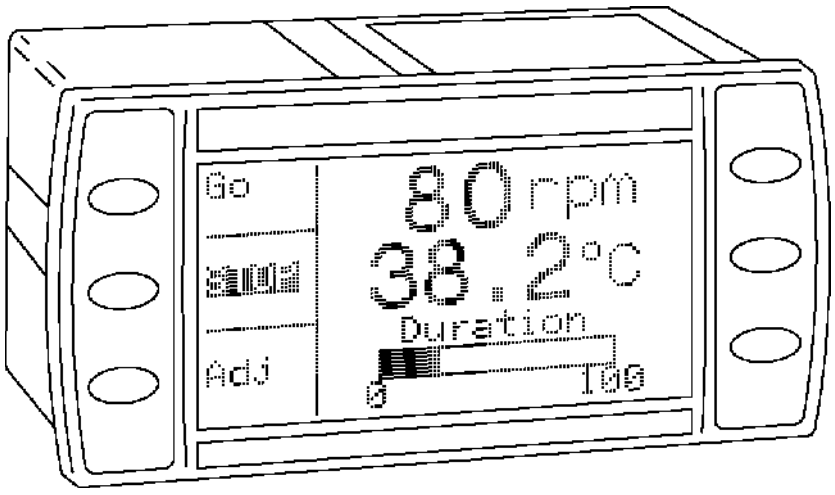
BA488C & BA484D
Serial text display

Programming Guide



Issue 4
10 December 2003

Contents

Introduction

This guide describes the BEKA Mode Protocol for the BA488C and BA484D Serial Text Displays. This information is only required when programming a host to communicate with these displays; it is not required by the end user. The target audience for this guide are software programmers with some experience in communicating with ASCII devices.

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

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.

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 the Instruction Manuals

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

Other sources of information

Our website at ***www.beka.co.uk*** has several files available to download:

- All of the examples in this guide
- Demonstration programs showing the capabilities of the display
- A 'Virtual Instrument' – a PC based simulator that behaves exactly like the real thing. This can be used during program development or to demonstrate the application to end users

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

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 display 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.

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

There are two switch outputs available, which are under total control of the host. These are totally isolated and can be energised or de-energised independently of each other.

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!

Operational Modes	Refers to the communication protocol between the host and the instrument These can range from Mode 0 (the simplest) to Mode 4 (the most complex). This mode essentially determines how much error checking is applied to the data during transmission. See the Protocol section (Page 8) for a detailed explanation
Row and Pixel Modes	Refers to the way text and graphics are positioned on the screen 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 Pixel Mode allows objects to be placed anywhere – but the drawback is that it takes a bit longer for the display to be updated See the <RM> Row Mode and <PM> Pixel Mode commands for further information
Write Modes	Refers to the way text and graphics are written on the screen 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 See the Write Mode section and also the <WM> command
Background Modes	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 See the <BM> Background Mode, <FL> Flashing and <EF> Enable Flashing commands
Key Modes	Refers to the format of the key-press data that is returned to the host. Mode 0 is the simplest, where data is returned as a single byte describing the last key pressed. Mode 1 also returns a single byte, but this time individual key status is returned as the six least significant bits of this byte. Mode 2 returns 6 individual bytes showing the status of each key as an ASCII 0 or 1 See the Response format section (Page 9)

The “Command Reference” section (Page 17) 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"

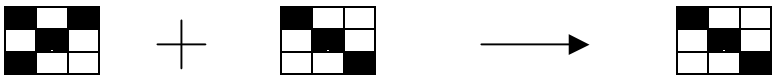
Write Modes

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

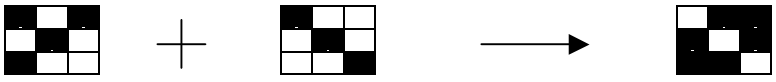
For example:



Existing Screen Display | New Object to be written | Resultant Screen Display

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:



Existing Screen Display | New Object to be written | Resultant Screen Display

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:

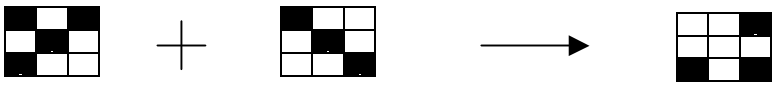
Existing Screen Display | New Object to be written | Resultant Screen Display

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:



Existing Screen Display | New Object to be written | Resultant Screen Display

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.

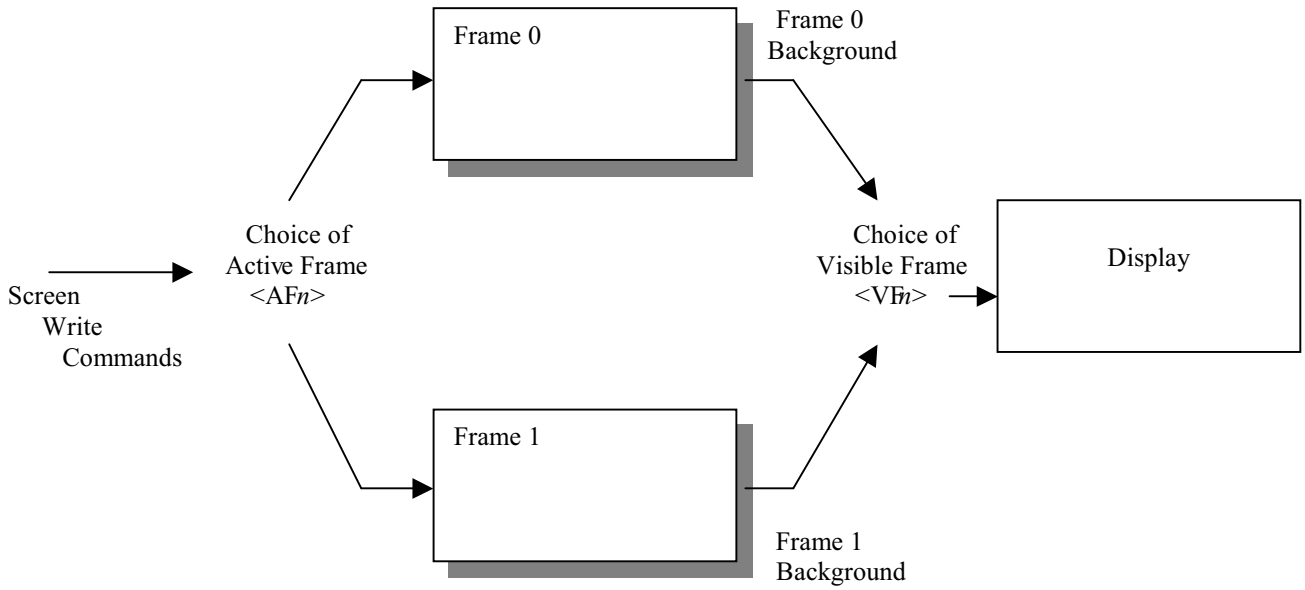
Rather than force the host to do all this work, the background is updated by the instrument automatically. The <BM> Background Mode command is used to control whether the background is clear, black or the inverse of what's written. Once the <BM> command is issued, the background is updated automatically 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, with one notable exception, <RL> Restore Logo.

Saved Frame Locations

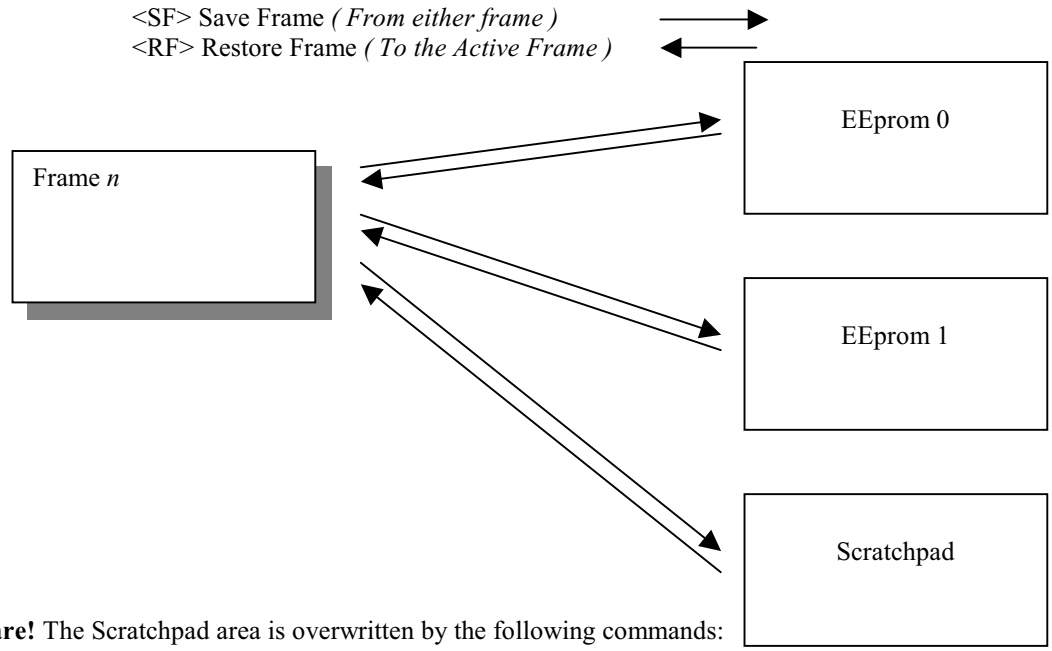
It is possible to store the screen contents for later use by saving the Visible Frame to memory via the <SF*nm*> Save Frame command. (This command can actually save either frame, but for simplicity disregard this for now)

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 two independent EEprom locations that may be used as required.

