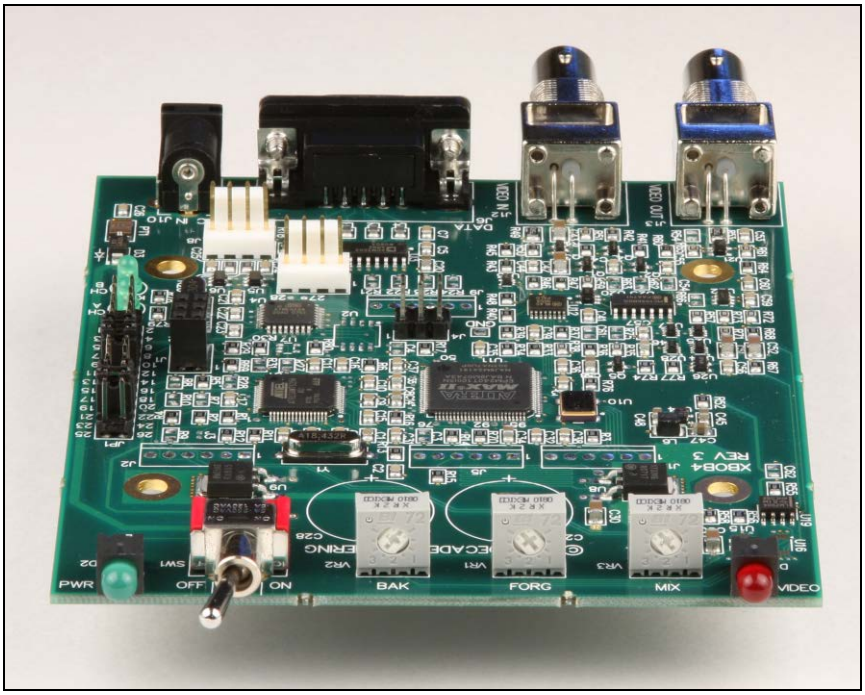
XBOB-4 Application Guide
Firmware V4.3.5 ~ 14 September 2015

*See **www.decadenet.com** for the latest revision of this document.*

XBOB-4GC



XBOB-4G PCB Assembly (example version)

Introduction

XBOB-4 is Decade's fourth-generation low-cost video information overlay board. XBOB-4 lets your PC or other host computer display text and vector graphics on standard TV monitors. With large user-definable character sets, XBOB-4 also supports bitmap graphics and multiple languages. XBOB-4 generates black background video on-board, or automatically genlocks to your video source and superimposes characters over the image. Printable characters and commands drive XBOB-4 through an RS-232 data link, much like a serial terminal or printer. NTSC and PAL video standards are customer-configurable. Field firmware upgrades are supported via PC connection.

XBOB-4 hookup is simple, requiring only serial data, video, and DC power for most applications. See ***System Hookup*** for full details. In its default configuration, XBOB-4 immediately prints plain ASCII characters starting in the upper left corner of the screen. Escape commands are used to modify the print position, change fonts, draw lines, clear the screen, etc. See ***Application Programming*** for details on this topic.

XBOB-4 is a functional replacement for XBOB, the third-generation (BOB-3 core technology) video character overlay device from Decade Engineering. For the sake of clarity in this document, we use "XBOB-3" when referring to the older model, but a Decade Engineering product with that model reference never existed.

Cautions

ESD (electro-static discharge) safety precautions MUST be followed at all times when handling non-enclosed XBOB-4 boards. Use a grounded wrist strap and grounded work surface. Non-enclosed XBOB-4 boards must be stored and shipped in static-shield (metallic, not pink poly) packaging.

To increase awareness of problems that can occur in XBOB-4 application development, refer directly to the *Troubleshooting* section of this document. It may not be pretty, but it's worth a few minutes of your time now if it prevents a big headache later!

Table of Contents

Important Symbolic Conventions

In this document, as in C language compilers, a number prefixed by “0x” is given as a hexadecimal value. For instance, “0x0A” represents 0A hex or 10 decimal.

