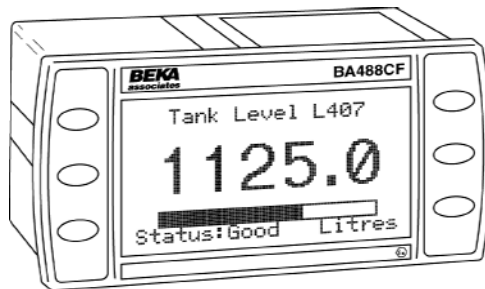
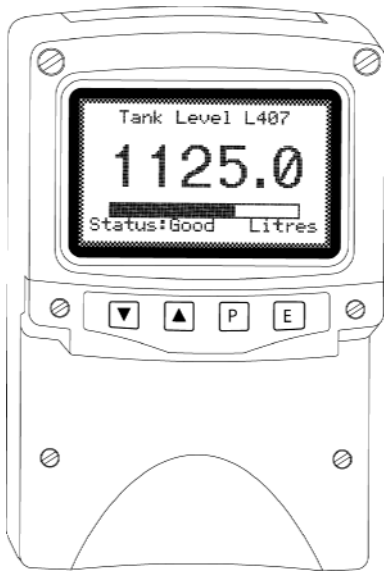




Fieldbus Display Programming Guide

[Version 2 Firmware]



This guide applies to the following models:

BA488CF - *Panel mounted, Intrinsically Safe*

BA484DF - *Field mounted, Intrinsically Safe*

BA688CF - *Panel mounted, Safe Area*

BA684DF - *Field mounted, Safe Area*

Contents

Introduction	1
<i>Model Variants.....</i>	<i>1</i>
<i>What's in this Programming Guide.....</i>	<i>2</i>
<i>What's in each Fieldbus Interface Guide.....</i>	<i>2</i>
<i>What's in the Instruction Manuals.....</i>	<i>2</i>
<i>Other sources of information.....</i>	<i>2</i>
Instrument Features	3
<i>Display.....</i>	<i>3</i>
<i>Analogue Input Display.....</i>	<i>3</i>
<i>Switch Inputs.....</i>	<i>3</i>
<i>Switch Outputs</i>	<i>3</i>
A few words about Modes... ..	4
Display Features	5
<i>Write Modes.....</i>	<i>5</i>
<i>Background Modes</i>	<i>6</i>
Frames	7
<i>Active and Visible Frames</i>	<i>7</i>
<i>Saved Frame Locations.....</i>	<i>8</i>
The BEKA Protocol.....	9
<i>Command format</i>	<i>9</i>
<i>Result format.....</i>	<i>10</i>
<i>Command and Result Timing.....</i>	<i>10</i>
<i>Key Press Information</i>	<i>10</i>
Graphics Transfers.....	11
<i>File Format</i>	<i>11</i>
<i>Downloads.....</i>	<i>11</i>
<i>Uploads.....</i>	<i>12</i>
Standard Screens	13
Command Summary	14
<i>Screen Handling & Text.....</i>	<i>14</i>
<i>Attributes</i>	<i>15</i>
<i>Pixel Graphics.....</i>	<i>15</i>
<i>Line Graphics.....</i>	<i>16</i>
<i>System.....</i>	<i>16</i>
<i>Mapped Fieldbus Variables</i>	<i>17</i>

Command Reference	18
<.>	Command..... 19
<AA>	Alarm Activation..... 20
<AF>	Active Frame..... 21
<AH>	Alarm Hysteresis..... 22
<AL>	Alarm Lower Limit..... 23
<AM>	Alarm Mapping..... 24
<AU>	Alarm Upper Limit..... 25
<BDy,x,l>	Box Draw..... 26
<BM>	Background Mode..... 27
<BS>	Boot Screen..... 28
<CA>	Centre Align..... 29
<CE>	Configuration Enable..... 30
<CI>	Command Implement..... 31
<CL>	Clear Line..... 32
<CM>	Cursor Move..... 33
<CP>	Configuration Prohibit..... 34
<CS>	Clear Screen..... 35
<CW>	Clear Window..... 36
<DBmnpqr>	Define Bargraph..... 37
<DDn,m>	Define Decimal..... 38
<DF>	Download Font..... 39
<DG>	Download Graphic..... 40
<DL>	Define Limits..... 41
<DS>	Download Screen..... 42
<DTn,string>	Define Tag..... 43
<DUn,string>	Define Units..... 44
<DVmnpq>	Define Variable..... 45
<DWyt,yb,xl,xr>	Define Window..... 46
<EB>	Erase Bargraph..... 47
<EF>	Enable Flashing..... 48
<EL>	Erase Line..... 49
<EV>	Erase Variable..... 50
<F1>	Font 1..... 51
<F2>	Font 2..... 52
<F3>	Font 3..... 53
<F4>	Font 4..... 54
<F5>	Font 5..... 55
<FL>	Flashing..... 56
<FR>	Font Restore..... 57
<FS>	Fill Screen..... 58
<FW>	Fill Window..... 59
<GB>	Graphics Block..... 60
<HBm,n>	Horizontal Bargraph..... 61
<HC>	Home Cursor..... 62
<HSm,n,p,q,r,s,t>	Horizontal Scroll..... 63
<IF>	Inhibit Flashing..... 64
<ISm,n,p>	Input Scaling..... 65
<KF>	Keep Fonts..... 66
<LA>	Left Align..... 67
<LF>	Line Feed..... 68
<LHx,l>	Line Horizontal..... 69
<LN>	Line New..... 70
<LVy,l>	Line Vertical..... 71

<NA>	No Align.....	72
<NL>	No Linefeed.....	73
<NS>	New Screen.....	74
<NU>	No Underline.....	75
<ODn>	Output De-energised.....	76
<OEn>	Output Energised.....	77
<PM>	Pixel Mode.....	78
<RA>	Right Align.....	79
<RFn>	Restore Frame.....	80
<RM>	Row Mode.....	81
<SBn>	Set Backlight.....	82
<SD>	Screen Defaults.....	83
<SFm,n>	Save Frame.....	84
<SO n>	Screen Option.....	86
<SSn>	Screens to Scroll.....	87
<ST>	Steady.....	88
<SVn>	Show Variable.....	89
<SW>	Smart Wrap.....	90
<TO n>	Time Out.....	91
<TW>	Text Wrap.....	92
	UnderLine.....	93
<US>	Upload Screen.....	94
<VBm,n>	Vertical Bargraph.....	95
<VFn>	Visible Frame.....	96
<WMn>	Write Mode.....	97
<WSn>	Write Soft character.....	98
<WTstring>	Write Text.....	99

Appendix.....	100
Single Variable Screen.....	100
Dual Variable Screen.....	100
Four Variable Screen.....	100
Single Variable with Bargraph.....	101
Example Custom Screen.....	101

Introduction

This guide describes the BEKA Mode Protocol for the BA488CF, BA484DF, BA688CF and BA684DF Fieldbus Displays. This information is only required when customised screens are to be generated by programming the host; it is not required by the end user. The target audience for this guide are software application programmers.

- For hardware installation information, please refer to the separate instruction manuals available for each model.
- For an overview of how to use these displays on a fieldbus system, please refer to the appropriate “Fieldbus Interface Guide”.

The BEKA protocol is very straightforward, being loosely based on the principals of HTML. Simple text messages can be displayed by using only a handful of commands. However, with a bit more perseverance, some quite advanced displays can be created.

Model Variants

Each model may be supplied in one of two variants, depending on the desired protocol and function blocks:

Variant	Protocol	Device Revision	Function Blocks	Transducer Block*	Max Length of Command string	Max Length of Graphic data
-P	Profibus PA		A0 x 8	GRAPHIC	118 bytes	118 bytes
-F	Foundation Fieldbus	1	MAO	GRAPHIC	118 bytes	118 bytes
		2	IS x 2	DISP8	32 bytes	64 bytes

* The Transducer Block noted in the above table is the one that contains the command parameters that are used to communicate using the BEKA protocol. Other transducer blocks are present but are not used when programming the display.

There are two “Device Revisions” supported by the Foundation Fieldbus variant, as some hosts do not support the MAO function block. The display is normally dispatched from the factory pre-configured to work with the customer’s host, but this may be changed locally by means of the “Restore Defaults” menu item. (See the relevant instruction guide for full details). Please note that the corresponding CFF and DD files must be obtained from the Foundation Fieldbus website (www.fieldbus.org) or directly from us (www.beka.co.uk).

Note that the maximum length of the “Command string” and “Graphic data” have been reduced in the Device Revision 2 variant in order to make the overall system more responsive. These maximum lengths can not be exceeded, and the issued commands will fail.

There are two Fieldbus Interface Guides available, one for the Profibus PA protocol (covering the –P variant), and one for the Foundation Fieldbus protocol (covering the –F variant). The appropriate guide should be read in conjunction with this manual in order to understand the underlying parameters and data structures.

What's in this Programming Guide

- A description of the instrument 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 each Fieldbus Interface Guide

- An overview of the instrument
- A description of the Resource, Function and Transducer blocks that are available for that fieldbus protocol
- A description of the various data types that are used on that fieldbus protocol
- Instructions on how to use the instrument in its standard non-programmed modes

What's in the Instruction Manuals

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

Other sources of information

Our website at [*www.beka.co.uk*](http://www.beka.co.uk) has several files available to download:

- All of the examples in the appendices
- A Demonstration program of our Serial Text Display that can be used to develop custom screens

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.

Existing customers - Please note:

This product is based on our Serial Text display (BA488C / BA484D) and we endeavoured to keep as many facilities and commands the same. However, some commands have had to be removed, added and changed in order to fit in with fieldbus conventions – Please check that any existing code ported from previous applications uses valid instructions!

The updated firmware has added some facilities that were previously unavailable. The new products are backwardly compatible with the older versions, but to simplify commissioning and maintenance it may be preferable to upgrade older devices to the latest version. Please contact our sales department for advice on how this may be accomplished.

Please do not hesitate to contact us if you need assistance.

Instrument Features

A detailed overview of the instrument is given in the instruction manual for each product. This should be read before implementing any system using this instrument. 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 variables that exist on the fieldbus. Four pre-programmed screen layouts are available to display one, two or four variables simultaneously (See the instruction manual and Fieldbus Guide for more information).

For applications that require a customised display, the unit can be programmed by following the instructions in this guide. In order to take full advantage of the fieldbus hierarchy it is possible to map a number of analogue values to a corresponding set of formatted text strings such that they are automatically updated without any further intervention. Therefore, as far as the fieldbus system is concerned, this display behaves as any other conventional device would.

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.

Switch Outputs

As an optional accessory (available only at the time of ordering), there can be six switch outputs available which are totally isolated and can be energised or de-energised independently of each other. They can be driven either by direct commands from the fieldbus, or alarm set-point values can be assigned such that they operate automatically.

A few words about Modes...

It is worth reviewing the different modes that are referred to in this manual - these can become confusing if taken into the wrong context!

Row and Pixel Modes	<p>Refers to the way text and graphics are positioned on the screen</p> <p>The simplest and quickest mode is Row Mode – think of it as being able to position objects on a page with ruled lines. In this mode the screen is split up into eight horizontal rows each eight pixels high. Text is then aligned with these rows</p> <p>Pixel Mode allows objects to be placed anywhere – but the drawback is that it takes a bit longer for the display to be updated</p> <p>See the <RM> Row Mode and <PM> Pixel Mode commands for further information</p>
Write Modes	<p>Refers to the way text and graphics are written on the screen</p> <p>Mode 0 is normal : objects appear as a black image on a clear background Mode 3 is inverse : objects appear as a clear image on a black background Modes 1 and 2 are more complex and are used for special effects</p> <p>See the Write Mode section and also the <WM> command</p>
Background Modes	<p>Refers to the image that appears when the screen is flashed A text or graphic object can be flashed against a clear background, a black background or an inverse of that image</p> <p>See the <BM> Background Mode, <FL> Flashing and <EF> Enable Flashing commands</p>
Text Display Mode	<p>These are not strictly modes, but reference is made to the unit being in “Text Display Mode” as opposed to the “Indicator Mode”. The instructions in this guide are only used when the unit is in the “Text Display Mode” and a customised screen display is desired.</p> <p>No programming is required to use the “Indicator Mode” – please see the Instruction Manual and Fieldbus Guide for more information.</p>
Indicator Mode	<p>The operating mode is set using the configuration menu, or via the fieldbus.</p> <p>See the <SO> Screen Option command</p>

The “Command Reference” section (Page 18) shows which modes are applicable to each command.

Display Features

Some powerful features are built into the display that allow relatively complex visual effects to be generated with only a few simple commands. The command reference section of this programming guide has many examples of what can be achieved with a little creativity and lateral thought.

One of the most important concepts to understand is the mechanism of writing to the display.

The display has a foreground and a background. Objects are written to the foreground by sending commands to the instrument. The background is updated automatically, although commands are available to control what is actually written there. These choices are described as the “Background Modes”

When an object needs to be written to the foreground there are a number of choices available that affect the appearance of that object. These choices will also effect what is written to the background, so these choices are described as the “Write Modes”

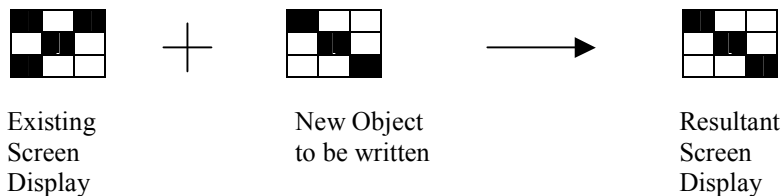
Write Modes

A new object can be added to the screen in four ways, each being associated with a particular Write Mode. However, the write mode is ignored in two cases where it is not considered appropriate, namely restored frames and bargraphs. In these cases, changing the appearance of such items may render them meaningless.

The four modes are:

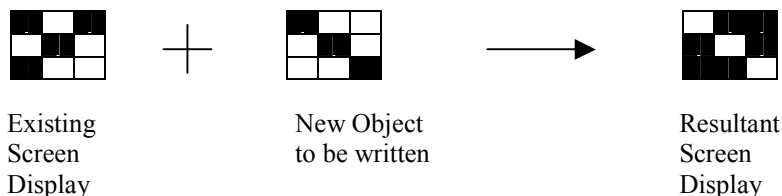
Write Mode 0 is the ‘normal’ method of updating the screen. The object is written to the screen where it over-writes the current screen contents i.e. if a pixel is set on the new object being written, then the corresponding pixel is set on the screen. If a pixel is not set on the new object, it is cleared on the screen

For example:



Write Mode 3 is almost the same as Mode 0, except that the resulting image is the inverse of the new object. The object is written to the screen where it over-writes the current screen contents i.e. if a pixel is set on the new object being written, then the corresponding pixel is cleared on the screen. If a pixel is not set on the new object, it is set on the screen

For example:

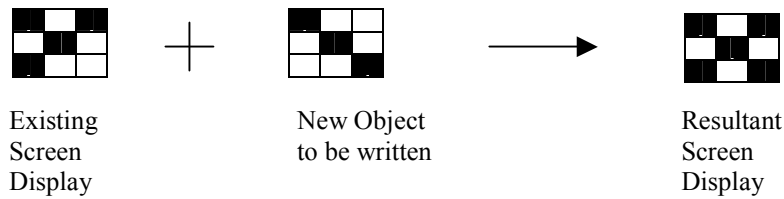


Write Mode 1 is slightly more complex in that the new object is ‘ORed’ with the existing screen contents i.e. if a pixel is set on the new object being written OR the corresponding pixel is set on the existing screen, then the pixel is set on the screen. The pixel is only ever cleared if both the new object and existing screen are clear.

This can be summarised in a table as follows:

Existing screen display	New Object to be written	Resultant screen display
not set	not set	not set
not set	set	set
set	not set	set
set	set	set

For example:

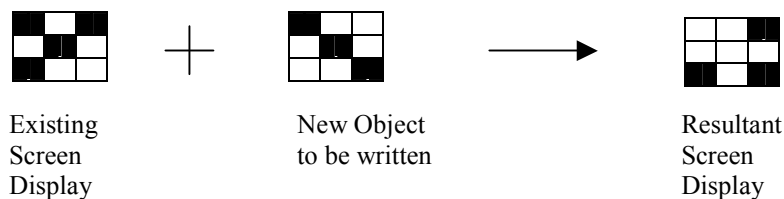


Write Mode 2 is the most complex in that the new object is ‘XORed’ with the existing screen contents i.e. if a pixel is set on the new object being written OR the corresponding pixel is set on the existing screen, then the pixel is set on the screen BUT if *both* are set then the pixel is cleared. The pixel is also cleared if both the new object and existing screen are clear.

This can be summarised in a table as follows:

Existing screen display	New Object to be written	Resultant screen display
not set	not set	not set
not set	set	set
set	not set	set
set	set	not set

For example:



Background Modes

The background is only ever visible when the screen is set to flash; the foreground image alternates with the background image every second i.e. If the background is clear, then some text on the foreground will disappear and re-appear every second. Alternatively, the background can be made all black. This gives a totally different visual effect which can be more noticeable. However by modifying the background so that it is the inverse of the foreground will make a very eye-catching effect.

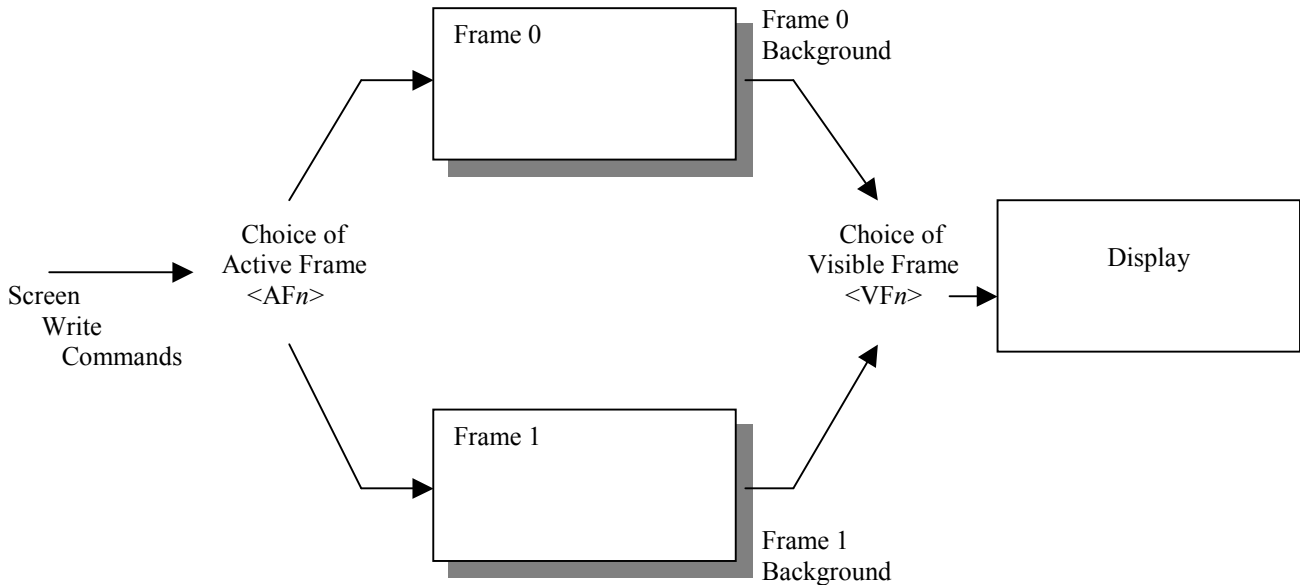
Rather than force the host to do all this work, the background is updated by the instrument automatically. The <BM> Background Mode command is used to control whether the background is clear, black or the inverse of what’s written. Once the <BM> command is issued, the background is updated automatically by each new screen object. Therefore it is possible to have all three flashing effects on the screen at once, simply by changing the Background Mode during the construction of the screen.

Frames

Active and Visible Frames

Another concept to grasp is the commands never actually write directly to the screen. Instead there are two “display buffers” that we refer to as ‘Frame 0’ and ‘Frame 1’. Only one of these frames is visible at any time, which is selected by the <VF*n*> Visible Frame command.

Similarly, only one of the frames is “Active” – that is, becomes the destination for all screen write commands. The destination is selected by the <AF*n*> Active Frame command.



Whilst this may seem complex at first, it is actually a very powerful method of displaying one message while building up another screen of data hidden from view. This hidden screen can then be made visible by issuing a single command. This is especially useful where the host cannot sustain a high data rate, or where very complex screens are being generated. As far as the operator is concerned the display updates almost immediately, even though it may have taken several seconds to construct.

For simple applications, frames can be disregarded. The unit powers up with both the Active Frame and Visible Frame set to 0. If the <AF*n*> and <VF*n*> commands are not issued, then the instrument behaves as if screen writes act directly on the display.

Important Note

For brevity this manual simply refers to commands writing to the screen. This has been done to keep the description of each command as simple as possible, so as to convey the main principals of that command. In reality, all commands write to the active frame.

