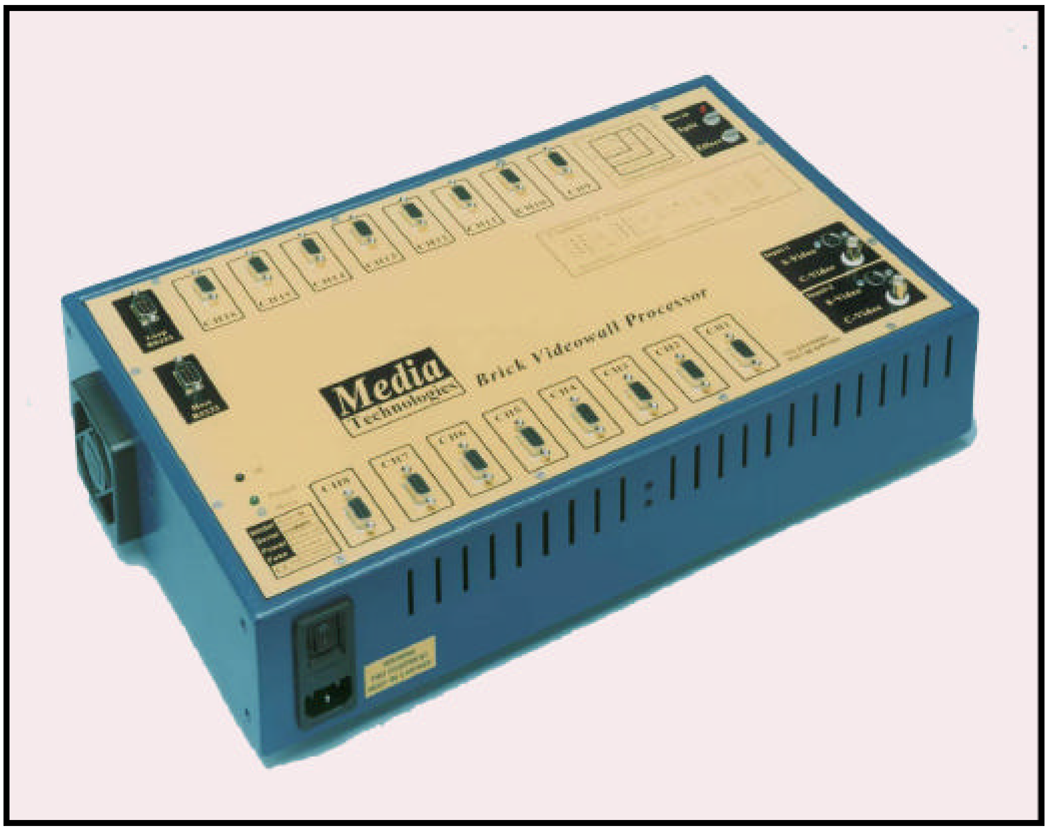


Brick Videowall Processor

USER MANUAL

Media

Technologies

For further information on our products, please contact us:

Media Technologies
361 Mill Road
Colchester
Essex, CO4 5GG
United Kingdom

Telephone	:	+44 7885 256665
Facsimile	:	+44 1 206 752451
Email	:	sales@mediat.co.uk
Internet	:	http://www.mediat.co.uk

CONTENTS

Page

1 PROCESSOR ARCHITECTURE 4

2 RS232 LINK PROTOCOL 5

PHYSICAL CONNECTION 5
FRAME STRUCTURE 5

3 CONNECTING THE PROCESSOR 8

4 EFFECTS COMMANDS 10

SYSTEM COMMANDS 10
VIDEO INPUT COMMANDS 12
VIDEO OUTPUT COMMANDS 14
EFFECTS COMMANDS 15
TIMEBASE CORRECTOR (TBC) FUNCTIONS 20
SEQUENCE EXECUTION FUNCTIONS 21
GRAPHICS FUNCTIONS 22

5 CONTROL SOFTWARE 27

COMPUTER REQUIREMENTS 27
INSTALLING THE SOFTWARE 27
PROGRAM EXECUTION 28
SEQUENCE FILE EXECUTION 28

6 SOFTWARE USER GUIDE 29

QUIT 29
HELP 29
INITIAL MENU 29
VIDEO MENU 29
SPLIT MENU 31
PROJECT MENU 31
SEQUENCE MENU 31
CONFIGURE MENU 34
ALIGNMENT MENU 36

7 CREATING SEQUENCES 38

SEQUENCE FILE FORMAT 38
COMMAND WORDS 38

8 DOS COMMAND LINE EXECUTION 46

COMMAND LINE INVOCATION 46

9 FLASH MEMORY UTILITY 48

FLASH MEMORY COMMANDS 48

10 LOW LEVEL MONITOR 1

BOOT PROCESS 1
FLASH FILE SYSTEM RECOVERY 1

11 SPECIFICATION 2

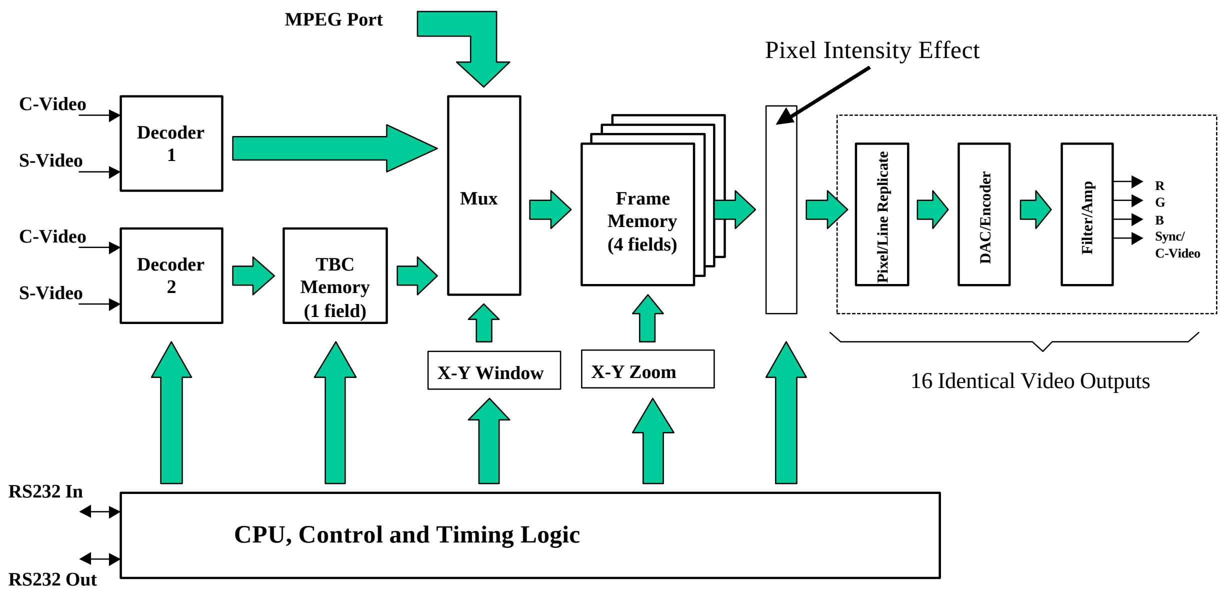
1 PROCESSOR ARCHITECTURE

The brick videowall processor architecture is shown below.

The Brick consists of two independent video digitisers, the second video digitiser has a field memory to provide synchronisation with the first (primary) video digitiser. Video fields are streamed into the video frame memory which can store up to four video fields. The buffer is configured to enable pixels to be read independently of being written to provide pixel mapping to the output replicators which carry out the X-Y magnification. The output of the frame buffer is fed to the 16 video output processors which magnify the portion of the image to be displayed on the particular monitor in the videowall array.

Various pixel switching and pixel intensity effects are provided for in hardware. The video output processors include colour wash registers to enable each monitor in the videowall array to display a different colour wash. These colour washes block the display of video and can be dynamically switched to enable wipes between wash and video.

A unique feature of the Brick is the ability of the powerful on-board CPU to write directly into the video memory to enable on screen messages and simple graphics to be displayed without any additional hardware. Bitmaps can also be downloaded and displayed to enable for example your logo to be called up on the videowall array.



Brick Videowall Processor Architecture

Communications with the processor are via a simple RS232 interface.

2 RS232 LINK PROTOCOL

Communications with the Brick is carried out via the RS232 interface. The protocol is designed to be robust and reliable and is based on the well know HDLC technique. It is byte transparent, that is, any 8-bit value can be sent to the processor using the framing characters. A simple single character command mode is supported to enable pre-loaded effects sequences to be triggered using (for example) single keyboard characters in conjunction with a standard terminal emulator such as that provided with Windows^{TM}.

PHYSICAL CONNECTION

The wiring of the cable for connection between a PC/AT 9-pin 'D' serial connector and the Processor is as follows:

Computer		**Videowall Processor**
Pin **2**	----------------	Pin **2**
Pin **3**	----------------	Pin **3**
Pin **5**	----------------	Pin **5**

The protocol is intended to be used with RS232 serial communications devices. The default communications parameters are as follows:

Speed	19200 baud
Data bits	8
Stop Bits	1
Parity	None

FRAME STRUCTURE

The frame structure is common to both directions of transmission. Messages are delimited by start and end of message flags. A single byte checksum is appended to the end of the message. The message is preceded by two bytes of additional data, a control and address byte as shown below:

<SOM> , <Control>, <Address>, [<Message data...>], <Checksum>, <EOM>

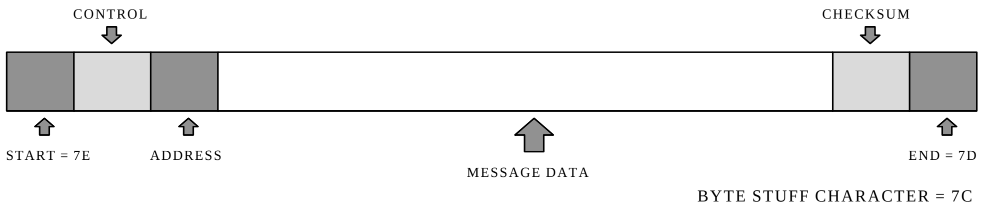
Frames are delimited by a start and end of message character. The body of the frame comprises the control and address bytes the message (which is optional), and a checksum. Frames not containing any message bytes are referred to as 'null' frames.

The start and end of messages are special character values which must not occur in the data bytes between them to ensure correct message framing. To achieve this a third value, the byte stuff character, (BSF) is used. This is also a unique character value which also must not appear as a data byte.

The byte values of these three special characters are:

SOM	=	0x7E
EOM	=	0x7D
BSF	=	0x7C

Byte transparency is achieved by replacing any of the three special characters which occur as data bytes with a two byte sequence headed by the BSF character. The second character is the character X-ORed with the 'smudge' character, 0x20, which prevents recognition by the receiver as a special byte. e.g. a data byte 0x7E is replaced with the byte pair: 0x7C,0x5E.



RS232 Link Protocol Frame Format

CONTROL AND ADDRESS BYTES

The control byte comprises two fields. The upper 4 bits are mode bits controlling the using of the addressing information. The lower 4 bits are the Device Address. In the receive direction (Processor to PC) the mode bits are not used. (Unused bits are marked in the table below with an 'x') As a convention, unused bits should be set to zero, but receiving software should not reject messages if they are not.