If a key combination is given, e.g. “Ctrl-J”, this generally means that a terminal program can transmit the desired control code byte by holding down the “Ctrl” key while typing the “J” key.

Single code bytes or byte sequences may also be identified by mnemonics in corner brackets, e.g. “<LF>”. This is shorthand for “Line Feed”, which is a single ASCII code byte with the value 0A hex. Thus “Line Feed”, “<LF>”, “0x0A”, and “Ctrl-J” are just different context-sensitive ways to express a single concept.

“<CSI>” (control sequence introducer) is an important example of a two-byte code sequence (0x1B, 0x5B) more widely known as an “Escape Sequence” because 0x1B is the ASCII escape code. The escape code by itself is often labeled “<ESC>”. Terminals (and terminal emulation programs) normally emit this code when the “Esc” key is struck. If there’s no Escape key on the keyboard (or even if there is), Ctrl-[often works as well. The second byte in <CSI> is a printable character code; the open-bracket or left-square-bracket symbol: “[“. Nearly all XBOB-4 commands are prefixed by <CSI>, so it’s critical to understand how this code sequence can be generated in your host system. See additional comments in the discussion on **Application Programming**.

Specifications

Physical	XBOB-4 PCB nominal dimensions are 4.825 (L) x 3.944 (W) inches, exclusive of controls and connectors. Overall height is less than 0.75 inches for the standard version. See Appendix A for PCB dimensional details. Weight is about 3.56 ounces (101 grams). Ambient operating temperature range is 0~50°C until further notice. For XBOB-4C, the cabinet measures 5.29 (L) x 5.32 (W) x 2.01 (H) inches, exclusive of feet, controls and connectors. Total weight is about 10.94 ounces (310g).
Power Supply	8~15 VDC at 130 mA typical; 150mA max. Requires industry-standard coaxial DC power input plug with 2.1 mm ID and 5.5 mm OD. Center pin connection is positive. Substantially higher power supply current ratings are recommended, to assure adequate rise time and good regulation. The specified 15 VDC upper limit must not be exceeded on a continuous basis.
Data I/O	The main data port is RS-232 serial with XBOB-3 compatible ‘standard’ rates of 1200, 2400, 4800, 9600, 19.2k, 38.4k, 76.8k, and 153.6k bits/S, using eight data bits, no parity, and one stop bit (8N1). The default rate is 9600. Other XBOB-3 standard rates are selectable via configuration shunts. 115.2k bits/S is also available. Arbitrary rates are selectable via software from 46 to 460k bits/S. Most other UART setup parameters are also subject to reconfiguration by command. Software (XON/XOFF) and hardware (RTS/CTS) flow control schemes are implemented, but use is not mandatory. A debug serial port is implemented on the XBOB-4 PCB. This port is hard-coded to run at 115.2k bits/S (8N1), and it requires a 3.3V logic interface. No handshaking is provided.
Video I/O	XBOB-4’s video environment is RS-170A (NTSC) or PAL-B composite baseband, 1Vpp (±10%) 75 ohms unbalanced. The video input tolerates up to 2.5VDC bias mixed with incoming video. The video output contains a small DC bias (+1V), which is common to many video sources and is well tolerated at the inputs to most video equipment. A ‘local’ video signal (black background) is generated by default if video input is not supplied, but users can enforce genlock or local video modes.