Saved Frame Locations

It is possible to store the screen contents for later use by saving the Visible Frame to memory via the `<SFnm>` Save Frame command. (This command can actually save either frame, but for simplicity disregard this for now)

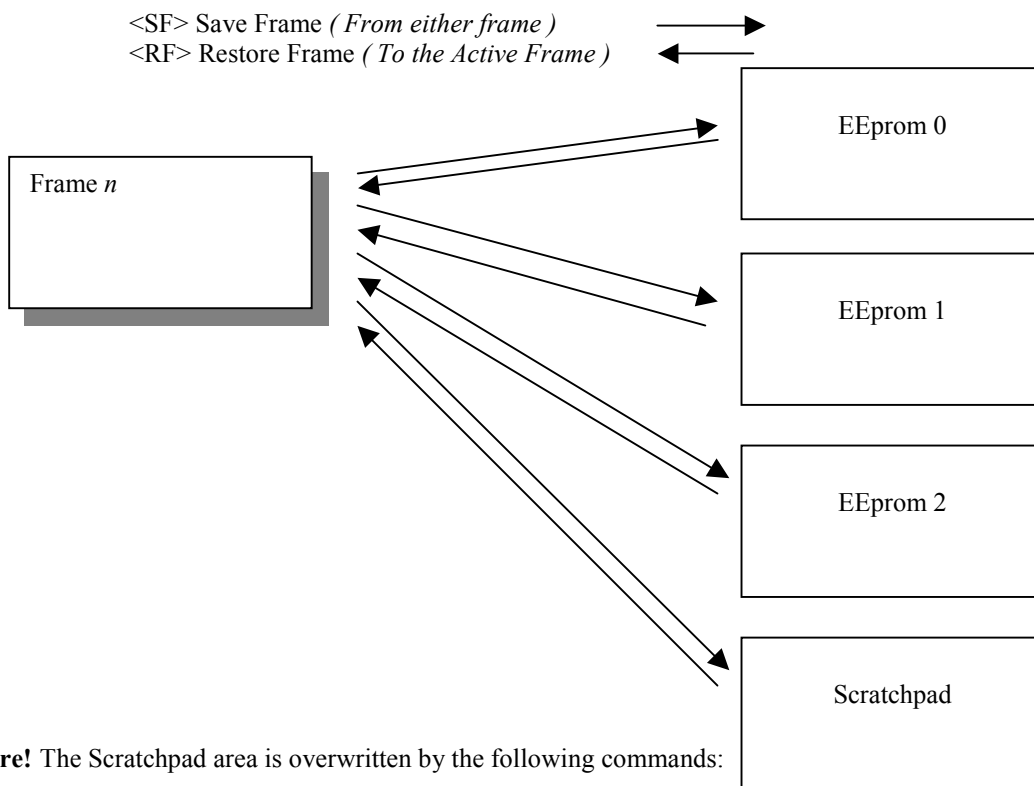
There are two types of memory available for saves:

- Non-volatile EEprom that is retained on power fail and
- A Scratchpad area in RAM that is lost when power is removed from the unit.

It is important to be aware that saves to EEprom take about 3 seconds, whilst saves to the scratchpad are immediate.

There are three independent EEprom locations that may be used as required.

Any frame can be used to store a power-on logo. This is a full screen graphic (with or without mapped variables) that appears when the unit is first turned on, or after the unit is re-booted. The `<BSn>` command selects which frame will be used, or whether the default logo should appear.



Beware! The Scratchpad area is overwritten by the following commands:

`<BD>`, `<DF>`, `<DG>`, `<LH>`, `<LV>`, `<SF>`, `<US>`
and the combination `<SO4><BS3>`

Restoring a frame after any of these commands will give unpredictable results.

The `<SSn>` command sets the number of saved frames to scroll through manually when the up/down keys are pressed. This can be set from 0 (where the arrow keys have no effect) to 3 (where all three screens are available). The screens are restored to the current active frame and this frame is then made visible.

This command allows a multi-page machine interface to be constructed within the display and viewed on demand by the operator.

More information can be found in the Command Reference section (Page 18).

The BEKA Protocol

The BEKA protocol is very loosely based on the principals behind HTML. Fundamentally, the intention is to make the scripts that generate a screen display “human readable”, in the same way that the source for a web page may be read.

The main features that achieve this are:

- It is a pure ASCII protocol except for graphics downloads
- Commands are always two characters, case insensitive, enclosed in angled brackets
- All commands are active until overridden by another command
- Some commands require parameters
 - Parameters follow the command directly
 - Multiple parameters are separated by commas
 - Any detected parameter error causes the command to be ignored, and an error result to be produced
 - A command and its parameters are enclosed within a single set of angled brackets
- No spaces are allowed in commands or parameter strings (except for written text strings)

Features have been added to maintain the data integrity between host and display. These allow the host to be confident that the display is actually showing valid data that has not become corrupted during transmission.

Command format

Commands are given to the display by writing to the *COMMAND_STRING* parameter in the appropriate Transducer Block. The name of the Transducer block is different for each supported protocol in the product range, as detailed in the Introduction on page 1 . In addition, the relevant “Fieldbus Interface Guide” contains further detailed information on the available parameters and their data structures.

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

where:

AB is the command.

[] indicates optional parameters separated by commas

example:

<CS>	Clear Screen
<CM4,90>	Cursor Move to Row 4 Column 90
<CI>	Command Implement

The commands are written to the *COMMAND_STRING* parameter in the appropriate Transducer Block. They may be written either singly, or several may be grouped together into one long string. The maximum length of a command string is 118 bytes (or 32 bytes in the FF Device Revision 2 variant).

N.B. Every command (or group of combined commands) has to be followed 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.

Result format

Result format is: 0,1,2,4,8 or 128

where:

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.

0x80 indicates that a previous command is still being processed.

The result is obtained by reading the **RESULT** parameter in the appropriate Transducer Block

Command and Result Timing

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 parameters in the appropriate Transducer Block 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** parameter.
2. Write the command string (including the terminating <CI> command) to the **COMMAND_STRING** parameter
3. Continually read the **RESULT_ID** parameter until it equals the value 'n' set in the **COMMAND_ID** parameter
4. Read the **RESULT** parameter: This is the result given by the command string

Key Press Information

- Key press information can be read using the Key_Status parameter in the appropriate transducer block. Key_Status = 0x00 means no key pressed. If key 1 has been pressed bit 0 (lsb) of Key_Status is set. If key 2 has been pressed bit 1 is set, and so on. Reading Key_Status clears the latched keypress information.

Individual key status is returned as the six least significant bits of this byte.

In binary notation the format of this returned byte is as follows:

msb							lsb
b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	0	0	0	0

b0 represents the status of Key 1 (0 =key open, 1=key closed)

b1 represents the status of Key 2

..

..

b5 represents the status of Key 6

b6 is always cleared (0)

b7 is always cleared (0)

Graphics Transfers

File Format

In all cases the file format used is a two colour (black and white) bitmap in standard Windows™ / OS2 format. These commonly have a .BMP extension on most PC applications.

Downloads

The protocol is extended as follows to cover the simple graphics download commands <DS> and <DG> and <DF n >

To avoid confusion, a download is defined as being from the host to the display

<DS> Download Screen command

A graphics file to be download must first be loaded into the **GRAPHIC_DATA** parameter in the appropriate Transducer Block. The size of this block is limited to only 118 bytes (or 64 bytes in the FF Device Revision 2 variant), which is the maximum allowed by the fieldbus protocols. Therefore, files must be split and loaded in segments of this number of bytes.

The <GB n > command is used to specify the segment that a subsequent write to the **GRAPHIC_DATA** parameter goes into. The value n can be in the range of 0 to 9 (or 16 in the FF Device Revision 2 variant). 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 .BMP file. Once the desired number of segments are filled with data, the <DS> command is then issued; The downloaded screen is then displayed

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

The image must be exactly 120x64 pixels and two colour (black and white) in standard Windows/OS2 format. An error result is produced if these requirements are not satisfied.

<DG> Download Graphic command

- Command <DG> follows exactly the same mechanism as the <DS> command above, but any size of image can be sent up to 120x64. Files in excess of this size will cause an error result to be produced.
- The display must be in Pixel Mode <PM> and the downloaded image is displayed at the current cursor position.
- The image dimensions are computed from the .BMP file sent.
- The image is drawn upwards from and to the right of the current cursor position. If any part of the image exceeds the display bounds the image is NOT displayed and an error result is produced.
- The .BMP format must still be two colour, standard Windows/OS2 format. An error result is produced if these requirements are not satisfied
- The downloaded image adopts the display attributes currently in force (Normal, OR, XOR, Inverse, Flashing, Steady)

The display must be in Pixel Mode <PM> and the downloaded image is displayed at the current cursor position.

The image dimensions are computed from the bitmap file that is sent; no parameters are necessary.

The image is drawn upwards and to the right of the current cursor position. If any part of the image exceeds the display bounds the image is NOT displayed and an error result is produced.

The downloaded image adopts the display attributes currently in force (Normal, OR, XOR, Inverse, Flashing, Steady), and the Write Mode setting is taken into account.

Please note:

The <DS> command is just a special case of the <DG> command but because of its fixed size is executed much more quickly.

Graphics can be uploaded to a hidden frame using the <AF> command to select the destination, and the <VF> command to make it visible when complete.

<DFn> Download Font command

The display has the capacity of storing four user defined characters for each font size. These “Soft Characters” can then be written onto the screen by using the <WSn> command. They may also be underlined and flashed using attributes, as any other character.

Before the <DFn> command is issued, the binary download of the soft character should be sent to the **GRAPHIC_DATA** parameter making use of the <GBn> mechanism if necessary. The required image size depends on the currently active font.

Font:	F1	Image Size (v x h):	8 x 6 pixels
	F2		16 x 10 pixels
	F3		24 x 15 pixels
	F4		32 x 19 pixels
	F5		48 x 29 pixels.

The image must be exactly as defined above otherwise an error result is produced.

Nothing is drawn to the screen during this command.

Uploads

The protocol is also extended to give the facility of obtaining a screen dump from the display. The main use for this is in the preparation of instruction manuals, but it could also be used in a debugging role.

<US> Upload Screen command

The 1086 byte data block, once saved to file, is a graphics image of the screen in 2-colour Windows/OS2 bitmap (.BMP) format.

Once the <US> command has been issued the data is available in the **GRAPHICS_DATA** parameter and accessed in 118 byte (64 byte for the -V variant) segments using the <GBn> command.

Standard Screens

There are nine 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 later in this Programming Guide.

The screen format is selected by either using the local menu (as described in the Instruction Manual) or by using the <SO> Screen Option BEKA protocol command. One of nine standard display formats can be selected as shown in the following table:

Screen Option 1	<pre> Inst1 Tag 21.835 Status:Good Units </pre>	Screen Option 2	<pre> Inst1 Tag Units 21.8350 Inst2 Tag Units 529.3300 </pre>
Screen Option 3	<pre> Inst1 Tag Units Inst3 Tag Units 21.835 -3.105 Inst2 Tag Units Inst4 Tag Units 529.33 -5600. </pre>	Screen Option 4	<pre> Inst1 Tag 21.835 Status:Good Units </pre>
Screen Option 5	<pre> Temperature °C [Bar] 21.46 Pressure Pa [Bar] 1.7500 </pre>	Screen Option 6	<pre> Temperature 25.25 °C [Bar] </pre>
Screen Option 7	<pre> Temp Pressure 25.22 1.750 °C Pa </pre>	Screen Option 8	<pre> Temp Press Flow 24.46 1.7500 48.9 °C Pa l/min </pre>
Screen Option 9	<pre> TEMP Pres Flow Fill 22.73 1.750 45.4 11.36 </pre>	Screen Option 0	Custom screens

Setting Screen Option to Zero <SO0> will allow custom screens to be displayed by using BEKA protocol commands.

Command Summary

There are 79 commands that can be arranged into 6 functional groups:

Screen Handling & Text	- used to control the screen in text mode
Mapped Fieldbus Variables	- link text and graphics to fieldbus values
Attributes	- affect the appearance of text and graphics
Line Graphics	- draw lines and boxes on the screen
Pixel Graphics	- draw graphical objects on the screen
System	- affect the operation of the text display

Screen Handling & Text

Command	Meaning
<CLn>	Clear Line
<CM _{y,x} >	Cursor Move
<CS>	Clear Screen
<CW>	Clear Window
<EL>	Erase Line
<FS>	Fill Screen
<FW>	Fill Window
<HC>	Home Cursor
<LN>	Line New
<SD>	Screen Defaults
<WS _n >	Write Soft character
<WTstring>	Write Text

Attributes

Command	Meaning
<BMn>	Background Mode
<CA>	Centre Align
<DWyt,yb,xl,xr>	Define Window
<EF>	Enable Flashing
<F1>	Font 1
<F2>	Font 2
<F3>	Font 3
<F4>	Font 4
<F5>	Font 5
<FL>	Flashing
<IF>	Inhibit Flashing
<LA>	Left Align
<LF>	Line Feed
<NA>	No Align
<NL>	No Linefeed
<NU>	No Underline
<RA>	Right Align
<ST>	Steady
<SW>	Smart Wrap
<TW>	Text Wrap
	UnderLine
<WMn>	Write Mode

Pixel Graphics

Command	Meaning
<DG>	Download Graphic
<DS>	Download Screen
<HSn,m,r,s,t,u,v>	Horizontal Scroll
<US>	Upload Screen

Line Graphics

Command	Meaning
<BDylength,xlength,lwidth>	Box Draw
<HBmn>	Horizontal Bargraph
<LHxlength, lwidth>	Line Horizontal
<LVylength, lwidth>	Line Vertical
<VBmn>	Vertical Bargraph

System

Command	Meaning
<AA n>	Alarm Activation
<AF n>	Active Frame
<AH mnp>	Alarm Hysteresis
<AL mnp>	Alarm Lower Limit
<AM mn>	Alarm Mapping
<AU mnp>	Alarm Upper Limit
<BS n>	Boot Screen
<CE>	Configuration Enable
<CI>	Command Implement
<CP>	Configuration Prohibit
<DF n>	Download Font
<FR>	Font Restore
<GB n>	Graphics Block
<KF>	Keep Fonts
<OD n>	Output De-energised
<OE n>	Output Energised
<PM>	Pixel Mode
<RF m>	Restore Frame
<RM>	Row Mode
<SB n>	Set Backlight
<SF mn>	Save Frame
<SO n>	Screen Option
<SS n>	Screens to Scroll
<TO n>	Time Out
<VF n>	Visible Frame

Mapped Fieldbus Variables

Command	Meaning
<DBn,m,p,q,r>	Define Bargraph
<DDn,m>	Define Decimal
<DLn,m,p>	Define Limits
<DTn,string>	Define Tag
<DUn,string>	Define Units
<DVn,m,p,q>	Define Variable
<EBn>	Erase Bargraph
<EVn>	Erase Variable
<NS>	New Screen
<SVn>	Show Variable

Command Reference

The following section lists each command in alphabetical order. Each page is formatted in the same way so that commands can be compared and reviewed easily.

The following page explains the format of each page:

<..>

Command

Group

Description	This is a brief description of the command
Parameters	The allowable range of values
Initial Value	The value at initialisation, if applicable
Modes	Some commands are only available in certain modes
Notes	Detailed comments
Uses	Describes where the command may be used
Example	<p>A simple example showing how to use the command Be aware that most examples assume a <SD> command has been issued to clear the screen first</p> <p>Also note that the <CI> commands have not been shown in any of the examples.</p>
Gotchas!	Common pitfalls to be aware of
See Also	Other related commands

Description	Specify that the alarm outputs can be controlled by bad data
Parameters	$n = 0$ - Alarm on good data ONLY $n = 1$ - Alarm on good and bad data
Initial Value	The default behaviour is to alarm on good data only i.e. $n = 0$
Modes	All Screen Modes
Notes	The design of the overall fieldbus system will dictate whether an alarm indication is required if the data is bad. This command gives the user the necessary flexibility
Uses	The <AA> command allows: <ul style="list-style-type: none">• The local alarm contacts to be conditioned as required
Gotchas!	If the alarm outputs are assigned set-point values then they can no longer be directly controlled by the host (via the OD and OE commands)
See Also	AA Alarm Activation AH Alarm Hysteresis AL Alarm Lower Limit AM Alarm Mapping AU Alarm Upper Limit OD Output Enable OE Output Disable

Description	Specify that all writes are directed to Frame <i>n</i>
Parameters	<i>n</i> = 0 or 1 - frame number
Initial Value	Frame 0 is the default at power up, or after a <SD> Screen Defaults command
Modes	Pixel and Row Modes only
Notes	All commands write to the Active Frame, not the Visible Frame. This gives the flexibility to make complex data screens appear relatively quickly, to provide an intuitive user interface Detailed information about the use of frames can be found in the Frames Section (Page 7).
Uses	The <AF> command allows: <ul style="list-style-type: none"> • Complex screens to be drawn and then displayed when they are complete • Rapid switching between two different information screens
Example	<div style="background-color: #e0ffe0; padding: 5px; text-align: center; margin-bottom: 10px;"> Original Data Screen </div> <p>Assume the display is showing some data, and the active frame and visible frame are both set to 0</p> <p><AF1> Set active frame to 1; LCD display screen unaltered</p> <p><CS> Clear the hidden frame; LCD display screen unaltered</p> <p><SW> Set smart wrap formatting on to cope with a long line of text</p> <p><WTThis text is written on a hidden frame and can displayed using a <VF1>> command></p> <p>Write out the text to the hidden frame; LCD display screen unaltered</p> <div style="background-color: #e0ffe0; padding: 5px; text-align: center; margin-bottom: 10px;"> Original Data Screen </div> <p>LCD display screen at this point</p> <p><VF1> Make frame 1 visible</p> <div style="background-color: #e0ffe0; padding: 5px; text-align: center;"> This text is written on a hidden frame and can be displayed using a <VF1> command </div>
Gotchas!	Make sure that the section on Frames (Page 7) is read and understood Frame <i>n</i> may or may not currently be visible. Use the <VF> command to achieve the desired result
See Also	VF Visible Frame RF Restore Frame SF Save Frame

Description This command allows local alarm to be set and adjusted without having to use the built in menus

Parameters *m* = 1 to 6 - the output alarm number
n = 0 or 1 - Hysteresis enable . 0 = disabled, 1 = enabled.
p = Any Valid Value - Hysteresis value i.e. the value the input should differ from the set-point in order to de-activate the alarm

Note: Valid values are 10 characters maximum, including the minus sign and decimal point
i.e. Between the range -999,999,999 to 9,999,999,999 (NB Commas only shown for clarity)

Initial Value All alarms are disabled.
The settings are defined using the local configuration menu

Modes All Screen Modes

Notes Alarm conditions are indicated by flashing variables and bargraphs as well as an open circuit on the specified output channel.

The user may choose to have alarms activated only by variables that have a “good” status.
See the <AA> command

Uses The <AH> command :

- **MUST NOT BE USED FOR SAFETY CRITICAL APPLICATIONS**
- adds a hysteresis band to an alarm set-point
- Facilitates local indication/annunciation of potential problems

Example

<AM3, 7>	Maps Alarm output 3 to <i>IN_7</i>
<AL3, 1, 199.9>	Alarm output 3 has a lower limit of 199.9 enabled i.e. it signals when the input falls below a value of 199.9
<AU3, 1, 263.5>	Alarm output 3 has an upper limit of 263.5 enabled i.e. it signals when the input rises below a value of 263.5
<AH3, 1, 10.0>	Alarm output 3 has hysteresis enabled, and is set to a value of 10.0 i.e. an alarm is signalled when the <i>IN_7</i> value falls below 199.9, and continues to do so until the value rises above 209.9 In addition, an alarm is signalled when the <i>IN_7</i> value rises above 263.5, and continues to do so until the value falls below 253.5
<AM3, 0>	Alarm output 3 is no longer mapped, and is disabled.
<AM0, 0>	All alarm mappings are deleted and all alarms are disabled

Gotchas! If particular parameters are disabled the corresponding values are ignored, but must still be entered in a correct format in the command or a parameter error will be detected and the whole command ignored.

Alarms can also be set up via the configuration menus; the latest alteration takes precedence.