Transmit Bits							
7	6	5	4	3	2	1	0
M1	M0	x	x	A3	A2	A1	A0

Receive Bits							
7	6	5	4	3	2	1	0
x	x	x	x	A3	A2	A1	A0

The following Device Address code has been allocated to the Brick Videowall processor:

Device Address [A3 to A0]	Device Type
0 - 6	RESERVED
7	Brick Processor

The two mode bits M1 and M0 allow for 4 addressing modes:

M1	M0	Card Type
0	0	Normal Address
0	1	Response (Poll)
1	0	Global of type (A3 to A0)
1	1	Global all units.

NORMAL ADDRESS MODE

In this mode, the address byte is used to select a specific device (one of 256), of the type set by the device address field in the control byte. Only one unique device will therefore act on the message.

RESPONSE (POLL) MODE

The message will be received by the addressed device in the same manner as the Normal Address mode, however the addressed device will respond with a return frame. If there is no message pending in the addressed device the response will be a null frame.

GLOBAL TYPE ADDRESS MODE

All devices of the type defined in the control bytes unit field will act upon this message. It is not possible to globally poll devices since more than one would respond and the messages collide. In this mode the address byte is redundant, but it is still sent to maintain a simple frame structure. By convention the address byte should be zero in this case.

GLOBAL ADDRESS MODE

This is similar to the previous case except that all devices of any type will act on the message. The usefulness of this mode is really restricted to generic commands such as 'RESET' and 'BAUDRATE'

CHECKSUM

The checksum is calculated by summing modulo-256, all of the bytes between the start and end of message characters, prior to the byte stuffing process, and subtracting this figure from zero. The receiver checks the received data by summing all the bytes following the start of message (after expanding the 'stuffed' bytes). When the end of message character is received the sum should be zero.

MESSAGE CONTENTS

The length of the message is not limited by the protocol. As described above, it can be of zero length for use in polling and null responses, where there is otherwise no data to transfer. The message format is the same for both directions of transmission, comprising a function code in the range 0 to 255 followed by an optional parameter list. The convention for parameters greater then 255 is least significant byte first.

Function codes are device specific and detailed in separate manuals for each device. The function codes from 0 to 15 are reserved for generic operations, such as returning software version numbers, which all devices support. In addition there are generic function codes to support extended addressing techniques for selecting multiple devices.

PROTOCOL VIOLATIONS

Any protocol violations should cause the current message to be abandoned. The receiver will revert to a mode waiting for the start of message character. If the violation was an out of sequence start of message character, a new message will be assumed. Message frames with checksum errors are ignored.

Although the BSF character would normally only precede a 'smudged' version of one of the 3 special byte codes, (SOM, EOM, BSF) the presence of any other character is not considered a violation. The 'un-smudging' process should be applied regardless of the 'smudged' character code provided it is not either SOM, EOM or BSF.

USING SEND TO DECODE MESSAGE FRAMES

Software shipped with Media Technologies videowall processors includes a message programme called SEND.EXE.

Type SEND at the DOS prompt to display the command line switches available. Use the following command to see what bytes are transmitted by a given videowall command sequence:

```
SEND EXAMPLE.SEQ -COM1: -L
```

Instead of "EXAMPLE.SEQ" use any example sequence file assembled using the videowall programming software.

This creates a text file called TX.LOG which contains information about the bytes sent to the serial port.

Use COM2: if necessary for your installation.

3 CONNECTING THE PROCESSOR

The physical layout of Brick connectors is shown below.

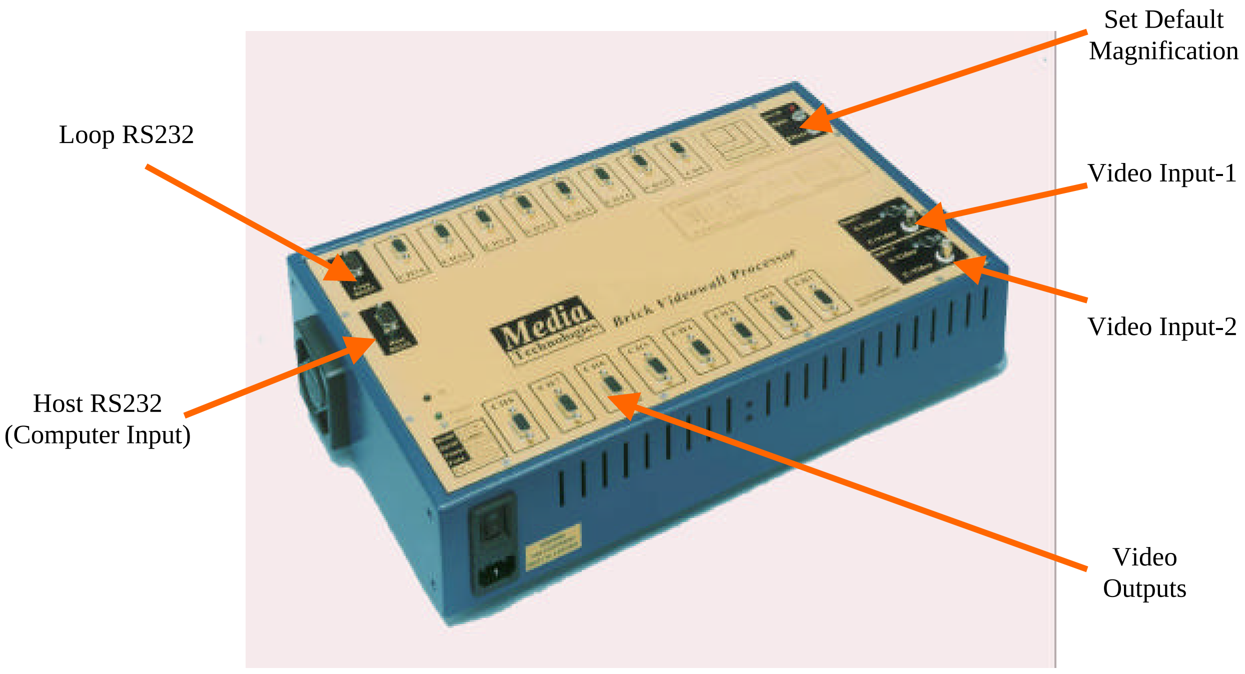


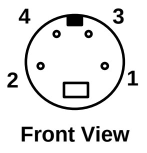
Figure-2 : Layout of Processor Connectors

The Brick videowall processor is controlled via the Host RS232 connector. The connector is a standard 9-pin 'D' connector with a one-one pin mapping to the standard IBM PC COM port connectors - only 3 wires are required for correct operation. The Loop RS232 connector is only required when other RS232 devices, typically stacked Brick processors need to be controlled from a single PC COM port. The Loop RS232 connector is a standard 9-pin 'D' connector with a crossover pin mapping relative to Host RS232 connector, to enable units to be daisy chained (Loop RS232 <-> Host RS232, etc.).

Pin assignments of all the connectors are given below.

S-Video input connector pin configuration [4-pin Mini DIN]

Pin	Signal	Type
1	'Y' Screen / GND	-
2	'C' Screen / GND	-
3	'Y' input	Input
4	'C' input	Input



Video output connector pin configuration [9-pin Female 'D']

Pin	Signal	Type
1	**Red**	Output
2	**Green**	Output
3	**Blue**	Output
4	Dual Function - i) Composite Video ii) RGB : Negative Composite Sync	Output
5 - 9	**GND**	-

Host RS232 connector pin configuration [9-pin Male 'D']

Pin	Signal	Type
1	Not used	
2	**Host-TX**	Output
3	**Host-RX**	Input
4	Not used	
5	**GND**	-
6 - 9	Not used	

Loop RS232 connector pin configuration [9-pin Male 'D']

Pin	Signal	Type
1	Not used	
2	**Host-RX**	Input
3	**Host-TX**	Output
4	Not used	
5	**GND**	-
6 - 9	Not used	

4 EFFECTS COMMANDS

This section is for users who wish to create advanced video sequences. Simple configurations of magnifications and colour washes can be achieved directly from the control software supplied without the need to understand these commands. Examples are provided on the distribution disk which should be consulted to gain a further insight into the application of these commands.

Function codes are used in messages with either the BRICK.EXE or SEND.EXE programs. The functions may have a number of parameters and are preceded by the address of the device to which they are directed, e.g. for the Brick processor.

```
send 7 0 FN_CODE p1 .. pn
```

send	: Builds and sends a message to the selected device.
7	: The device type for the BRICK processor
0	: The sub-address of the device type.
FN_CODE	: Command code, 0 to 255 binary
p1 .. pn	: Optional parameters

For example, to set the global split to 4x4

```
send 7 0 40 4
```

Where 40 is the function code to set a split factor.

Some functions require 16 bit parameters. These can be created by preceding them with the '&' character, e.g. to position the graphics cursor:

```
send 7 0 85 &100 &100
```

String parameters are required by some functions. These are terminated with a zero character which must be done explicitly using send. For example:

```
send device address FN_CODE "string" 0
```

Commands and parameter names in CAPITALS are the function names used in sequence programme scripts, they are defined in the system file "STD.INC". It is possible (but not recommended) to edit "STD.INC" to customise the names for function commands, for example to substitute words in another language.

For example, the sample command above to set the split has been defined as follows to make it easier to use.

```
SETSPLIT 4
```

The following function codes are available to the user. Other codes not specified here are either reserved or not intended to be directly accessible by the user.

SYSTEM COMMANDS

RESET

Resets the addressed device. All internal parameters are set to their power on reset state. The device configuration is derived from a combination of front panel switches and configurations stored in on-board Flash memory.

Parameters

Byte	Parameter	Comment
Code	**RESET**	Function code = 00

BAUDRATE

Sets the baud rate for the addressed device serial communications. On reset the system baud rate is always set to 19200. This command should be sent to all devices in the system to set a new communications rate.

Parameters

Byte	Parameter	Comment
Code	**BAUDRATE**	Function code = 01
1	0 1 2 3 4 5 6 7	1200 2400 4800 9600 19200 38400 300 19200 **(Default)**

CONFIG

Causes the addressed device to place data in the output buffer containing the state of the card configuration switches. The data is read using the 'POLL' command. There are no parameters associated with this command.

Parameters

Byte	Parameter	Comment
Code	**GETCFG**	Function code = 03

Returns

Byte	Parameter	Comment
Code	**GETCFG**	Function code = 0 3
1	0..15	Hex switch 1 setting (Split)
2	0..15	Hex switch 2 setting (Effects)

RELEASE

Causes the addressed device to place data in the output buffer containing the firmware version. The data is read using the 'POLL' command. There are no parameters associated with this command.

Parameters

Byte	Parameter	Comment
Code	**GETRLS**	Function code = 04

Returned data

Byte	Parameter	Comment
Code	**GETRLS**	Function code = 04
1	Major	Version Number MAJOR
2	Minor	Version Number MINOR
3 .. n	ASCII Text Message	Software release name

VIDEO INPUT COMMANDS

SELECT INPUT

Each input can select either a Composite Video Input (via the BNC connectors) or S-Video via the S-Video connectors.