Print Speed	Small printable characters from system fonts are normally written to display RAM within a few microseconds after the stop bit is received, so total print delay time is essentially that of the serial interface at low to moderate data rates (e.g. 521uS per character at 19,200 bits/S). XBOB-4 can print more than 7,000 chars/S continuously, but large characters and graphic objects consume increased CPU time roughly proportional to screen area, potentially reducing this rate. Characters may not appear in the display until the next vertical scan cycle, depending on when they are written. If single-frame print timing accuracy is required, host data transmission activity should be triggered from the start of vertical blanking and display position should be near screen bottom. To assist in achieving smooth animation, XBOB-4 can message the host when vertical blanking begins.
Character Format	Character bitmaps can be of arbitrary dimensions up to 255x255, limited only by font storage space. As in BOB-4 modules, 62kB of flash memory is always available for custom fonts (shared with bitmap graphics) as Device 4. XBOB-4 also provides 512kB of supplemental font memory as SPI Device 0. Proportional fonts are supported. Font depth can be one or two bits per pixel. 2bpp fonts support character outline and background features, but render as 1bpp if blinking is enabled. English fonts of assorted size and appearance reside in permanent 'system' memory. Only the XBOB-3 look-alike font, which is the default system font, currently includes European language support. New XBOB-4 fonts can be created or imported and edited with the BOB-4 Conscriptor, a PC program supplied without charge by Decade Engineering. In XBOB-3 compatibility mode, 34 columns and 17 (NTSC) or 19 (PAL) rows of characters may be displayed on overscanning monitors. 40 columns are available if the full raster is used, which is possible with LCD video monitors. 304 character patterns are provided as 12x13 pixel bitmaps, including upper & lower case, italics, European language support, and a set of graphics characters useful for lines, bar graphs, etc. The character set closely replicates that of XBOB-3 prior to firmware version 4, including the default RAM font (but now it's in flash memory).
Graphics	A set of vector graphics drawing commands is provided, including a special command to draw binary data as white/black regions on individual scan lines within the vertical blanking interval. See the *Vector Graphics Commands* section of the XBOB-4 Application Guide for a full description and examples. Bitmap graphics objects up to 255x255 pixels can be imported by the BOB-4 Conscriptor program and stored in XBOB-4 user font memory space (flash). Limitations apply; see BOB-4 Conscriptor Help system for additional information. These objects are treated internally as fonts with a single character: <SP>.
Display Features	Overlay resolution for square pixels is 320x240 in NTSC mode, or 384x288 in PAL mode. Higher pixel rates yield increased display density (up to 480 pixels/line). Only monochrome text and graphics are available. Characters are displayed by default in white with a thin halftone (reduced video intensity) outline. Halftone and black character cell backgrounds are optional, along with many other character rendering variations. Overlay foreground and background levels and overlay transparency are screwdriver-adjustable at the front panel. In local video mode, a full-screen black matte background is automatically supplied. Blinking is selectable by character. "Mirror image" display reversal is available on command, beginning with firmware V4.2.16. The text display window may be reduced to any desired portion of the screen. Vertical scrolling is automatic. A single crawl (horizontal scroll) line can display up to 4096 characters sequentially without disturbing other display elements. The information display may be toggled on or off without affecting the contents of display RAM. Writing to display RAM is permitted with display on or off. A non-volatile boot script memory stores up to 900 (was 512 prior to V4.2.16) characters that may be used to configure XBOB-4 and automatically generate a display at power-up time. The boot script memory has an endurance limit of 10,000 write cycles.

Note: Product specifications, policies and prices are subject to change without notice. Contact Decade Engineering to confirm current status if any specified parameter is critical to your application.

Front Panel Controls

XBOB-4 provides a power switch and green power indicator LED, as well as a red LED to indicate missing video at the video input connector.

Screwdriver adjustments are provided for character foreground and background brightness. They are factory-set to white (full clockwise) and black (full CCW), respectively. A third screwdriver adjustment is provided for setting overlay transparency (MIX). The factory setting is full clockwise, for maximum overlay contrast. The full CCW setting yields zero contrast, making the video overlay disappear. This control may be freely adjusted for best results in each application. Mid-range settings allow background video to show through superimposed characters, and also reduce the crawling effect sometimes seen around character edges when they're placed over intensely colored regions of the image.

Video Modes

This document makes references to the video operating modes offered by XBOB-4. The basic modes are "Local" and "Genlock". Genlock mode may also be called *Overlay* mode, because video generator synchronization (genlock) must be achieved in order to superimpose characters on the image. A third video operating mode, "Automatic," derives from XBOB-4's ability to switch between the basic modes by detecting video input.