See Also

AA	Alarm Activation
AL	Alarm Lower Limit
AM	Alarm Mapping
AU	Alarm Upper Limit
OD	Output Enable
OE	Output Disable

Description	This command allows local alarm to be set and adjusted without having to use the built in menus	
Parameters	<i>m</i> = 1 to 6	- the output alarm number
	<i>n</i> = 0 or 1	- Minimum Alarm enable 0 = disabled 1 = enabled.
	<i>p</i> = Any Valid Value	- Minimum value. The value below which the alarm is activated.
	Note: Valid values are 10 characters maximum, including the minus sign and decimal point i.e. Between the range -999,999,999 to 9,999,999,999 (NB Commas only shown for clarity)	
Initial Value	All alarms are disabled. The settings are defined using the local configuration menu	
Modes	All Screen Modes	
Notes	Alarm conditions are indicated by flashing variables and bargraphs as well as an open circuit on the specified output channel. The user may choose to have alarms activated only by variables that have a “good” status. See the <AA> command	
Uses	The <AL> command : <ul style="list-style-type: none"> • MUST NOT BE USED FOR SAFETY CRITICAL APPLICATIONS • Provides local indication/annunciation of potential problems • Multiple alarms may be set on the same input to provide low-low and high-high alarms • Inherits the limitations of the fieldbus variables which are transferred across the fieldbus as 4-byte (32bit) IEEE floats. The limited resolution means values may not be exact. 	
Example	<AM3 , 7>	Maps Alarm output 3 to <i>IN_7</i>
	<AL3 , 1 , 199 . 9>	Alarm output 3 has a lower limit of 199.9 enabled i.e. it signals when the input falls below a value of 199.9
	<AU3 , 1 , 263 . 5>	Alarm output 3 has an upper limit of 263.5 enabled i.e. it signals when the input rises below a value of 263.5
	<AH3 , 1 , 10 . 0>	Alarm output 3 has hysteresis enabled, and is set to a value of 10.0 i.e. an alarm is signalled when the <i>IN_7</i> value falls below 199.9, and continues to do so until the value rises above 209.9 In addition, an alarm is signalled when the <i>IN_7</i> value rises above 263.5, and continues to do so until the value falls below 253.5
	<AM3 , 0>	Alarm output 3 is no longer mapped, and is disabled.
	<AM0 , 0>	All alarm mappings are deleted and all alarms are disabled
Gotchas!	If particular parameters are disabled the corresponding values are ignored, but must still be entered in a correct format in the command or a parameter error will be detected and the whole command ignored. A user may choose to have alarms only activated by variables that have a “good” status. This is the default condition, but it can be over-ridden to enable alarms irrespective of the variables status. Alarms can also be set up via the configuration menus; the latest alteration takes precedence.	
See Also	AA	Alarm Activation
	AH	Alarm Hysteresis
	AM	Alarm Mapping
	AU	Alarm Upper Limit
	OD	Output Enable
	OE	Output Disable

Description	This command allows local alarm to be set and adjusted without having to use the built in menus	
Parameters	<i>m</i> = 0 to 8	- the input number <i>IN_1</i> to <i>IN_8</i>
	<i>n</i> = 0 to 6	- the output alarm number
Initial Value	All alarms are disabled. The settings are defined using the local configuration menu	
Modes	All Screen Modes	
Notes	Alarm conditions are indicated by flashing variables and bargraphs as well as an open circuit on the specified output channel.	
	The user may choose to have alarms activated only by variables that have a “good” status. See the <AA> command.	
	Issuing the command with <i>m</i> = 0 will delete ALL mappings and disable all alarms. Issuing the command with <i>n</i> = 0 will delete the specified mapping <i>m</i> and disable the alarm.	
Uses	The <AM> command : <ul style="list-style-type: none"> • MUST NOT BE USED FOR SAFETY CRITICAL APPLICATIONS • Provides local indication/annunciation of potential problems • Multiple alarms may be set on the same input to provide low-low and high-high alarms 	
Example	<AM3 , 7>	Maps Alarm output 3 to <i>IN_7</i>
	<AL3 , 1 , 199 . 9>	Alarm output 3 has a lower limit of 199.9 enabled i.e. it signals when the input falls below a value of 199.9
	<AU3 , 1 , 263 . 5>	Alarm output 3 has an upper limit of 263.5 enabled i.e. it signals when the input rises below a value of 263.5
	<AH3 , 1 , 10 . 0>	Alarm output 3 has hysteresis enabled, and is set to a value of 10.0 i.e. an alarm is signalled when the <i>IN_7</i> value falls below 199.9, and continues to do so until the value rises above 209.9 In addition, an alarm is signalled when the <i>IN_7</i> value rises above 263.5, and continues to do so until the value falls below 253.5
	<AM3 , 0>	Alarm output 3 is no longer mapped, and is disabled.
	<AM0 , 0>	All alarm mappings are deleted and all alarms are disabled
Gotchas!	If particular parameters are disabled the corresponding values are ignored, but must still be entered in a correct format in the command or a parameter error will be detected and the whole command ignored.	
	A user may choose to have alarms only activated by variables that have a “good” status. This is the default condition, but it can be over-ridden to enable alarms irrespective of the variables status.	
	Alarms can also be set up via the configuration menus; the latest alteration takes precedence.	
See Also	AA	Alarm Activation
	AH	Alarm Hysteresis
	AL	Alarm Lower Limit
	AU	Alarm Upper Limit
	OD	Output Enable
	OE	Output Disable

Description	This command allows local alarm to be set and adjusted without having to use the built in menus	
Parameters	<i>m</i> = 1 to 6	- the output alarm number
	<i>n</i> = 0 or 1	- Upper Alarm enable 0 = disabled 1 = enabled.
	<i>p</i> = Any Valid Value	- Maximum value. The value above which the alarm is activated.
	Note: Valid values are 10 characters maximum, including the minus sign and decimal point i.e. Between the range -999,999,999 to 9,999,999,999 (NB Commas only shown for clarity)	
Initial Value	All alarms are disabled. The settings are defined using the local configuration menu	
Modes	All Screen Modes	
Notes	Alarm conditions are indicated by flashing variables and bargraphs as well as an open circuit on the specified output channel. The user may choose to have alarms activated only by variables that have a “good” status. See the <AA> command	
Uses	The <AU> command : <ul style="list-style-type: none"> • MUST NOT BE USED FOR SAFETY CRITICAL APPLICATIONS • Provides local indication/annunciation of potential problems • Multiple alarms may be set on the same input to provide low-low and high-high alarms • Inherits the limitations of the fieldbus variables which are transferred across the fieldbus as 4-byte (32bit) IEEE floats. The limited resolution means values may not be exact. 	
Example	<AM3 , 7>	Maps Alarm output 3 to <i>IN_7</i>
	<AL3 , 1 , 199 . 9>	Alarm output 3 has a lower limit of 199.9 enabled i.e. it signals when the input falls below a value of 199.9
	<AU3 , 1 , 263 . 5>	Alarm output 3 has an upper limit of 263.5 enabled i.e. it signals when the input rises below a value of 263.5
	<AH3 , 1 , 10 . 0>	Alarm output 3 has hysteresis enabled, and is set to a value of 10.0 i.e. an alarm is signalled when the <i>IN_7</i> value falls below 199.9, and continues to do so until the value rises above 209.9 In addition, an alarm is signalled when the <i>IN_7</i> value rises above 263.5, and continues to do so until the value falls below 253.5
	<AM3 , 0>	Alarm output 3 is no longer mapped, and is disabled.
	<AM0 , 0>	All alarm mappings are deleted and all alarms are disabled
Gotchas!	If particular parameters are disabled the corresponding values are ignored, but must still be entered in a correct format in the command or a parameter error will be detected and the whole command ignored. A user may choose to have alarms only activated by variables that have a “good” status. This is the default condition, but it can be over-ridden to enable alarms irrespective of the variables status. Alarms can also be set up via the configuration menus; the latest alteration takes precedence.	
See Also	AA	Alarm Activation
	AH	Alarm Hysteresis
	AL	Alarm Lower Limit
	AM	Alarm Mapping
	OD	Output Enable
	OE	Output Disable

Description Draws a box y pixels high, x pixels wide with a line thickness of l

Parameters $y = 1$ to 64 - height
 $x = 1$ to 120 - width
 $l = 1$ to 32 - line thickness

Modes Pixel mode only

Notes The box is drawn from the current cursor position upwards and to the right.

The cursor position is unchanged after the command

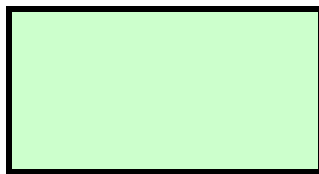
The parameters may be any value that will keep the box being drawn on-screen. If any part of the defined box is off-screen, then the box is not drawn and an error result is produced.

Uses The <BD> command allows:

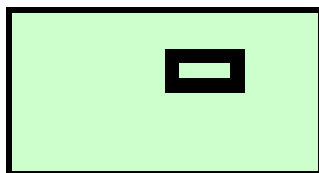
- information to be segmented
- borders to be drawn
- line images to be constructed

Example

<CS>	Clear Screen
<PM>	Set Pixel Mode
<CM63, 0>	Move the cursor to the bottom left had corner of the display LCD
<BD64, 120, 1>	A box ,a single pixel thick, is drawn round the edge of the display LCD



<CM31, 60>	Move the cursor to the centre of the LCD display
<BD16, 30, 5>	A box 16 by 30 pixels, 5 pixels thick, is drawn with its bottom left hand corner in the centre of the LCD display



Gotchas! The entire box must fit on the screen, otherwise nothing will be drawn and an error result is produced

See Also **LH** Line Horizontal
LV Line Vertical

<BM*n*>

Background Mode

Attributes

Description Defines the appearance of the 'flashing' attribute

Parameters $n = 0$ to 2 - flashing style

Initial Value 0

Modes Pixel and Row Modes only

Notes The flash background is defined by the value n
 $n = 0$ sets all pixels off
 $n = 1$ sets all pixels on
 $n = 2$ sets the pixels to the inverse of the character or graphic being written

To use this command the flashing attribute <FL> must be set for each object, and then the enable flash <EF> command sent

Uses The <BM> command allows:

- attention grabbing messages
- special effects

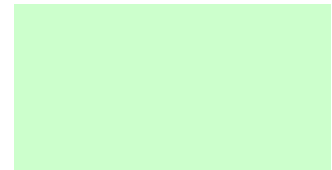
Example

<BM0> Background to flashing characters is all pixels off

<WTFFLASH> Write the text FLASH to the screen

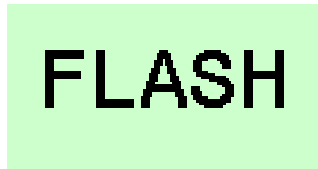


↔ Alternates each second with



<BM1> Background to flashing characters is all pixels on

<WTFFLASH> Write the text FLASH to the screen

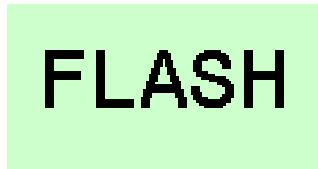


↔ Alternates each second with



<BM2> Background to flashing characters is all pixels are the inverse of the image being flashed

<WTFFLASH> Write the text FLASH to the screen



↔ Alternates each second with



Gotchas!

To use these effects successfully, the background mode must be set **before** the screen is written to.

See Also

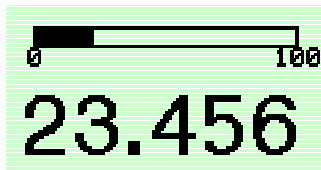
EF Enable Flash
FL Flashing

Description	This command specifies which stored frame is shown when the display is powered up
Parameters	$n = 0$ to 3 - frame number (0 – 2) or BEKA logo (3)
Initial Value	The default action is to show the BEKA logo i.e. <BS3>
Modes	This command is only applicable when in Text Display Mode
Notes	The BEKA logo is scrolled if no commands are received. There is no way to achieve this scrolling on any other screen.

To show a logo until a button is pressed, then see the SS n command.

Uses	The <BS> command : <ul style="list-style-type: none"> • Automatically restores a custom screen on power-on • Provides a simple way to show an OEM Logo
-------------	--



Example	<SD>	Start from a known state
	<CM1, 10>	Move to Row 1
	<DL1, 0.0, 100.0>	Define the bargraph limits for IN_1 as 0.0 to 100.0
	<DB1, 100, 0, 0, 0>	Define a horizontal bargraph 100 pixels long using IN_1, and the limits specified by the DL command
	<CM2, 8>	Move to Row 2 to write in the scale
	<WT0>	Write “0”
	<CM2, 101>	Now move to the other end
	<WT100>	Write “100”
	<F4>	Specify a large font (32 x 19 pixels)
	<CM7, 5>	Move to Row 7
	<DV1, 6, 3, 0>	Define a variable using IN_1, 6 character field, 3 chars after the decimal point, and written left aligned
	<SF0, 2>	Save the screen definition to EEprom store 2
	<BS2>	Set saved frame 2 to be shown on power-up.



The fieldbus variable linked to IN_1 in this example has a “good” status and a value of “23.456”

Gotchas!	This command is only applicable when in Text Display Mode The default BEKA logo is briefly shown on power up in Indicator Mode
-----------------	---

See Also	SF Save Frame SO Screen Option
-----------------	---

Description	Set the attribute so that written text is aligned horizontally within the screen or defined window	
Parameters	None	
Initial Value	Not aligned; Text appears at the current cursor position	
Modes	All Screen Modes	
Notes	<p>This command only affects text written after the attribute has been set.</p> <p>In Pixel Mode <PM> the centring is always based on the full screen. In row mode the text is centred in the currently defined window, which by default is the full screen.</p> <p>The attribute is cancelled by the <NA> command or any of the other text alignment commands <LA>, <RA>, <SW> & <TW></p>	
Uses	<p>The <CA> command allows:</p> <ul style="list-style-type: none"> • Text to be automatically aligned without the need for cursor move commands • Tidy screen presentation 	
Example	<pre><PM> <CM40,0> <CA> <WTThis is centred></pre>	<p>Set Pixel Mode</p> <p>Set up the vertical position: move the cursor to 40 pixels down from the top of the screen, on the left hand side of the screen</p> <p>Align all following text centrally</p> <p>Write the message</p>  <p>The horizontal position is calculated from the length and size of the text</p> <pre><RM> <DW0,7,60,119> <CM1,0> <CA> <WTThis> <LN> <WTis> <LN> <WTcentred></pre> <p>Set Row Mode</p> <p>Define a window as the right hand half of the screen</p> <p>Move the cursor to row 1 (one row from the top)</p> <p>Align all following text centrally</p> <p>Write the text</p> <p>Move the cursor down one line</p> <p>Write the text</p> <p>Move the cursor down one line</p> <p>Write the text</p> 
See Also	<p>LA Left Align</p> <p>NA No Align</p> <p>RA Right Align</p> <p>SW Smart Wrap</p> <p>TW Text Wrap</p>	

Description	Control access to the configuration menus	
Parameters	None	
Initial Value	This is the default	
Modes	All Screen Modes	
Notes	<p>The <CE> and <CP> commands control access to the main and quick access menus used for unit configuration.</p> <p>The <CP> command will prevent user access to the configuration menus via the dual P-E and P-UpArrow key presses.</p> <p>The <CE> command will re-enable user access to the menus.</p> <p>The Quick Access menu can also be disabled within the ‘Display’ section of the main menu. When it is disabled in this way the <CE> command has no effect on the access to this menu.</p> <p>The commands have no effect on the LCD display screen.</p> <p>The instrument provides a different response to commands from the host when it is in the programming menus.</p>	
Uses	<p>The <CE> command allows:</p> <ul style="list-style-type: none"> • changes to instrument configuration to be made after a <CP> Configuration Prohibit command 	
Example	<p><CP></p> <p>Any</p> <p><CE></p>	<p>Lock out the menus</p> <p>A set of commands or operator instructions that should not be interrupted by adjustments to the display configuration be made</p> <p>Re-enable access to the menus for maintenance</p> <p style="text-align: center;">There is no effect on the display LCD screen when these commands are used</p>
See Also	CP	Configuration Prohibit

<CI>

Command Implement

System

Description This is the command terminator

Parameters None

Modes All Screen Modes

Notes The <CI> terminator is the signal to action the preceding command string.

It has to be appended to the command string so that the display can work out how many of the commands in the buffer to action.

Uses The <CI> command has to be used in all communications with the display

Example The command string:

```
<CS><FS><CS><FS><CI>
```

will clear the display LCD, then turn all the pixels on, clear the display again, and turn all the pixels on again. The command string <CS><FS><CS><FS> is only actioned when the <CI> command is received.

See Also

Description Clears a complete line on the screen

Parameters $n = 0$ to 7 - line number

Modes Row Mode Only

Notes There are 8 lines on the screen numbered 0 to 7, 0 being the top line.

The command clears a number of lines upwards from the stated line, depending on the current font:
 For font 1 <F1> only one line is cleared
 For font 2 <F2> two lines are cleared, and so on.

This command is window aware. If a window is in use, the line numbers are relative to the window. That is, line 0 is the top line of the window.

Cursor position is unchanged by this command

Uses The <CLn> command allows:

- Message/status information to be cleared
- Ensures new messages can be written without leaving part of an old message in place
- Clearing of lines in a window

Example

Assume the initial display is:

```
line0
line1
line2
line3
line4
line5
line6
line7
```

<F1>

Make sure we know the font in use

<CL5>

Clear line 5

```
line0
line1
line2
line3
line4

line6
line7
```

Gotchas!

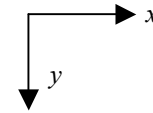
The current font size must be taken into account before issuing this command. In the example above, if Font 2 were in use then line 5 and line 4 would be blanked.

See Also

DW Define Window
EL Erase Line
Fn Font n

Description Moves the cursor on the screen

Parameters Row Mode: $y = 0$ to 7
 $x = 0$ to 119 Pixel Mode: $y = 0$ to 63
 $x = 0$ to 119



In both modes, co-ordinate 0,0 is at the top left of the screen

Modes Pixel and Row Modes only

Notes This command moves the cursor to the position defined by the parameters y and x . The cursor is never visible; it can be considered an insertion point on the screen for text and graphics.

Text and graphics are always drawn upwards and to the right of the current cursor position.

When text is written, the cursor is placed at the end of the inserted text. When graphics images are written to the screen, the cursor position is unaltered.

In Row mode this command is window aware. If a window is in use the parameters y and x are relative to the current window.

Uses The <CM> command allows:

- Positioning of text and graphics objects

Example

<CS>	Clear Screen
<RM>	Set Row Mode
<F1>	Small 8x6 pixel font
<CM1, 20>	Move the cursor to the second row, 20 pixels from the left edge of screen
<WTFlow Rate:>	Write a heading
<F3>	
<CM6, 0>	Move the cursor to the next to bottom row, on left edge of screen
<WT20.543>	Write in process value
<F1>	Small font again
<CM5, 90>	Move cursor to row 5, 30 pixels from right had side of screen
<WT1/s>	Write in units

```

Flow rate:
20.543 1/s

```

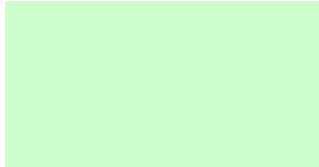
Gotchas! The horizontal resolution is always 1 pixel.

See Also DW Define Window

Description	Control access to the configuration menus	
Parameters	None	
Initial Value	<CE> Configuration Enable is active on power up	
Modes	Pixel and Row Modes only	
Notes	<p>The <CE> and <CP> commands control access to the main and quick access menus used for unit configuration.</p> <p>The <CP> command will prevent user access to the configuration menus via the dual P-E and P-UpArrow key presses.</p> <p>The <CE> command will re-enable user access to the menus.</p> <p>The Quick Access menu can also be disabled within the 'Display' section of the main menu. When it is disabled in this way the <CE> command has no effect on the access to this menu.</p> <p>The commands have no effect on the display LCD screen.</p> <p>The instrument signals a different response to commands from the host when it is in the programming menus.</p>	
Uses	<p>The <CP> command allows:</p> <ul style="list-style-type: none"> • The prevention of unauthorised changes to instrument configuration • The prevention of operators missing messages from the host, due to the instrument being in programming mode 	
Example	<p><CP> Lock out the menus</p> <p>< Anything > A set of commands or operator instructions that should not be interrupted by adjustments to the display configuration be made</p> <p><CE> Re-enable access to the menus for maintenance</p> <div style="background-color: #90EE90; width: 200px; height: 60px; margin-top: 10px;"></div> <p style="text-align: center;">There is no effect on the display LCD screen when these commands are used</p>	
See Also	CE	Configuration Enable

Description	Turn all pixels off, creating a blank screen
Parameters	None
Modes	Pixel and Row Modes only
Notes	This command also: <ul style="list-style-type: none">• Removes any windows that may be defined (equivalent to issuing a <DW0,7,0,119> command)• Homes the cursor (equivalent to issuing a <HC> command)
Uses	The <CS> command provides: <ul style="list-style-type: none">• A known starting point before drawing a new screen

Example `<CS>` Clear Screen



Gotchas! Defined variables and bargraphs are NOT cleared by this command – or rather they are, but are updated again automatically. Use the RV and RB commands to remove the mappings.

If windows are being used, they must be defined after this command

See Also

CW	Clear Window
DW	Define Window
FS	Fill Screen
HC	Home Cursor
NS	New Screen
RB	Remove Bargraph
RV	Remove Variable

Description Turn all pixels off within a defined window

Parameters None

Modes Row Mode only

Notes This command also homes the cursor in the defined window area, equivalent to issuing a <HC> command.

Apart from its main use in just clearing the contents of a window, it can also be used to create frames to contain text and graphics. Using this technique is much faster than using a <BD> Box Draw command in Pixel Mode.

Uses The <CW> command allows:

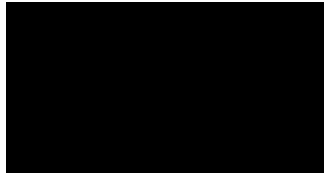
- A known starting point before updating a window
- Creation of a simple border

Example**<RM>**

Set Row Mode

<FS>

Turn all screen pixels on

**<DW2, 5, 20, 100>**

Define a window two rows from top and bottom, 20 pixels in from both sides

<CW>

Set all the pixels in the window area to off

**See Also**

BD Box Draw
CS Clear Screen
FW Fill Window
HC Home Cursor

Description	This command specifies how a fieldbus variable is written as a bargraph to the screen	
Parameters	<i>m</i> = 1 to 8	- the input number <i>IN_1</i> to <i>IN_8</i>
	<i>n</i> = 5 to 120	- the total length of the bargraph in pixels
	<i>p</i> = 0 to 8	- Lower limit of bargraph 0 = use static limit defined by the <DL> command 1 - 8 = Use the fieldbus variable connected to the specified input <i>IN_1</i> to <i>IN_8</i>
	<i>q</i> = 0 to 8	- Upper limit of the bargraph 0 = use static limit defined by the <DL> command 1 - 8 = Use the fieldbus variable connected to the specified input <i>IN_1</i> to <i>IN_8</i>
	<i>r</i> = 0 or 1	- Alignment: 0 = Horizontal bar, 1 = Vertical bar.

Modes Row Mode only

Notes The bargraph variable defined by this command is drawn at the current cursor position. The variable is automatically updated as new fieldbus data is received.

An empty bargraph with a dotted outline indicates that data with “bad” status is being displayed, or that the input value is outside the defined limits.

A flashing bargraph indicates an alarm that is set on this variable has been activated.

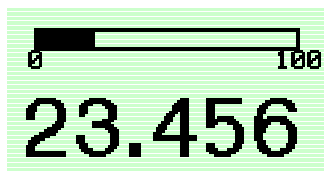
Only one instance of a particular input can be on the displayed at a time. If the command is issued when there is already a bargraph of the specified input on-screen, the variable is moved from the old to the new position.