Byte	Parameter	Comment
Code	**VIDEOINPUT**	Function code = 16
1	0 1	Video Source 1 Video Source 2
2	0 1	Composite Video Input S-Video Input

BRIGHTNESS

Byte	Parameter	Comment
Code	**BRIGHTNESS**	Function code = 17
1	0 1	Video Source 1 Video Source 2
2	0 .. 255	Brightness

CONTRAST

Byte	Parameter	Comment
Code	**CONTRAST**	Function code = 18
1	0 1	Video Source 1 Video Source 2
2	0 .. 255	Contrast

SATURATION

Byte	Parameter	Comment
Code	**SATURATION**	Function code = 19
1	0 1	Video Source 1 Video Source 2
2	0 .. 255	Saturation

HUE

Byte	Parameter	Comment
Code	**HUE**	Function code = 20
1	0 1	Video Source 1 Video Source 2
2	0 .. 255	Hue

COLOUR SYSTEM

Byte	Parameter	Comment
Code	**VIDEOSYSTEM**	Function code = 21
1	0 1	Video Source 1 Video Source 2
2	0 1 2	PAL NTSC SECAM

SYNC OFFSET

This only applies to Video Source 1. The effect is to alter the delay beteen sync and video, providing a measure of adjustment in the horizontal picture position. This affects ALL monitors in the array.

Byte	Parameter	Comment
Code	**SYNCOFFSET**	Function code = 23
1	Signed byte	Offset

VIDEO SOURCE SAVE

Save video source settings in FLASH memory. Saved settings are applied automatically on power-up or software reset.

Byte	Parameter	Comment
Code	**SAVEINPUT**	Function code = 24

READ VIDEO SOURCE SETTINGS

Reads the current video source settings for both channels.

Byte	Parameter	Comment
Code	**READINPUT**	Function code = 25

Returns:

Byte	Parameter	Comment
Code	**READINPUT**	Function code = 25
1	Source-A	
2	Brightness-A	
3	Contrast-A	
4	Saturation-A	
5	Hue-A	
6	Colour system-A	
7	Source-B (TBC)	
8	Brightness-B	
9	Contrast-B	
10	Saturation-B	
11	Hue-B	
12	Colour system-B (TBC)	
13	Sync offset	

VIDEO OUTPUT COMMANDS

VIDEO OUTPUT MODE

The video outputs can be configured for RGB, Composite Video, or disabled. All outputs channels will be set to the same configuration. Colour Systems supported are PAL, NTSC or SECAM.

Byte	Parameter	Comment
Code	**DISPLAYMODE**	Function code = 30
1	0 1 2 3 4	RGB PAL NTSC SECAM Output disabled [Set to 0V]

GENERATE COLOUR BARS

Displays a colour bar test pattern on the selected monitor, or all monitors simultaneously.

Byte	Parameter	Comment
Code	**BARS**	Function code = 31
1	Monitor	0 .. 15 or 255 for all monitors
2	0 1	OFF ON

SAVE VIDEO OUTPUT SETTINGS

Save the current video output settings in a file in FLASH memory. These settings will automatically be applied on power-up or reset.

Byte	Parameter	Comment
Code	**SAVEOUTPUT**	Function code = 34

READ VIDEO OUTPUT SETTINGS

This command reads back the current video output settings. The returned parameter corresponds to byte 1 of function 30.

Byte	Parameter	Comment
Code	**READOUTPUT**	Function code = 35

ADJUST VIDEO OUTPUT HORIZONTAL ALIGNMENT

Move a vertical column of monitor images left or right. The offset is specified in pixels but is actually only possible in increments of magnified pixels.

Byte	Parameter	Comment
Code	**ALIGNH**	Function code = 36
1	Column	0 .. 3
2	Offset	Signed byte

ADJUST VIDEO OUTPUT VERTICAL ALIGNMENT

Move a horizontal row of images up or down. The offset is specified in pixels but is actually only possible in increments of magnified lines.

Byte	Parameter	Comment
Code	**ALIGNV**	Function code = 37
1	Row	0 .. 3
2	Offset	Signed byte

SAVE VIDEO OUTPUT ALIGNMENT IN FLASH MEMORY

Saves the current alignment settings in a file in FLASH memory in the processor. These settings will be applied automatically on power-up or reset.

Byte	Parameter	Comment
Code	**SAVEALIGN**	Function code = 38

READ VIDEO OUTPUT ALIGNMENT

Reads back the current alignment settings.

Byte	Parameter	Comment
Code	**READALIGN**	Function code = 39

Returns:

Byte	Parameter	Comment
Code	**READALIGN**	Function code = 39
1	Row 1	Signed byte
2	Column 1	
3	Row 1	
4	Column 1	
5	Row 1	
6	Column 1	
7	Row 1	
8	Column 1	

EFFECTS COMMANDS

SET SPLIT

Set the global split factor. This will magnify the image by the factor set globally. For example a split of 2 on a 4x4 configuration will result in 4 duplicate 2x2 images.

The vertical and horizontal shift parameters are optional. If supplied and set to 1 they cause the image to be offset by half a monitor size. In the case of a vertical shift this moves the image up. A horizontal shift move the image to the left. A typical application of this function is to position a ‘letter box’ source across a subset of monitors, e.g. a 4x4 magnified image on a 4x3 videowall. The command to achieve this would be:

```
SETSPLIT 4 1
```

Parameters

Byte	Parameter	Comment
Code	**SETSPLIT**	Function code = 40
1	Split	1 to 8
2	0 = No shift 1 = Shift	Optional Vertical shift
3	0 = No shift 1 = Shift	Optional Horizontal shift

ASPLIT

Set an asymmetric split. This command is similar to SETSPLIT but the horizontal and vertical magnifications can be different. No monitor shifts are supported by this command.

The following command sets a horizontal split of 4 and a vertical split of 1 for a vertically squashed image:

```
ASPLIT 4 2
```

Parameters

Byte	Parameter	Comment
Code	**ASPLIT**	Function code = 100
1	Horizontal Split	1 to 8
2	Vertical Split	1 to 8

WASHCOLOUR

Set the colour wash for a given monitor or all monitors simultaneously. The command will have no direct visual effect if a wash is not selected (i.e. a magnified image is currently displayed). When a monitor display mode is set to WASH the colour set with this command will be used. If a wash is currently displayed the on-screen colour will be updated.

Parameters

Byte	Parameter	Comment
Code	**WASHCOLOUR**	Function code = 41
1	Monitor	0 .. 15 or 255 for all monitors
2	Red Intensity	0 .. 255
3	Green Intensity	0 .. 255
4	Blue Intensity	0 .. 255

NORMAL

This command resets the pixel mode to normal and is used following a video effect command such as SOLARISE, NEGATIVE, Etc.

Parameters

Byte	Parameter	Comment
Code	**NORMAL**	Function code = 96
1	0	0 = Normal Video

SOLARISE

This command resets the pixel mode to truncate the lower 5 bits to give the video a halo appearance. Remove using the NORMAL command.

Parameters

Byte	Parameter	Comment
Code	**SOLARISE**	Function code = 96
1	1	1 = Solarised Video

NEGATIVE

This command resets the pixel mode to a negative image. Remove using the NORMAL command.

Parameters

Byte	Parameter	Comment
Code	**NEGATIVE**	Function code = 96
1	2	2 = Negative Video

DAZZLE

This command resets the pixel mode to an intensity highlight effect. Remove using the NORMAL command.

Parameters

Byte	Parameter	Comment
Code	**DAZZLE**	Function code = 96
1	4	4 = Dazzle Video

DISPLAY

This command sets the display mode for a given monitor, which can be either a magnified image or a colour wash

Parameters

Byte	Parameter	Comment
Code	**DISPLAY**	Function code = 42
1	Monitor	0 .. 15 or 255 for all monitors
2	VIDEO WASH	0 1

FREEZE

This command freezes (or un-freezes) the primary frame store and affects all monitors displaying a magnified picture. The frozen image can be either a single field or a full frame (interlaced image).

Parameters

Byte	Parameter	Comment
Code	**FREEZE**	Function code = 44
1	OFF ODD EVEN BOTH	0 = Unfreeze - Full Motion Video 1 = Freeze odd field 2 = Freeze even field 3 = Freeze both fields

STROBE

This command freezes (or un-freezes) the primary frame store and affects all monitors displaying a magnified picture. The frozen image can be either a single field or a full frame (interlaced image).

Parameters

Byte	Parameter	Comment
Code	**STROBE**	Function code = 45
1	OFF ON	0 = Unfreeze - Full Motion Video 1 = Strobe Video
2	Speed	0 - 255, Increasing number = Decreasing Speed

TEXT/INTERLACE

Sets the global operation to text mode. This is equivalent to non-interlaced. Text when displayed in interlaced mode can flicker due to the strong horizontal components in the image.

Parameters

Byte	Parameter	Comment
Code	**TEXTMODE**	Function code = 46
1	OFF ON	0 = Interlaced 1 = Text mode (non-interlaced)

INTERLACE OFFSET

When interlaced mode is selected it is possible to set the number of magnified lines between fields for each split factor. Under some conditions it is possible to improve the visual impact by adjusting this offset. WARNING - if the interlace offset is set to an illegal value for a given split factor, the display will flicker at the field rate.

Byte	Parameter	Comment
Code	**IOFFSET**	Function code = 48
1	Split 1..4	Split factor
2	Offset +/- N	Signed offset

SELECT MONITOR

Using this command it is possible to create an arbitrary set of monitors which will respond to global commands. This reduces the number of messages necessary to achieve a given effect. By default all monitors are selected. It remains possible to directly access a specific monitor by its address.

Parameters

Byte	Parameter	Comment
Code	**SELECTMON**	Function code = 47
1	Flag [Monitor 0]	1 = Selected, 0 = Deselected
2	Flag [Monitor 1]	1 = Selected, 0 = Deselected
3	Flag [Monitor 2]	1 = Selected, 0 = Deselected
4	Flag [Monitor 3]	1 = Selected, 0 = Deselected
5	Flag [Monitor 4]	1 = Selected, 0 = Deselected
6	Flag [Monitor 5]	1 = Selected, 0 = Deselected
7	Flag [Monitor 6]	1 = Selected, 0 = Deselected
8	Flag [Monitor 7]	1 = Selected, 0 = Deselected
9	Flag [Monitor 8]	1 = Selected, 0 = Deselected
10	Flag [Monitor 9]	1 = Selected, 0 = Deselected
11	Flag [Monitor 10]	1 = Selected, 0 = Deselected
12	Flag [Monitor 11]	1 = Selected, 0 = Deselected
13	Flag [Monitor 12]	1 = Selected, 0 = Deselected
14	Flag [Monitor 13]	1 = Selected, 0 = Deselected
15	Flag [Monitor 14]	1 = Selected, 0 = Deselected
16	Flag [Monitor 15]	1 = Selected, 0 = Deselected

TEST PATTERN

This function will draw a test pattern on the display using the graphics facilities. Each monitor has two diagonal lines drawn from opposite corners and a circle with the channel number in the centre. Channels are numbered from left to right, top to bottom assuming a split of 4. The channel numbers are written using a font “TMS.FNT”. If this file is not present in FLASH sector 4, the test pattern will still be drawn but no channel numbers will be displayed.

Byte	Parameter	Comment
Code	**TESTPATTERN**	Function code = 51
1	0 1	Use field 0 Use field 1
2	0 1 2 3 4	Use current split Draw for split = 1 Draw for split = 2 Draw for split = 3 Draw for split = 4

WIPE

This command provides wipes between a graphics colour mask (similar effect to a WASH) and video. And also wipes between the two video sources.