XBOB-4 powers up in Automatic. If there's no video input, it selects local mode. In this case, XBOB-4 generates video and characters appear on a black matte background. If video input is present, XBOB-4 switches to genlock mode so that characters are superimposed on the 'remote' (externally generated) video signal. XBOB-4 continues to monitor incoming video and switch between the basic modes as required to maintain video output.

Application programmers can force XBOB-4 to stay in local or genlock modes if desired. Undesired mode switching (to local mode) due to incoming video signal dropouts or glitches can be avoided by forcing genlock mode.

System Hookup

For connection to a PC COM port, XBOB-4 requires a 9-pin "D-subminiature" or "DB9" male/female cable assembly with all pins wired straight through. This is often described as a DCE or modem-style cable. Do not use a null-modem style hookup cable. Here's the RS-232 serial port connector pin assignment for XBOB-4:

Pin	Function
1	Linked to 4 & 6
2	TX data out
3	RX data in
4	Linked to 1 & 6
5	Ground
6	Linked to 1 & 4
7	CTS input
8	RTS output
9	Not connected

The video output of XBOB-4 must be connected to the video input jack of a TV or video monitor, using 75-ohm coaxial cable with a BNC style plug at XBOB-4. TV inputs marked "Cable" or "Antenna" are not suitable. It's not necessary to connect anything to the video input jack on XBOB-4 for a quick operating test, because XBOB-4 generates video when there's no input (the red LED will be lit). To overlay text on video, connect the composite video output of a camera, or equivalent video source, to XBOB-4's video input jack. This connection also requires 75-ohm coaxial cable terminated with a BNC plug.

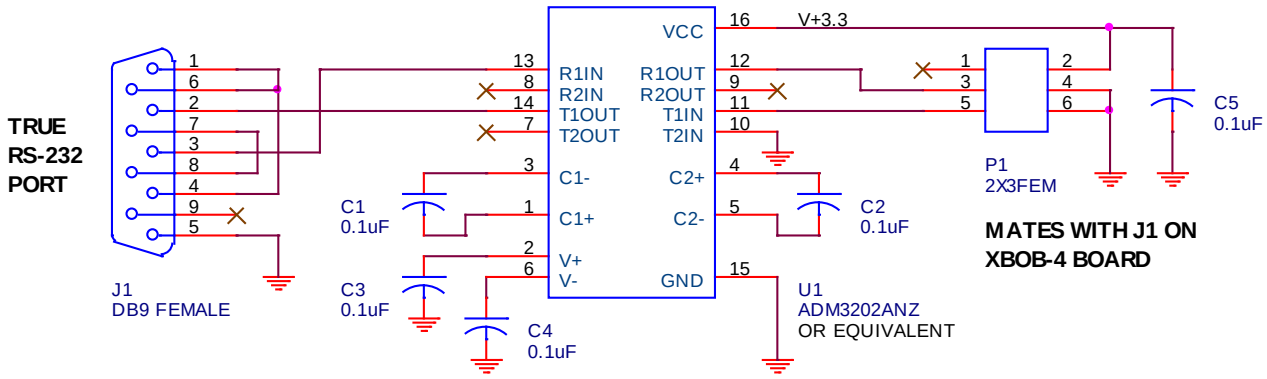
Nominal power supply voltage range is 8~15VDC at 130mA (typical). Decade Engineering recommends the use of power supplies with higher current ratings, e.g. 500mA~1000mA, for better voltage regulation. Use a standard coaxial DC power input plug with 2.1mm ID and 5.5mm OD, and wire the center pin positive. *RadioShack.com* part number 910-0902 is a suitable DC power input plug. Note that unregulated "9VDC" power supplies typically deliver about 12VDC with a light load, and "12VDC" supplies often exceed 15VDC output!