The command will fail and a parameter error result is produced if any part of the defined bargraph is off-screen

Uses The <DB> command allows:

- A fieldbus variable to be shown on screen as a continuously updated bargraph
- Up to 8 bargraphs to be displayed on a single screen
- Bargraph definitions to be saved together with on-screen text and graphics using the <SF> command

Example	<SD><CM1,10>	Start from a known screen and move to Row 1
	<DL1,0.0,100.0>	Define the bargraph limits for <i>IN_1</i> as 0.0 and 100.0
	<DB1,100,0,0,0>	Define a horizontal bargraph 100 pixels long using <i>IN_1</i> , and the limits specified by the DL command
	<CM2,8><WT0>	Move to row 2 and write in the scale “0”
	<CM2,101><WT100>	Move cursor to the right and write “100”
	<F4>	Specify a large font (32x19 pixels)
	<CM7,5>	Move to Row 7
	<DV1,6,3,0>	Define a variable using <i>IN_1</i> , 6 character field, 3 chars after the decimal point, and written left aligned



The fieldbus variable linked to *IN_1* in this example has a “good” status and a value of “23.456”

Gotchas! Only one instance of a particular input can be on the displayed at a time, although a bargraph can be displayed along with its numeric value..

See Also **DV** Define Variable
RB Remove Bargraph

<DDn,m>

Define Decimal

Mapped Variables

Description Set the number of decimal places displayed on standard screens

Parameters $n = 1$ to 8 - Input variable
 $m = 0$ to 5 - Number of decimal places
0 = Show no decimal places
1 - 4 = Show 1 - 4 decimal places
5 = Automatically format to show the maximum number of decimal places

Initial Value All input variables are set to Auto format
Once changed, these settings are retained in non-volatile memory

Modes Indicator mode (i.e. when showing Standard Screens)

Notes The number of decimal places that are displayed is selectable between a value of 0 and 5.

Each input variable can be configured to have a different number of decimal places, according to the application requirements.

A value of $m = 0$ will display no decimal places, $m = 1$ will display one decimal place and so on. However, a value of $m = 5$ will select an automatic mode whereby the maximum number of decimal places is shown, dependant on the available space.

Information written in this way is saved to non-volatile memory and is retained if power to the display is lost.

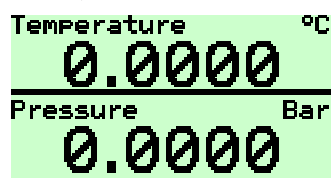
Uses The <DD> command allows:

- The standard screens to show the operator the required precision
- Automatic formatting of the numeric data without additional programming

Example

<SO2>
<SV1> Set Screen Option to 2

<CV1, 0>
<CV2, 0> Send data to variables 1 and 2

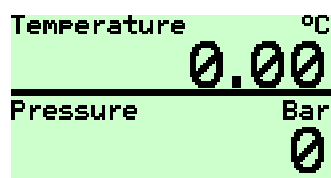


Temperature °C
0.0000

Pressure Bar
0.0000

The default appearance is to display as many decimals as possible on a given screen

<DD1, 2>
<DD2, 0> Set the desired number of decimals

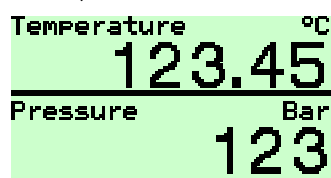


Temperature °C
0.00

Pressure Bar
0

The same data is presented as desired

<CV1, 123.457>
<CV2, 123.456> Send more data to variables 1 and 2



Temperature °C
123.45

Pressure Bar
123

The number of decimals is preserved as the data is updated

See Also DV Define Variable

Description Download soft fonts to the display

Parameters $n = 0$ to 3 - soft font character

Modes Pixel and Row Modes only

Notes A soft font is any user defined image that is the same size as the current font.
The display will store 4 soft fonts ($n = 0$ to 3) for each font F1 to F5.

Soft characters are written to the screen by using the **<WSn>** command and may be used in both Row and Pixel Modes. They may also be underlined and flashed using attributes, as any other character.

Before the **<DFn>** command is issued, the binary download of the soft character should be sent via **GRAPHIC_DATA** making use of the **<GBn>** mechanism as necessary. Detailed information is in the Graphics Transfer Section (Page 11). The required image size depends on the currently active font:

Font:	F1	Image Size (v x h):	8 x 6 pixels
	F2		16 x 10 pixels
	F3		24 x 15 pixels
	F4		32 x 19 pixels
	F5		48 x 29 pixels

Soft fonts are lost when power is removed from the display. All fonts can be saved / restored as a block using the **<KF>** Keep Fonts and **<FR>** Font Restore commands

Uses The **<DF>** command allows:

- Any special character to be stored in the display so that it can be written to the screen just like any other character

Example

```

<CS><F5>          Clear Screen & Set the largest font size

Send a .BMP file of the required soft character to the display. Here a
<GB0>             48 x 29 pixel image of a GBP symbol (£) is sent in three segments,
                  as the actual size of the image in BMP format is 254 bytes.

First 118* bytes written to GRAPHIC_DATA
<GB1>             (Please note that FF Device Revision 2 variants use 64 byte
                  segments, not 118)

Second 118 bytes written to GRAPHIC_DATA
<GB2>

Last 18 bytes written to GRAPHIC_DATA
<DF0>             Tell the display that a soft character number 0 (for Font 5) has been
                  downloaded
<WS0><WT500>      Write the soft character to the screen & Write normal text

```



Gotchas! Make sure that the section on Graphics Transfer (Page 11) is read and understood!

Soft characters can be underlined with the **** attribute – care should be taken when designing fonts if this attribute is to be used.

See Also

DG	Download Graphic	FR	Font Restore
KF	Keep Font	WS	Write Soft Character

Description Download a graphics image to the screen and display it at the current cursor position

Parameters None

Modes Pixel Mode Only

Notes The size of the image is computed from the data sent. If any part of the image would be off-screen when drawn then nothing is drawn on the screen and an error result is produced.

The download mechanism is identical to the <DFn> Download Font and <DS> Download Screen commands. Detailed information is in the Graphics Transfer Section (Page 11).

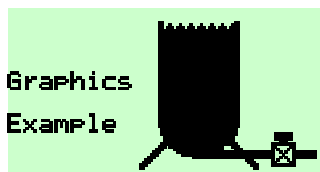
The cursor position is unchanged by this command.

Uses The <DG> command allows:

- Complex images to be generated on a PC and then downloaded to the display.
- Images can form a backdrop onto which standard text or data is then added.
- Pictures can sometimes convey simple messages more easily than text.

Example

<CS>	Clear Screen
<RM>	Set Row Mode
<F1>	Small 8x6 pixel font
<CM3,0>	Move the cursor to the start of the 4th row.
<WTGraphics>	Write the word "Graphics"
<CM5,0>	Move the cursor to the start of the 6th row.
<WTEExample>	Write the word "Example"
<PM>	Change to Pixel Mode
<CM60,50>	Move the cursor to three pixels from the bottom of the screen, 50 pixels from the left hand side
Binary download of .BMP file	Download a 56 x 67 pixel image of a tank to the display, using the <GBn> command (See DFn Example)
<DG>	Tell the display to interpret the graphics image download



Gotchas! Make sure that the section on Graphics Transfer (Page 11) is read and understood

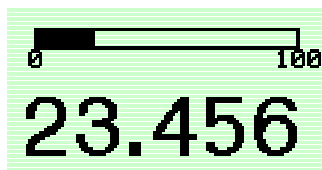
If any part of the graphic would be off-screen, then nothing is drawn and an error result is produced.

See Also

DS	Download Screen
US	Upload Screen

Description	This command defines static minimum and maximum limits for bargraph variables so that optimum scaling can be chosen
Parameters	<p>$m = 0$ to 8 - Input number IN_1 to IN_8</p> <p>$n =$ Any Valid Value - Lower limit for the bargraph variable</p> <p>$p =$ Any Valid Value - Upper limit for the bargraph variable</p> <p>Note: Valid values are 10 characters maximum, including the minus sign and decimal point i.e. Between the range $-999,999,999$ to $9,999,999,999$ (<i>NB Commas only shown for clarity</i>)</p>
Initial Value	None
Modes	Pixel and Row Modes only
Notes	Variables outside the lower or upper limits are shown by empty dotted outline bargraphs
Uses	<p>The <DL> command allows:</p> <ul style="list-style-type: none"> • Optimum scaling of bargraphs • Displaying of fieldbus variables in the form of bargraphs when used in conjunction with the <DB> command

Example	<pre> <SD> Start from a known screen <CM1,10> Move to Row 1 <DL1,0.0,100.0> Define the bargraph limits for IN_1 as 0.0 and 100.0 <DB1,100,0,0,0> Define a horizontal bargraph 100 pixels long using IN_1, and the limits specified by the DL command <CM2,8> Move to row 2 and write in the scale <WT0> Write "0" <CM2,101> Move cursor to the right <WT100> Write "100" <F4> Specify a large font (32x19 pixels) <CM7,5> Move to Row 7 <DV1,6,3,0> Define a variable using IN_1, 6 character field, 3 chars after the decimal point, and written left aligned </pre>
----------------	--



The fieldbus variable linked to IN_1 in this example has a “good” status and a value of “23.456”

Gotchas! This command also affects all bargraphs that are mapped to that fieldbus variable on hidden and saved frames.

See Also **DB** Define Bargraph

Description Download a full-screen 64 x 120 pixel graphic image to the screen.

Parameters None

Modes Pixel and Row Modes only
The <WMn> Write Mode has no effect on this command

Notes This command is really just a special case of the <DG> command, but because of the fixed size is executed much faster.

This command ignores the current Write Mode setting, and draws the downloaded image to the screen normally.

All other attributes are ignored, except those concerned with the ability to Flash the image.

The download mechanism is identical to the <DFn> Download Font and <DG> Download Graphic commands. Detailed information is in the Graphics Transfer Section (Page 11).

The cursor position is unchanged by the command.

Uses The <DS> command allows:

- A full screen image to form a backdrop onto which standard text or data is then added
- A customised logo to appear at power on when used with the <SF> and <BS> commands

Example

<CS> Clear Screen

Send a .BMP file of the required 64 x 120 pixel graphics image that it should display full screen. This is sent in ten segments, as the actual size of the image in BMP format is 1086 bytes.

<GB0>

Note that FF Device Revision 2 variants use 64 byte segments, not 118, so 17 segments are required [i.e.GB0-GB16]

First 118 bytes written to GRAPHIC_DATA

<GB1>

Next 118 bytes written to GRAPHIC_DATA

<GB2>

...etc, until

<GB9>

Last 24 bytes written to GRAPHIC_DATA

<DS> Tell the display to process the data, and display it on the screen.

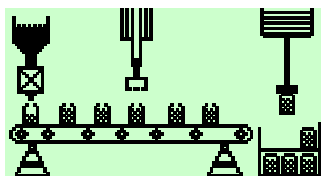


Image is displayed when received

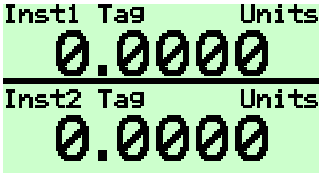
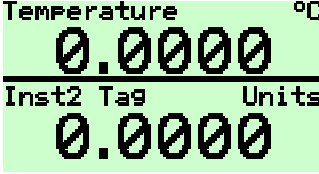
Gotchas! The logo is not stored in EEprom unless the <SF> Save Frame command is issued

The FF Device Revision 2 variant use 64 byte segments as the GRAPHIC_DATA parameter is smaller

See Also RF Restore Frame
SF Save Frame

<DTn,string> Define Tag

Mapped Variables

Description	Set Tag information to be displayed next to the input variable on standard screens
Parameters	<i>n</i> = 1 to 8 - the input variable number <i>IN_1</i> to <i>IN_8</i> <i>string</i> - any 7-bit ASCII string up to 16 characters long
Initial Value	“Inst1_Tag”, “Inst2_Tag” ... “Inst8_Tag” Once changed, these settings are retained in non-volatile memory
Modes	Indicator mode (i.e. when showing Standard Screens)
Notes	Each of the eight process variables may be displayed with an identification tag that can have up to sixteen alphanumeric characters. Numbers, punctuation plus upper & lower case letters are available. Information written in this way is saved to non-volatile memory and is retained if power to the display is lost. The <DT> command is used to send an ASCII string with a maximum length of 16 characters.
Uses	The <DT> command allows: <ul style="list-style-type: none">• Displayed variables to have an associated description on screen• Standard screens to be used for the convenient display of up to eight process variables with little application programming
Example	<pre><SO2> Select Screen format 2 <SV1> Show the first variable <CV1,0> <CV2,0> Send data to the input variables</pre>  <pre><DT1, Temperature> Change the Tag to read “Temperature” Change the Units to read “ °C ” <DU1, `C> Note the use of the special character ` which is converted to the degree symbol (See the <DU> command)</pre> 
Gotchas!	The string cannot include '>' as this is the command terminator Some Standard Screens are unable to show the full length of the string due to lack of space
See Also	DU Define Units

<DU n ,string> Define Units

Mapped Variables

Description	Set “Units of Measure” information to be displayed next to the input variable on standard screens
Parameters	$n = 1$ to 8 - the input variable number IN_1 to IN_8 <i>string</i> - any 7-bit ASCII string up to 8 characters long
Initial Value	“Units” Once changed, these settings are retained in non-volatile memory
Modes	Indicator mode (i.e. when showing Standard Screens)
Notes	Each of the eight process variables may be displayed with units of measure that can have up to eight alphanumeric characters. Numbers, punctuation plus upper and lower case letters are available. Information written in this way is saved to non-volatile memory and is retained if power to the display is lost.

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**

The <DU> command is used to send an ASCII string with a maximum length of 8 characters. The command needs to be repeated for each input variable, in order to set the appropriate units of measure.

Uses	The <DU> command allows: <ul style="list-style-type: none">• Displayed variables to have an associated unit of measure on screen• Standard screens to be used for the convenient display of up to eight process variables with little application programming
-------------	--

Example	<SO2> Select Screen format 2
	<SV1> Show the first variable
	<CV1,0> Send data to the input variables
	<CV2,0>

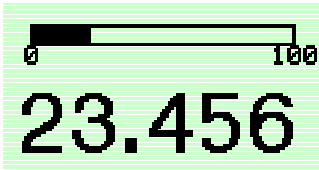
```
Inst1 Tag Units
  0.0000
-----
Inst2 Tag Units
  0.0000
```

<DT1, Temperature>	Change the Tag to read “Temperature”
	Change the Units to read “°C”
<DU1, `C>	Note the use of the special character ` which is converted to the degree symbol



```
Temperature °C
  0.0000
-----
Inst2 Tag Units
  0.0000
```

Gotchas!	The string cannot include '>' as this is the command terminator Some Standard Screens are unable to show the full length of the string due to lack of space
-----------------	--

See Also	DT Define Tag
-----------------	----------------------

Description	This command specifies how a fieldbus variable is written to the screen	
Parameters	<i>m</i> = 1 to 8	- the input number <i>IN_1</i> to <i>IN_8</i>
	<i>n</i> = 1 to 10	- the total variable string length
	<i>p</i> = 0 to 9	- the maximum number of digits after the decimal point
	<i>q</i> = 0 or 1	- Alignment. 0 = left align, 1 = right align in the defined field.
Initial Value	None	
Modes	Row Mode only	
Notes	<p>The variable defined by this command is written at the current cursor position and in the current font. The variable is automatically updated as new fieldbus data is received.</p> <p>Fieldbus data with “bad” status is written in “inverse” i.e. White numbers on a Black background. Flashing values indicate an alarm set on this variable has been activated.</p> <p>Only one instance of a particular input can be displayed at a time. If the command is issued when there is already a variable from the specified input on-screen, the variable is moved from the old to the new position.</p> <p>The command will fail and a parameter error result is produced if any part of the defined variable is off-screen.</p>	
Uses	<p>The <DV> command allows:</p> <ul style="list-style-type: none"> • A fieldbus variable to be shown and continuously updated on screen • The status and alarm condition of the variable to be indicated • Up to 8 variables to be displayed on a single screen • Variable definitions to be saved together with on-screen text and graphics using the <SF> command 	
Example	<pre><SD> <CM1,10> <DL1,0.0,100.0> <DB1,100,0,0,0> <CM2,8><WT0> <CM2,101> <WT100> <F4> <CM7,5> <DV1,6,3,0></pre>	<p>Start from a known screen</p> <p>Move to Row 1</p> <p>Define the bargraph limits for <i>IN_1</i> as 0.0 and 100.0</p> <p>Define a horizontal bargraph 100 pixels long using <i>IN_1</i>, and the limits specified by the DL command</p> <p>Move to row 2 and write in the scale “0”</p> <p>Move cursor to the right</p> <p>Write “100”</p> <p>Specify a large font (32x19 pixels)</p> <p>Move to Row 7</p> <p>Define a variable using <i>IN_1</i>, 6 character field, 3 chars after the decimal point, and written left aligned</p>
		<p>The fieldbus variable linked to <i>IN_1</i> in this example has a “good” status and a value of “23.456”</p>
Gotchas!	<p>Parameter “<i>n</i>” overrides parameter “<i>p</i>”</p> <p>A Defined Variable cannot be removed by a Clear Screen or Screen Defaults command – Use the <RV> Remove Variable command instead</p>	
See Also	RV	Remove Variable
	DB	Define Bargraph

<DWyt,yb,xl,xr> Define Window

Description	Defines an area of the screen into which certain screen write commands are constrained	
Parameters	<i>yt</i> = 0 to 7	- top row of the window area
	<i>yb</i> = 0 to 7	- bottom row of the window area
	<i>xl</i> = 0 to 119	- pixel column of the left hand side of the window
	<i>xr</i> = 0 to 119	- pixel column of the right hand side of the window
Initial Value	The initial window size is the full screen	
Modes	Row Mode Only	
Notes	When a window is in use, all cursor related commands are relative to the window area. For example: <HC> will home the cursor in the window area <CM0,0> will move the cursor to the top row, left hand side of the window area. The window may be redefined at any time without affecting the screen contents. In this way a window can be removed by defining the whole screen as a new window i.e.<DW0,7,0,119> The <CS> Clear Screen and <PM> Pixel Mode commands also remove a window definition.	
Uses	The <DW> command allows: <ul style="list-style-type: none">• Text to be scrolled in a window• Trend graphs to be drawn by combining the use of the Horizontal Scroll <HS> command• Static text and graphics to be protected; headings, footers or titles can be left in place while different messages are displayed and cleared within a window	
Example	<pre><FS> Fill Screen <RM> Set Row Mode <F2> 16x10 pixel font <CM4 , 0> Move the cursor to fifth row from the top, at the left of the screen <WM3> Write characters in Inverse mode (clear character on black background) <WTF1ow:> Write the text "Flow:" <WM0> Back to normal write mode (black character on white background) <DW3 , 5 , 60 , 115> Define window for a value to be written <F3> Larger font <CW> Clear the window <WT2 . 74> Write out a value</pre>	
		Subsequent values then only need:
	<pre><HC> Home the cursor in the window area <WT3 . 18> Write out the new value</pre>	
		
Gotchas!	The <CS> Clear Screen and <PM> Pixel Mode commands remove any defined windows	
See Also	CW	Clear Window

Description	This command erases bargraphs created with the <DB> command from the screen.
Parameters	<i>n</i> = 0 - Erases all defined bargraphs from the screen <i>n</i> = 1 to 8 - Erases only the specified bargraph from the screen
Initial Value	None
Modes	Pixel and Row Modes only
Notes	This command acts in two different ways depending on the value of “ <i>n</i> ” There is no need to erase a bargraph if you just want to change its position. Just set up the new attributes and issue another <DB> command To erase bargraphs from stored frames, restore the frame, erase the bargraphs and re-save the frame.
Uses	The <EB> command allows: <ul style="list-style-type: none">• Bargraphs to be permanently erase ed from the display. The <CS> Clear Screen command does not work on defined variables (not for long anyway!)
Example	Self explanatory
Gotchas!	
See Also	DB Define Bargraph EV Erase Variable NS New Screen

Description Flash text and graphics written with the Flashing <FL> attribute set

Parameters None

Initial Value Inhibited (No Flashing)

Modes Pixel and Row Modes only

Notes The <EF> is a global command that affects the whole screen
The opposite of this command is the <IF> Inhibit Flash command

Uses The <EF> command allows:

- Flashing text generally attracts attention.
- Sending the <EF> command after the text is written ensures that all the text (written with the flashing attribute set) starts flashing at the same time.