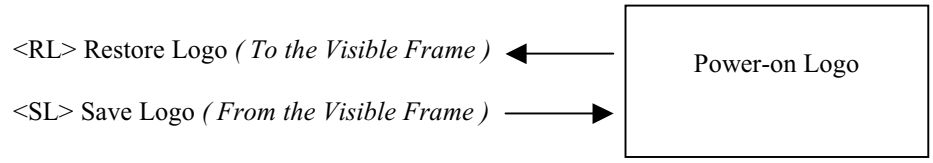
In addition to the two EEprom locations there is a totally separate location, also in EEprom, that is used to store a power-on logo. This is a full screen graphic that appears when the unit is first turned on, or after the unit is re-booted. It is possible to use this area as another storage location, but whatever is in that location will also be used as the logo.



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

<BD>, <DF>, <DG>, <DL>, <LH>, <LV>, <RB>, <RL> & <SL>

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



More information can be found in the Command Reference (Page 17) section.

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, checksums and CRCs
- 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 is returned
    - 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)
- Any characters not enclosed in angled brackets are written directly to the screen at the current cursor position, unless error checking is active

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. The level of checking is adjustable, depending on the application.

- The unit's response to received messages is programmable. Modes are:
    - No response
    - Response to every correctly formatted command
    - Response to a combination of correctly formatted commands
- Where a response is returned, a user must wait for the response before sending further commands
- Message error checking is programmable. Modes are:
    - No error checking
    - Simple checksum
    - 16bit Cyclic Redundancy Check
- Switch status is encoded into the returned message, or can be explicitly requested

Key presses are latched and are sent back to the host with each message. Once sent, the latches are cleared automatically.

Three Key Modes are available according to the application's requirements:

> Mode 0 is the default, and shows the last key that was pressed.
>
> Mode 1 can be used in applications that need to determine if more than one key has been pressed. It can also be used if the external keys are push-to-break (normally closed).
>
> The major disadvantage is that the returned byte is not human readable using a standard terminal.
>
> Mode 2 is provided to overcome this limitation by transmitting six ASCII bytes, each one representing a key, instead of one binary byte. This is useful for debugging, but the relatively high transmission overhead makes it undesirable in general use.

The Key Mode is configured using the display keypad and in-built menus. Refer to the instruction manual for more information.

In all cases it should be remembered that the key status shows the keys that have been pressed since the last response. To determine whether a key is being pressed at any given time, the application should check the second of two consecutive responses.

Command format

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

where:

AB is the command.
[ ] indicates optional parameters separated by comas

example:

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

Response format

Key Mode 0 (Default)

The response format is: Ka or Ea or ?a or P0

where:

K indicates that the previous command/command set has been accepted.
E indicates a parameter or communications error has been detected in the previous command string.
? indicates that the command is unrecognised.
P indicates that a message has been received but NOT actioned, as the unit is being configured by a local user
"a" returns the key status i.e. the number of the key that was last pressed (1=Key 1, 2=Key 2, ….. 6=Key 6)

example:

K0	Command accepted, no keys pressed
E4	Command error detected, key 4 pressed
?6	Command unrecognised, key 6 pressed
P0	Command rejected, unit being configured (Any key presses are discarded)

Note: The unit is configured by a local user accessing the configuration menu directly on the instrument front panel. Access to this menu can be denied by issuing the <CP> Configuration Prohibit command.

Key Mode 1

The response format is the same as Mode 0 but the meaning of "a" is modified. In this mode "a" is not a printable ASCII character. Individual key status is returned as the six least significant bits of this byte. The most significant bit of the byte is always set so that the returned data can be distinguished from the Mode 0 data.

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

msb							lsb
b7	b6	b5	b4	b3	b2	b1	b0
1	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 set (1)

Key Mode 2

The response format is similar to Mode 1 but the “a” is replaced with six consecutive ASCII characters. It is intended mainly for debugging purposes and hosts with limited processing capability.

In this mode the key press information is returned as 6 individual bytes. If a key has been pressed then the ASCII character “1” (hex 0x31, decimal 49) is returned else ASCII character “0” (hex 0x30, decimal 48) is returned. As the returned data is in ASCII notation a dumb terminal can be used to view the data stream.

Key1 is transmitted first.

example:

K000000	Command accepted, no keys pressed
K100010	Command accepted, Keys 1 and 5 have been pressed
E000100	Command error detected, key 4 pressed
?000001	Command unrecognised, key 6 pressed
P000000	Command rejected, unit being configured (Any key presses are discarded)

Operational Modes

The unit can be configured to expect data in a certain format. These formats are termed “Operational Modes” and range from a simple VDU like mode (0) to a fully error checked mode (4).

The operational mode is configured using the display keypad and in-built menus. Refer to the instruction manual for more information.

The modes are:

Mode 0: Commands are executed immediately, no reply message except when specifically requested. Plain text is written directly to the screen, no reply message.

Mode 1: Commands are executed immediately, a response is returned to each command. Plain text is written directly to the screen, no reply message.

Mode 2: Multiple commands can be sent, but these are not executed until a “Command Implement” <CI> command is sent. One reply is returned for each set of commands. An error in any of the commands will result in a Command Error response. Plain text is ignored.

Mode 3: As Mode 2 but the <CI> command is replaced by a <CCn> command where n is a single byte simple checksum of all characters sent (including spaces) up to, but not including the <CCn> command. The returned command has a similar single byte checksum appended to the end of the response. The command string is not actioned if the checksum of the data received does not match the parameter of the <CCn> command. Plain text is ignored

Mode 4: As Mode 3 but the <CCn> is replaced by <CRnn> where nn is a 16-bit CRC code.

CRC Generation

The 16-bit CRC used in the protocol is the same as used for the well known Modbus Protocol. Details are as follows:

The CRC is started by first preloading a 16-bit register to all 1's. Then a process begins of applying successive eight-bit bytes of the message to the current contents of the register. Only the eight bits of data in each character are used for generating the CRC. Start and stop bits, and the parity bit, do not apply to the CRC.

During generation of the CRC, each eight-bit character is exclusive ORed with the register contents. The result is shifted in the direction of the least significant bit (LSB), with a zero filled into the most significant bit (MSB) position.

The LSB is extracted and examined. If the LSB was a 1, the register is then exclusive ORed with a preset, fixed value (A001 hex). If the LSB was a 0, no exclusive OR takes place.

This process is repeated until eight shifts have been performed. After the last (eighth) shift, the next eight-bit character is exclusive ORed with the register's current value, and the process repeats for eight more shifts as described above. The final contents of the register, after all the characters of the message have been applied, is the CRC value.

Generating a CRC

Step 1	Load a 16-bit register with FFFF hex (all 1's). Call this the CRC register.
Step 2	Exclusive OR the first eight-bit byte of the message with the low order byte of the 16-bit CRC register, putting the result in the CRC register.
Step 3	Shift the CRC register one bit to the right (toward the LSB), zero filling the MSB. Extract and examine the LSB.
Step 4	If the LSB is 0, repeat Step 3 (another shift). If the LSB is 1, Exclusive OR the CRC register with the polynomial value A001 hex (1010 0000 0000 0001).
Step 5	Repeat Steps 3 and 4 until eight shifts have been performed. When this is done, a complete eight-bit byte will have been processed.
Step 6	Repeat Steps 2 ... 5 for the next eight-bit byte of the message. Continue doing this until all bytes have been processed.
Result	The final contents of the CRC register is the CRC value.

This CRC value is then appended to the message. The LSB of the CRC is sent first followed by the MSB.

Multidrop Operation:

Multidrop operation is possible. A unique unit address between 1 and 47 has to be given to each instrument by using the display keypad and in-built menus. Refer to the instruction manual for more information.

Command <MC*n*> Make Connection is used to define the instrument address (*n*) to which subsequent commands are directed. This “virtual connection” remains in force until it is explicitly released by issuing the <RC> release Connection command. These two commands are necessary in order to confirm that all instruments receive and react to the commands successfully. Once connected, the units respond in exactly the same way as a single units would, with error responses being issued if a problem occurs.

If a unit has a non-zero address, then on power-up, a <MCn> command must be sent prior to any other command, even if it is the only unit on the line.

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

The command <DS> is issued with any additional bytes (checksum, CRC etc) as required by the current operational mode. The command is acknowledged if correctly received.

A binary download (from the host to the display) of the graphic file is then expected. The image must be exactly 120x64 pixels and if not an error response is returned.

After the file has been downloaded the <CI>, <CCn>, or <CRnn> command must be sent as per the current operational mode, the check byte(s) being calculated from all of the bytes in the .BMP file

The download is acknowledged if it was correctly received (including checksum or CRC checks) and the image is displayed. The downloaded image disregards the Write Mode setting and is displayed normally i.e. as though it was preceded by a <WM0> command. In addition, it only adopts the display attributes concerned with Flashing. All other attributes are ignored.

There is a 2 second timeout for the download operation, during which time if no bytes are received the download is aborted and an error response is returned. Of course, the actual total download time depends on the speed of the serial link.

<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 response.

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 response is returned.

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.

<DF*n*> 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 <WS*n*> command. They may also be underlined and flashed using attributes, as any other character

After the <DFn> command is issued, the display expects a binary download of the soft character. 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 response is returned.

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.

<UE> Upload Enable command

Because a graphic upload generates a significant amount of data, there must be safeguards in place to ensure that the data is really required. The <US> Upload Screen command will therefore not respond unless it is immediately preceded with the <UE> Upload Enable command.

<US> Upload Screen command

The command <US> is issued with any additional bytes (checksum, CRC etc) as required by the current operational mode. The command is acknowledged if correctly received.

After a short delay of 500ms, a 1086 byte block of data is sent to the host.
(This delay is introduced to allow the host to set itself up for data reception).

A command acknowledge then follows with the check bytes as per the current operational mode. The check bytes include the data block bytes and the acknowledge, but not the check bytes themselves.