Byte	Parameter	Comment
Code	**WIPE**	Function code = 52
1	0 1 2	Wipe to source A Wipe to source B Wipe to graphics colour
2	0 1 2 3	Left Right Down Up
3	Speed	Range 1 - 255, increasing speed

TIMEBASE CORRECTOR (TBC) FUNCTIONS

The second video input source is processed internally using a time base corrector/synchronisation (TBC) circuit, which allows the two video sources to be non-synchronous. The second source is therefore referred to as "TBC" for the purposes of programming.

TBC ALTERNATE FIELD

The TBC is selected to store alternate fields or every field. Use this command to reduce interlace flicker on the second source.

Byte	**Parameter**	**Comment**
Code	**TBCALT**	Function code = 55
1	0 1	OFF ON

TBC WINDOW POSITION

When TBC MODE (see below) is set to Quarter Screen, this command sets the display quadrant position.

Byte	Parameter	Comment
Code	**TBCWINDOW**	Function code = 56
1	0 1 2 3	Top Left Top Right Bottom Left Bottom right

TBC ENABLE

This command enables and disables display of the second source in full screen or quarter screen mode as set by the command TBC MODE.

Byte	Parameter	Comment
Code	**TBCENABLE**	Function code = 57
1	0	OFF
	1	ON

TBC SIZE

The second source is selected to fill the screen, or a quarter of the screen. When Quarter screen mode is selected, the source will appear to have a coarse appearance due to the dropping of alternate vertical and horizontal input pixels.

Byte	Parameter	Comment
Code	**TBCSIZE**	Function code = 58
1	0 1	Full Screen Quarter Screen

TBC FREEZE

The second source can be frozen independently of the primary source.

Byte	Parameter	Comment
Code	**TBCFREEZE**	Function code = 59
1	0 1	OFF ON

SEQUENCE EXECUTION FUNCTIONS

The following commands are used to initiate sequences which have been downloaded to the processor FLASH memory using the download menu command.

RUN INFRA-RED SEQUENCE

This command executes a sequence which has previously been downloaded to processor and stored in FLASH memory. The file 2:$IRMAP.TXT in the FLASH memory maps key codes to sequence file names. The infra-red handset returns key codes 0 to 15. Other codes can be used to map files which can only be invoked by the code number using this function. Sequence files can always be executed by name using function code 77.

Byte	**Parameter**	**Comment**
Code	**IRSEQ**	Function code = 75
1	Key-code	ASCII key-code

READ SEQUENCE TERMINATION CONDITION

The reason for termination of a sequence file can be determined using this command. A non-zero condition indicates either an error or forced termination using function code 79. Errors usually occur because a dependent file is missing, e.g. if the INCLUDE command is used and the file does not exist, or the file name has been entered incorrectly.

Sequences will also terminate if recursive loops are used causing too many files to be opened for example.

Byte	Parameter	Comment
Code	**SEQERR**	Function code = 76

Returns:

Byte	Parameter	Comment
Code	**SEQERR**	Function code = 76
1	0	Normal termination
	1	Stopped by user (FN 79)
	2	Error reading FLASH
	3	Error opening FLASH file
	4	FLASH file seek error
	5	Illegal function
	6	Function code unsupported
	7	Include name
	8	Include open
	9	Include depth

RUN SEQUENCE FILE

Executes a sequence file by name. Sequence files are stored by default in FLASH Sector 3.

Byte	Parameter	Comment
Code	**RUNSEQ**	Function code = 77
1	FLASH Sector	1 .. 5 (typically 3)
2	Character string	First character
N-1		Last character
N	0	String termination (null)

READ EXECUTING SEQUENCE FILE NAME

Byte	Parameter	Comment
Code	**READSEQ**	Function code = 78

Returns:

Byte	Parameter	Comment
Code	**READSEQ**	Function code = 78
1	FLASH Sector	1 .. 5 (typically 2)
2	Character string	First character
N-1		Last character
N	0	String termination (null)

CANCEL EXECUTING SEQUENCE

Byte	Parameter	Comment
Code	**STOPSEQ**	Function code = 79

GRAPHICS FUNCTIONS

A limited set of graphics functions are provided. These write pixels directly to the common video frame store. It is therefore necessary to freeze the image. Two pages are provided, effectively the even and odd video fields.

Each page has a set of independent variables:

- Drawing colour
- Cursor (X,Y)
- Font

When page 1 is displayed, page 2 is accessible using the following functions and vice-versa. Both pages can be displayed sequentially as an interlaced full frame picture. This is not generally recommended as this can introduce a strong flicker.

The frame store holds images in YUV format. Colours can be specified either in RGB or YUV format. The nominal displayed image is 720x288 pixels with a 50Hz field rate (PAL/SECAM) or 720x240 at 60Hz (NTSC). In practice the horizontal pixel range is halved (360) to simplify access to the video memory which is in YUV 4:2:2 format. Some additional vertical over scan is provided which extends the vertical range to 300. The accessible range for graphics functions is therefore 360x300. The actual displayed range depends on the frame rate and amount of over scan on the display monitors. At 360x300 resolution pixels are roughly square.

Text messages in any colour can be displayed over the frozen image. The fonts used are loaded in FLASH memory. Two fonts are installed by default in FLASH Sector 4, TMS.FNT (a serif font) and HLV.FNT. (sans serif).

Bit mapped graphics files in two colour '.BMP' format, held in the FLASH memory, can be copied into the video frame buffer. This feature is useful for displaying small graphic images e.g. company logos.

INITIALISE GRAPHICS

Initialises the graphics system page variables as follows:

Cursor X,Y	:	0,0
Font	:	None
Colour	:	White

Byte	Parameter	Comment
Code	**INITGX**	Function code = 80

SELECT GRAPHICS PAGE

Selects the graphics page. The Freeze function (44) can be used to the same effect.

Byte	Parameter	Comment
Code	**PAGE**	Function code = 81
1	1 2	Page 1 Page 2

SELECT FONT

Selects the current font for the selected page. Two fonts are provided as standard TMS.FNT and HLV.FNT. Other fonts can be downloaded to FLASH if required. The font for each drawing page is selected independently.

Byte	Parameter	Comment
Code	**FONT**	Function code = 82
1	1 2	Page 1 Page 2
2	Character string	First character
N-1		Last character
N	0	String termination (null)

PRINT TEXT

Prints a text string at the current cursor location in the current font in the current drawing colour. E.g.

```
Send 7 0 FN_PRINT “Message” 0
```

Byte	Parameter	Comment
Code	**PRINT**	Function code = 83
1	Character string	First character
N-1		Last character
N	0	String termination (null)

DRAW LINE

Draws a line in the current drawing colour to location X,Y from the current cursor location (See also DRAW LINE RELATIVE). The current cursor location is updated to X,Y. If the specified location is outside the drawing area no line is drawn and the cursor remains at the current location. E.g.

```
Send 7 0 FN_LINETO &100 &50
```

Byte	Parameter	Comment
Code	**LINE**	Function code = 84
1,2	X	X location (0..359)
3,4	Y	Y location (0..299)

MOVE CURSOR

Moves the cursor to location X,Y if it is within the drawing page limits (See also MOVE CURSOR RELATIVE).

Byte	Parameter	Comment
Code	**CURSOR**	Function code = 85
1,2	X	X location (0..359)
3,4	Y	Y location (0..299)

DRAW LINE RELATIVE

Draws a line from the current cursor position to a location relative to the current cursor if the new location is within the drawing limits.

Byte	Parameter	Comment
Code	**RLINE**	Function code = 86
1,2	DX	X offset (+/- 359)
3,4	DY	Y offset (+/- 299)

MOVE CURSOR RELATIVE

Moves the cursor to a location relative to the current cursor location. If the new location is outside the drawing limits the cursor location is not changed.

Byte	Parameter	Comment
Code	**RCURSOR**	Function code = 87

1,2	DX	X offset (+/- 359)
3,4	DY	Y offset (+/- 299)

FILL PAGE

The currently selected page is filled with the current drawing colour.

Byte	Parameter	Comment
Code	**FILL**	Function code = 88

DRAWING COLOUR RGB

Sets the current drawing colour using the RGB colour scheme. This is converted to the internal YUV system.

Byte	Parameter	Comment
Code	**COLOUR**	Function code = 89
1	Red	Red component 0..255
2	Green	Green component 0..255
3	Blue	Blue component 0..255

PLOT BITMAP

Plots a two colour (1-bit) bitmap. Pixels in the file set to ‘1’ are written as the current drawing colour. Pixels set to ‘0’ are unchanged. E.g.

```
Send 7 0 90 5 “LOGO.BMP” 0
```

Byte	Parameter	Comment
Code	**BITMAP**	Function code = 90
1	Sector	FLASH Sector 1..5
2	Character string	First character
N-1		Last character
N	0	String termination (null)

BMP files can be loaded into FLASH memory using the supplied utility FLASH.EXE in the C:\BRICK\EXE directory.

```
PUT C:\yourdirectory\yourlogo.BMP 5:MYLOGO.BMP
```

Installs the file “MYLOGO.BMP” in FLASH sector 5. Names of files stored in FLASH memory are case sensitive.

To plot pictures with multiple colours, use a photo editor or similar package to separate the colours in your multiple colour bitmap into separate 1-bit bitmap files and download them separately into the processor. Use multiple bitmap commands to overlay them on the display (bitmap commands overwrite previous bitmaps).

DRAW ELLIPSE

Draws an ellipse centred on the current cursor location. The horizontal distance from the centre is specified by DX and the height from the centre by DY. The cursor remains unchanged.

Byte	Parameter	Comment
Code	**ELLIPSE**	Function code = 91

1,2	DX	X Radius (0..180)
3,4	DY	Y Radius (0..150)

DRAW RECTANGLE

Draws a filled rectangle at the current cursor of length DX and depth DY. The cursor is updated. If the rectangle extends beyond the drawing limits it is not displayed and the cursor is unchanged.

Byte	Parameter	Comment
Code	**RECTANGLE**	Function code = 92
1,2	DX	X length (1..360)
3,4	DY	Y length (1..300)

DRAWING COLOUR YUV

Sets the current drawing colour using the YUV colour space as defined in CCIR 601.

Byte	Parameter	Comment
Code	**YUVCOLOUR**	Function code = 93
1	Y	Y component 0..255
2	U	U component 0..255
3	V	V component 0..255

5 CONTROL SOFTWARE

The control software enables all of the hardware features to be configured remotely using a single serial link between a PC and the processor. It is used to create, maintain and execute video effects sequences. These may be timed using the PC real-time clock, or synchronised to time codes associated with the video image using a standard time code reader which can be installed in the PC.

The user interface is graphical and makes use of a mouse which may be a standard 2 or 3 button type.

COMPUTER REQUIREMENTS

The minimum requirements for the computer running the program are:

- IBM/PC AT compatible, preferably 486 processor, or better
- 640KB RAM
- Hard disk with at least 2 MB free.
- One serial port
- VGA display
- Microsoft compatible mouse
- MS-DOS 3.3 or greater.

The program will run under Windows an MS-DOS application.

The program requires at least 540kB of main MS-DOS memory to operate reliably.

INSTALLING THE SOFTWARE

To install the software invoke Windows Explorer and double click on the file called 'INSTALL.BAT' which creates the following directories

```
C:\BRICK
    EXE
    EXAMPLES
    FIRMWARE
    MANUAL
```

This batch file will not modify the AUTOEXEC.BAT file.

Edit the batch file to automate installation from disks other than A: and for hard disks other than C: or simply use Windows Explorer to copy all the directories across to the target drive.