Example

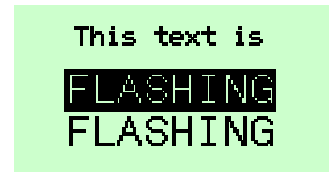
<SD>	Set a known state for the display
<CM1, 0>	Move the cursor to the first line down from the top of the screen
<CA>	All text is aligned centrally
<WTThis text is>	Write out the text
<BM2>	Set the flash background to the inverse of the foreground image
<F2>	Font size 2, 16 x 10 pixels
<CM4, 0>	Move the cursor to the fourth line down from the top of the screen
<FL>	Set the Flashing attribute, so any text written will flash if flashing is enabled
<WTFLASHING>	Write out the text
<CM6, 0>	Move the cursor to the sixth line down from the top of the screen
<WM3>	Write the foreground text as inverse (white on black background)
<WTFLASHING>	Write out some text



<EF> Enable the flashing for the whole screen



↔ Alternating each second with



Gotchas! Attributes are not saved and restored when screens are moved to and from memory with the <SFnm> and <RFm> commands

See Also

BM	Background Mode
FL	Flashing
IF	Inhibit Flashing
ST	Steady

Description	Erase any text or graphics from the current cursor position to the end of the row
Parameters	None
Modes	Row Mode only
Notes	<p>The command erases a number of lines upwards from the current cursor position, depending on the current font:</p> <p style="padding-left: 40px;">For font 1 <F1> only one line is erased For font 2 <F2> two lines are erased, and so on.</p> <p>This command is window aware. If a window is in use the command erases only to the end of the window row.</p> <p>Cursor position is unchanged by this command</p>
Uses	<p>The <CLn> command allows:</p> <ul style="list-style-type: none"> • Message/status information of variable length to be erased. • Ensures new messages can be written without leaving part of an old message in place
Example	<pre> <F1> Sets the single row font, 8 x 6 pixels <CA> Turn on the centre align attribute <WTText line 0> Write out a line of text <LN> Down to the next line ... Last two commands repeated up to..... <WTText line 7> Text line 0 Text line 1 Text line 2 Text line 3 Text line 4 Text line 5 Text line 6 Text line 7 <CM3, 50> Move the cursor to row 3, 50 pixels in from the left of the screen <EL> Text erased from this cursor position to the end of the screen. Text line 0 Text line 1 Text line 2 Text Text line 4 Text line 5 Text line 6 Text line 7 </pre>
Gotchas!	The current font size must be taken into account before issuing this command.
See Also	<p>CL Clear Line</p> <p>DW Define Window</p> <p>Fn Font <i>n</i></p>

Description	This command erases variables created with the <DV> command from the screen.	
Parameters	<i>n</i> = 0	- Erases all defined variables from the screen.
	<i>n</i> = 1 to 8	- Erases only the specified variable from the screen
Initial Value	None	
Modes	Pixel and Row Modes only	
Notes	This command acts in two different ways depending on the value of “ <i>n</i> ”	
	There is no need to erase a variable if you just want to change its position or font. Just set up the new attributes and issue another <DV> command.	
	To erase variables from stored frames, restore the frame, erase the variables and re-save the frame.	
Uses	The <EV> command allows:	
	<ul style="list-style-type: none">• Mapped Variables to be permanently erased from the display. The <CS> Clear Screen command does not work on defined variables (not for long anyway!)	
Example	Self explanatory	
Gotchas!		
See Also	DV	Define Variable
	EB	Erase Bargraph
	NS	New Screen

Description Define the text size written by the <WT> command as 8 x 6 pixels

Parameters None

Initial Value Font 1 is the default used on initialisation

Modes Pixel and Row Modes only

Notes Font 1 is a single row font, each character being 8 pixels high by 6 pixels wide.
 Font 1 does NOT have true descenders
 Font 1 has a full 7-bit ASCII character set
 This command also homes the cursor to the top left character position.

There are five separate commands that define the text size written by the <WT> command. They also affect free text written in Operational Modes 0 and 1.

The font sizes are as follows:

F1	Single row font	8 x 6 pixels
F2	Two row font	16 x 10 pixels
F3	Three row font	24 x 15 pixels
F4	Four row font	32 x 19 pixels
F5	Six row font	48 x 29 pixels

All fonts have a full 7-bit character set, except F5
 All fonts have true descenders, except <F1>

Uses The <F1> command allows:

- The maximum number of characters on the screen
- Creation of long messages without resorting to multiple screens

Example

<CS>	Clear Screen
<F1>	Select font 1
<CM7, 0>	Move cursor to the lower left of the display
<WT12YZ>	Write "12YZ"

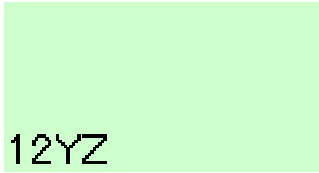


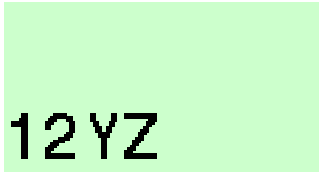
12YZ

Gotchas! The cursor is homed by this command, and may need to be moved to the desired position before writing anything

See Also

DFn	Download Font
WSn	Write Soft Character
WT	Write Text

Description	Define the text size written by the <WT> command as 16 x 10 pixels															
Parameters	None															
Initial Value	Font 1 is the default used on initialisation															
Modes	Pixel and Row Modes only															
Notes	<p>Font 2 is a two-row font, each character being 16 pixels high by 10 pixels wide. Font 2 has true decenders Font 2 has a full 7-bit ASCII character set This command also homes the cursor to the top left character position.</p> <p>There are five separate commands that define the text size written by the <WT> command. They also affect free text written in Operational Modes 0 and 1.</p> <p>The font sizes are as follows:</p> <table border="0" style="margin-left: 40px;"> <tr> <td>F1</td> <td>Single row font</td> <td>8 x 6 pixels</td> </tr> <tr> <td>F2</td> <td>Two row font</td> <td>16 x 10 pixels</td> </tr> <tr> <td>F3</td> <td>Three row font</td> <td>24 x 15 pixels</td> </tr> <tr> <td>F4</td> <td>Four row font</td> <td>32 x 19 pixels</td> </tr> <tr> <td>F5</td> <td>Six row font</td> <td>48 x 29 pixels</td> </tr> </table> <p>All fonts have a full 7-bit character set, except F5 All fonts have true decenders, except <F1></p>	F1	Single row font	8 x 6 pixels	F2	Two row font	16 x 10 pixels	F3	Three row font	24 x 15 pixels	F4	Four row font	32 x 19 pixels	F5	Six row font	48 x 29 pixels
F1	Single row font	8 x 6 pixels														
F2	Two row font	16 x 10 pixels														
F3	Three row font	24 x 15 pixels														
F4	Four row font	32 x 19 pixels														
F5	Six row font	48 x 29 pixels														
Uses	<p>The <F2> command allows:</p> <ul style="list-style-type: none"> • Improved readability over font 1 															
Example	<table border="0" style="margin-left: 20px;"> <tr> <td style="padding-right: 20px;"><CS></td> <td>Clear Screen</td> </tr> <tr> <td style="padding-right: 20px;"><F2></td> <td>Select font 2</td> </tr> <tr> <td style="padding-right: 20px;"><CM7, 0></td> <td>Move cursor to the lower left of the display</td> </tr> <tr> <td style="padding-right: 20px;"><WT12YZ></td> <td>Write "12YZ"</td> </tr> </table> <div style="margin-left: 20px; text-align: center;">  </div>	<CS>	Clear Screen	<F2>	Select font 2	<CM7, 0>	Move cursor to the lower left of the display	<WT12YZ>	Write "12YZ"							
<CS>	Clear Screen															
<F2>	Select font 2															
<CM7, 0>	Move cursor to the lower left of the display															
<WT12YZ>	Write "12YZ"															
Gotchas!	The cursor is homed by this command, and may need to be moved to the desired position before writing anything															
See Also	<table border="0" style="margin-left: 20px;"> <tr> <td style="padding-right: 10px;">DFn</td> <td>Download Font</td> </tr> <tr> <td style="padding-right: 10px;">WSn</td> <td>Write Soft Character</td> </tr> <tr> <td style="padding-right: 10px;">WT</td> <td>Write Text</td> </tr> </table>	DFn	Download Font	WSn	Write Soft Character	WT	Write Text									
DFn	Download Font															
WSn	Write Soft Character															
WT	Write Text															

Description	Define the text size written by the <WT> command as 24 x 15 pixels															
Parameters	None															
Initial Value	Font 1 is the default used on initialisation															
Modes	Pixel and Row Modes only															
Notes	<p>Font 3 is a three-row font, each character being 24 pixels high by 15 pixels wide. Font 3 has true decenders Font 3 has a full 7-bit ASCII character set This command also homes the cursor to the top left character position.</p> <p>There are five separate commands that define the text size written by the <WT> command. They also affect free text written in Operational Modes 0 and 1.</p> <p>The font sizes are as follows:</p> <table border="0" style="margin-left: 40px;"> <tr> <td>F1</td> <td>Single row font</td> <td>8 x 6 pixels</td> </tr> <tr> <td>F2</td> <td>Two row font</td> <td>16 x 10 pixels</td> </tr> <tr> <td>F3</td> <td>Three row font</td> <td>24 x 15 pixels</td> </tr> <tr> <td>F4</td> <td>Four row font</td> <td>32 x 19 pixels</td> </tr> <tr> <td>F5</td> <td>Six row font</td> <td>48 x 29 pixels</td> </tr> </table> <p>All fonts have a full 7-bit character set, except F5 All fonts have true decenders, except <F1></p>	F1	Single row font	8 x 6 pixels	F2	Two row font	16 x 10 pixels	F3	Three row font	24 x 15 pixels	F4	Four row font	32 x 19 pixels	F5	Six row font	48 x 29 pixels
F1	Single row font	8 x 6 pixels														
F2	Two row font	16 x 10 pixels														
F3	Three row font	24 x 15 pixels														
F4	Four row font	32 x 19 pixels														
F5	Six row font	48 x 29 pixels														
Uses	<p>The <F3> command allows:</p> <ul style="list-style-type: none"> • Improved readability over font 2 															
Example	<table border="0" style="margin-left: 20px;"> <tr> <td style="padding-right: 20px;"><CS></td> <td>Clear Screen</td> </tr> <tr> <td style="padding-right: 20px;"><F3></td> <td>Select font 3</td> </tr> <tr> <td style="padding-right: 20px;"><CM7, 0></td> <td>Move cursor to the lower left of the display</td> </tr> <tr> <td style="padding-right: 20px;"><WT12YZ></td> <td>Write "12YZ"</td> </tr> </table> <div style="margin-left: 20px; text-align: center;">  </div>	<CS>	Clear Screen	<F3>	Select font 3	<CM7, 0>	Move cursor to the lower left of the display	<WT12YZ>	Write "12YZ"							
<CS>	Clear Screen															
<F3>	Select font 3															
<CM7, 0>	Move cursor to the lower left of the display															
<WT12YZ>	Write "12YZ"															
Gotchas!	The cursor is homed by this command, and may need to be moved to the desired position before writing anything															
See Also	<table border="0" style="margin-left: 20px;"> <tr> <td style="padding-right: 10px;">DFn</td> <td>Download Font</td> </tr> <tr> <td style="padding-right: 10px;">WSn</td> <td>Write Soft Character</td> </tr> <tr> <td style="padding-right: 10px;">WT</td> <td>Write Text</td> </tr> </table>	DFn	Download Font	WSn	Write Soft Character	WT	Write Text									
DFn	Download Font															
WSn	Write Soft Character															
WT	Write Text															

Description Define the text size written by the <WT> command as 32 x 19 pixels

Parameters None

Initial Value Font 1 is the default used on initialisation

Modes Pixel and Row Modes only

Notes Font 4 is a four-row font, each character being 32 pixels high by 19 pixels wide.
 Font 4 has true decenders
 Font 4 has a full 7-bit ASCII character set
 This command also homes the cursor to the top left character position.

There are five separate commands that define the text size written by the <WT> command. They also affect free text written in Operational Modes 0 and 1.

The font sizes are as follows:

F1	Single row font	8 x 6 pixels
F2	Two row font	16 x 10 pixels
F3	Three row font	24 x 15 pixels
F4	Four row font	32 x 19 pixels
F5	Six row font	48 x 29 pixels

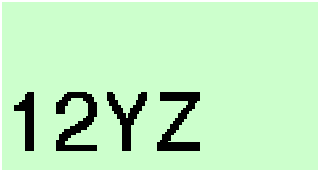
All fonts have a full 7-bit character set, except F5
 All fonts have true decenders, except <F1>

Uses The <F4> command allows:

- An important parameter to dominate the screen layout

Example

<CS>	Clear Screen
<F4>	Select font 4
<CM7, 0>	Move cursor to the lower left of the display
<WT12YZ>	Write "12YZ"



12YZ

Gotchas! The cursor is homed by this command, and may need to be moved to the desired position before writing anything

See Also

DFn	Download Font
WSn	Write Soft Character
WT	Write Text

Description Define the text size written by the <WT> command as 48 x 29 pixels

Parameters None

Initial Value Font 1 is the default used on initialisation

Modes Pixel and Row Modes only

Notes Font 5 is a six-row font, each character being 48 pixels high by 29 pixels wide.
Font 5 has true decenders
Font 5 has a limited 7-bit ASCII character set consisting of the following:

0 to 9, A to Z, space, comma, full-stop, plus, minus.

This command also homes the cursor to the top left character position.

There are five separate commands that define the text size written by the <WT> command. They also affect free text written in Operational Modes 0 and 1.

The font sizes are as follows:

F1	Single row font	8 x 6 pixels
F2	Two row font	16 x 10 pixels
F3	Three row font	24 x 15 pixels
F4	Four row font	32 x 19 pixels
F5	Six row font	48 x 29 pixels


All fonts have a full 7-bit character set, except F5
All fonts have true decenders, except <F1>

Uses The <F5> command allows:

- Maximum visibility
- Eye-catching warnings when used with the <FL> flashing attribute
- Display of one critical process variable

Example

<CS>	Clear Screen
<F5>	Select font 5
<CM7 , 0>	Move cursor to the lower left of the display
<WT12YZ>	Write "12YZ"



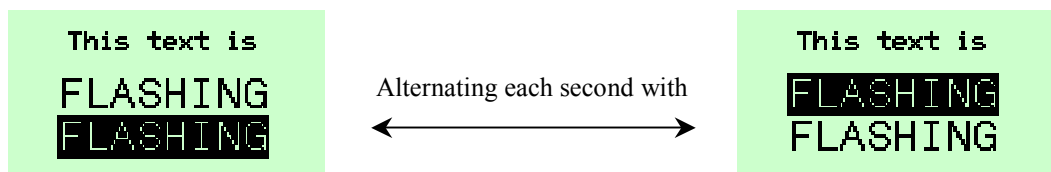
Gotchas! The cursor is homed by this command, and may need to be moved to the desired position before writing anything
F5 has a limited character set

See Also

DFn	Download Font
WSn	Write Soft Character
WT	Write Text

Description	Set the flashing attribute, so that any subsequently written text or graphic will flash when the global attribute <EF> is set.
Parameters	None
Initial Value	Steady (No Flashing)
Modes	Pixel and Row Modes only
Notes	The <BMn> Background Mode attribute controls what background appears when the image flashes. This attribute applies to all writes to the screen except bargraphs. The Flashing attribute is cancelled by the <ST> STeady attribute.
Uses	The <FL> command allows: <ul style="list-style-type: none"> • Attention grabbing messages to be displayed • Screens to be built with both flashing and non-flashing text and graphics • Sending the <EF> command after the text is written ensures that all the text (written with the flashing attribute set) starts flashing at the same time.

Example	<SD>	Set a known state for the display
	<CM1, 0>	Move the cursor to the first line down from the top of the screen
	<CA>	All text is aligned centrally
	<WTThis text is>	Write out the text
	<BM2>	Set the flash background to the inverse of the foreground image
	<F2>	Font size 2, 16 x 10 pixels
	<CM4, 0>	Move the cursor to the fourth line down from the top of the screen
	<FL>	Set the Flashing attribute, so any text written will flash if flashing is enabled
	<WTFLASHING>	Write out the text
	<CM6, 0>	Move the cursor to the sixth line down from the top of the screen
	<WM3>	Write the foreground text as inverse (white on black background)
	<WTFLASHING>	Write out some text
	<EF>	Enable the flashing for the whole screen



Gotchas! Flashing messages with a blank background can cause the message to be missed on a glance at the display. If this could be a problem, use a flash background which is the inverse of the of the image <BM2> as in the example above.

Attributes are not saved and restored when screens are moved to and from memory with the <SFnm> and <RFm> commands

See Also	BM	Background Mode
	EF	Enable Flashing
	IF	Inhibit Flashing
	ST	Steady

Description Recover previously stored soft fonts from EEprom

Parameters None

Modes Pixel and Row Modes only

Notes This command recovers all the soft fonts in all sizes F1 to F5 from EEprom, overwriting any that may have been downloaded, but not kept. This command is required as all currently defined soft fonts (except those stored in EEprom) are lost when power is removed from the instrument.

Fonts can only be recovered as an entire block. There is no provision for restoring a single soft font number, or a single font size.

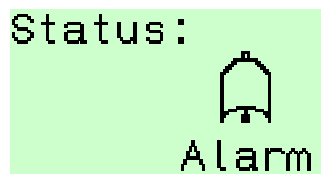
This command is used in conjunction with the Download Font <DF*n*> and Keep Fonts <KF> commands.

Uses The <FR> command allows:

- Time to be saved, rather than having to download soft font sets after a power down

Example

<CS>	Clear the screen
<F2>	Define the font size required
<WTStatus:>	Write out the text "Status:"
<CM7, 65>	Bottom line of screen, 65 pixels from the left of the screen
<WTAlarm>	Write the text "Alarm"
<FR>	Recover soft fonts. N.B. The position of the command is unimportant. The command could have been issued at any time after power-up and before the Write Soft <WS> command.
<F4>	Choose the font size
<CM5, 80>	Go to the position to write the character
<WS3>	Writes character number 3 (bell) in soft font size F4 to the screen



```
Status:
      [Bell Icon]
Alarm
```

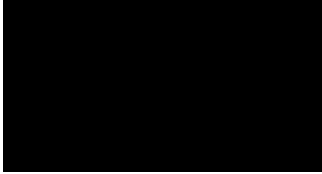
Gotchas!

This is an 'all-or-nothing' command – all fonts of all sizes are restored at once

Performing a <FR> Font Restore without first downloading and saving the desired characters will yield unpredictable results

See Also

DF Download Font
KF Keep Font

Description	Turn all pixels on, creating a black screen
Parameters	None
Initial Value	All pixels are off
Modes	Pixel and Row Modes only
Notes	This command also: Removes any windows that may be defined (equivalent to issuing a <DW0,7,0,119> command) Homes the cursor (equivalent to issuing a <HC> command)
Uses	The <FS> command provides: <ul style="list-style-type: none">• A known starting point before drawing a new screen
Example	<FS> Fill Screen 
Gotchas!	If windows are being used, they must be defined after this command
See Also	CS Clear Screen CW Clear Window DW Define Window HC Home Cursor

Description Turn all pixels on within a defined window

Parameters None

Initial Value All pixels are off

Modes Pixel and Row Modes only

Notes This command also homes the cursor in the defined window area, equivalent to issuing a <HC> command.

Apart from its main use in just filling the contents of a window, it can also be used to create inverse frames to contain text and graphics. Using this technique is much faster than using a Box Draw <BD> command in Pixel Mode.

Uses The <FW> command allows:

- A known starting point before updating a window
- Creation of a simple border

Example

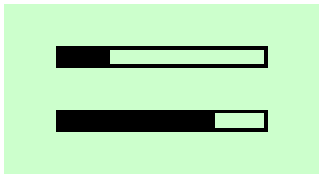
<RM>	Set Row Mode
<CW>	Turn all screen pixels on
<DW1, 6, 10, 110>	Define a window one row from top and bottom, 10 pixels in from both sides
<FW>	Set all the pixels in the window area to on
<F2>	Set required font size
<CM3, 0>	Move the cursor to the third row down in the window
<CA>	Centre align the following text
<WM3>	Set inverse mode
<WTInverse>	Write out the text



See Also

BD	Box Draw
CS	Clear Screen
DW	Define Window
HC	Home Cursor

Description	This command specifies the graphics block that can be accessed through the GRAPHIC_DATA parameter in the Transducer block	
Parameters	$n = 0$ to 9	- Block segment number
	or	
	$n = 0$ to 16	- Block segment number (FF Device Revision 2 variant)
Initial Value	$n = 0$	
Modes	All Screen Modes	
Notes	<p>The parameter size is limited to a maximum of 118 bytes (or 64 bytes in the FF Device Revision 2 variant). Graphics data sent to and received by the display is often much larger than this parameter size, so the data has to be split into segments.</p> <p>The <GBn> parameter specifies which segment (i.e. block of 118 or 64 bytes) is being sent or read from the display</p> <p>Graphics commands will ignore any data past the end of file in a block and data in any higher numbered blocks</p>	
Uses	<p>The <GB> command allows:</p> <ul style="list-style-type: none"> • The data length limitations in transferring graphics data to be overcome 	
Example	<GB0>	<p>Set Graphics Block 0</p> <p>Read/Write first segment of data to the display using the GRAPHIC_DATA parameter</p>
	<GB1>	<p>Set Graphics Block 1</p> <p>Read/Write the next segment of data to the display using the GRAPHIC_DATA parameter</p>
	<GB2>	<p>Set Graphics Block 2</p> <p>and so on until all the data has been transferred</p>
	<DG>	<p>Now issue the command to process all previously transmitted blocks.</p>
Gotchas!	<p>Make sure that the section on Graphics Transfer (Page 11) is read and understood</p> <p>The parameter size is different between some model variants</p>	
See Also	DF	Download Font
	DG	Download Graphic
	DS	Download Screen
	US	Upload Screen

Description	Draw a horizontal bargraph n pixels long with m pixels filled	
Parameters	$m = 3$ to 120	- Length of bargraph
	$n = 0$ to n	- Number of filled pixels, starting from the left
Modes	Row Mode only The <WM n > Write Mode has no effect on this command	
Notes	The horizontal bargraph is drawn at the current cursor position. The cursor is restored to its original position after the command. The number of filled pixels has to be less than or equal to the overall length of the bargraph. Note that the first and last pixels are always filled in to form the frame, so <HB80,0> and <HB80,1> are visually identical, as are <HB80,79> and <HB80,80>	
Uses	The <HB> command allows: <ul style="list-style-type: none"> • Simple graphical representation of values or progress • Bargraphs to be combined without restriction with other text and graphics 	
Example	<SD>	Return screen to known state
	<CM2 , 20>	Move cursor to the second row down, 20 pixels from the left of the screen
	<HB80 , 20>	Draw a horizontal bargraph 80 pixels long of which 20 pixels are filled. (25% fill)
	<CM5 , 20>	Move the cursor to the fifth row down
	<HB80 , 60>	Draw another horizontal bargraph 80 pixels long but this time with 60 pixels filled (75% fill)
		
Gotchas!	Bargraphs created by this command are static and are not linked to fieldbus variables	
See Also	VB	Vertical Bargraph

Description Return the cursor to the top left of the screen

Parameters None

Modes Pixel and Row Modes only

Notes This command is a special case of the <CM> Cursor Move command. The vertical position of the cursor is set such that the currently active font will display normally at the top left of the screen.