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

Command Summary

There are 68 commands that can be arranged into 5 functional groups:

Screen Handling & Text	- used to control the screen in text mode
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
<CMy,x>	Cursor Move
<CS>	Clear Screen
<CW>	Clear Window
<EL>	Erase Line
<FS>	Fill Screen
<FW>	Fill Window
<HC>	Home Cursor
<LN>	Line New
<RS>	Request Status
<SD>	Screen Defaults
<WSn>	Write Soft character
<WTthis is a message>	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
<UL>	UnderLine
<WMn>	Write Mode

Line Graphics

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

Pixel Graphics

Command	Meaning
<DG>	Download Graphic
<DS>	Download Screen
<UE>	Upload Enable
<US>	Upload Screen

System

Command	Meaning
<AFn>	Active Frame
<CCn>	Check Code
<CE>	Configuration Enable
<CI>	Command Implement
<CP>	Configuration Prohibit
<CRnm>	Cyclic Redundancy check
<DFn>	Download Font
<FR>	Font Restore
<HSmnrstuv>	Horizontal Scroll
<KF>	Keep Fonts
<MCn>	Make Connection
<ODn>	Output De-energised
<OEn>	Output Energised
<PM>	Pixel Mode
<RB>	ReBoot
<RC>	Release Connection
<RFm>	Restore Frame
<RLx>	Restore Logo
<RM>	Row Mode
<SBn>	Set Backlight
<SFnm>	Save Frame
<SL>	Save Logo
<TOn>	Time Out
<VFn>	Visible Frame

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 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

Gotchas! Common pitfalls to be aware of

See Also Other related commands

<AF*n*> Active Frame

System Command

Description Specify that all writes are directed to Frame *n*

Parameters *n* = 0 or 1 - frame number

Initial Value Frame 0 is the default at power up, or after a <SD> Screen Defaults command

Modes All Modes

Notes All commands (with the notable exception of <RL*n*>) 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 6).

Uses The <AF> command allows:

- Complex screens to be drawn and then displayed when they are complete
- Rapid switching between two different information screens

Example

Original Data Screen	Assume the display is showing some data, and the active frame and visible frame are both set to 0
<AF1>	Set active frame to 1; LCD display screen unaltered
<CS>	Clear the hidden frame; LCD display screen unaltered
<SW>	Set smart wrap formatting on to cope with a long line of text
<WTThis text is written on a hidden frame and can displayed using a <VF1>> command>	Write out the text to the hidden frame; LCD display screen unaltered
Original Data Screen	LCD display screen at this point
<VF1>	Make frame 1 visible
This text is written on a hidden frame and can be displayed using a <VF1> command	

Gotchas! Make sure that the section on Frames (Page 6) is read and understood

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

See Also **VF** Visible Frame

<BDy,x,l> Box Draw

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

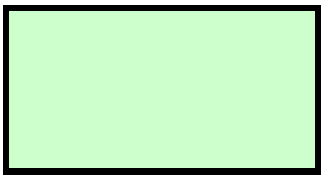
The parameters may be any value that will keep the box being drawn on-screen. If any part of the defined box is off-screen, then the box is not drawn and an error response it returned to the host.

Uses The <BD> command allows:

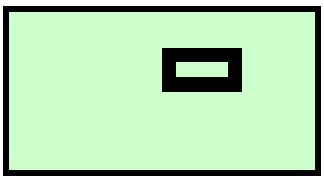
- information to be segmented
- borders to be drawn
- line images to be constructed

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 response generated

See Also

LH Line Horizontal
LV Line Vertical

<BM*n*> Background Mode

Description Defines the appearance of the 'flashing' attribute

Parameters *n* = 0 to 2 - flashing style

Initial Value 0

Modes All Modes

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

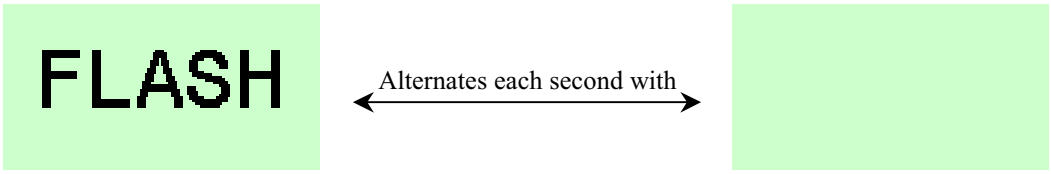
Uses The <BM> command allows:

- attention grabbing messages
- special effects

Example

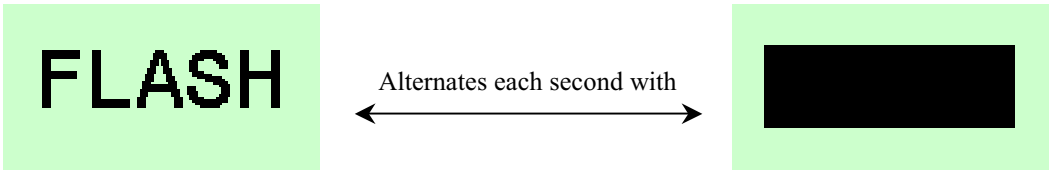
<BM0> Background to flashing characters is all pixels off

<WTFLASH> Write the text FLASH to the screen



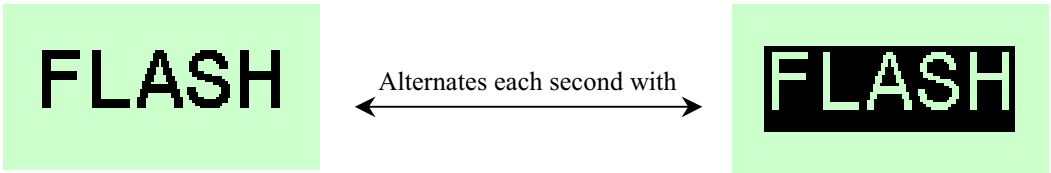
<BM1> Background to flashing characters is all pixels on

<WTFLASH> Write the text FLASH to the screen



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

<WTFLASH> Write the text FLASH to the screen



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

<CA> Centre Align

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 Modes

Notes This command only affects text written after the attribute has been set.

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.

The attribute is cancelled by the <NA> command or any of the other text alignment commands <LA>, <RA>, <SW> & <TW>

Uses The <CA> command allows:

- Text to be automatically aligned without the need for cursor move commands
- Tidy screen presentation

Example

<PM>	Set Pixel Mode
<CM40,0>	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
<CA>	Align all following text centrally
<WTThis is centred>	Write the message
This is centred	The horizontal position is calculated from the length and size of the text
<RM>	Set Row Mode
<DW0,7,60,119>	Define a window as the right hand half of the screen
<CM1,0>	Move the cursor to row 1 (one row from the top)
<CA>	Align all following text centrally
<WTThis>	Write the text
<LN>	Move the cursor down one line
<WTis>	Write the text
<LN>	Move the cursor down one line
<WTcentred>	Write the text

```
This
 is
centred
```

See Also

LA	Left Align
NA	No Align
RA	Right Align
SW	Smart Wrap
TW	Text Wrap

<CC*n*> Check Code

Description Command terminator with 8-bit checksum

Parameters The parameter is an 8 bit checksum of all the characters in the preceding command string.
To calculate the value of the parameter, sum the ASCII values of all the characters in the command string up to but not including the <CC*n*> command. Divide this sum by 256 decimal (0x100 hex). The checksum is the remainder after the division and is sent as a single byte.

Modes Operational Mode 3 only

Notes This command is the signal to verify the checksum of the preceding command string and, if correct, action the commands.

If the checksum is not correct, no commands are actioned and an error response is returned.

The <CC*n*> command terminator is used in the most basic of the error checked modes, Operational Mode 3. If higher data security is required consider using Operational Mode 4 which has two parameters in the <CR*nn*> command terminator, representing a 16 bit CRC of the preceding command string.

Uses The <CC*n*> command allows:

- Commands to be queued but not actioned until required
- Basic message error checking

Example Assume the text display is in operational mode 3.

In order to clear the screen, a <CS> command must be sent followed by the <CC*n*> command where n represents the 8-bit sum of the characters "<", "C", "S", ">"

ASCII values of the example command are:

"<"	=	60 decimal	(3C hex)
"C"	=	67 decimal	(43 hex)
"S"	=	83 decimal	(53 hex)
">"	=	62 decimal	(3E hex)

In decimal notation:

The sum is 60+67+83+62 = 272 and the checksum is the remainder after division by 256.
Hence the checksum is remainder of the division 272/256 = 16

In hexadecimal notation:

The sum is 3C+43+53+3E =110 hex and the checksum is the Least Significant Byte (LSB) of this sum.
Hence the checksum is 10 hex.

The checksum must be sent as a single byte 16 decimal (10 hex).

Hence the full command string is: **<CS><CC[16]>**

Note! The square brackets are not sent, they are just there to emphasise that a single byte 16 is sent. The checksum, as in this case, may not be a printable ASCII character.

Gotchas! The checksum is always a single byte, and may be an unprintable character.

See Also

CI Command Implement
CR Cyclic Redundancy Check

<CE> Configuration Enable

Description Control access to the configuration menus

Parameters None

Initial Value This is the default

Modes All Modes

Notes The <CE> and <CP> commands control access to the main and quick access menus used for unit configuration.

The <CP> command will prevent user access to the configuration menus via the dual P-E and P-UpArrow key presses.

The <CE> command will re-enable user access to the menus.

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.

The commands have no effect on the LCD display screen.

The instrument sends a different response to commands from the host when it is in the programming menus.

Uses The <CE> command allows:

- changes to instrument configuration to be made after a <CP> Configuration Prohibit command

Example

<CP>	Lock out the menus
Any	A set of commands or operator instructions that should not be interrupted by adjustments to the display configuration be made
<CE>	Re-enable access to the menus for maintenance

There is no effect on the display LCD screen when these commands are used

See Also **CP** Configuration Prohibit

<CI> Command Implement

Description Command terminator without any checksum

Parameters None

Modes Operational Mode 2 only

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

It is used only in Operational Mode 2 where a string of commands can be "queued" and then actioned at the same time. This is essentially the same as the more complex Operational Modes 3 or 4, but without any error checking

Uses The <CI> command allows:

- Commands to be queued but not actioned until required
- Commands to be as actioned as quickly as possible as there is no error checking or responses to the host until all the commands have been actioned.
- Complex screens to be designed and tested just using a terminal emulator

Example Assume the display is in Operational Mode 2.