To run the software, create a link on your Windows desktop by right clicking on the desktop and selecting 'new shortcut'.

The program is C:\BRICK\EXE\BRICK.EXE

We suggest selecting the standard Windows icon that looks like a blue window frame with four panes. Then after the shortcut is created you can optionally change to the Media Technologies custom icon by right clicking on the short cut, select program, select change icon, select browse and click on file C:\BRICK\EXE\MT.ico

The windows default settings should be Ok for running the software.

A selection of sample videowall effects sequences are provided in the directory called C:\BRICK\EXAMPLES which should be automatically selected when you first run the software.

On running the software, a graphical screen will appear with menus accessed using the mouse and left clicking on the buttons as follows.

Select 'SEQUENCES' to see the examples we have provided. If you get the message 'No sequences defined' use the 'PROJECT' menu to browse the directory structure. You need to click on the file C:\BRICK\EXAMPLES\WALL.EFX

The software uses PC port COM1: as the default. If you wish to use COM2: you will need to access the 'CONFIGURE/SERIAL PORT' menu in the software and select COM2:

This selection will be automatically saved ready for next time the software is used.

You can use the 'PROJECT' menu to create your own projects - create another directory, for example C:\BRICK\MYSTUFF using Windows Explorer and copy all the files in C:\BRICK\EXAMPLES across to this new directory, then using the software, create your own buttons, deleting the examples you do not want.

PROGRAM EXECUTION

The BRICK program is executed from MS-DOS by entering ‘BRICK’ from the MS-DOS command line or clicking on the icon on the Windows desktop. The program will execute and select the last used project. Communication with the hardware is not necessary at this stage.

There are optional command line switches which can be appended to the command line:

• **-r**<filename.seq>	: Force execution of 'filename.seq' sequence after start-up
• **-p**	: Do not select last selected project directory

SEQUENCE FILE EXECUTION

If desired the program will execute a named sequence file after start-up. The name of the sequence file follows the "-r" switch without any space. For example to run the sequence file "DEMO.SEQ" on start-up:

```
BRICK -rDEMO.SEQ
```

This feature is useful in situations where it is desirable for the Videowall to begin execution of a sequence immediately on power-up without any other manual intervention - by incorporating the above command in the "AUTOEXEC.BAT" file.

Note: It is possible to configure the Brick processor to perform the same function by downloading a sequence file to FLASH memory and installing it as the "startup" sequence.

DEFAULT PROJECT DIRECTORY

Normally the program will select the last used directory as the default "project" directory. This feature can be disabled by appending the -p switch to the command line.

6 SOFTWARE USER GUIDE

The program is mouse-oriented with pull-down menus. Menus are comprised of a set of graphical button icons. Active buttons are indicated by a bright central legend. When the left mouse button is pressed with the mouse over a button icon, the menu option is activated. The right mouse button is used to exit from menus. It has the same effect as the 'escape' key.

The initial screen consists of 2 buttons across the top line which are always present. A second line of 5 buttons comprise the initial menu. The top-line buttons are:

- **Quit**
- **Help**

QUIT

When selected the user is presented with a 'yes'/'no' option. If 'yes' is selected the program terminates and control reverts to MS-DOS or Windows.

HELP

The help system for the BRICK program is context sensitive. A help file is associated with each menu level. When help is requested, help text is displayed in a window pulled down below the help button.

INITIAL MENU

There are 5 buttons in the second row of buttons on the initial window which form the top level menu. They are:

- **Video**
- **Project**
- **Split**
- **Sequences**
- **Configure**

VIDEO MENU

Selects the video input and output modes:

- **Video Input**
- **Video Output**

VIDEO INPUT MENU

This allows the contrast, brightness etc. to be remotely adjusted. Settings can be stored remotely in FLASH memory or saved in a notepad file for inclusion in a sequence file.

The following features are provided:

INPUT CHANNEL

Selects one of either of the two digitisers available in the Brick processor.

COLOUR SYSTEM

Enables the colour system to be set for the selected input channel. Three options are available:

- **PAL**
- **NTSC**
- **SECAM**

The selection of PAL/SECAM on the primary video input channel also selects 50Hz frame rate processing in the pixel replication logic. Selecting NTSC configures the pixel replication logic for 60Hz.

VIDEO INPUT SOURCE

Sets the video input signal type and video levels. There are two possible sources for each input channel:

- **Composite**
- **S-video**

Each input is independent, therefore it is possible to have different sources on the Composite Video and S-video inputs of either channel, and switch between them.

The sliders set the Brightness, Saturation, Contrast and Hue levels.

SAVE IN FLASH

This button will write the current settings into a file in the FLASH memory system (for both channels). If this button is not pressed, the processor will revert to the previously saved settings on power-up.

VIDEO OUTPUT MENU

This allows the output video mode to be selected.

The following features are provided:

- **RGB**
- **Composite Video**
- **Interlaced**
- **Non-Interlaced**
- **Save in Flash**

RGB

Selects RGB output mode - see the connectors section for the pin assignment of the output connector.

COMPOSITE VIDEO

Selects Composite Video output mode - see the connectors section for the pin assignment of the output connector.

INTERLACED

Every incoming video field is displayed, may result in flicker if a lot of text or horizontal features present in the video source material.

NON-INTERLACED

Alternate incoming video fields are displayed, reduces flicker if a lot of text or horizontal features present in the video source material.

SAVE IN FLASH

This button will write the current settings into a file in the FLASH memory system (for both channels). If this button is not pressed, the processor will revert to the previously saved settings on power-up.

SPLIT MENU

The SPLIT menu allows different splits to be configured on the array. This over-rides the switch on the processor front panel, but the processor will revert to the default value on the panel switch value on power-up.

PROJECT MENU

A project is a sub-directory containing sequences files, (".SEQ") sub-project files (".EFX") and data files containing the non- default configuration data.

The file which defines the sequence file buttons is called by default "WALL.EFX" Additional files with the extension ".EFX" can be used for further sub-projects in the project directory. The default settings and wall dimension will be the same for all ".EFX" files in a sub-directory.

The initial project selected will be set to the last used project on start-up unless the -p switch is used to suppress this feature, in which case a file "WALL.EFX" in the current directory is used.

To select a new project enter the directory and path name after the 'New project: ' prompt and press return.

To leave without changing the current project directory simply press return, or 'click' on the 'CANCEL' button.

If you wish to browse the directory to search for an existing project file, click on the 'BROWSE' button. This will pull down a window containing a list of current ".EFX" files, MS-DOS sub-directories and disk drives. The 'UP' and 'DN' buttons page the directory listing in the window.

Clicking on the text in the window will select the file or sub-directory/drive. If the selected item is a sub-directory/drive the directory will be re-listed with the new path. To use the selected file click on 'OK'. This will place the full file path name in the text buffer and close the window.

If a sub-directory instead of a file is specified the filename "WALL.EFX" is assumed. If the project directory selected is new, a prompt will appear querying whether the current project system files should be copied to the new one.

The system files are:

WALL.EFX
WALLDIM.INC
WALLED0.MAC

Note: Not all of these files may exist in any given project directory.

SEQUENCE MENU

When the 'SEQUENCE' button is selected the sequence file control panel will be displayed. This contains a double row of control buttons above a field of sequence buttons which relate to the user's sequence files. These sequence files contain the commands necessary to execute a timed sequence of effects on the wall. Using the upper rows of control buttons it is possible to add, delete, and edit the sequence files, and alter the position and page on which the sequence buttons appear.

SEQUENCE COMMAND SUMMARY

The command format of the sequence files is described in a separate section of this manual. The following is a summary of the available commands:

INCLUDE	Includes text from another file in the current file.
EXIT IF	Terminate an 'include' file if a macro name is defined
DEFINE	Define macro text substitution.

FORGET	Delete a macro/procedure.
PROC	Procedure definition.
ENDPROC	Marks the end of a procedure.
SEND	Sends a single protocol command.
OUTPUT	Outputs unstructured bytes directly to the serial port.
WAIT	Waits for a period (relative to PC clock or time code)
REPEAT	Loop construct, start.
AGAIN	Loop construct termination.
UNTIL	Conditional loop construct termination.
REM	Issue a comment to PC screen scrolled in a window.
SPEED	Change serial port speed.
POLL	Request data from the hardware.
RADIX	Change the default number base.
SHELL	Use MS-DOS command shell to execute a MS-DOS program.
EXEC	Suspend the software and execute a MS-DOS program
END	End of file delimiter.

ADD

Before a sequence file itself can be created using the editor directly, a button must created using the 'ADDSEQ' option.

The button definition has two parts:

LEGEND	: The name that will appear on the button.
FILENAME	: The sequence file associated with the button.

Both the legend and the filename must be defined. A filename may be automatically generated by clicking on the 'AUTO' button.

Sequence files have the extension type ".SEQ". If an extension is not used the program will automatically append ".SEQ" to the filename. The file does not have to exist before it is associated with the button.
A single sequence file can be invoked by any number of buttons.

DELETE

When selected any sequence button when 'clicked' will be deleted with (optionally) the associated sequence file.

EDIT

The Editor is invoked by selecting 'EDIT' on the toolbar menu and then 'clicking' on a sequence button. The file associated with the button will be loaded into memory and the editor window opened. To leave the Editor click on 'EXIT'. If you wish to abandon changes to a file without leaving the Editor click on 'CANCEL'. Button assignments for the editor as follows:

Pg-Up	**:** Scrolls up one page.
Pg-Dn	**:** Scrolls down one page.
Up	**:** Scrolls up one line.
Dn	**:** Scrolls down one line.
File	**:** File presents a menu with three options:

FILE	**:** Open a new file
MERGE	**:** Merge an existing file at the current cursor position.
SAVE	**:** Update file with latest edits.

Cut **:** Cut a block of text lines and save in a temporary buffer. To select the lines move the cursor to the start line then press and hold the left mouse button. Move the cursor to the end of the block and release the mouse button. The selected block will be highlighted. Select 'Cut' to delete and save it in the buffer.

Copy **:** As 'Cut' but the selected blocks are not deleted.

Paste **:** Copies the previously selected block to the file at the current cursor location.

Find **:** Find a string in the current file.

Line **:** Move the cursor to a selected line number.

Cancel **:** Cancel any changes since the start of editing the current file.

Exit **:** Leave the editor. If the file has changed a dialogue box will appearing allowing the file to be saved before exit, to exit with no changes, or to return to the editor.

CONTROL KEYS

Editor control key functions are as follows:

Control + Key	Function Key	Action
^R	Pg-Up	Page up
^C	Pg-Dn	Page down
	Home	Start of line
	End	End of line
^QR	Ctrl + Home	Top of file
^QC	Ctrl + End	End of file
^S	Left arrow	Cursor left
^D	Right arrow	Cursor right
^E	Up arrow	Cursor up
^X	Down arrow	Cursor Down
^V	Insert	Insert mode On/Off
^G	Delete	Delete character
^H	Backspace	Cursor left + delete character
^I	TAB	Tab (4 spaces)
	ESC	Exit editor
^W		Scroll window up
^Z		Scroll window down
^T		Delete word left
^Y		Delete line
^U		Undelete line
^L		Search next
^N		Insert new line
^QF		Search
^QI		Go to line number

MACRO BUTTONS

If the '#' button at the top right of the editor screen is clicked a column of 'macro' buttons will appear. At the top of the column is the 'Notepad' button. If this is clicked, the contents of the notepad will be copied to the file. The notepad holds temporary data generated from the effects windows. The remaining buttons are defined in a macro file. This file has the name "WALLED0.MAC" and contains up to 10 macro lines. The format for these files is:

"Macro text.... ", buttonlegend

"@filename" , buttonlegend

Where 'Macro text' is any text string and 'Buttonlegend' is the legend to appear on the associated button. If the first character of the macro text is an '@' then the text is treated as a filename which will be merged when the macro button is pressed.