For example, with <F1> active <HC> is equivalent to <CM0,0>. Similarly with <F5> active <HC> is equivalent to <CM4,0> (in Row Mode)

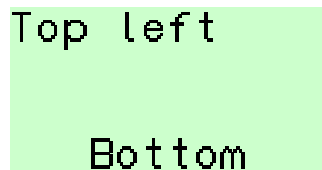
Home cursor is done automatically by commands such as <CS>, <FS>, <CW>, <FW> and setting any font size

Uses The <HC> command allows:

- Any subsequently written text to be easily positioned at the top of the display
- A starting point for constructing new screens

Example

<SD>	Put the display in a known state
<F2>	Set the font size
<CM7, 30>	Cursor down to the bottom of the screen
<WTBottom>	Write out some text
<HC>	Home the cursor
<WTTop left>	Write out some text showing the effect on the cursor of the <HC> command



Gotchas! Make sure that the font size is selected before issuing the <HC> command

See Also

CS	Clear Screen
CW	Clear Window
Fn	Font
FS	Fill Screen
FW	Fill Window

<HSm,n,p,q,r,s,t> Horizontal Scroll

System

Description Scrolls a defined area of the screen by one pixel

Parameters

$m = 0$ or 1	- scrolls the screen either Left ($m = 0$) or Right ($m = 1$)
$n = 0$ to 7	- The first row to scroll.
$p = 0$ to 7	- The last row to scroll
$q = 0$ to 64	- Starting position of line 1
$r = 0$ to 64	- Length of line 1
$s = 0$ to 64	- Starting position of line 2
$t = 0$ to 64	- Length of line 2

Modes Row Mode only

Notes This command scrolls a defined area of the screen, left or right by one pixel. In addition, two vertical lines of any length may be drawn in the 'new' pixel column.

The parameters q and s define the starting positions of these two new lines, in pixels above the bottom of Row p . The length of these lines are defined by the corresponding parameters r and t , again in pixels. These lines are drawn in the blank pixel column created by the left or right pixel block move. The lines may overlap if necessary.

If no lines are required, set q,r,s,t to zero.

By default the command acts on the whole width of the screen, but as it is a window aware command, the effective width may be controlled by setting up a suitable window.

Uses The <HS> command allows:

- Line, bar, block charts that scroll with time
- Visual effects

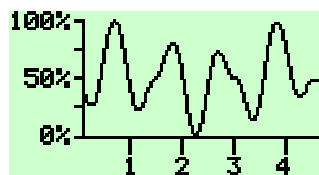
Example

To use this command effectively takes more commands than can easily be listed here. However consider the following screen which will illustrate the possibilities.

This illustrates the use of the command in displaying a trend graph.

The 'y' axis is drawn with the <LH> and <LV> commands

Similarly, the initial 'x' axis is drawn in the same way, but the <HS> command can be used to 'move' the axis with the data if desired.



A window is set up just to the right of the vertical 'y' axis and the whole height of the screen.

The command <HS0,0,7,0,0,0,0> will scroll the graph area and the horizontal 'x' axis left by one pixel.

The line draw parameters are used to:

1. Draw in the next point on the graph
2. Draw in the 'x' axis and its marker lines

The scale values are created with normal cursor moves and write text commands

Gotchas! If the command is used in a window, the parameters are relative to that window.

See Also

DW	Define Window
LH	Line Horizontal
LV	Line Vertical

Description Inhibit the automatic 1 second flash of any text or graphics drawn with the <FL> attribute

Parameters None

Initial Value Inhibited (No Flashing)

Modes Pixel and Row Modes only

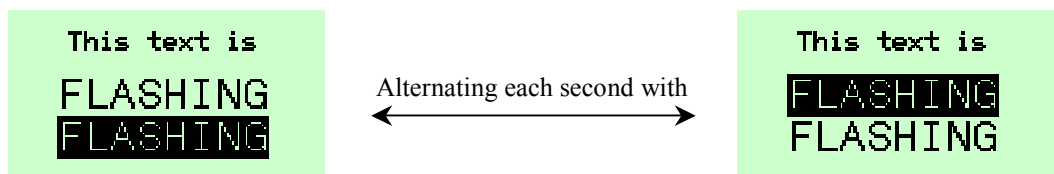
Notes This command acts on the whole screen.
Flashing can be re-enabled by using the <EF> command.

Uses The <IF> command allows:

- A menu structure to be built up of flashing and static screens
- A simple method of acknowledging operator input

Example

<SD>	Set a known state for the display
<EF>	Enable the flashing for the whole screen
<CM1, 0>	Move the cursor to the first line down from the top of the screen
<CA>	All text is aligned centrally
<WTThis text is>	Write out the text
<BM2>	Set the flash background to the inverse of the foreground image
<F2>	Font size 2, 16 x 10 pixels
<CM4, 0>	Move the cursor to the fourth line down from the top of the screen
<FL>	Set the Flashing attribute, so any text written will flash if flashing is enabled
<WTFLASHING>	Write out the text
<CM6, 0>	Move the cursor to the sixth line down from the top of the screen
<WM3>	Write the foreground text as inverse (white on black background)
<WTFLASHING>	Write out some text



<IF> Now inhibit the flashing



Gotchas! When this command is received, the foreground image will be immediately displayed, even if the background was actually on screen at that time

See Also

EF	Enable Flashing
FL	Flashing
IF	Inhibit Flashing
ST	Steady

Description Applies a scaling factor to the input variable before displaying it on the screen

Parameters $m = 1$ to 8 - Input number IN_1 to IN_8
 $n =$ Any Valid Value - Zero Offset for the variable
 $p =$ Any Valid Value - Gain Factor for the variable

Note: Valid values are 10 characters maximum, including the minus sign and decimal point
i.e. Between the range $-999,999,999$ to $9,999,999,999$ (NB Commas only shown for clarity)

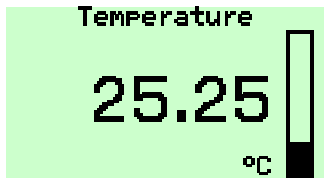
Modes All Screen Modes

Notes Scaling is applied to the specified input data prior to its display. Both numeric display and bargraphs are affected on both custom and standard screens.

The scaling calculation is as follows:
Displayed Value = (Input Value x Gain Factor) + Zero Offset

Uses The <IS> command allows:
• Variables to be shown in different units to those used by the host.

Example



Standard Screen 6 is being used to display an IN_1 value in degrees Celcius.

If we want to display this in degrees Fahrenheit we need to use a conversion formula of $T_f = (T_c \times (9/5)) + 32$

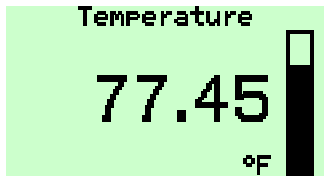
<IS1, 32, 1.8>

Convert from C to F

<DU1, `F>

Change the Units to read "°F"

Note the use of the special character ` which is converted to the degree symbol



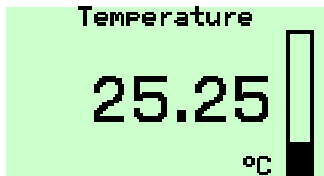
[Note that the bargraph limits may need to be changed, depending on the application]

<IS1, 0, 0>

Revert back to Celcius

<DU1, `C>


Change the Units to read "°C"



Gotchas! Input Scaling may also be applied using the local menus. These should be checked if incorrect readings are being displayed.

See Also DV Define Variable

Description	Save previously download soft fonts (F1-F5) to EEprom										
Parameters	None										
Modes	Pixel and Row Modes only										
Notes	<p>This command causes all the soft fonts in all font sizes F1 to F5 to be saved to EEprom.</p> <p>Downloaded soft fonts not stored in this way are lost when power is removed.</p> <p>It is not possible to save just an individual soft font number or even all the soft fonts in a given size. Soft fonts are restored with the <FR> command. This is not done automatically on power-up.</p> <p>The font data is written as a block and will overwrite any previously stored data.</p> <p>To add a soft font definition to the current stored values they should be restored to the instrument memory first. The new font can then be downloaded and the entire new font set re-saved</p>										
Uses	<p>The <KF> command allows:</p> <ul style="list-style-type: none"> • A quicker method of providing soft fonts after power-on. 										
Example	<table border="0"> <tr> <td style="padding-right: 20px;"><FR></td> <td>Get any existing font data</td> </tr> <tr> <td><F2></td> <td>Set required font size</td> </tr> <tr> <td>102 bytes written to GRAPHIC_DATA</td> <td>Binary download of 16 x 10 pixel image</td> </tr> <tr> <td><DF3></td> <td>Process the data as character 3 of Font 2</td> </tr> <tr> <td><KF></td> <td>Save fonts</td> </tr> </table>	<FR>	Get any existing font data	<F2>	Set required font size	102 bytes written to GRAPHIC_DATA	Binary download of 16 x 10 pixel image	<DF3>	Process the data as character 3 of Font 2	<KF>	Save fonts
<FR>	Get any existing font data										
<F2>	Set required font size										
102 bytes written to GRAPHIC_DATA	Binary download of 16 x 10 pixel image										
<DF3>	Process the data as character 3 of Font 2										
<KF>	Save fonts										
Gotchas!	<p>Fonts are not restored automatically on power-up</p> <p>Earlier versions of this product was unable to store and recall Font 5</p>										
See Also	<table border="0"> <tr> <td style="padding-right: 20px;">DF</td> <td>Download Font</td> </tr> <tr> <td>FR</td> <td>Font Restore</td> </tr> </table>	DF	Download Font	FR	Font Restore						
DF	Download Font										
FR	Font Restore										

Description	Set the attribute so that written text is aligned to the left of the display or defined window	
Parameters	None	
Initial Value	Not aligned; Text appears at the current cursor position	
Modes	Pixel and Row Modes only	
Notes	<p>This command sets the attribute that causes text written with the <WT> command to be aligned at the left hand side of the screen (or window, if defined).</p> <p>It only affects text written after the attribute has been set.</p> <p>The attribute is cancelled by the <NA> command or any of the other text alignment commands <CA>, <RA>, <SW> & <TW></p>	
Uses	<p>The <LA> command allows:</p> <ul style="list-style-type: none"> • Text to be automatically aligned without the need for cursor move commands • Tidy screen presentation 	
Example	<pre><SD> <CM3, 60> <LA> <WTLeft> <RA> <WTRight> <LN> <CA> <WTMiddle></pre>	<pre>Set the display to a known state Move the cursor to the middle of row 3 Set left align attribute Left align the word 'Left' on the current row Set the right align attribute Right align the word 'Right' on the current row. Move cursor one row down Set centre align attribute The word 'Middle' is written centre aligned on the current row.</pre>
		
See Also	<pre>CA Centre Align NA No Align RA Right Align SW Smart Wrap TW Text Wrap</pre>	

Description	Add a line feed character after a carriage return character has been received
Parameters	None
Initial Value	This attribute is cleared; Line Feed and Carriage Return are independent actions
Modes	Row Mode only
Notes	<p>This command causes the display to add a line feed character after a carriage return character has been received.</p> <p>This has the effect of moving the cursor to the beginning of the next row down when a single 'carriage return' character (13 decimal, 0x0D in hex) is received.</p> <p>If the cursor is already on the bottom line of the display or window, the current screen is scrolled up one line and the cursor positioned at the beginning of the bottom line.</p> <p>The <NL> command cancels this attribute, making LF and CR independent actions. <NL> is the default condition.</p>
Uses	<p>The <LF> command allows:</p> <ul style="list-style-type: none">• The display to be used as a dumb terminal• Hosts that only send CR instead of CR+LF to be accommodated
See Also	NL No LineFeed

Description	Draw a horizontal line <i>x</i> pixels long with a line thickness of <i>l</i>
Parameters	<i>x</i> = 1 to 120 - length <i>l</i> = 1 to 64 - line thickness
Modes	Pixel mode only
Notes	The line is drawn from the current cursor position upwards and to the right. The cursor position is unchanged after the command The parameters may be any value that will keep the line being drawn on-screen. If any part of the defined line is off-screen, then the line is not drawn and an error result is produced.
Uses	The <LH> command allows: <ul style="list-style-type: none"> • information to be segmented • borders to be drawn • line images to be constructed
Example	<pre> <SD> Set the display to a known state <PM> Set display to Pixel Mode <CM33,0> Move the cursor to pixel row 33, at the left of the screen <LH120,4> Draw a horizontal line 120 pixels long and 4 pixels wide <RM> Back to Row Mode <SW> Turn Smart Wrap attribute on. Text wraps without splitting words <HC> Home the cursor to top left of screen <WTThis is the top half of the screen> Write out some text <CM5,0> Cursor move to sixth row down <WT ... and this is the bottom half of the screen> Write out some more text This is the top half of the screen _____ ... and this is the bottom half of the screen </pre>
Gotchas!	The entire line must fit on the screen, otherwise nothing will be drawn and an error response generated
See Also	BD Box Draw LV Line Vertical

<LN>

Line New

Description Send a 'CR + LF' to move the cursor down one line and to the left hand side of the screen or window

Parameters None

Modes Row Mode Only

Notes This command sends a 'Carriage Return' + 'Line Feed' to the display so that the cursor is moved down one line and to the left hand side of the screen or window.

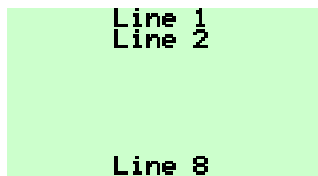
If the cursor is already on the bottom line the display will scroll up one line, leaving the cursor on the new bottom line.

Uses The <LN> command allows:

- A vertical scroll of text (and graphics) to occur if the cursor is already on the bottom line
- A quicker but more limited version of the Cursor Move command

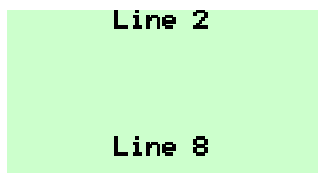
Example

<SD>	Set the display to a known state
<CA>	Align all following text centrally
<WTLine 1>	Write some text
<LN>	Move the cursor down one line
<WTLine 2>	Write some more text
<CM7,0>	Move to the bottom line
<WTLine 8>	Write some more text



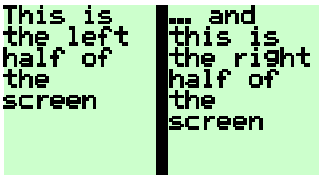
A terminal window with a light green background. The text 'Line 1' and 'Line 2' is centered on the first two lines. There is a large gap, and then 'Line 8' is centered on the eighth line.

<LN> Move the cursor down one line



A terminal window with a light green background. The text 'Line 2' is centered on the second line. There is a large gap, and then 'Line 8' is centered on the eighth line.

See Also SW Smart Wrap

Description	Draw a vertical line <i>y</i> pixels high with a line thickness of <i>l</i>	
Parameters	<i>y</i> = 1 to 64	- height
	<i>l</i> = 1 to 120	- line thickness
Modes	Pixel mode only	
Notes	<p>The line is drawn from the current cursor position upwards and to the right.</p> <p>The cursor position is unchanged after the command</p> <p>The parameters may be any value that will keep the line being drawn on-screen. If any part of the defined line is off-screen, then the line is not drawn and an error result is produced.</p>	
Uses	<p>The <LV> command allows:</p> <ul style="list-style-type: none"> • information to be segmented • borders to be drawn • line images to be constructed 	
Example	<pre><SD> <PM> <CM63,58> <LV64,4> <RM> <DW0,7,0,57> <SW> <HC> <WTThis is the left half of the screen> <DW0,7,63,119> <HC> <WT ... and this is the right half of the screen></pre>	<pre>Set the display to a known state Set display to Pixel Mode Move the cursor to pixel row 63, in the middle of the screen Draw a vertical line 64 pixels long and 4 pixels wide Back to Row Mode Define a window on the left half of the screen Turn Smart Wrap attribute on. Text wraps without splitting words Home the cursor to top left of window Write out some text Define a window on the right half of the screen Home the cursor to the top left of the window Write out some more text</pre>
		
Gotchas!	The entire line must fit on the screen, otherwise nothing will be drawn and an error response generated	
See Also	BD	Box Draw
	LH	Line Horizontal

Description	Cancel all of the text alignment attributes <LA>, <RA>, <CA>, <SW> and <TW>																
Parameters	None																
Initial Value	This is the default																
Modes	Pixel and Row Modes only																
Notes	<p>This command clears all alignment attributes so that text written with the <WT> command appears at the current cursor position.</p> <p>It only affects text written after the attribute has been set.</p>																
Uses	<p>The <NA> command allows:</p> <ul style="list-style-type: none"> • manual formatting after special alignment attributes have been used 																
Example	<table border="0"> <tr> <td style="padding-right: 20px;"><code><SD></code></td> <td>Set screen to known state</td> </tr> <tr> <td style="padding-right: 20px;"><code><RA></code></td> <td>Set right alignment attribute on</td> </tr> <tr> <td style="padding-right: 20px;"><code><WTThis text is></code></td> <td>Write out some text</td> </tr> <tr> <td style="padding-right: 20px;"><code><LN></code></td> <td>Cursor to next line down, left of screen</td> </tr> <tr> <td style="padding-right: 20px;"><code><WTright aligned></code></td> <td>Write some more text</td> </tr> <tr> <td style="padding-right: 20px;"><code><LN></code></td> <td>Cursor to next line down, left of screen</td> </tr> <tr> <td style="padding-right: 20px;"><code><NA></code></td> <td>Cancel text alignment attribute</td> </tr> <tr> <td style="padding-right: 20px;"><code><WTThis is not.></code></td> <td>Write some text, this time it appears at the current cursor position.</td> </tr> </table> <pre style="background-color: #e0ffe0; padding: 5px;"> This text is right aligned This is not.</pre>	<code><SD></code>	Set screen to known state	<code><RA></code>	Set right alignment attribute on	<code><WTThis text is></code>	Write out some text	<code><LN></code>	Cursor to next line down, left of screen	<code><WTright aligned></code>	Write some more text	<code><LN></code>	Cursor to next line down, left of screen	<code><NA></code>	Cancel text alignment attribute	<code><WTThis is not.></code>	Write some text, this time it appears at the current cursor position.
<code><SD></code>	Set screen to known state																
<code><RA></code>	Set right alignment attribute on																
<code><WTThis text is></code>	Write out some text																
<code><LN></code>	Cursor to next line down, left of screen																
<code><WTright aligned></code>	Write some more text																
<code><LN></code>	Cursor to next line down, left of screen																
<code><NA></code>	Cancel text alignment attribute																
<code><WTThis is not.></code>	Write some text, this time it appears at the current cursor position.																
Gotchas!	<NA> also cancels <SW> Smart Wrap and <TW> Text Wrap																
See Also	<table border="0"> <tr> <td style="padding-right: 20px;">CA</td> <td>Centre Align</td> </tr> <tr> <td style="padding-right: 20px;">LA</td> <td>Left Align</td> </tr> <tr> <td style="padding-right: 20px;">RA</td> <td>Right Align</td> </tr> <tr> <td style="padding-right: 20px;">SW</td> <td>Smart Wrap</td> </tr> <tr> <td style="padding-right: 20px;">TW</td> <td>Text Wrap</td> </tr> </table>	CA	Centre Align	LA	Left Align	RA	Right Align	SW	Smart Wrap	TW	Text Wrap						
CA	Centre Align																
LA	Left Align																
RA	Right Align																
SW	Smart Wrap																
TW	Text Wrap																

<NL>

No Linefeed

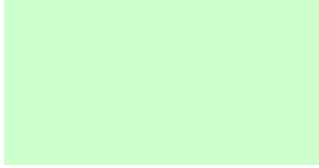
Attributes

Description	Cancel the automatic execution of a ‘CR + LF’ when just a single ‘CR’ is received
Parameters	None
Initial Value	This is the default
Modes	Pixel and Row Modes only
Notes	This command reverses the action of the <LF> command by cancelling the automatic execution of a ‘carriage return’ + ‘linefeed’ when just a single ‘carriage return’ is received.
Uses	The <NL> command allows: <ul style="list-style-type: none">• The display to be used as a dumb terminal• Hosts that send CR and LF separately to be accommodated
Example	<pre><SD> Set screen to known state <LF> Set Linefeed attribute on <WTFirst line of text> Write a line of text Send a [CR] character, with the Line Feed attribute set, this would be interpreted as [CR]+[LF] <WT [CR]> <i>Note! The square brackets are not sent, they are just there to show that a Carriage Return character (ASCII 13) is sent.</i> <WTMore text> This text written on the line below First line of text More text <NL> Turn off line feed attribute <WT [CR]> Send another [CR] character As the Line Feed attribute has been turned off, the display has only actioned the [CR] so this text overwrites the “More Text” string sent earlier. First line of text Last line of text</pre>
See Also	LF Line Feed

<NS>

New Screen

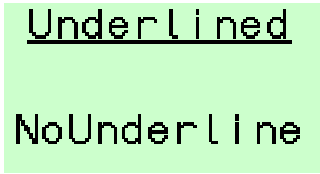
Mapped Variables

Description	Clears the screen and removes all mapped variables and bargraphs
Parameters	None
Initial Value	None
Modes	All Modes
Notes	<p>The <NS> command is functionally equivalent to issuing the three commands <CS><EV0><EB0> in succession.</p> <p>This command also:</p> <ul style="list-style-type: none">• Removes any windows that may be defined (equivalent to issuing a <DW0,7,0,119> command)• Homes the cursor (equivalent to issuing a <HC> command)
Uses	<p>The <NS> command allows:</p> <ul style="list-style-type: none">• An efficient method of clearing the screen after mapped variables have been used
Example	<p><NS> New Screen</p> 
Gotchas!	If windows are being used, they must be defined after this command
See Also	<p>CS Clear Screen EB Erase Bargraph EV Erase Variable</p>