Debug port:

XBOB-4 offers a debug serial port at **J1** on the PCB (cabinet must be removed). Programmers and engineers can use this port as a troubleshooting aid during system development. It may also be used for firmware upgrades and custom font loading, but the main serial port provides these functions more conveniently. Here's the pinout for J1:

Function	Pin		Function
TEST (do not connect!)	1	2	+3.3 VDC output
Debug RXD input	3	4	Ground
Debug TXD output	5	6	Ground

The debug port requires a 3.3V logic interface. Pin 2 provides +3.3V auxiliary power sufficient for typical 3.3V RS-232 interface chips. Here's an interface circuit example:



Using XBOB-4 in a cable TV system:

It's not possible to insert a single XBOB-4 in a cable TV system and display the same text on all channels at once. There are a number of reasons for this. In a cable system, video signals are modulated onto RF carriers at different frequencies (that's how the TV tuner is able to pick out just one). The signals often originate at widely separated locations with no regard for scan synchronization, and individual signal strengths may be poorly controlled.

XBOB-4's input and output are both baseband video. This means that incoming TV channels must be demodulated from RF to baseband in order to place a text overlay on the image. To display the output from XBOB-4 on a standard TV receiver, you must use an RF modulator to re-create a TV broadcast channel, which may then be fed into your cable system for distribution to as many TV sets as required. Each TV set must be tuned to the modulator's output channel in order to view the text.

Of course, you need a demodulator (tuner), a XBOB-4, and a modulator for each TV channel requiring a text overlay. A side-benefit of this arrangement is that you may freely restructure channel assignments in your local cable system. A potential problem is that low-cost modulators are often poorly filtered and may generate interference on adjacent channels. Be sure to use modulators that are designed for adjacent channel operation, or else leave dead channels between the ones you place in service.

Hardware Configuration

XBOB-4 relies on an internal shunt header, or 'jumper platform' marked "**JP1**" for basic hardware configuration. These tables define JP1 shunt options:

Shunt	Name	Default	Purpose
1~2	DF0	X	Choose SPI chip-select signal for supplementary font memory IC.
3~4	DF1		
5~6	DF2		
7~8	SEI0		N/A
9~10	SEI1		
11~12	SEI2		
13~14	EPWR		N/A
15~16	EFIG0		N/A
17~18	EFIG1		
19~20	ACR0		Shunts link configuration lines to ground. Lines are pulled high without shunts. See table below for configuration details.
21~22	ACR1	X	
23~24	ACR2	X	
25~26	ACR3		

ACR0	ACR1	ACR2	ACR3	Baud Rate & Configuration
X	X	X	X	1200bps XON/XOFF
X	X		X	2400bps XON/XOFF
	X		X	4800bps XON/XOFF
			X	9600bps XON/XOFF
X			X	19.2kbps XON/XOFF
		X	X	38.4kbps XON/XOFF
X		X	X	76.8kbps XON/XOFF
	X	X	X	153.6kbps XON/XOFF
	X	X		Default setup: **SPI Master** enabled for supplemental font memory; main serial port defaults to 9600bps; software configuration of UART is allowed.
X				115.2kbps RTS/CTS (hardware flow control)
				115.2kbps XON/XOFF

"X" indicates a shunt plug installed at the designated location. Hardware configuration lines are scanned only once during XBOB-4 system initialization. SPI master mode **must** be selected in order to use the 512kB supplementary font memory chip (Device 0) supplied in XBOB-4. This is the factory default selection.

The BOB-4 Conscriptor loads custom font data into the supplemental memory (Device 0) or a reserved portion of the system processor's on-board flash memory (Device 4) according to the user's selection. See the **v** command with n=56~63 to enable the supplemental memory and configure the SPI clock rate, or use the Conscriptor's Configuration Editor (Comm tab). Additional information can be found in the BOB-4 Conscriptor Help system. Save the new configuration and reboot XBOB-4 before attempting to load custom fonts. After loading, XBOB-4 discovers the new font files and makes them available for printing.

For main serial port UART software setup, see **v** command (n=40~42). Save the new configuration and reboot. Note: The RS-232 interface chip used in XBOB-4 production boards has a maximum bit rate specification of 460kbps.

Communication Basics

Main serial port communication parameters default to 9600bps **8N1** (eight data bits, no parity, one stop bit) and no echo. Communication options are configured with shunt plugs at JP1. See **Hardware Configuration** tables. See **v** command (n=40~42) for main port configuration via software (when allowed by shunts). Configuration shunts are sampled only at system initialization time. Debug port baud rate is fixed to 115.2kbps (8N1).