A "WALLED0.HLP" file is associated with the macro file. If such a file exists, with the MACRO buttons displayed, the contents of the help file will be displayed when the 'HELP' button is selected. If not, the normal editor help file will be displayed.

DOWNLOAD

Sequences may be down loaded to the embedded system controller to be stored in FLASH memory and subsequently executed directly on the system.

Important Note:

> **The download-sequence compiler does not currently support all the sequence commands available, and 'WAIT' commands must be relative (a time code reader is not present on the processor).**
>
> **Also, procedures can only be used without parameters.**

When 'Download' is selected and a sequence button then selected, the sequence compiler will be invoked. If the sequence compiles successfully it is downloaded into the FLASH memory.

The current contents of the FLASH memory can be displayed by selecting 'List' in the 'Controller' menu, which is itself located in the 'Controls' menu. The name of the FLASH file is the same as the first 8 characters of the MS-DOS filename but with the extension ".SFB".

Note: Each sequence filename in the FLASH memory must be unique.

PROJECT

The 'Project' button at this menu level has the same effect as the 'Project' button from the top level menu.

RENAME

A selected sequence button can be renamed. Both the legend on the button and the associated sequence file may be changed.

CUT

When selected, any sequence button can be temporarily deleted and saved. The 'Paste' mode will then be automatically selected. The button may then be 'pasted' back in any position by clicking in the appropriate place. The page may be changed while in paste mode. Only one button may be 'Cut' at a time. If a second button is cut, it will replace the earlier one.

PASTE

If a button has previously been 'Cut', selecting 'Paste' will allow it to be pasted back onto the page. The button may then be pasted back in any position by clicking in the appropriate place. The page may be changed while in paste mode to transfer or replicate the button on another page.

EXIT

Leaves the 'Sequence' menu. If there were any changes made to sequence buttons a prompt will be displayed requesting confirmation whether or not the changes are to be made permanent.

The sequence menu (and all menus) can also be exited by right clicking on the mouse.

CONFIGURE MENU

The configure menu pulls down a further set of buttons comprising:

- Controller
- Serial Port
- Firmware Upgrade
- Hardware Reset
- Alignment
- Program info

CONTROLLER MENU

Selects the System Control function, from which sequences installed in the Brick's FLASH memory can be listed and executed. The following are available from the controller menu:

- Keypad
- Run
- Stop
- List
- Delete
- Startup
- Show
- Map

KEYPAD MENU

Brings up a keypad to enable already downloaded sequences mapped to key numbers (see below) to be invoked.

RUN MENU

When selected a menu of installed sequences in FLASH memory will be displayed. If a file is selected if will execute on the processor independently of the PC.

STOP MENU

When selected stops any executing sequence running remotely on the Brick processor.

LIST MENU

Produces a list of currently installed sequences and shows the file sizes. This assumes that sequences are stored in Sector 3 of the FLASH filing system. This is the case for all sequences stored using the graphical interface which always assumes sector 3. However it is possible using the FLASH.EXE utility to store sequence files in other sectors.

DELETE MENU

Brings up a list of sequence files. If a file is selected, then it will be deleted when OK is pressed.

START-UP MENU

Pops up a further sub-menu. This allows a specific sequence stored on the PC to be installed as a special start-up file invoked on power-up. The file can also be deleted, and executed from this menu. By pressing "Execute" the sequence can be invoked without resetting the unit, to test it for example.

SHOW MENU

Shows the filename of the currently executing sequence or "none" if no sequence currently running.

MAP MENU

When this button is pressed an editor is invoked to edit a text file stored in FLASH memory which maps a key from the Infra-red handset (or front panel Effect switch) to a particular downloaded file. The format of this mapping file is:

```
<keycode> = sequence.sfb
```

The standard handset produces codes which are interpreted by the Brick software as key codes 0 to 15. The handset keys are mapped as follows:

Key	Keycode
1	0
2	1
3	2
A	3
4	4
5	5
6	6
B	7

Key	Keycode
7	8
8	9
9	10
C	11
	12
0	13
.	14
D	15

E.g. to map key 1 to file MYSEQ.SFB edit the file to include the line: 1 = MYSEQ.SFB

The effects mapped to the Infra red handset keys can also be invoked manually using the front panel switch labelled "Effect" and pushing the "Store ID" button.

If the Infra-red option has not been requested, a blank file is automatically created ready for editing. Additional mappings can be included using key code 16 to 255. These can only be invoked using a command via the serial link.

SERIAL PORT MENU

This button selects COM1: or COM2: to be used by the software. The selection will be automatically saved ready for next time the software is used.

FIRMWARE UPGRADE MENU

New firmware can be installed in the Brick hardware using this menu option. Detailed use of this facility will be described in the documents shipped with firmware upgrades.

When selected, a prompt appears for a file containing the binary image of the new firmware. The path and filename can be entered directly, or the Browse button pressed to locate the desired file. Once selected, download begins when OK is pressed. To abandon download press cancel.

The DOS file selected it will always be renamed to SYSTEM.COM in the FLASH filing system. A file of this name will usually exist, in which case a prompt will appear warning that this is about to be overwritten. If ‘Yes’ is pressed, download will proceed over writing the existing SYSTEM.COM file. As the download progresses a graphical thermometer type display will indicate progress.

Once complete it is necessary to reset the unit to run the new code. If for any reason the download fails do not reset the unit. Instead re-attempt the download. Once it is successful its then ok to execute a reset. If the unit has been reset without properly upgrading the code then it is necessary to use the boot failure recovery procedure described in a separate section.

HARDWARE RESET MENU

This button sends a global system reset command to the hardware. Before the command is sent a 'Yes or No' confirmation dialogue box will appear. The reset command will be sent if 'Yes' is selected. The hardware reset operation will take a few seconds to complete.

ALIGNMENT MENU

This menu option presents an alignment adjustment screen. This shows graphically the monitor channel numbers and their position in the wall. A row or column can be selected using the letter/number boxes on the Top and Left of the wall representation. Once selected the Adjust slider can be used to move the row or column in relation to the other monitors.

This procedure should only be used after all of the videowall monitors have been aligned individually to compensate for under or over scanning of the magnified fragments displayed on the monitors.

Once the wall has been aligned the settings can be saved in FLASH memory to be automatically applied on start-up. It is also possible to save the settings in text to the notepad for subsequent inclusion in a sequence file.

The image on the wall is by default a magnified image. If the ‘Test Pattern’ button is pressed a pattern which identifies the channel and draws a line from opposing corners of each monitor is displayed, to assist in aligning the image.

Pressing the VIDEO button will cause the input video signal to be displayed.

You can select 2x2, 3x3 or 4x4 magnification mode by pressing the appropriate buttons on this menu page.

PROGRAM INFORMATION

A software maintenance facility - it shows the amount of free memory and free file capacity. It also displays the program version number.

7 CREATING SEQUENCES

The ‘BRICK.EXE' videowall control program includes a real-time sequence control language. Video effects are invoked using timed sequences of commands to the hardware. The language commands are held in standard ASCII text files. The sequence files may be created with the integrated editor or an external text editor/word processor.

Control commands are decoded and sent as messages to the hardware via the PC serial ports.

The control language uses a basic command and parameter list format. A simple line based macro processor is included. In the descriptions of the commands uses a simple language notation. Optional parameters are contained in square brackets. A parameter which must be one of a set is indicated by the set enclosed in curly brackets, each parameter separated by a vertical bar. Numeric parameters are contained in angle brackets. Parameters are separated by commas.

SEQUENCE FILE FORMAT

Each line of the command file consists of either a blank line, a comment, or a command. Only one command may be invoked per line. Most commands are followed by a variable length list of parameters dependent on the command type.

Numeric parameters can be in either decimal, hex, or binary. Hex numbers may be represented in either the 'C' format as 0xnn or the 'assembler' format using a leading decimal digit and trailing 'H' Binary numbers are designated by a trailing 'B'. Any number may have leading zeros. Unlike 'C' this does not designate base 8. Hex numbers may use upper or lower case letters.

For example:	Decimal	:	255
	16-bit Decimal	:	&16430
	Binary	:	11111111B
	8-bit Hexadecimal	:	0xFF, 0x0ff or 0FFH
	16-bit Hexadecimal	:	&0x800F

All commands and parameters should be delimited by a space, comma, or tab character. In addition the default base may be changed using the RADIX command to any base from 2 to 16.

Macros may be defined to substitute text strings with key words.

Command words can be in either upper or lower case. Macros are case sensitive. Command lines before and after macro expansion must be less than 132 characters.

COMMAND WORDS

PAL (NTSC)

When compiling sequences to be downloaded to FLASH memory in the processor for stand alone operation, it is necessary for the compiler to adjust the number of video fields counted in ‘WAIT’ commands, to ensure that time delays are equivalent for both PAL (50 Hz) and NTSC (60 Hz) operation.

Normally, the appropriate sequence command ‘PAL’ or ‘NTSC’ is included at the beginning of the STD.INC file depending on the country where the system was purchased. In the event that it is required to operate in PAL mode in an NTSC country, the user should create a version of STD.INC with the command ‘PAL’ substituted for the command ‘NTSC’, and vice-versa.

INCLUDE

The include command 'includes' text from another file in the current file.

Command syntax:

```
      INCLUDE { <filename.typ> | ["][d:][\path\]filename[.typ]["] }
```

The INCLUDE command has one parameter, either a filename in angle brackets or a conventional MS-DOS filename. Quotes are optional on the conventional filename. Included filenames in angle brackets are prefixed by the MS-DOS environment variable VIDEOWALL.

For example in the system AUTOEXEC.BAT file the SET command may be used to prefix all the SEND program include files enclosed in angle brackets with the subdirectory C:\BRICK using,

```
SET VIDEOWALL=C:\BRICK
```

Conventional filenames without the angle brackets can have optionally drive letters, path descriptions and file types.

Includes can be nested to a depth of 10. The nesting depth is the only method by which recursive includes are trapped. Files may be 'included' at any point in a command file.

The file STD.INC is a special case. If a file of this name is found in the current directory it will be included before the sequence file is executed. If not, the directory defined using the VIDEOWALL environment variable will be searched.

SEND

The SEND command sends a single command to the hardware. The send command expects at least 3 parameters. (See the hardware interface description for details of commands.)

Command syntax:

```
SEND <p1>,<p2>,<p3> [,<p4> .. <pN>]
```

The parameters p1 to pN are numbers in the range 0 - 255. A parameter from p4 onwards may be preceded by the '&' character to force it to occupy two bytes, LSB first.

The first 3 parameters common to all SEND commands are as follows:

p1 = Module type.
p2 = Module address.
p3 = Message function code.

Module type is a number with the following allocation,

0	=	DIGITISER
1	=	MULTIPLEXOR
2	=	FRAMESTORE
3	=	CONTROL CARD
4	=	QUAD OUTPUT CARD
5	=	MASTER FRAMESTORE CARD
6	=	DUAL DIGITISER
7	=	BRICK PROCESSOR

Module address is a number from 0 to 255. The message function code is decoded by the addressed module which executes the function using any parameters which follow. The BRICK processor has no internal modules so requires only one sub-address.