<NU>

No Underline

Attributes

Description	Cancel the Underline attribute																
Parameters	None																
Initial Value	This is the default																
Modes	Pixel and Row Modes only																
Notes	<p>This command cancels the ‘Underline’ attribute so that text written with the <WT> command appears without being underlined</p> <p>It only affects text written after the attribute has been set.</p>																
Uses	<p>The <NU> command allows:</p> <ul style="list-style-type: none">• A combination of underlined and plain text to appear on the same screen																
Example	<table><tr><td><SD></td><td>Set screen to known state</td></tr><tr><td><CA></td><td>Centre align the text</td></tr><tr><td></td><td>Set Underline attribute on</td></tr><tr><td><F2></td><td>Choose a font size (not F1)</td></tr><tr><td><WTUnderlined></td><td>Write out some text that is underlined</td></tr><tr><td><NU></td><td>Cancel the underline attribute</td></tr><tr><td><CM6, 0></td><td>Move the cursor down</td></tr><tr><td><WTNoUnderline></td><td>Write out some more text which is not underlined</td></tr></table> 	<SD>	Set screen to known state	<CA>	Centre align the text		Set Underline attribute on	<F2>	Choose a font size (not F1)	<WTUnderlined>	Write out some text that is underlined	<NU>	Cancel the underline attribute	<CM6, 0>	Move the cursor down	<WTNoUnderline>	Write out some more text which is not underlined
<SD>	Set screen to known state																
<CA>	Centre align the text																
	Set Underline attribute on																
<F2>	Choose a font size (not F1)																
<WTUnderlined>	Write out some text that is underlined																
<NU>	Cancel the underline attribute																
<CM6, 0>	Move the cursor down																
<WTNoUnderline>	Write out some more text which is not underlined																
See Also	UL Underline																

Description	Control the state of the output contacts, making it de-energised	
Parameters	$n = 1$ to 6	- Output number
Initial Value	De-energised (open circuit) on power up	
Modes	Pixel and Row Modes only	
Notes	<p>These commands allow the user to control the state of the output contacts. There are six isolated solid state contacts per display</p> <p>The parameter n selects which output is being controlled: $n = 1$ controls the output Alarm 1 $n = 2$ controls the output Alarm 2 and so on ..</p> <p>The command <ODn> turns off (de-energises) output n The command <OEn> turns on (energises) output n</p> <p>This command cannot be used in conjunction with the <AM> Alarm Mapping command. If the output is already being controlled by an alarm setpoint then a command error result is produced if direct control is attempted.</p>	
Uses	<p>The <OD> command allows:</p> <ul style="list-style-type: none"> • The display to control alarms, annunciators, sounders etc. under program control 	
Example	<pre><OE1> <OE2> <OD1> <OD2></pre>	<p>Output Alarm1 is energised (short circuit)</p> <p>Output Alarm2 is energised (short circuit)</p> <p>Output Alarm1 is de-energised (open circuit)</p> <p>Output Alarm2 is de-energised (open circuit)</p>
		<p>There is no effect on the display LCD screen when these commands are used</p>
See Also	<p>AA Alarm Activation</p> <p>AH Alarm Hysteresis</p> <p>AL Alarm Lower Limit</p> <p>AM Alarm Mapping</p> <p>AU Alarm Upper Limit</p> <p>OE Output Energised</p>	

Description	Control the state of the output contacts, making it energised	
Parameters	$n = 1$ to 6	- Output number
Initial Value	De-energised (open circuit) on power up	
Modes	Pixel and Row Modes only	
Notes	<p>These commands allow the user to control the state of the output contacts. There are two isolated solid state contacts per display.</p> <p>The parameter n selects which output is being controlled: $n = 1$ controls the output Alarm1 $n = 2$ controls the output Alarm2 and so on ..</p> <p>The command <OEn> turns on (energises) output n The command <ODn> turns off (de-energises) output n</p> <p>This command cannot be used in conjunction with the <AM> Alarm Mapping command. If the output is already being controlled by an alarm setpoint then a command error result is produced if direct control is attempted.</p>	
Uses	<p>The <OE> command allows:</p> <ul style="list-style-type: none"> • The display to control alarms, annunciators, sounders etc. under program control 	
Example	<p><OE1> Output Alarm1 is energised (short circuit)</p> <p><OE2> Output Alarm2 is energised (short circuit)</p> <p><OD1> Output Alarm1 is de-energised (open circuit)</p> <p><OD2> Output Alarm2 is de-energised (open circuit)</p>	<p>There is no effect on the display LCD screen when these commands are used</p>
See Also	<p>AA Alarm Activation</p> <p>AH Alarm Hysteresis</p> <p>AL Alarm Lower Limit</p> <p>AM Alarm Mapping</p> <p>AU Alarm Upper Limit</p> <p>OD Output De-energised</p>	

Description	Put the unit into Pixel Mode
Parameters	None
Initial Value	Row Mode
Modes	Pixel and Row Modes only
Notes	This command allows all text to have pixel positional resolution in both vertical and horizontal directions, rather than being constrained into rows as with Row Mode.

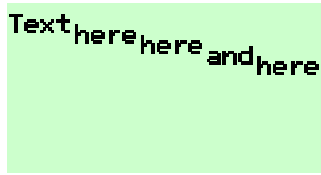
Most graphics commands require the display to be in Pixel Mode.

The vertical parameters for the cursor move command <CM> are 0 to 63 when in Pixel Mode.

Pixel modes writes to the screen are always slower than the corresponding Row Mode write. It is recommended that Row Mode operations are used whenever possible to optimise the response time. Alternatively, complex screens can be written to the non-active frame and then made visible; This gives the appearance of a fast redraw after a short pause.

Uses	The <PM> command allows: <ul style="list-style-type: none"> • Flexibility of text and graphics positioning • Tidy screen presentation
-------------	---

Example	<PM>	Set Pixel mode
	<CM11 , 1>	Move the cursor to Line 11, Row 1
	<WTText>	Write the word 'Text'
	<CM15 , 26>	Move the cursor to Line 15, Row 26
	<WThere>	Write the word 'here'
	<CM19 , 51>	Move the cursor to Line 19, Row 51
	<WThere>	Write the word 'here'
	<CM23 , 76>	Move the cursor to Line 23, Row 76
	<WTand>	Write the word 'and'
	<CM27 , 95>	Move the cursor to Line 27, Row 95
	<WThere>	Write the word 'here'

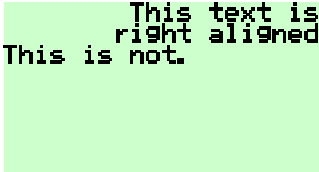


```
Text here here and here
```

Gotchas!	Pixel mode is much slower than Row mode
-----------------	---

The <PM> Pixel Mode command clears any currently defined window

See Also	AF	Active Frame
	RM	Row Mode
	VF	Visible Frame

Description	Set the attribute so that written text is aligned to the right of the display or defined window
Parameters	None
Initial Value	Not aligned; Text appears at the current cursor position
Modes	Pixel and Row Modes only
Notes	<p>This command sets the attribute that causes text written with the <WT> command to be aligned at the right hand side of the screen (or window, if defined). Effectively, the horizontal cursor position is ignored and the text is automatically positioned such that it ends on the right hand edge.</p> <p>It only affects text written after the attribute has been set.</p> <p>The command is cancelled by the <NA> command or any of the other text alignment commands <CA>, <LA>, <SW> & <TW></p>
Uses	<p>The <RA> command allows:</p> <ul style="list-style-type: none"> • Labelling the right hand ‘soft keys’ • Constraining text away from text or images on the left of the screen • Text to be automatically aligned without the need for cursor move commands
Example	<pre> <SD> Set screen to known state <RA> Set right alignment attribute on <WTThis text is> Write out some text <LN> Cursor to next line down, left of screen <WTright aligned> Write some more text <LN> Cursor to next line down, left of screen <NA> Cancel text alignment attribute <WTThis is not.> Write some text, this time it appears at the current cursor position. </pre>  <pre> This text is right aligned This is not. </pre>
See Also	<p>CA Centre Align LA Left Align NA No Align SW Smart Wrap TW Text Wrap</p>

Description	Restore a previously saved frame to the currently active frame
Parameters	$n = 0$ to 3 - Saved Frame memory location
Modes	Pixel and Row Modes only The <WMn> Write Mode has no effect on this command
Notes	This command restores a frame image saved with the <SF> command to the currently active frame. Any mapped variables (text and bargraphs) that are part of the saved frame are also restored.

The parameter n specifies which memory location the stored frame is recovered from:

$n = 0$ specifies EEprom area 0

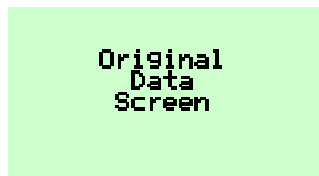
$n = 1$ specifies EEprom area 1

$n = 2$ specifies EEprom area 2

$n = 3$ specifies the scratchpad area in RAM

The scratchpad area is faster than the EEprom areas, but must be used with care as some commands will overwrite this location. See the <SF> Save Frame command for details

Uses	The <RF> command allows: <ul style="list-style-type: none"> • Images to be transferred from one frame to another • Any data screen to be flashed using the sequence <SF0,3><FL><EF><BM2><RF3>
-------------	---

Example

Assume the display is showing some data, and our active frame is set to the same value as the visible frame

<SF0,3>

Save frame 0 to scratchpad RAM

<CS>

Clear screen for new message

<LN>

Move down a line

<WTImportant>

Write out a message....

<LN><WTMessage>

<CM7,0>

<WTAny key to confirm>



Wait for an operator response by checking the KEY_STATUS parameter in the transducer block.


<RF3>

Restore the original screen from scratchpad



Gotchas!	Attributes are not restored The currently Active Frame may not be the currently Visible Frame
-----------------	--

See Also	AF Active Frame
	SF Save Frame
	VF Visible Frame

Description	Put the unit into Row Mode	
Parameters	None	
Modes	Pixel and Row Modes only	
Notes	<p>This command enables Row Mode. In this mode the screen is split up into eight horizontal rows each eight pixels high. Text is then aligned with these rows</p> <p>In this mode the vertical position in the Cursor Move command is limited to 0 to 7.</p> <p>Windows are available in Row Mode to constrain and align text.</p> <p>Writes to the display in Row Mode are always faster than Pixel Mode operation, and should be used wherever possible</p>	
Uses	<p>The <RM> command allows:</p> <ul style="list-style-type: none"> • Rapid display of text messages • Simple text alignment 	
Example	<pre> <CS> <RM> <CA> <WTPlease> <F2> <CM3,0> <WTPRESS KEY 6> <F1> <CM5,0> <WTwhen the operation> <LN> <WT is complete> </pre>	<pre> Clear screen for new message Set Row Mode Centre align the text Write out message.... Use a larger font size Move the cursor to row 3 Write more text Back to the small font Move the cursor to row 5 Write out more text Next line down Write out final line of text </pre>
		
See Also	PM	Pixel Mode
	DW	Define Window

Description	Alter the intensity of the backlight
Parameters	$n = 0$ to 40 - Backlight Intensity 0 = Off, 40 = full on
Initial Value	Dependant on setting made in configuration menus
Modes	Pixel and Row Modes only
Notes	<p>This command alters the intensity of the backlight depending on the parameter n:</p> <p>$n = 0$ backlight off. $n = 40$ backlight fully on.</p> <p>The new backlight intensity is not saved in EEprom. If permanent changes to the backlight intensity are required, use the configuration or quick access menus</p>
Uses	<p>The <SB> command allows:</p> <ul style="list-style-type: none">• The backlight to be flashed to attract attention• Panel illumination to be controlled by the host
Example	<p><SB0> Turn the backlight off</p> <p><SB40> Turn the backlight to full intensity</p>
Gotchas!	The current backlight intensity cannot be read back from the display, nor can the defaults be changed by the host
See Also	

Description Cancels all attributes and returns the display to a known configuration

Parameters None

Initial Value This is the default at power up

Modes Pixel and Row Modes only

Notes This command behaves as if the following commands were received by the display:

<AF0>	Active frame = 0
<VF0>	Visible frame = 0
<F1>	Small font 8 x 6 Pixels
<CS>	Clear Screen
<HC>	Cursor homed
<WM0>	Normal text
<RM>	Row Mode
<IF>	Inhibit Flashing
<ST>	Text Steady attribute
<NA>	No Text Alignment or Wrap
<BM0>	Background Mode = 0
<NU>	No Underline

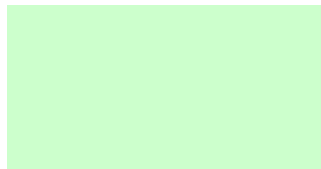
As a consequence, the screen is cleared, window definitions are removed, display scrolling is turned off and key press data cleared

Uses The <SD> command allows:

- A known starting point for the creation of each screen

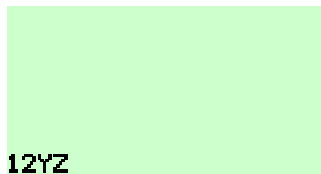
Example

<SD> Set Screen defaults



<CM7, 0> Move cursor to the lower left of the display

<WT12YZ> Write "12YZ"



12YZ

Gotchas!

Use the <CS> Clear Screen for a less drastic initialisation
Mapped variables assigned to text and bargraphs are not cleared

See Also

CS	Clear Screen
NS	New Screen
RB	Remove Bargraph
RV	Remove Variable

Description Save the specified frame m to memory location n

Parameters $m = 0$ or 1 - frame number
 $n = 0$ to 3 - memory location

Initial Value None

Modes Pixel and Row Modes only

Notes The save frame command allows the specified frame m to be saved to memory location n .
 $n = 0$ saves the frame m to EEprom area 0
 $n = 1$ saves the frame m to EEprom area 1
 $n = 2$ saves the frame m to EEprom area 2
 $n = 3$ saves the frame m to scratchpad RAM

Saved frames are restored with the <RF n > command.

The scratchpad RAM area is also used by the following commands:

<BD>, <DF>, <DG>, <LH>, <LV>, <SF>, <US> and the combination <SO4><BS3>

Use of any of these commands will corrupt a saved image in scratchpad ram.

Any mapped variables (text and bargraphs) that are part of the frame are also saved.

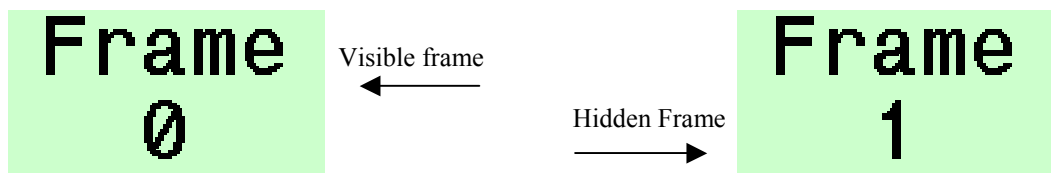
Detailed information about the use of frames can be found in the Frames Section (Page 7).

Uses The <SF> command allows:

- Complex screen backdrops to be saved, to which live data can then be added
- Temporary frame storage while another message is displayed
- Images to be moved between frames
- Normally static frames to flash, by saving them and then restoring them with the <FL> and <EF> attributes turned on. This is a simple way of indicating an alarm condition.

Example 1

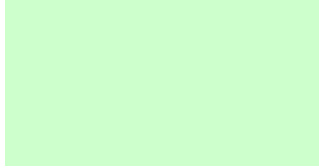
<SD>	Start with the active frame and visible frame set to 0
<CS>	Clear frame 0
<F4>	Set the required font
<CA>	Let the display centre the text automatically
<WTF $frame$ >	Write out the word "Frame"
<LN>	Down a row
<WT0>	Write out the number "0"
<AF1>	Switch to the hidden frame
<CS>	Clear the hidden frame
<WTF $frame$ >	Write out the word "Frame"
<LN>	Down a row
<WT1>	Write out the text to the hidden frame; LCD display screen unaltered
<SF0, 1>	Save frame 0 to EEprom area 1
<SF1, 3>	Save frame 1 to scratchpad RAM.



Example 2

<SD>

Sets active and visible frames to 0 and clears the screen



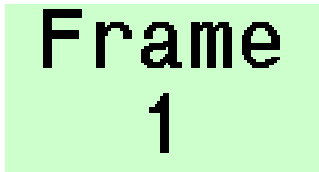
<RF1>

The text “Frame 0” is restored to the screen from EEprom



<RF3>

The text “Frame 1” is restored to the screen from scratchpad RAM



Note: If this last sequence is repeated after the power has been removed and restored, then only the RF0 will restore the saved image correctly as the scratchpad ram contents will be undefined.

Gotchas!

Make sure that the section on Frames (Page 7) is read and understood

Be aware of the limitations regarding scratchpad RAM – unexpected results may easily occur

Frame *m* may or may not currently be visible. Use the <VF> command to achieve the desired result

See Also

RF Restore Frame
VF Visible Frame

Description This command allows the screen type to be changed remotely.

Parameters

- $n = 0$ - Text Display Mode: programmable to generate custom screens
- $n = 1$ - Standard screen; single variable displayed
- $n = 2$ - Standard screen; two variables displayed
- $n = 3$ - Standard screen; four variables displayed
- $n = 4$ - Standard screen; single variable and a horizontal bargraph displayed
- $n = 5$ - Standard screen; two variables and two horizontal bargraphs displayed
- $n = 6$ - Standard screen; single variable and a vertical bargraph displayed
- $n = 7$ - Standard screen; two variables and two vertical bargraphs displayed
- $n = 8$ - Standard screen; three variables and three vertical bargraphs displayed
- $n = 9$ - Standard screen; four variables and four vertical bargraphs displayed

Initial Value Set by the configuration menu

Modes All Screen Modes

Notes This command function is also found in the configuration menu

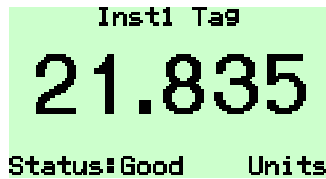
See the “Standard Screens” section on Page 13 for further details.

Uses The <SO> command allows:

- The displayed screen to be changed without having to enter the configuration menus

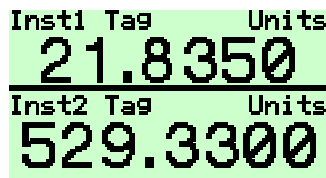
Example

<SO0>



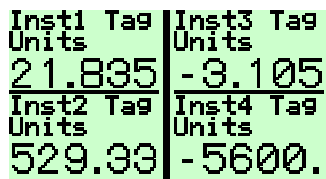
Standard screen, single variable displayed at a time

<SO1>



Standard screen, two variables displayed at a time

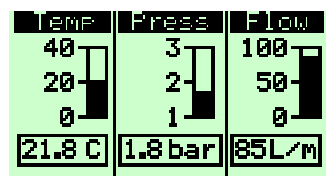
<SO2>



Standard screen, four variables displayed at a time

<SO0>

<SO0> on its own changes to the boot screen. You have to program the display to your requirements.



Text Display Mode – An example of a custom screen (See the appendix for the code used to generate this screen)

Gotchas! Changing Screen Option causes the unit to show the boot screen if changing to Text Display Mode

Description	This command specifies how many of the saved frames can be scrolled through when the arrow keys on the display are pressed.								
Parameters	Values of n allowed and their meanings are: <table><tr><td><i>n</i> = 0</td><td>- No screens are available. Arrow keys have no effect</td></tr><tr><td><i>n</i> = 1</td><td>- Only one screen accessible (Saved frame 0).</td></tr><tr><td><i>n</i> = 2</td><td>- Two screens are available using the arrow keys. (Saved frames 0 and 1)</td></tr><tr><td><i>n</i> = 3</td><td>- Three screens are available using the arrow keys. (Saved frames 0, 1 and 2)</td></tr></table>	<i>n</i> = 0	- No screens are available. Arrow keys have no effect	<i>n</i> = 1	- Only one screen accessible (Saved frame 0).	<i>n</i> = 2	- Two screens are available using the arrow keys. (Saved frames 0 and 1)	<i>n</i> = 3	- Three screens are available using the arrow keys. (Saved frames 0, 1 and 2)
<i>n</i> = 0	- No screens are available. Arrow keys have no effect								
<i>n</i> = 1	- Only one screen accessible (Saved frame 0).								
<i>n</i> = 2	- Two screens are available using the arrow keys. (Saved frames 0 and 1)								
<i>n</i> = 3	- Three screens are available using the arrow keys. (Saved frames 0, 1 and 2)								
Initial Value	Default is <SS0>								
Modes	Pixel and Row Modes only								
Notes	Screens are restored to the current active frame. This frame is then made visible								
Uses	The <SO> command allows: <ul style="list-style-type: none">• A custom screen to be made visible by a single keypress• Unused screens to be hidden								
Example	This command can be used to “hide” a user logo in the following way: <ul style="list-style-type: none">- Download the logo using the <GB> and <DS> commands- Save the logo in EEprom area 2- Set <SS2>- Set <BS2> <p>The user logo will be displayed on power-up. The arrow keys restore the saved operational screens. The user logo cannot be accessed once an arrow key has been pressed. This only works in Text Display Mode with custom screens. There is no way to display a user logo in Screen Options 0 to 3</p>								
Gotchas!	This only works in Text Display Mode with custom screens								
See Also	BS Boot Screen								

<ST>

Steady

Attributes

Description Cancel the flashing attribute set with the <FL> command

Parameters None

Initial Value Steady (No Flashing)

Modes Pixel and Row Modes only

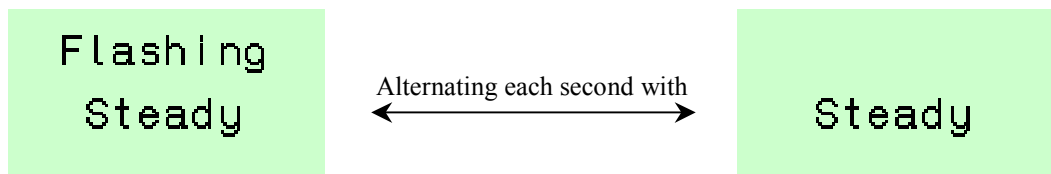
Notes

Uses The <ST> command allows:

- Screens to be built with both flashing and non-flashing text and graphics

Example

<SD>	Set the display in to a known state
<FL>	Set the flashing attribute
<EF>	Enable flashing
<F2>	Use font 2
<CA>	Align the text in the centre of the screen
<CM2, 0>	Down to row 2
<WTFashing>	Write the word "Flashing"
<ST>	Cancel the flashing attribute
<CM5, 0>	Down to row 5
<WTSteady>	Write the word "Steady"



See Also

BM	Background Mode
EF	Enable Flashing
FL	Flashing
IF	Inhibit Flashing

Description Displays the selected input variable using the current Standard Screen

Parameters n = 1 to 8 - Input Variable number

Modes All, when showing Standard Screens

Notes The operator can select which input variable is displayed on the screen by pressing the up and down arrow keys. If necessary, the host can control which input variable is being shown by using this command.