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

CC Check Code
CR Cyclic Redundancy Check

<CL*n*> Clear Line

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
F*n*	Font *n*

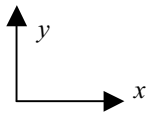
<CM*y,x*> Cursor Move

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 All Modes

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.

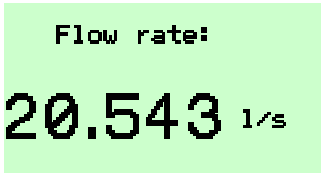
In Row mode this command is window aware. If a window is in use the parameters *y* and *x* are relative to the current window.

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
<WTl/s>	Write in units



Gotchas! The horizontal resolution is always 1 pixel.

See Also **DW** Define Window

<CP> Configuration Prohibit

Description Control access to the configuration menus

Parameters None

Initial Value <CE> Configuration Enable is active on power up

Modes All Modes

Notes The <CE> and <CP> commands control access to the main and quick access menus used for unit configuration.

The <CP> command will prevent user access to the configuration menus via the dual P-E and P-UpArrow key presses.

The <CE> command will re-enable user access to the menus.

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.

The commands have no effect on the display LCD screen.

The instrument sends a different response to commands from the host when it is in the programming menus.

Uses The <CP> command allows:

- 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

<CP>	Lock out the menus
< Anything >	A set of commands or operator instructions that should not be interrupted by adjustments to the display configuration be made
<CE>	Re-enable access to the menus for maintenance

There is no effect on the display LCD screen when these commands are used

See Also **CE** Configuration Enable

<CR*n,m*> Cyclic Redundancy Check

Description Command terminator with 16-bit checksum

Parameters The parameters *n* and *m* are two 8 bit bytes of the 16 bit checksum of all the characters in the preceding command string, *n* being the LSB, and *m* the MSB.

To calculate the value of the parameter, use the ASCII values of all the characters in the command string up to but not including the <CR*nm*> command. See the section on CRC Generation (page 10) .

Modes Operational Mode 4 only

Notes This command is the signal to verify the checksum of the preceding command string and, if correct, action the commands.

If the checksum is not correct, no commands are actioned and an error response is returned.

Uses The <CR*nm*> command allows:

- Commands to be queued but not actioned until required
- Rigorous message error checking

Example Assume the text display is in operational mode 4.

To clear the screen the following command needs to be sent:

<CS><CRnm>

The parameters n and m are calculated by running the ASCII value of the characters "<", "C", "S", ">" through the CRC algorithm. When this is done the CRC code generated is 0x8040

Hence n = 40 hex (64 decimal), m = 80 hex (128 decimal).

The full command is then:

<CS><CR[64][128]>

Note! The square brackets are not sent, they are just there to emphasise that n and m are single 8-bit bytes. The check bytes may not be printable ASCII characters.

The CRC code for the message <WTHello World> is 0x721B

Hence n = 1B hex (27 decimal), m = 72 hex (114 decimal).

The full command is then:

<WTHello World> <CR[27][114]>

Gotchas! Make sure that the section on CRC Generation (page 10) is read and understood!

See Also

CC Check Code
CI Command Implement

<CS> Clear Screen

Description Turn all pixels off, creating a blank screen

Parameters None

Initial Value All pixels are off

Modes All Modes

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 <CS> command provides:

- A known starting point before drawing a new screen

Example

```
<CS>
```

Clear Screen

Gotchas! 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

<CW> Clear Window

Description Turn all pixels off within a defined window

Parameters None

Initial Value All pixels are off

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

<DF*n*> Download Font

Description Download soft fonts to the display

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

Modes All Modes

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 <WS*n*> command and may be used in both Row and Pixel Modes. They may also be underlined and flashed using attributes, as any other character.

After the <DF*n*> command is issued, the display expects a download of a Windows 2-colour BMP file of the soft character. The required image size depends on the currently active font

Font:		Image Size (v x h):
	F1	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 download mechanism is identical to the <DG> Download Graphic and <DS> Download Screen commands. Detailed information is in the Graphics Transfer Section (Page 12).

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

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>	Clear Screen
<F5>	Set largest font size
<DF0>	Tell the display that a soft character number 0 (for Font 5) is going to be downloaded
Binary download of graphics file	Send a .BMP file of the required soft character to the display. In this case a 48 x 29 pixel image of a GBP symbol (£)
<WS0>	Write the soft character to the screen
<WT500>	Write normal text

Gotchas! Make sure that the section on Graphics Transfer (Page 12) is read and understood!

Font 5 characters cannot be saved to EEprom with the <KF> command

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

See Also

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

<DG> Download Graphic

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 response returned to the host.

The download mechanism is identical to the <DF*n*> Download Font and <DS> Download Screen commands. Detailed information is in the Graphics Transfer Section (Page 12).

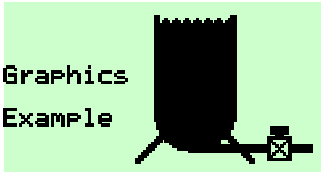
The cursor position is unchanged by this command.

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.
<WTExample>	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
<DG>	Tell the display to expect a graphics image download
Binary download of .BMP file	Download a 56 x 67 pixel image of a tank to the display



Gotchas! Make sure that the section on Graphics Transfer (Page 12) is read and understood

If any part of the graphic would be off-screen, then nothing is drawn and an error response is returned to the host

See Also

DS Download Screen
US Upload Screen

<DS> Download Screen

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

Parameters None

Modes All Modes
The <WM*n*> 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.

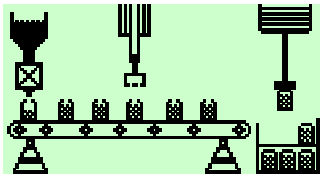
The download mechanism is identical to the <DF*n*> Download Font and <DG> Download Graphic commands. Detailed information is in the Graphics Transfer Section (Page 12).

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 <SL> Save Logo command

Example

<CS>	Clear Screen
<DS>	Tell the display to expect a 64 x 120 pixel graphics image that it should display full screen.
Binary download of .BMP file is now sent	
	Image is displayed when received

Gotchas! The logo is not stored in EEprom unless the <SL> Save Logo command is issued.

See Also

RL Restore Logo
SL Save Logo

<DW*yt*,*yb*,*xl*,*xr*> Define Window

Description Defines an area of the screen into which certain screen write commands are constrained

Parameters

yt = 0 to 7	- top row of the window area
yb = 0 to 7	- bottom row of the window area
xl = 0 to 119	- pixel column of the left hand side of the window
xr = 0 to 119	- pixel column of the right hand side of the window

Initial Value The initial window size is the full screen <DW0,7,0,119>

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:

- 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

<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)
<WTFlow:>	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

Subsequent values then only need:

<HC>	Home the cursor in the window area
<WT3.18>	Write out the new value

Gotchas! The <CS> Clear Screen and <PM> Pixel Mode commands remove any defined windows

See Also **CW** Clear Window

<EF> Enable Flashing

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

Parameters None

Initial Value Inhibited (No Flashing)

Modes All Modes

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

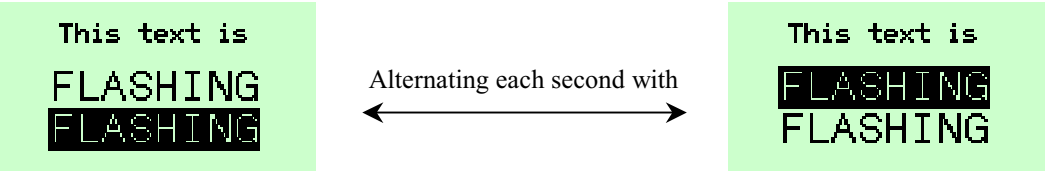
Uses The <EF> command allows:

- 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



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

<EL> Erase Line

Description Erase any text or graphics from the current cursor position to the end of the row

Parameters None

Modes Row Mode only

Notes The command erases a number of lines upwards from the current cursor position, depending on the current font:

For font 1 <F1> only one line is erased
For font 2 <F2> two lines are erased, and so on.

This command is window aware. If a window is in use the command erases only to the end of the window row.

Cursor position is unchanged by this command

Uses The <CLn> command allows:

- 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

<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
```

Gotchas! The current font size must be taken into account before issuing this command.

See Also

CL Clear Line
DW Define Window
F*n* Font *n*

<F1> Font 1

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 All Modes

Notes Font 1 is a single row font, each character being 8 pixels high by 6 pixels wide.
Font 1 does NOT have true decenders
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

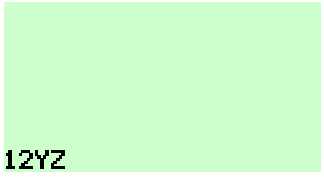
All fonts have a full 7-bit character set, except F5
All fonts have true decenders, except <F1>

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”



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

See Also

DF*n*	Download Font
WS*n*	Write Soft Character
WT	Write Text

<F2> Font 2

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 All Modes

Notes 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.

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 <F2> command allows:

- Improved readability over font 1

Example

<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

DF*n*	Download Font
WS*n*	Write Soft Character
WT	Write Text

<F3> Font 3

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 All Modes

Notes

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.

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

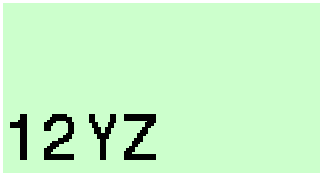
All fonts have a full 7-bit character set, except F5
All fonts have true decenders, except <F1>

Uses The <F3> command allows:

- Improved readability over font 2

Example

<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

DF*n*	Download Font
WS*n*	Write Soft Character
WT	Write Text

<F4> Font 4

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 All Modes

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

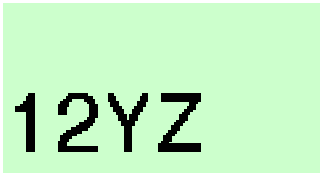
All fonts have a full 7-bit character set, except F5
All fonts have true decenders, except <F1>

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”



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

See Also

DF*n*	Download Font
WS*n*	Write Soft Character
WT	Write Text

<F5> Font 5

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 All Modes

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

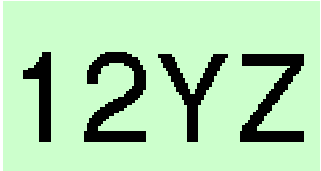
All fonts have a full 7-bit character set, except F5
All fonts have true decenders, except <F1>

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

DF*n*	Download Font
WS*n*	Write Soft Character
WT	Write Text

<FL> Flashing

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 All Modes

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.

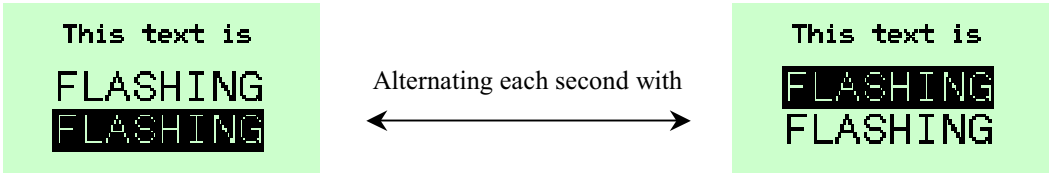
The Flashing attribute is cancelled by the <ST> STeady attribute.

Uses The <FL> command allows:

- 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

<FR> Font Restore

Description Recover previously stored soft fonts from EEprom

Parameters None

Modes All Modes

Notes This command recovers all the soft fonts in sizes F1 to F4 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.

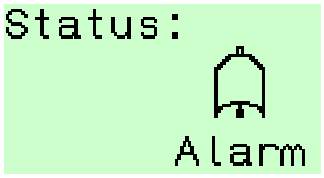
This command is used in conjunction with the Download Font <DF*n*> and Keep Fonts <KF> commands.

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



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

<FS> Fill Screen

Description Turn all pixels on, creating a black screen

Parameters None

Initial Value All pixels are off

Modes All Modes

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:

- 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

<FW> Fill Window

Description Turn all pixels on within a defined window

Parameters None

Initial Value All pixels are off

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 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

<HB*n*,*m*> Horizontal Bargraph

Description Draw a horizontal bargraph *n* pixels long with *m* pixels filled

Parameters
n = 3 to 120 - Length of bargraph
m = 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.

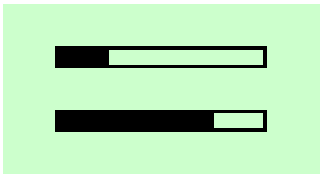
The cursor is restored to its original position after the command.

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:
- 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)



See Also **VB** Vertical Bargraph

<HC> Home Cursor

Description Return the cursor to the top left of the screen

Parameters None

Modes All Modes

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)

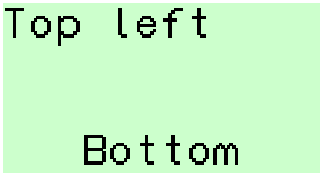
Home cursor is done automatically by commands such as <CS>, <FS>, <CW>, <FW> and setting any font size

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
F*n*	Font
FS	Fill Screen
FW	Fill Window

<HS*m,n,r,s,t,u,v*> Horizontal Scroll

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.
r = 0 to 7	- The last row to scroll
s = 0 to 64	- Starting position of line 1
t = 0 to 64	- Length of line 1
u = 0 to 64	- Starting position of line 2
v = 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 *s* and *u* define the starting positions of these two new lines, in pixels above the bottom of Row *r*. The length of these lines are defined by the corresponding parameters *t* and *v*, 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 *s,t,u,v* 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:

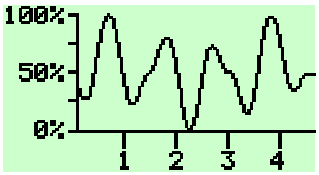
- Line, bar, block charts that scroll with time
- Visual effects

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

<IF> Inhibit Flashing

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 All Modes

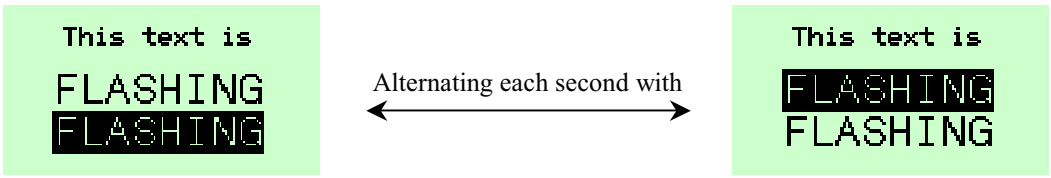
Notes This command acts on the whole screen.
Flashing can be re-enabled by using the <EF> command.

Uses The <IF> command allows:

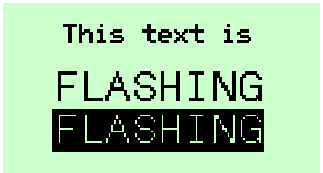
- A menu structure to be built up of flashing and static screens
- A simple method of acknowledging operator input

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



(Steady image)

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

<KF> Keep Fonts

Description Save previously download soft fonts (F1-F4) to EEprom

Parameters None

Modes All Modes

Notes This command causes all the soft fonts in font sizes F1 to F4 to be saved to EEprom. Due to space limitations, font size F5 soft characters cannot be saved or restored.

Downloaded soft fonts not stored in this way are lost when power is removed.

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.

The font data is written as a block and will overwrite any previously stored data.

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

Uses The <KF> command allows:

- A quicker method of providing soft fonts after power-on.

Example

<FR>	Get any existing font data
<F4>	Set required font size
<DF3>	Tell the display to expect a BMP file download of font data
Binary download of 32 x 19 pixel image	Send the file
<KF>	Save fonts

Gotchas! Font size F5 soft characters cannot be saved or restored

See Also

DF Download Font
FR Font Restore

<LA> Left Align

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 All Modes

Notes 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).

It only affects text written after the attribute has been set.

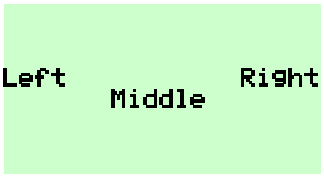
The attribute is cancelled by the <NA> command or any of the other text alignment commands <CA>, <RA>, <SW> & <TW>

Uses The <LA> command allows:

- Text to be automatically aligned without the need for cursor move commands
- Tidy screen presentation

Example

<SD>	Set the display to a known state
<CM3,60>	Move the cursor to the middle of row 3
<LA>	Set left align attribute
<WTLeft>	Left align the word ‘Left’ on the current row
<RA>	Set the right align attribute
<WTRight>	Right align the word ‘Right’ on the current row.
<LN>	Move cursor one row down
<CA>	Set centre align attribute
<WTMiddle>	The word ‘Middle’ is written centre aligned on the current row.



See Also

CA	Centre Align
NA	No Align
RA	Right Align
SW	Smart Wrap
TW	Text Wrap

<LF> Line Feed

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 This command causes the display to add a line feed character after a carriage return character has been received.

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.

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.

The <NL> command cancels this attribute, making LF and CR independent actions. <NL> is the default condition.

Uses The <LF> command allows:

- 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

<LH*x*,*l*> Line Horizontal

Description Draw a horizontal line *x* pixels long with a line thickness of *l*

Parameters
x = 1 to 120 - length
l = 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

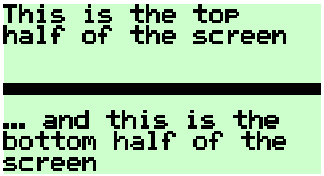
The parameters may be any value that will keep the line being drawn on-screen. If any part of the defined line is off-screen, then the line is not drawn and an error response it returned to the host.

Uses The <LH> command allows:

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

Example

<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



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.

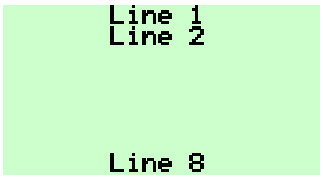
If the cursor is already on the bottom line the display will scroll up one line, leaving the cursor on the new bottom line.

Uses The <LN> command allows:

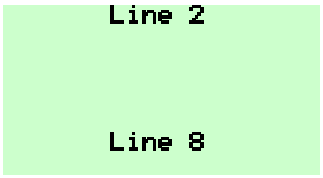
- A vertical scroll of text (and graphics) to occur if the cursor is already on the bottom line
- A quicker but more limited version of the Cursor Move command

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



<LN>	Move the cursor down one line



See Also **SW** Smart Wrap

<LV*y,l*> Line Vertical

Description Draw a vertical line *y* pixels high with a line thickness of *l*

Parameters
y = 1 to 64 - height
l = 1 to 120 - 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

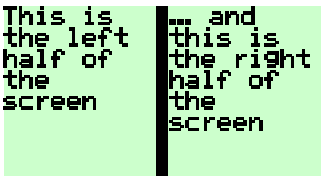
The parameters may be any value that will keep the line being drawn on-screen. If any part of the defined line is off-screen, then the line is not drawn and an error response it returned to the host.

Uses The <LV> command allows:

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

Example

<SD>	Set the display to a known state
<PM>	Set display to Pixel Mode
<CM63,58>	Move the cursor to pixel row 63, in the middle of the screen
<LV64,4>	Draw a vertical line 64 pixels long and 4 pixels wide
<RM>	Back to Row Mode
<DW0,7,0,57>	Define a window on the left half of the screen
<SW>	Turn Smart Wrap attribute on. Text wraps without splitting words
<HC>	Home the cursor to top left of window
<WTThis is the left half of the screen>	Write out some text
<DW0,7,63,119>	Define a window on the right half of the screen
<HC>	Home the cursor to the top left of the window
<WT … and this is the right half of the screen>	Write out some more text



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

<MC*n*> Make Connection

Description The following commands are intended for the instrument with address '*n*'

Parameters *n* = 1 to 47 - address range

Modes This command is used in multidrop or multiple instrument configurations

Notes Only the instrument with address '*n*' will acknowledge this command. Each instrument must have a unique address; commands cannot be 'broadcast' to several displays at once.

This command remains in force until cancelled by a <RC> Release Connection command. After an <RC> command has been confirmed by the currently active instrument, no instruments will respond to any commands until a further <MC*n*> command is sent to a valid instrument.

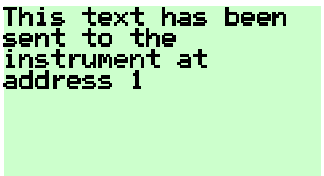
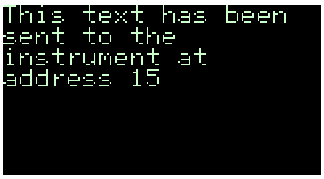
Multiple instrument configurations must send a valid <MC*n*> command when powered up as all instruments with a non-zero address will initially assume they are not 'connected'.

Single instrument configurations with address 0 will return an error response to this command.

Uses The <MC> command allows:

- Multiple instruments to be connected to one host port

Example

<MC1>	Make connect to the instrument with address 1
<CS>	Clear the screen on instrument address 1
<SW>	Set the smart wrap attribute on instrument address 1
<WTThis text has been sent to the instrument at address 1>	Send this text to the instrument with address 1
<RC>	Release the 'connection' to instrument 1
<MC15>	Make a 'connection' to the instrument with address 15
<WM3>	Set inverse write mode on the instrument at address 15
<FS>	Fill the screen on instrument address 15
<SW>	Set the smart wrap attribute on instrument address 15
<WTThis text has been sent to the instrument at address 15>	Send this text to the instrument with address 15
<RC>	Release the 'connection' to instrument 15
	Instrument address 1:
	Instrument address 15:

Gotchas! There is no such thing as a 'broadcast address'.

See Also **RC** Release Connection

<NA> No Align

Description Cancel all of the text alignment attributes <LA>, <RA>, <CA>, <SW> and <TW>

Parameters None

Initial Value This is the default

Modes All Modes

Notes This command clears all alignment attributes so that text written with the <WT> command appears at the current cursor position.

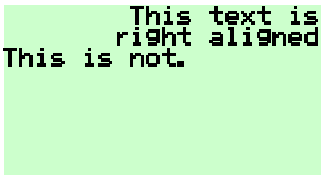
It only affects text written after the attribute has been set.

Uses The <NA> command allows:

- manual formatting after special alignment attributes have been used

Example

<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.



Gotchas! <NA> also cancels <SW> Smart Wrap and <TW> Text Wrap

See Also

CA	Centre Align
LA	Left Align
RA	Right Align
SW	Smart Wrap
TW	Text Wrap

<NL> No Linefeed

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 All Modes

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:

- The display to be used as a dumb terminal
- Hosts that send CR and LF separately to be accommodated

Example

<SD>	Set screen to known state
<LF>	Set Linefeed attribute on
<WTFirst line of text>	Write a line of text
<WT[CR]>	Send a [CR] character, with the Line Feed attribute set, this would be interpreted as [CR]+[LF] ***Note! The square brackets are not sent, they are just there to show that a Carriage Return character (ASCII 13) is sent.***
<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
<WTLast line of text>	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
```