Parameters p4 to pN are placed in the data bytes section of a message. An ASCII text string can be used in the parameter list. Each character of the string will be converted to a number in the range 32..127 and used in the same way as numeric parameters.

OUTPUT

The OUTPUT command is a lower level version of the SEND command. It will send any number of parameters supplied as bytes to the hardware (limited by the maximum line length) without formatting the data into a structured message.

Command syntax: `OUTPUT <p1> [<p2> .. <pN>]`

A parameter may be preceded by the '&' character to force it to occupy two bytes, LSB first. ASCII strings may be sent by enclosing them in double quotes, For example:

```
OUTPUT "PLAY",0x0D
```

WAIT

This is the primary time dependent command word. Parameters supplied determine whether the control program will wait for a set period, until the elapsed time since the start of the file, the absolute time or on reaching a time code for the selected video source.

Command syntax:

```
  WAIT [{FOR | UNTIL | TIMECODE | ELAPSED}] <time> [,<offset> [,<rate>]]
```

```
Where: <time>     = hh:mm:ss[{.dd|:ff|-ff}] (dd = decimal fraction, 01..99)
       <offset>   = hh:mm:ss[{.dd|:ff|-ff}] (ff = Fields 1..24 for PAL)
       <rate>     = 0..65535
```

The default if no sub-command is supplied is the same as FOR. The time and offset are defined in 24 hour clock format, hours, minutes, seconds and optionally hundredths of a second if a dot is used to separate the seconds or fields if a colon or dash is used. There are 25 fields per second for PAL and 30 for NTSC. The actual time is calculated by multiplying the rate and offset then adding this to the base time and compensating according to the PAL or NTSC command included in the STD.INC file. See also 38 PAL (NTSC) command.

The offset and rate are optional. These are mainly used in loops, for example:

```
REPEAT 9
  SEND BRICK,@i,EFFECT,P1,P2      ; Prime hardware with an effect
  WAIT TIMECODE 0:0:10,0:0:1,@i   ; Wait at time codes 0:0:10,0:0:11 etc.
AGAIN
```

WAIT FOR

The WAIT FOR subcommand will wait for the selected time on decoding the command. If any key on the keyboard is pressed the wait command will terminate.

WAIT ELAPSED

The WAIT UNTIL subcommand will wait until the elapsed time from the start of the command file is greater than the time defined. If any key on the keyboard is pressed the wait command will terminate.

WAIT UNTIL

The WAIT UNTIL subcommand will wait until the absolute time (PC Clock) is greater than the time defined. If any key on the keyboard is pressed the wait command will terminate.

WAIT TIMECODE

The WAIT TIMECODE subcommand will wait until the time read from the time code reader is greater than the time parameter. If any key on the keyboard is pressed the wait command will terminate.

If a time code reader is not installed an error message is issued and command file interpretation is aborted.

DEFINE

Macros can be defined, using the DEFINE command, which will be expanded on a line by line basis when encountered.

Command syntax: `DEFINE NAME [=] text....`

The macro NAME can have up to 16 characters. There is no restriction on the characters of the macro name apart from delimiters space, tab and comma.

Macros are expanded as they are encountered in the input file by the command processor except if they are parameters to a macro definition. Macros must be defined before they can be used.

The macro text can comprise delimited parameters. A simple parameter substitution mechanism is provided. This allows up to 9 parameters in the text portion of the macro definition to be undefined. These are designated by a question mark followed by a decimal digit in the range 1 to 9. Parameters following the macro name on expansion will be substituted in the order defined by the number following the question mark.

For example the macro defined thus:

```
DEFINE SWAP = ?2,?1
```

On expansion will reverse the parameters supplied, so that for example:

```
OUTPUT SWAP 200,100
```

Expands to:

```
OUTPUT 100 200
```

Macros can be nested to any depth. On expansion however the line length must not exceed 132 characters.

Macro expansion is dependent to a certain extent on the command. Commands such as SHELL and EXEC will stop macro expansion as soon as the command word is decoded. This is particularly true in the case of the DEFINE command itself.

For example:

```
DEFINE ONE 1            ; Define macro with value of 1
SHELL ONE               ; ONE is used directly and not converted

DEFINE NEWSHELL SHELL   ; Redefine SHELL command
NEWSHELL ONE            ; Macro expansion takes place to give: SHELL 1
```

FORGET

Macros and procedures can be deleted from the symbol table at any point in the file using the FORGET command. If previously defined macros depend on the macro which has been deleted they will cause errors on expansion later.

Command syntax: `FORGET name`

PROCEDURES

Procedures are similar to macros in may ways except that they span more than one line and may contain a number of commands. Up to 9 parameters can be passed to each procedure. The procedure is defined starting with the PROC keyword and terminates with the ENDPROC keyword. Both keywords must be on a separate line.

Command syntax:

```
PROC name [p1 .. pn]
    commands...
ENDPROC
```

Procedures can have up to 9 parameters supplied. These can be used to substitute parameters to commands within the body of the procedure, or passed as further parameters to procedures calls within the procedure body.

For example:

```
PROC EFFECT start,middle,finish      ; Procedure EFFECT with 3 parameters
   SEND 7 0 1,2,start                ; Use each parameter in turn for the
   SEND 7 0 1,2,middle               ; same basic command
   SEND 7 0 1,2,finish
ENDPROC                              ; End of procedure

EFFECT 1,2,3                         ; Execute procedure
EFFECT 4,5,6                         ; Again with different arguments
```

The dummy arguments: 'start, middle, finish' define the number of parameters the procedure will accept. When the procedure is used the number of parameter arguments supplied is checked. If the wrong number are supplied an error message is displayed and execution stops.

REMARKS

The REM command allows remarks to be displayed on the PC screen as the commands are sent to the hardware. All the text following the REM statement is displayed on the screen. As soon as the macro expander recognises the REM command, macro expansion ceases.

In addition a number of escape codes are provided.

@t	Current system time.
@d	Current system date.
@s	File start time.
@e	Elapsed time.
@f	The current sequence file name.
@r	Read and insert current time code from time code reader (if installed).
@b	Bell character.

Two further escape codes may be used in REM commands to show the two innermost loop counts generated by the REPEAT command.

@i	Inner loop count
@j	Second level loop count

REPEAT

Command syntax: `REPEAT [FOREVER | <n>]`

The REPEAT command marks the beginning of a program loop. REPEAT commands must be matched by a corresponding AGAIN or UNTIL command.

An optional loop count can be supplied. If not FOREVER is assumed. A loop can be terminated using a parameter to the UNTIL command even if it is started with a REPEAT FOREVER command.

For example:

```
REPEAT 20
  ..more commands on any number of lines..
AGAIN
```

Or...

```
REPEAT FOREVER
       .
       . body of loop
       .
```

```
      UNTIL TIMECODE = 00:20:00
```

REPEAT commands can be nested to any depth (limited by only the PCs memory). For example:

```
      REPEAT                     ; Start of outer loop
        REPEAT 20                ; Repeat a sequence 20 times
              .
              . body of sequence  ; Commands comprising the sequence
              .
        AGAIN                    ; Loop back to nearest REPEAT
      UNTIL TIMECODE 0:0:10      ; Keep going until timecode is passed
```

Two special macros are provided which are converted to the current loop index when used. These are @i and @j. The following shows the effect,

```
      REPEAT 3
        REPEAT 3
          REM Loop = @j,@i
        AGAIN
      AGAIN
```

This double loop will display in a window the following messages:

```
      Loop = 1,1
      Loop = 1,2
      Loop = 1,3
      Loop = 2,1
      (etc... until)
      Loop = 3,3
```

These special macros can be used as parameters to macros and procedures.

AGAIN

This command marks the end of a looping program block. There are no parameters. See above REPEAT command.

UNTIL

The UNTIL command marks the end of a sequence loop. The loop is repeated conditionally with one of three conditions, TIMECODE, ELAPSED, or LOOP.

Command syntax:

```
UNTIL [TIMECODE[=] hh:mm:ss[{.dd|:ff|-ff}]]
UNTIL [ELAPSED [TIME[=]] hh:mm:ss[{.dd|:ff|-ff}]]
UNTIL [LOOP [TIME[=]] hh:mm:ss[{.dd|:ff|-ff}]]
```

TIMECODE is followed by time parameters. The loop will continue until the current time code is greater than the time code parameters specified.

ELAPSED TIME is followed by time parameters. The loop will continue until the elapsed time since the start of execution of the file is greater than the time parameter.

UNTIL LOOP TIME is also followed by time parameters. This will loop until the elapsed time since the start of the loop is greater than the supplied time parameter.

SPEED

This command can be used to change the baud rate of the serial communications port. By default this is 19200. If other devices are connected to the same port it is possible to send commands to them using the OUTPUT command to build the command messages and SPEED to select the correct transmission rate.

Command syntax: `SPEED <rate>`

The rate parameter can be any rate supported by an IBM PC/AT up to 19200 bps. Typical rates are:

300, 600, 1200, 2400, 4800, 9600, 19200

The data format is : 8 data bits, 1 stop bit, with no parity.

POLL

Command syntax: `POLL <card>,<address>`

This command requests information from hardware. If the POLLed device has any data ready for transmission it will be returned on receipt of this command. Devices will only have data if they have been sent a command previously requesting it. If no data is ready a 'NULL POLL' message will be returned. There will be no displayed information in response to 'NULL POLL'

If the card addressed does not exists then after a short time-out period a message will be displayed showing that the POLL command failed.

Returned messages are defined in the product specific documentation.

RADIX

Normally the SEND program uses decimal numbers by default. This may changed using the RADIX command to any base from 2 to 16. The base selection parameter is always in decimal regardless of the current radix.

Command syntax: `RADIX <n>`

For example to select HEX as the default base:

```
RADIX 16
```

SHELL

Executes a MS-DOS command using the MS-DOS command shell interpreter. Control reverts back to the BRICK program on completion, not the MS-DOS command line. SHELL expects a string which is passed to the MS-DOS command shell.

On return the serial port is reset and the transmission rate is set to the default rate of 19200 baud.

For example to display the current directory,

```
SHELL DIR *.* /w
```

or using a MACRO...

```
DEFINE FILES SHELL DIR C:\WALL\*.SEQ /W
  ....
FILES
```

The SHELL command should not be used if the program requires any keyboard input. This is because the output is held in a temporary file and only displayed in a window when control reverts to the BRICK program. Any keyboard input would not be echoed back to the user until the program ended.

Also the SHELLed program must not change the video mode of the screen otherwise the BRICK program display will be corrupted.

Programs which require input or write directly to the screen can be executed directly from the BRICK program using the EXEC command.

EXEC

This command completely suspends the BRICK program and executes the specified MS-DOS command. The screen contents are saved. This allows programs such as word processors which will reconfigure the screen to be executed from a button.

For example to execute WORDSTAR

```
EXEC WS
```

END

This command will terminate sequence file execution. Any succeeding commands will be lost. This has the same effect as an end of file and is therefore optional. The command should not be used in procedures.

EXIT IF

To enable include files to be 'included' more than once without causing re-definition errors, a conditional EXIT command is provided. Command syntax:

```
EXIT IF [ DEFINED ] symbol
```

If the symbol is defined 'including' is abandoned and control continues from the 'calling' file.