The <SVn> command forces the unit to display input variable n using the current screen format i.e. if 2 variables per screen are being shown, then issuing a <SV4> command will show input variables 3 and 4

Uses The <SV> command allows:

- An operator to be alerted to a particular value
- A host can display a sequence of input variables

Example

<SO2> Start with a 4 variable display

```

Inst1 Ta9  Inst3 Ta9
Units      Units
21.835    -3.105
Inst2 Ta9  Inst4 Ta9
Units      Units
529.33    -5600.

```

<SO2> Change to a 2 variable screen

```

Inst1 Ta9      Units
21.8350
Inst2 Ta9      Units
529.3300

```

<SV3> Now show input variable 3

```

Inst3 Ta9      Units
-3.1050
Inst4 Ta9      Units
-5600.00

```

The standard screen showing input variable 3 is displayed

See Also SO Screen Option

Description	Force text that cannot fit on the current line, to be written on the next line without splitting words	
Parameters	None	
Initial Value	<NA> No Alignment	
Modes	Row Mode only	
Notes	<p>With the <SW> attribute set, the <WT> command will automatically wrap long lines of text without splitting words. It means that the programmer does not have to worry about the formatting as long as the text all fits on the screen. The display will scroll in order to display all the text sent.</p> <p>Smart Wrap is a text alignment attribute that cannot be used in conjunction with any other alignment command <CA>, <LA>, <RA> or <TW>. It is cancelled by the <NA> command.</p> <p><SW> can be used with either the full screen, or within a window.</p>	
Uses	<p>The <SW> command allows:</p> <ul style="list-style-type: none"> • Simple formatting of text strings 	
Example	<pre><SD> <SW> <WTThis is a very long line of text that shows how the Smart Wrap attribute automatically formats the text.></pre>	<p>Set the display to a known state</p> <p>Set the Smart Wrap</p> <p>Write a lot of text. It all fits on screen</p>
	<pre>This is a very long line of text that shows how the Smart Wrap attribute automatically formats the text.</pre>	
	<pre><CS> <DW0,7,20,100> <WTThis is a very long line of text that shows how the Smart Wrap attribute automatically formats the text.></pre>	<p>Clear the screen</p> <p>Define a window</p> <p>Send the same line of text, but because of the narrowed window it does not all fit on screen. The display scrolls to accommodate all the text.</p>
	<pre>line of text that shows how the Smart Wrap attribute automatically formats the text.</pre>	<p>Note that the display has scrolled</p>
See Also	<p>CA Centre Align</p> <p>LA Left Align</p> <p>NA No Align</p> <p>RA Right Align</p> <p>TW Text Wrap</p>	

Description Activate a timer that warns if communications from the host ceases for a (*n* x 10) Seconds

Parameters *n* = 0 to 255 - Multiples of 10 Seconds

Initial Value 0, no timeout active

Modes Pixel and Row Modes only

Notes This command activates a timer that warns via a screen message that there has been no write to the **COMMAND_STRING** parameter for a defined period of time.

Note that cyclic communications to defined variables and bargraphs do not affect the timeout.

The parameter *n* sets a timeout period of *n* x 10 seconds.
n = 0 deactivates the timeout function.

In order to reset the timer, a valid command must be received the display.

Uses The <TO> command allows:

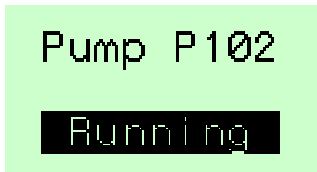
- Users to be warned that the message displayed may be out of date

Example

<T02>

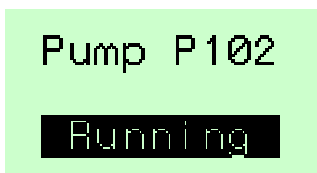
Sets a timeout period of 2 x 10 = 20 seconds

Assume that the following screen was being displayed



If no communication was received for more than 20 seconds the warning screen will alternate every second with the original screen.

When a communication is received, the warning message will not be displayed again until the timeout period has been exceeded once again.



↔ Alternating each second with



Gotchas!

This does not indicate that cyclic data has been interrupted. In these cases, the data is marked "Bad" by being shown in inverse colours.

In normal operation, make sure that the host communicates at least once every timeout period. The <CI> Command Implement command on its own may be used for this purpose

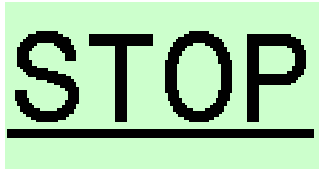
See Also

CI Command Implement

Description	Force text that cannot fit on the current line, to be written on the next line
Parameters	None
Initial Value	<NA> No Align
Modes	Pixel and Row Modes only
Notes	<p>This attributes forces any text that will not fit on the current line to be written on the following line. The operation is not intelligent in any way, the decision of whether to wrap to the next line is made on a character by character basis. This means words will usually flow across two lines.</p> <p>Text written off the end of the bottom line will cause the screen to scroll.</p> <p>The Text Wrap attribute may be used with the whole screen or constrained within a window.</p> <p>It is cancelled by the <NA> No Align command.</p> <p>Text that exceeds the line length without either the <TW> or <SW> attributes set will not be written to the screen and an error result is produced.</p>
Uses	<p>The <TW> command allows:</p> <ul style="list-style-type: none"> • Strings can be sent without worrying about their length • Maximum visible message size, albeit with poor formatting
Example	<pre><SD> Set the display to a known state <CM3,0> Cursor Move to line 3 <TW> Set Text Wrap attribute <WTThis text exceeds the line length> Send a long line of text, which exceeds the screen width</pre> <div style="background-color: #e0ffe0; padding: 5px; margin: 5px 0;"> <pre>This text exceeds th e line length</pre> </div> <p>Note that all the text is displayed without an error being produced, but the word "the" is split on to two lines.</p> <pre><NA> Cancel Text Wrap <CM3,0> Move back to the same starting point <WTThis is a long line of text that wraps on to three rows> Send another long line of text, which exceeds the screen width</pre> <div style="background-color: #e0ffe0; padding: 5px; margin: 5px 0;"> <pre>This is a long line e line length</pre> </div> <p>This time the first line is overwritten, but the second line is not because the text has not been wrapped. Also, an error result is produced to indicate that the write command failed</p>
Gotchas!	If text needs to wrap, but without splitting words, use the <SW> attribute instead.
See Also	<p>NA No Align</p> <p>SW Smart Wrap</p>

UnderLine

Attributes

Description	Set the Underline attribute, so that any subsequently written text is underlined.												
Parameters	None												
Initial Value	<NU> No Underline												
Modes	Pixel and Row Modes only												
Notes	<p>Once this attribute has been set, any text written in Fonts 2 to 5 are underlined in the descender area of the font. As Font 1 does not have descenders, this attribute is not recognised. If Font1 text really does need to be underlined, use a line draw command <LH> in pixel mode.</p> <p>Characters defined in the soft fonts are also underlined using this command. This should be born in mind when defining the characters.</p> <p>The Underline attribute is cancelled with the <NU> command.</p>												
Uses	<p>The command allow:</p> <ul style="list-style-type: none">• Attention to be focussed onto certain text• Screen presentation to be improved by the use of headings												
Example	<table><tr><td><SD></td><td>Set the display in to a known state</td></tr><tr><td><F5></td><td>Maximum font size</td></tr><tr><td><CM6 , 0></td><td>Down to row 6</td></tr><tr><td><CA></td><td>Set centre align attribute</td></tr><tr><td></td><td>Set underline attribute</td></tr><tr><td><WTSTOP></td><td>Write the message</td></tr></table> 	<SD>	Set the display in to a known state	<F5>	Maximum font size	<CM6 , 0>	Down to row 6	<CA>	Set centre align attribute		Set underline attribute	<WTSTOP>	Write the message
<SD>	Set the display in to a known state												
<F5>	Maximum font size												
<CM6 , 0>	Down to row 6												
<CA>	Set centre align attribute												
	Set underline attribute												
<WTSTOP>	Write the message												
Gotchas!	Font 1 cannot be underlined using this method												
See Also	NU No Underline												

Description Upload the current screen contents to the host.

Parameters None

Initial Value

Modes All Screen Modes

Notes Detailed information about the upload procedure is in the Graphics Transfer Section (Page 11).

The <US> command is sent in the normal way.

Uses The <US> commands allow:

- Screen contents to be uploaded to a host computer as a Windows format .BMP file. These screen captures can be included in operator user manuals and other documentation.

This combination of commands was used to generate the example screen-shots in this manual.

Example



Assume default logo is displayed

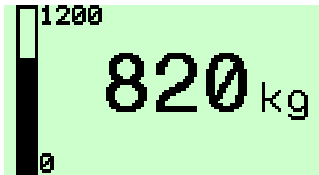
<US>

Bitmap file of screen image is made available to host in the GRAPHICS_DATA parameter



Gotchas! Make sure that the section on Graphics Transfer (Page 11) is read and understood

See Also DS Download Screen

Description	Draw a vertical bargraph n pixels high with m pixels filled
Parameters	$m = 0$ to 64 - Height of bargraph $n = 0$ to m - Number of filled pixels, starting from the bottom
Modes	Row Mode only The <WM n > Write Mode has no effect on this command
Notes	<p>The vertical bargraph is drawn at the current cursor position.</p> <p>The cursor is restored to its original position after the command.</p> <p>The number of filled pixels has to be less than or equal to the overall length of the bargraph. Note that the first and last pixels are always filled in to form the frame, so <VB60,0> and <VB60,1> are visually identical, as are <VB60,59> and <VB60,60></p>
Uses	<p>The <VB> command allows:</p> <ul style="list-style-type: none"> • Simple graphical representation of values or progress • Bargraphs to be combined without restriction with other text and graphics
Example	<pre> <SD> Set the display in to a known state <CM7, 5> Cursor down to the bottom row, five pixels in. <VB64, 44> Draw a vertical bar 64 pixels long with 44 pixels filled <CM7, 14> Cursor to bottom row 14 pixels in <WT0> Write a "0" as the lower scale value <CM0, 14> Cursor to top row, 14 pixels in <WT1200> Write "1200" as the max scale value <F4> Large font <CM5, 37> Cursor position for variable <WT820> Write out the value <F2> Smaller font <PM> Pixel mode so units label can be precisely positioned <CM42, 97> Position of units label <WTkg> Write out the units </pre> 
Gotchas!	Bargraphs created by this command are static and are not linked to fieldbus variables
See Also	HB Horizontal Bargraph

Description	Page frame n is made visible
Parameters	$n = 0$ or 1 - frame number
Initial Value	0
Modes	Pixel and Row Modes only
Notes	The display comprises of two virtual screens, screen 0 and screen 1. Only one of these screens is visible at a time. The <VF n > command is issued to make the required screen visible. It is used in conjunction with the <AF n > Active Frame command.
Uses	The <VF n > command allows <ul style="list-style-type: none"> • complex screens to be drawn while hidden and then instantly displayed • frequently used screens to be instantly restored • a single command to alternate two images

Example

```
This is the
foreground screen
prior to the
following example
```

<AF1>	All writes to the display after this command are directed to screen 1, which is currently hidden
<CS>	Screen 1 is cleared, display still shows the initial message
<F5>	Large Font enabled, display still shows the initial message
<WTSTOP>	The word STOP is written on the hidden screen, display still shows the initial message
<VF1>	Screen 1 now made visible. The word STOP appears on the LCD screen

```
STOP
```

Gotchas! Cursor positions are not saved or restored with frames

This command only makes the selected frame visible; it does not change the frame that is written to. Make sure that the Active Frame <AF n > command is issued appropriately

See Also **AF** Active Frame
RF Restore Frame

Description Determine how text or graphics is drawn on the screen

Parameters n = 0 to 3 - mode number

Modes Pixel and Row Modes only

Notes The write mode is defined by the value n
n = 0 data is written normally to the screen, over-writing the current screen contents
n = 1 data being written to the screen is 'ORed' with the current screen contents
n = 2 data being written to the screen is 'XORed' with the current screen contents
n = 3 the inverse of the data is written to the screen, over-writing the current screen contents

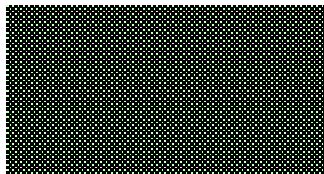
Detailed information is in the Display Features Section (Page 5)

Uses The <WM> command allows:

- Complete flexibility over the appearance of text and graphics
- Allows objects to be written that although they may overlap do not overwrite each other
- Inverse can be used to highlight
- XOR writes will undo what has been written

Example

Original screen

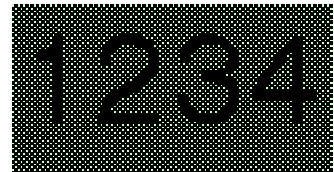


The following examples show the effect of writing the text '1234' in font 5 on a chequer-board background for the 4 write modes:

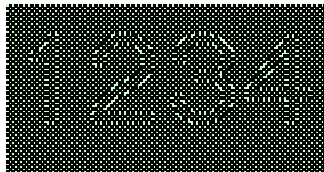
<WM0>



<WM1>



<WM2>

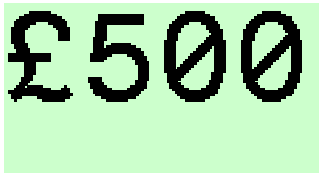


<WM3>



Gotchas! Write modes do not apply to Bargraphs or Restored Frames

See Also BM Background Mode

Description	Write the soft character number <i>n</i> of the current font at the current cursor position
Parameters	<i>n</i> = 0 to 3 - soft font character
Modes	Pixel and Row Modes only
Notes	<p>A soft font is any user defined image that is the same size as the current font. The display can accommodate 4 soft fonts (<i>n</i> = 0 to 3) for each font F1 to F5.</p> <p>The soft character written assume all the current attributes, just as any normal character.</p> <p>Although normally used for text characters or symbols that not in the normal character set, the soft characters can be used to store and write any image of the correct size.</p> <p>This command will assume that the soft font specified has already been downloaded or restored. No error is generated if a soft font does not exist, it just writes uninitialised data.</p> <p>Soft fonts are lost when power is removed from the display. Most fonts can be saved / restored as a block using the <KF> Keep Fonts and <FR> Font Restore commands</p>
Uses	<p>The <WS> command allows:</p> <ul style="list-style-type: none"> • Any special character to be written to the screen just like any other character
Example	<p><CS><F5> Clear Screen & Set the largest font size</p> <p>Send a .BMP file of the required soft character to the display. Here a 48 x 29 pixel image of a GBP symbol (£) is sent in three segments, as the actual size of the image in BMP format is 254 bytes.</p> <p><GB0></p> <p>First 118 bytes written to GRAPHIC_DATA</p> <p><GB1></p> <p>Second 118 bytes written to GRAPHIC_DATA</p> <p><GB2></p> <p>Last 18 bytes written to GRAPHIC_DATA</p> <p><DF0> Tell the display that a soft character number 0 (for Font 5) has been downloaded</p> <p><WS0><WT500> Write the soft character to the screen & Write normal text</p> <div style="text-align: center; margin-top: 10px;">  </div>
See Also	<p>KF Keep Font</p> <p>FR Font Restore</p>

<WTstring>

Write Text

Description Write text to the display, using any set attributes

Parameters *string* = any 7-bit ASCII string

Initial Value None

Modes Pixel and Row Modes only

Notes This command allows text to be written to the display and take advantage of all the attributes and formatting commands.

To simplify temperature display using Fonts 1 to 4, the ` character (alt+096) is mapped to the degrees symbol. For example, the string **Temp `C** is displayed as **Temp °C**

If the '>' character is required in a text string with the <WT> command the character should be included twice.


Text that exceeds the line length without either the <TW> or <SW> attributes set will not be written to the screen and an error result is produced.

Uses The <WT> command allows

- Text to be written !

Example

<SD>	Put the display into a known state
<CM4,0>	Cursor to row 4
<CA>	Align all following text centrally
<WT This is centred >	Write the message



Gotchas! Font 5 has a limited character set, and no degrees symbol

See Also **WS** Write Soft Character

Appendix

The following section lists the commands needed to generate approximations of the standard screens. Note that they will differ slightly because some functionality is not available through the command set. Remember that the parameter size is limited to either 118 or 32 bytes, so the code will need to split into a number of segments, each sent with a <CI> command.

Single Variable Screen

```
<SD>
<RB0>
<RV0>
<F4>
<CM5,3>
<DV1,6,4,1>
<F1>
<CM0,0>
<WTInstrument Tag>
<CM7,0>
<WTStatus:OK>
<RA>
<WTUnits>
```

```
Instrument Tag
 21.835
Status:OK      Units
```

Dual Variable Screen

```
<SD>
<RB0>
<RV0>
<F3>
<CM3,0>
<DV1,8,3,1>
<F1>
<CM0,0>
<WTInst. 1 Tag  Units>
<F3>
<CM7,0>
<DV2,8,3,1>
<F1>
<CM4,0>
<WTInst. 2 Tag  Units>
```

```
Inst. 1 Tag  Units
 21.835
Inst. 2 Tag  Units
 529.330
```

Four Variable Screen

```
<SD>
<RV0>
<RB0>
<PM>
<CM63,58>
<LV64,2>
<F1>
<RM>
<CM0,0>
<WTV3 Temp>
<CM1,0>
<WTDegC>
<F2>
<CM3,0>
<DV1,5,4,1>
<F1>
<CM4,0>
<WTInp Flow>
<CM5,0>
<WTlitres/s>
<F2>
<CM7,0>
<DV2,5,4,1>
<F1>
<CM0,62>
<WTStirer>
<CM1,62>
<WTrpm>
<F2>
<CM3,62>
<DV3,5,4,1>
<F1>
<CM4,62>
<WTV4 Press>
<CM5,62>
<WTkPa>
<F2>
<CM7,62>
<DV4,5,4,1>
```

```
U3 Temp  | Stirer
DegC     | RPM
 21.83   | 15.67
Inp Flow | U4 Press
litres/s | kPa
 529.3   | 5.930
```

Single Variable with Bargraph

```

<SD>
<RB0>
<RV0>
<F4>
<CM5,3>
<DV1,6,4,1>
<F1>
<CM0,0>
<WTInstrument Tag>
<CM7,0>
<WTStatus:OK>
<RA>
<WTUnits>
<CM6,10>
<DB1,100,7,8,0>
Instrument Tag
21.835
Status:OK Units

```

Example Custom Screen

```

<SD><RV0><RB0>
<CM0,2>
<HB36,36>
<CM0,41>
<HB39,39>
<CM0,83>
<HB35,35>
<PM>
<CM63,39>
<LV64,1>
<CM63,81>
<LV64,1>
<RM>
<CM6,5>
<DV1,4,1,0>
<CM6,29>
<WTC>
<PM>
<CM57,3>
<BD12,34,1>
<RM>
<CM0,9>
<WM3><WTTemp><WM0>
<CM4,29>
<DL1,0,40>
<DB1,27,0,0,1>
<PM>
<CM13,26>
<LH3,1>
<CM17,13>
<WT40>
<CM26,26>
<LH3,1>
<CM30,13>
<WT20>
<CM39,26>
<LH3,1>

```

```

<CM43,19>
<WT0>
<RM>
<CM6,43>
<DV2,3,1,0>
<CM6,60><WTbar>
<PM>
<CM57,41>
<BD12,39,1>
<RM>
<CM0,46>
<WM3><WTPress><WM0>
<CM4,69>
<DL2,1,3>
<DB2,27,0,0,1>
<PM>
<CM13,66>
<LH3,1>
<CM17,59>
<WT3>
<CM26,66>
<LH3,1>
<CM30,59>
<WT2>
<CM39,66>
<LH3,1>
<CM43,59>
<WT1>
<RM>
<CM6,85>
<DV3,2,0,0>
<CM6,98>
<WTL/m>
<PM>
<CM57,83>
<BD12,34,1>
<RM>
<CM0,89>
<WM3><WTFlow><WM0>
<CM4,109>
<DL3,0,100>
<DB3,27,0,0,1>
<PM>
<CM13,106>
<LH3,1>
<CM17,87>
<WT100>
<CM26,106>
<LH3,1>
<CM30,93>
<WT50>
<CM39,106>
<LH3,1>
<CM43,99>
<WT0>

```

Temp	Press	Flow
40	3	100
20	2	50
0	1	0
21.8 C	1.8 bar	85L/m



BEKA Associates
Old Charlton Road
Hitchin
Hertfordshire
SG5 2DA

Tel: +44 (0)1462 438301

Fax: +44 (0)1462 453971

Web: www.beka.co.uk

Email: support@beka.co.uk

or sales@beka.co.uk