See Also **LF** Line Feed

<NU> No Underline

Description Cancel the <UL> Underline attribute

Parameters None

Initial Value This is the default

Modes All Modes

Notes This command cancels the 'Underline' attribute so that text written with the <WT> command appears without being underlined

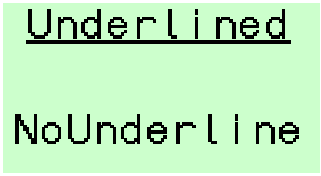
It only affects text written after the attribute has been set.

Uses The <NU> command allows:

- A combination of underlined and plain text to appear on the same screen

Example

<SD>	Set screen to known state
<CA>	Centre align the text
<UL>	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

<OD*n*> Output De-energised

Description Control the state of the output contacts, making it de-energised

Parameters *n* = 1 or 2 - Output number

Initial Value De-energised (open circuit) on power up

Modes All Modes

Notes These commands allow the user to control the state of the output contacts.
There are two isolated solid state contacts per display, A1 – A2 and A3 – A4

The parameter *n* selects which output is being controlled:
n = 1 controls the output A1-A2
n = 2 controls the output A3-A4

The command <OD*n*> turns off (de-energises) output *n*
The command <OE*n*> turns on (energises) output *n*

Uses The <OD> command allows:

- The display to control alarms, annunciators, sounders etc. under program control

Example

<OE1>	Output A1 – A2 is energised (short circuit)
<OE2>	Output A3 – A4 is energised (short circuit)
<OD1>	Output A1 – A2 is de-energised (open circuit)
<OD2>	Output A3 – A4 is de-energised (open circuit)

There is no effect on the display LCD screen when these commands are used

See Also **OE** Output Energised

<OE*n*> Output Energised

Description Control the state of the output contacts, making it energised

Parameters *n* = 1 or 2 - Output number

Initial Value De-energised (open circuit) on power up

Modes All Modes

Notes These commands allow the user to control the state of the output contacts.
There are two isolated solid state contacts per display, A1 – A2 and A3 – A4

The parameter *n* selects which output is being controlled:
n = 1 controls the output A1-A2
n = 2 controls the output A3-A4

The command <OE*n*> turns on (energises) output *n*
The command <OD*n*> turns off (de-energises) output *n*

Uses The <OE> command allows:

- The display to control alarms, annunciators, sounders etc. under program control

Example

<OE1>	Output A1 – A2 is energised (short circuit)
<OE2>	Output A3 – A4 is energised (short circuit)
<OD1>	Output A1 – A2 is de-energised (open circuit)
<OD2>	Output A3 – A4 is de-energised (open circuit)

There is no effect on the display LCD screen when these commands are used

See Also **OD** Output De-energised

<PM> Pixel Mode

Description Put the unit into Pixel Mode

Parameters None

Initial Value Row Mode

Modes All Operational Modes

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.

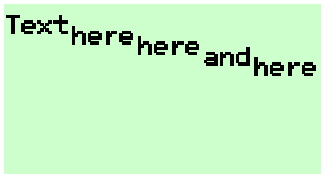
Pixel modes writes to the screen are always slower than the corresponding Row Mode write. It is recommended that Row Mode operations are used whenever possible to optimise the response time. Alternatively, complex screens can be written to the non-active frame and then made visible; This gives the appearance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

Uses The <PM> command allows:

- 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’



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

<RA> Right Align

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 All Modes

Notes 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.

It only affects text written after the attribute has been set.

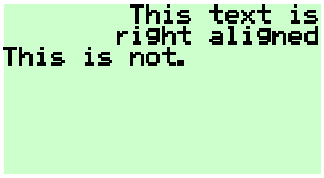
The command is cancelled by the <NA> command or any of the other text alignment commands <CA>, <LA>, <SW> & <TW>

Uses The <RA> command allows:

- 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

<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.



See Also

CA	Centre Align
LA	Left Align
NA	No Align
SW	Smart Wrap
TW	Text Wrap

<RB> Reboot

Description Cause a complete restart of the instrument, just as if it had been powered up

Parameters None

Modes All Modes

Notes This command causes a complete restart of the instrument, just as if it had been powered up after being switched off.

The receipt of this command is acknowledged in the normal way and then the instrument is restarted by causing a deliberate watchdog timeout.

This can be used to force a complete restart of the instrument, which may be needed if the host and display are independently powered.

An alternative is to issue the <SD> Screen Defaults command, which simply initialises the display to a known state

Uses The <RB> command allows:

- The entire instrument to be put into a known state

Example

<RB>	Reboot the instrument A delay of up to 2 seconds may elapse before the watchdog timeout restarts the hardware
	The unit reads its configuration settings from EEprom and displays the power-on logo

Gotchas! Soft fonts must be restored with the <FR> Font Restore command after a reboot

See Also **SD** Screen Default

<RC> Release Connection

Description Disconnect the currently 'connected' instrument

Parameters None

Initial Value All instruments with non-zero addresses power up with no connection active

Modes This command is used in multidrop or multiple instrument configurations

Notes After an <RC> command has been confirmed by the currently active instrument, no instruments will respond to any commands until a further <MC*n*> command is sent to a valid instrument.

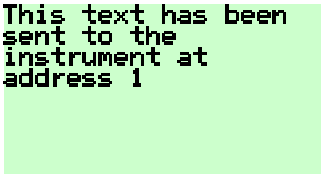
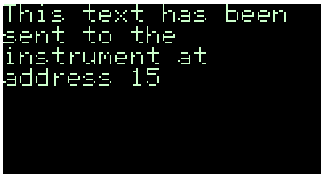
Multiple instrument configurations must send a valid <MC*n*> command when powered up as all instruments with a non-zero address will initially assume they are not 'connected'.

Single instrument configurations with address 0 will return an error response to this command.

Uses The <RC> command allows:

- Multiple instruments to be connected to one host port

Example

<MC1>	Make connect to the instrument with address 1
<CS>	Clear the screen on instrument address 1
<SW>	Set the smart wrap attribute on instrument address 1
<WTThis text has been sent to the instrument at address 1>	Send this text to the instrument with address 1
<RC>	Release the 'connection' to instrument 1
<MC15>	Make a 'connection' to the instrument with address 15
<WM3>	Set inverse write mode on the instrument at address 15
<FS>	Fill the screen on instrument address 15
<SW>	Set the smart wrap attribute on instrument address 15
<WTThis text has been sent to the instrument at address 15>	Send this text to the instrument with address 15
<RC>	Release the 'connection' to instrument 15
	Instrument address 1:
	Instrument address 15:

See Also **MC** Make Connection

<RF*m*> Restore Frame

Description Restore a previously saved frame to the currently active frame

Parameters *m* = 0 to 2 - Saved Frame memory location

Modes The <WM*n*> 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.

