8	Script String Variable 16	3, 16		

## **Supported Modbus Functions**

The Modbus functions that are supported by the Serial Text Display are as follows:

- |                             |                                  |
|-----------------------------|----------------------------------|
| 01 - Read Coils             | 06 - Write Single Register       |
| 02 - Read Discrete Inputs   | 15 - Write Multiple Coils        |
| 03 - Read Holding registers | 16 - Write Multiple Registers    |
| 04 - Read Input Registers   | 43 - Read Device Identification. |
| 05 - Write Single Coil      |                                  |



BEKA Associates  
Old Charlton Road  
Hitchin  
Hertfordshire  
SG5 2DA

Tel: +44 (0)1462 438301  
Fax: +44 (0)1462 453971

Web: [www.beka.co.uk](http://www.beka.co.uk)  
Email: [support@beka.co.uk](mailto:support@beka.co.uk)  
or [sales@beka.co.uk](mailto:sales@beka.co.uk)