For example:

```
EXIT IF FRED
DEFINE FRED = 1
```

COMMENTS

Comments can be included either on a blank line or at the end of a command line by preceding the comment with a semicolon. Commented lines must be less than the line maximum of 132 characters. Blank lines may be freely spread throughout a command file to improve readability.

8 DOS COMMAND LINE EXECUTION

A version of the control software - SEND.EXE - is provided which enables control of the hardware in MS-DOS batch files. Control commands are decoded and sent as messages to the hardware using the COM1: or COM2: serial ports. The language of the control file uses a basic command and parameter list format. A simple line-based macro processor is included.

The contents of the sequence files used by the SEND program are the same as those used in the BRICK program but without the overhead of the graphical user environment. (See previous section)

COMMAND LINE INVOCATION

SEND expects at least one argument: the command file to send. If no filename is supplied a short prompt message will appear such as:

```
Videowall command file processor. Version 1.25 Jul 26 1998
Copyright (c) Media Technologies, 1991 - 1998

Use:

SEND {filename.typ [-e | -t | -v]} | {-i} [-l] [-com1: | -com2:] [-b<rate>]

Where: -e       macro expansion mode.
       -t       test expansion mode.
       -v       Verbose execution mode.
       -i       Interactive execution mode.
       -l       Log I/O to files TX.LOG & RX.LOG
       -com1:   Use serial port 1
       -com2:   Use serial port 2
       -b<rate> Set speed of selected serial port

ERROR: Filename expected.
```

If a filename is supplied without any switches, SEND will sequentially process the file sending commands to the hardware. No output will appear on the PC screen except for the output from 'REM' commands. This is the normal execution mode.

SEND can be used additionally with one of four possible switches to select further operational modes described below.

TEST MODE [-t]

In this mode SEND will show each stage of the macro expansion process and the final bytes in HEX that would be sent to the hardware. No actual data is sent and the 'WAIT' command is ignored.

MACRO EXPANSION MODE [-e]

In this mode SEND will expand all macros to reduce the source file to its simplest form. All comments and blank lines are removed. The reduced output is sent the MS-DOS console device and so may re-directed to file using the '>' redirection facility of the MS-DOS command shell. No data is sent to the hardware.

VERBOSE EXECUTION MODE [-v]

This mode is much the same as normal execution mode except that the lines of the command file are displayed as they are interpreted. When a 'WAIT' command is encountered a wait prompt message is issued in addition to the command lcausing it being displayed.

INTERACTIVE MODE [-i]

In this mode the source text is taken from the keyboard instead of a file. Unlike the previous modes an error will not cause the program to exit to MS-DOS. To exit to MS-DOS enter ^Z (End of file.) or the command word 'END'.

LOG FILES

In either execution mode the -l switch can be used to log the messages sent to and from the serial port in two files,

TX.LOG	: Messages from the PC to the hardware
RX.LOG	: Messages from the hardware to the PC in response to a POLL

Each line of the log file contains the raw message in HEX preceded with a time stamp. E.g.

```
[hh:mm:ss.dd] 7E..message in hex..7D
```

SERIAL PORT

The SEND program uses COM1: by default. This can be overridden by specifying -COM2: if required.

BAUDRATE

The default baud rate is 19.2k bit/s. This can be overridden using the -b switch followed by a baud rate supported by the hardware:

- 300
- 1200
- 2400
- 4800
- 9600
- 19200
- 38400

For example to use 9600 instead of 19200.

```
-b9600
```

9 FLASH MEMORY UTILITY

FLASH MEMORY COMMANDS

The FLASH utility provides a means of administering the data held in the Brick hardware's FLASH memory.

The Brick processor contains a 512Kbyte FLASH memory device which is used to hold the basic operating system and data files. The device is divided into 8 logical sectors of 64K bytes. The basic operating code is held in Sector 0 and is protected from accidental erasure. Five sectors are available for use as a filing system accessible using the FLASH.EXE utility. The remaining two sectors are reserved for internal use.

File names are case sensitive and can be up to 12 characters. File extensions are not used but '.' is a legal character so file names in the MS/DOS 8.3 format can be used e.g. "SYSTEM.COM".

The utility is invoked by typing FLASH at the command line.

Commands available are:

DIR
ERASE
VER
TYPE
DUMP
PUT
GET
RESTORE
?

By default the serial port used is COM1: This can be changed by specifying the port in the FLASH.INI file. FLASH.EXE will look for the FLASH.INI file in the same sub-directory as the .EXE file.

?

To display a list of FLASH commands and their syntax enter ? at the prompt. The following text will be displayed:

```
DIR [n[:file]]
ERASE n:file
VER
TYPE n:file
DUMP n:file
PUT [d:\path]dosfile[.typ] n:file
GET n:file [d:\path\]dosfile[.typ]
RESTORE [1..5]
```

DIR

Lists the entry for a specific file, a sector or all sectors. Sectors are numbered 1 to 5. E.g.

```
DIR                 ; lists all files in all sectors
DIR 1:              ; lists all files in sector 1
DIR 1:SYSTEM.COM    ; shows file statistics for the specific file
```

Three parameters are shown with respect to each sector displayed related to the memory usage. These are the amount of memory used, the amount free and the amount wasted. Wasted memory is memory that cannot be used because a file has been deleted. To recover wasted memory, use the RESTORE command.

ERASE

Removes a specified file form the filing system. The file is defined in the form <sector>:filename. E.g. to delete file SYSTEM.COM from sector 1

```
ERASE 1:SYSTEM.COM
```

The filename will no longer be visible but no memory is released. In order to release the memory use the RESTORE command.

VER

Gets and displays the version of firmware running on the target hardware.

TYPE

Types an ASCII file stored in the FLASH filing system. E.g.

TYPE 1:$CONFIG

Output can be directed to a file on the PC using the usual '>' feature of MS/DOS.

DUMP

The same as FLASH TYPE but data in the file is presented in HEX.

PUT

Copies a file from the PC to the FLASH memory. The transfer is always in straight binary. It is necessary to ensure that sufficient space exists prior to performing the transfer. No file can exceed the limits of a single sector which is 64K bytes, less the minimum 512 byte directory overhead per sector. E.g.

```
PUT C:\BRICK\FIRMWARE\SYS101.BIN 1:SYSTEM.COM
```

GET

Does the reverse of PUT, e.g.

```
GET 1:SYSTEM.COM C:\BRICK\FIRMWARE\SYS101.BIN
```

RESTORE

Restores memory consumed by deleted files. This process takes a few seconds to complete for each sector. It is possible to perform a restore on just one sector. E.g. to restore Sector 1:

RESTORE 1

10 LOW LEVEL MONITOR

Rarely, such as a PC power failure during firmware upgrade, the operating system in the processor FLASH memory may be corrupted. In that event, the FLASH ROM must be rebuilt using the low level monitor.

If the "Store ID" button is pushed during power-up, the BRICK processor will not boot and instead drops into a low level monitor. Communication with the low level monitor is via the host serial link using a dumb terminal with the following settings:

9600, 8 bits, no parity.

A PC utility TALK.COM is supplied to perform this function on the distribution disk if the user does not already have a suitable terminal. The low level monitor accepts single line commands after the '>' prompt is presented. Entering ? at the prompt will display a list of possible commands. These commands allow the FLASH filing system to be manipulated via the terminal interface.

BOOT PROCESS

If the Store ID switch is not pushed during power-up the processor will boot as normal by searching for a file called $CONFIG in Sector 1 of the FLASH memory. This is a simple text file containing boot instructions in the form:

```
BOOT=<d>:filename
```

Each BOOT statement will be tried until a specified file is found. When this file is located it is copied into RAM and the processor begins execution of that code.

If there is no file $CONFIG or none of the BOOT files exists the file 1:SYSTEM.COM is assumed and the processor attempts to boot using this file. This is in fact the normal case. The BRICK.EXE program assumes this when performing a firmware upgrade. If the $CONFIG file has been used to point to a different boot image, it is necessary to either manually upgrade the firmware or delete the $CONFIG file before using the BRICK.EXE program to perform a firmware upgrade.

If a SYSTEM.COM does not exist or is corrupted the low level monitor exits the boot load process and returns a '>' prompt for user input from a terminal via the host serial link.

FLASH FILE SYSTEM RECOVERY

If the FLASH file system has been corrupted it is possible to rebuild it via the low level monitor using the following procedure. Power-up the processor with the Store ID button pressed. Run the programme TALK which is in the EXE directory of the software supplied with the system. If necessary, configure TALK using the menus to communicate with the processor using the correct COM port and baud rate.

A '>' prompt will appear if the ENTER key is pressed a few times and text commands can be entered as follows:

- Format the FLASH memory by entering PREP. This will erase the sectors used for the filing system and write empty directories to each sector. The low level operating code is in a protected sector and will not be affected.
- Copy the boot code using the HEX command. A file SYS101.HEX is supplied on the distribution dish for this purpose. E.g. enter HEX 1:SYSTEM.COM
- On the terminal press F9 and enter the filename to download e.g. C:\BRICK\FIRMWARE\SYS101.HEX A series of dots will be displayed as the code is downloaded.
- Reboot the Brick hardware without the Store ID button being pressed..

At this point exit the terminal emulator and use the FLASH utility to copy the remaining files. E.g.

```
FLASH PUT C:\BRICK\FIRMWARE\TMS.FNT 4:TMS.FNT
FLASH PUT C:\BRICK\FIRMWARE\HLV.FNT 4:HLV.FNT
FLASH PUT C:\BRICK\FIRMWARE\MEDIAT.BMP 5:MEDIAT.BMP
```

It is now possible to run the BRICK.EXE program to reconfigure the processor and reload your sequences.

11 SPECIFICATION

The following technical details are subject to change without notice.

Enclosure	Welded Steel, 430mm x 265mm x 85mm Suitable for standard 19” rack installation - shelf mounting
Video Inputs	2 Sources with internal video timing synchronisation
Video Input Format	Standard Video - 50/60Hz field rate, 15,625Hz line rate Composite Video, PAL,NTSC,SECAM, 1V p-p / 75Ω S-Video, 0.7V p-p / 75Ω
Video Conversion	27 MHz sampling [YUV 4:2:2] Software control of video levels - Brightness, Contrast, Saturation, Hue.
Video Outputs	Software selected type: RGB + Composite Sync, 0.7V p-p /75Ω Composite Video - Software selected 1V p-p / 75Ω PAL, NTSC, SECAM
Display Resolution	720 x 576 Interlaced. No motion artifacts
Number of Monitors	16 monitors - CRTs, Cubes, Projectors or Flat panels. Units can be stacked to support up to 64 monitors.
Effects Sequence Control	MS-DOS software package (runs under Windows ^{TM}). Download sequences for stand-alone operation. Sequences selected using panel switch or optional infra-red handset.
Power	100 VAC to 240 VAC, 47Hz or 440Hz /50W Max.
Weight	7 kg
Environment	Temperature range 5° - 30° (Integral Fan)
Effects Library	Magnify 1 – 8 Switch Video Inputs Colour wash - select different colours on different monitors Freeze Strobe Text and Graphics Overlays - download your logo bitmap, etc. Wipes between Video and Wash Wipes between video sources Zoom to small image Bounce small image Minify image Pixel intensity effects - 'solarise', 'negative' and 'dazzle' Set large pixels - 'pixellate' Other effects in development - see www.mediat.co.uk