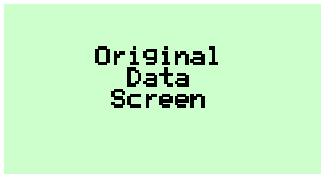
The parameter *m* specifies which memory location the stored frame is recovered from:
m = 0 specifies EEprom area 0
m = 1 specifies EEprom area 1
m = 2 specifies the scratchpad area in RAM

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:

- Images to be transferred from one frame to another
- Any data screen to be flashed using the sequence <SF0,2><FL><EF><BM2><RF2>

Example



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

```
<SF0,2>
<CS>
<LN>
<WTImportant>
<LN>
<WTMessage>
<CM7,0>
<WTAny key to
confirm>
```

Save frame 0 to scratchpad RAM
Clear screen for new message
Move down a line
Write out a message….


```
<RS>
```

Wait for an operator response.
N.B. Loop sending this command until either a response or timeout

```
<RF2>
```

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

<RL*n*> Restore Logo

Description Restore a logo that has been saved with the <SL> command

Parameters *n* = 0 or 1 - Static or Scrolled

Initial Value There is a default "BEKA associates" logo built in that appears if no user logo is defined.
(OEM versions of the product may have an alternative logo instead)

Modes All Modes

Notes This command is used to restore a logo that has been saved with the <SL> command. The parameter *n* specifies whether the logo scrolls, as on power up.
n = 0 specifies no scrolling
n = 1 specifies logo should scroll
Scrolling will start after 20 seconds and pause for 10 seconds between each screen scroll

If there is no saved logo, this command will restore the default BEKA logo.

The logo is always recovered to the current visible frame, overwriting the frame contents. Note this command is the only command that does not write to the current active frame.

Uses The <RL> command allows:

- A customised logo to appear if the system is not being used, or there have been no messages for a period of time
- A scrolling logo to reassure operators that the display is still functioning correctly, without any host programming

Example

```
<RL1>
```

Display the logo with scrolling enabled

Image is displayed when received

Gotchas! This command is the only one that does not write to the currently active frame.

See Also

DS Download Screen
RF Restore Frame
SL Save Logo

<RM> Row Mode

Description Put the unit into Row Mode

Parameters None

Initial Value Row Mode

Modes All Operational Modes

Notes 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

In this mode the vertical position in the Cursor Move command it limited to 0 to 7.

Windows are available in Row Mode to constrain and align text.

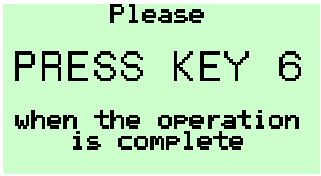
Writes to the display in Row Mode are always faster than Pixel Mode operation, and should be used wherever possible

Uses The <RM> command allows:

- Rapid display of text messages
- Simple text alignment

Example

<CS>	Clear screen for new message
<RM>	Set Row Mode
<CA>	Centre align the text
<WTPlease>	Write out message….
<F2>	Use a larger font size
<CM3,0>	Move the cursor to row 3
<WTPRESS KEY 6>	Write more text
<F1>	Back to the small font
<CM5,0>	Move the cursor to row 5
<WTwhen the operation>	Write out more text
<LN>	Next line down
<WT is complete>	Write out final line of text



See Also

PM Pixel Mode
DW Define Window

<RS> Request Status

Description Get key-press status information from the display

Parameters None

Initial Value None

Modes All Modes

Notes This command is used to get key-press status information from the display. It has no effect on the screen or any of the display settings.

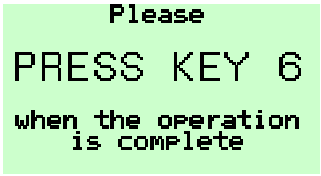
This command was included primarily to be able to read the keys in Operational Mode 0, where there is not normally a response to commands. However, it works in any mode and can be used in a loop waiting for an operator key-press.

Uses The <RS> command allows:

- Operator feedback
- The only method for checking the keys in Operational Mode 0

Example

<CS>	Clear screen for new message
<RM>	Set Row Mode
<CA>	Centre align the text
<WTPlease>	Write out message….
<F2>	Use a larger font size
<CM3,0>	Move the cursor to row 3
<WTPRESS KEY 6>	Write more text
<F1>	Back to the small font
<CM5,0>	Move the cursor to row 5
<WTwhen the operation>	Write out more text
<LN>	Next line down
<WT is complete>	Write out final line of text



<RS>	Wait for an operator response. N.B. Loop sending this command until either a response or timeout

See Also

CI Command Implement
CC Check Code
CR Cyclic Redundancy check

<SB*n*> Set Backlight

Description Alter the intensity of the backlight

Parameters *n* = 0 to 40 - Backlight Intensity

Initial Value Dependant on setting made in configuration menus

Modes All Modes

Notes This command alters the intensity of the backlight depending on the parameter *n*:

n = 0 backlight off.
n = 40 backlight fully on.

The actual brightness of the backlight depends on the single/multiple unit configuration. See the instruction manual for further information

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

The <RB> ReBoot command restores the backlight to the default value as part of the initialisation process

Uses The <SB> command allows:

- The backlight to be flashed to attract attention
- Panel illumination to be controlled by the host

Example

<SB0>	Turn the backlight off
<SB40>	Turn the backlight to full intensity

Gotchas! The current backlight intensity cannot be read back from the display, nor can the defaults be changed by the host

See Also **RB** ReBoot

<SD> Screen Defaults

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

Parameters None

Initial Value This is the default at power up

Modes All Modes

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"
```

Gotchas! Use the <CS> Clear Screen for a less drastic initialisation

See Also

CS	Clear Screen
RB	ReBoot

<SF*n,m*> Save Frame

Description Save the specified frame *n* to memory location *m*

Parameters
n = 0 or 1 - frame number
m = 0 to 2 - memory location

Initial Value None

Modes All Modes

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

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

If more non-volatile frame storage is required, the <SL> Save Logo command can be used, but a frame saved using this command is automatically displayed on power-on.

The scratchpad RAM area is also used by the following commands:
<DL><DG><RB><SL><RL><LH><LV><BD><DF>
Use of any of these commands will corrupt a saved image in scratchpad ram.

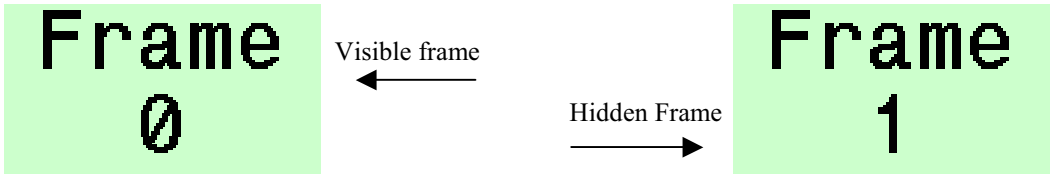
Detailed information about the use of frames can be found in the Frames Section (Page 6).

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

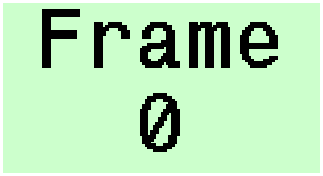
<SD>	Start with the active frame and visible frame set to 0
<CS>	Clear frame 0
<F4>	Set the required font
<CA>	Let the display centre the text automatically
<WTFrame>	Write out the word “Frame”
<LN>	Down a row
<WT0>	Write out the number “0”
<AF1>	Switch to the hidden frame
<CS>	Clear the hidden frame
<WTFrame>	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
<SF1,2>	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



<RF2> 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 6) is read and understood

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

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

See Also

RF Restore Frame
VF Visible Frame

<SL> Save Logo

Description Save the currently visible frame contents as the power-on logo

Parameters None

Initial Value There is a default "BEKA associates" logo built in that appears if no user logo is defined.
(OEM versions of the product may have an alternative logo instead)

Modes All Modes

Notes The screen may be drawn using the text and graphics commands or simply downloaded from the host as a .BMP file using <DS> or <DG>

A saved logo can be overwritten at any time by issuing another <SL> command.

If a user logo is no longer required, then clear the screen and execute the <SL> command. This will restore the default BEKA logo

Uses The <SL> command allows:

- A customised logo to appear at power on

Example

<CS>	Clear Screen
<DS>	Tell the display to expect a 64 x 120 pixel graphics image that it should display full screen.
Binary download of .BMP file is now sent	
	Image is displayed when received
<SL>	Save Logo to EEprom
<RB>	ReBoot the display
	User logo is shown (and scrolled) on power up
<CS>	Clear Screen
<SL>	Save Logo to EEprom
<RB>	ReBoot the display
	Default BEKA logo is shown (and scrolled) on power up

See Also

DS Download Screen
DG Download Graphic

<ST> Steady

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

Parameters None

Initial Value Steady (No Flashing)

Modes All Modes

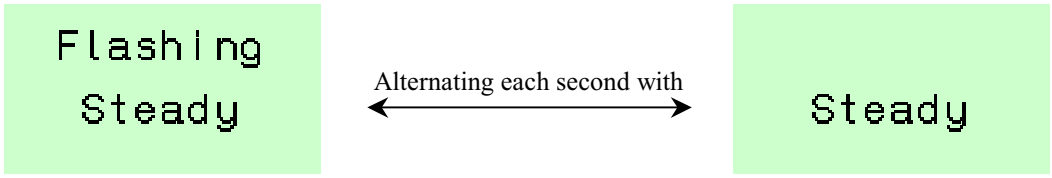
Notes

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
<WTFlashing>	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

<SW> Smart Wrap

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 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.

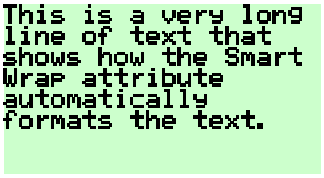
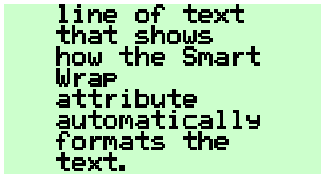
Smart Wrap is a text alignment attribute that cannot be used in conjunction with any other alignment command <CA>,<LA>, <RA> or <TW>. It is cancelled by the <NA> command.

<SW> can be used with either the full screen, or within a window.

Uses The <SW> command allows:

- Simple formatting of text strings

Example

`<SD>`	Set the display to a known state
`<SW>`	Set the Smart Wrap
`<WTThis is a very long line of text that shows how the Smart Wrap attribute automatically formats the text.>`	Write a lot of text. It all fits on screen
	
`<CS>`	Clear the screen
`<DW0,7,20,100>`	Define a window
`<WTThis is a very long line of text that shows how the Smart Wrap attribute automatically formats the text.>`	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.
	Note that the display has scrolled

See Also

CA	Centre Align
LA	Left Align
NA	No Align
RA	Right Align
TW	Text Wrap

<TO*n*> Time Out

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 All Modes

Notes This command activates a timer that warns via a screen message that there has been no communication from the host for a defined period of time.

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 with a correct checksum (if used) must be received and acknowledged by the display. In a multidrop application, each individual display must be communicated to within their timeout period.

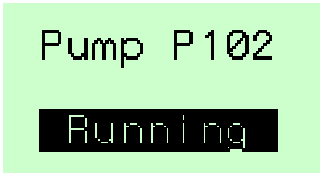
Uses The <TO> command allows:

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

Example

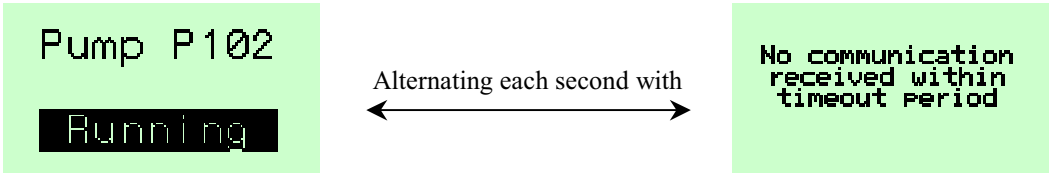
`<TO2>` Sets a timeout period of 2 x 10 = 20 seconds

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.



Gotchas! In normal operation, make sure that the host communicates at least once every timeout period. The <RS> Request Status command may be used for this purpose

See Also

RB ReBoot
RS Request Status

<TW> Text Wrap

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 Row Mode only

Notes 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.

Text written off the end of the bottom line will cause the screen to scroll.

The Text Wrap attribute may be used with the whole screen or constrained within a window.

It is cancelled by the <NA> No Align command.

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

Uses The <TW> command allows:

- Strings can be sent without worrying about their length
- Maximum visible message size, albeit with poor formatting

Example

<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
This text exceeds th e line length	Note that all the text is displayed without an error being returned to the host, but the word “the” is split on to two lines.
<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
This is a long line e line length	This time the first line is overwritten, but the second line is not because the text has not been wrapped. Also, an error response is returned to the host to indicate that the write command failed

Gotchas! If text needs to wrap, but without splitting words, use the <SW> attribute instead.

See Also

NA No Align
SW Smart Wrap

<UE> Upload Enable

Description Enables the use of the Upload Screen <US> command

Parameters None

Initial Value Not enabled. <US> command will return an error unless preceded by <UE>

Modes All Modes

Notes This command enables the use of the Upload Screen <US> command, and must be sent immediately prior to that command.

The Upload Screen <US> command is the only command that uploads data from the display, so this enable command is included to prevent accidental use of the <US> command which would disrupt normal communications for a few seconds.

Uses The <UE><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

```
<UE>
<US>
```

Bitmap file of screen image is returned to host

Gotchas! In Operational Modes greater than 0, command responses and checksums will surround the data

See Also **US** Upload Screen

<UL> UnderLine

Description Set the Underline attribute, so that any subsequently written text is underlined.

Parameters None

Initial Value <NU> No Underline

Modes All Modes

Notes Once this attribute has been set, any text written in Fonts 2 to 5 are underlined in the decender area of the font. As Font 1 does not have decenders, this attribute is not recognised. If Font1 text really does need to be underlined, use a line draw command <LH> in pixel mode.

Characters defined in the soft fonts are also underlined using this command. This should be born in mind when defining the characters.

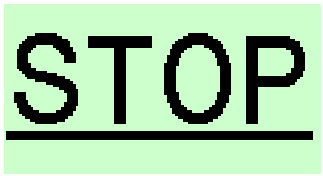
The Underline attribute is cancelled with the <NU> command.

Uses The <UL> command allow:

- Attention to be focussed onto certain text
- Screen presentation to be improved by the use of headings

Example

```
<SD>        Set the display in to a known state
<F5>        Maximum font size
<CM6,0>     Down to row 6
<CA>        Set centre align attribute
<UL>        Set underline attribute
<WTSTOP>    Write the message
```



Gotchas! Font 1 cannot be underlined using this method

See Also **US** Upload Screen

<US> Upload Screen

Description Upload the current screen contents to the host.

Parameters None

Initial Value Not enabled. <US> command will return an error unless preceded by <UE>

Modes All Modes

Notes Detailed information about the upload procedure is in the Graphics Transfer Section (Page 12).

The <US> command is acknowledged in the normal way. After a short gap (500ms), a 1086 byte block of data is sent to the host. A command acknowledge then follows with the check bytes as per the current operational mode. The check bytes include the data block bytes and the acknowledge, but not the check bytes themselves. The 1086 byte data block, saved to file is a graphics image of the screen in 2-colour Windows .BMP format

This command requires the Upload Enable <UE> command to be sent immediately prior to it.

Uses The <UE><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

```
<UE>
<US>
```

Bitmap file of screen image is returned to host

Gotchas! In Operational Modes greater than 0, command responses and checksums will surround the data

See Also **UE** Upload Enable

<VB*n*,*m*> Vertical Bargraph

Description Draw a vertical bargraph *n* pixels high with *m* pixels filled

Parameters
n = 0 to 64 - Height of bargraph
m = 0 to *n* - Number of filled pixels, starting from the bottom

Modes
Row Mode only
The <WM*n*> Write Mode has no effect on this command

Notes
The vertical 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 <VB60,0> and <VB60,1> are visually identical, as are <VB60,59> and <VB60,60>

Uses
The <VB> command allows:
- Simple graphical representation of values or progress
- Bargraphs to be combined without restriction with other text and graphics

Example

<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

See Also
HB Horizontal Bargraph

<VF*n*> Visible Frame

Description Page frame *n* is made visible

Parameters *n* = 0 or 1 - frame number

Initial Value 0

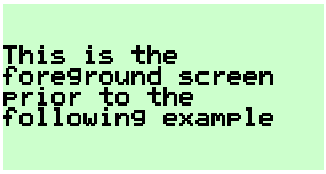
Modes All Modes

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.

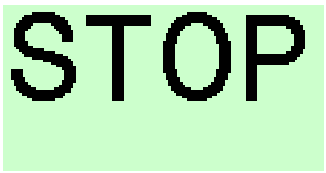
Uses The <VF*n*> command allows

- 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



<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



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 <AFn> command is issued appropriately

See Also

AF Active Frame
RF Restore Frame

<WM*n*> Write Mode

Description Determine how text or graphics is drawn on the screen

Parameters *n* = 0 to 3 - mode number

Modes All Modes

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

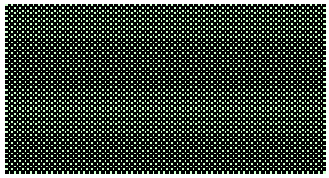
Detailed information is in the Display Features Section (Page 4)

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

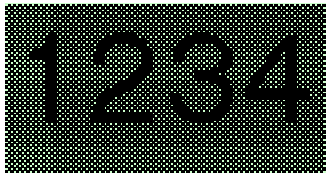
```
Original screen
```



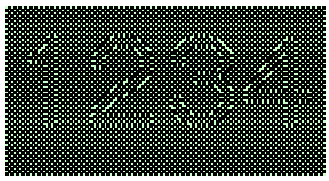
The following examples show the effect of writing the text '1234' in font 5 on a chequer-board background for the 4 write modes:

```
<WM0>
```


```
<WM1>
```




```
<WM2>
```



```
<WM3>
```

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

See Also **BM** Background Mode

<WS*n*> Write Soft character

Description Write the soft character number *n* of the current font at the current cursor position

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

Modes All Modes

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

The soft character written assume all the current attributes, just as any normal character.

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.

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.

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

Uses The <WS> command allows:

- Any special character to be written to the screen just like any other character

Example

<CS>	Clear Screen
<F5>	Set largest font size
<DF0>	Tell the display that a soft character number 0 (for Font 5) is going to be downloaded
Binary download of graphics file	Send a .BMP file of the required soft character to the display. In this case a 48 x 29 pixel image of a GBP symbol (£)
<WS0>	Write the soft character to the screen
<WT500>	Write normal text

See Also

KF Keep Font
FR Font Restore

<WTmessage> Write Text

Description Write text to the display, using any set attributes

Parameters *message* = any 7-bit ASCII string

Initial Value None

Modes All Modes

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

This command can be used in any mode, but it must be used in Operational Modes 2 to 4 in order to write text to the screen

Free text can be written to the screen in modes 0 and 1, but it is not confirmed and cannot be formatted

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

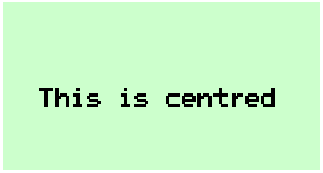
Text that exceeds the line length without either the <TW> or <SW> attributes set will not be written to the screen and an error is returned to the host.

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
<WTThis is centred>	Write the message



Gotchas! Font 5 has a limited character set

See Also **WS** Write Soft Character