After a brief start-up delay (<1S), XBOB-4 transmits <XON> to inform the host controller that it's on-line (even if RTS/CTS flow control is enabled). XBOB-4 may send a garbage character or two during initialization. XBOB-4 does **not** echo incoming characters back to the host by default. The debug port is capable of reporting all characters received on the main serial port.

Your host controller must manage data flow control handshaking correctly if you transmit data to XBOB-4 continuously at a high rate. XBOB-4 normally transmits very little data back to the host controller, so reverse flow control was implemented in the hardware method only. Two forward flow control methods are provided:

> **Software flow control:** By default, XBOB-4 transmits <XOFF> (0x13) when the receive data buffer is almost full, and transmits <XON> (0x11) when it's nearly empty. This is an industry-standard technique that is fully compatible with common PC terminal emulation programs such as HyperTerminal[tm]. When software flow control is selected, flow control operates only on the receive channel. That is, <XON> and <XOFF> are transmitted in accordance with receive buffer levels, but <XON> and <XOFF> received from the host are ignored. If you anticipate a large volume of data from XBOB-4, hardware flow control should be used. Since XBOB-4 is a display-oriented device, this is unlikely to cause a problem in most circumstances.
>
> **Hardware flow control:** RTS/CTS handshaking enjoys hardware support in most cases where the host system includes a fairly complete UART implementation. Hardware flow control is generally preferred over software methods due to greater reliability. See **v** command (n=41). Note: When XBOB-4 asserts RTS to stop character reception, it will correctly receive the current character in transit. The user ***must not*** send another character after RTS is asserted, because it will be ignored. Some devices, e.g. certain USB/serial adapter dongles, do not meet this requirement, so a character is lost. This is because the processor asserts RTS when the receive buffer is 100% full. The receive holding register permits only one more character to be received. The XBOB-4 processor's own transmitter meets this requirement when CTS is asserted at the input.

Application Programming

Much like a conventional serial terminal or printer, XBOB-4 accepts plain ASCII character codes to print English text. Other languages (and custom characters) are supported via user-installed fonts and extended character coding. By default, the first printable character appears in the upper left corner of the screen. Escape sequences are used to move the 'cursor' (print position), clear the screen, draw lines, etc, but there's nothing tricky about basic text printing—just send plain ASCII character data.

Standard XBOB-4 commands begin with an 'escape sequence' of two special code bytes: <ESC> (0x1B, Ctrl-[ ), and "[" (0x5B); otherwise known as the <CSI> (Control Sequence Introducer). The *String* start/end commands (see below) are exceptions; they're prefixed by <ESC> only. HyperTerminal generates <CSI> with just two keystrokes: "Esc" followed by "[". In this document, "<CSI>" is used interchangeably with "<ESC>[".

A note for C language programmers: <ESC> may be embedded in *printf* calls with "\033", which specifies the desired character code value in octal format. Anachronistic, but there it is.

XBOB-4 commands are a subset of the ECMA-048 standard. XBOB-4 recognizes most escape sequences in the DOS ANSI command set, except those inappropriate for XBOB-4. Command identifiers are case-sensitive and postfix (subsequent to parameters). Numeric arguments are transmitted in an intuitive variable-length ASCII decimal format, separated by semicolons. XBOB-4 allows negative arguments, but because the minus character (hyphen) is an "intermediate" character in ECMA-048, XBOB-4 uses "<" for a sign character. For instance, <ESC>[<4;30x positions the cursor at pixel position -4, 30. If some parameters are not required, they are simply omitted. Defective commands are generally ignored.

Carriage Return <CR> (0x0D) and Line Feed <LF> (0x0A, Ctrl-J) codes are treated literally in XBOB-4. Both are required to move the character 'cursor' (print position) to the beginning of the next line, unless a **v** (n=11) command has altered this behavior. BackSpace <BS> (0x08, Ctrl-H) moves the cursor back one space and deletes nothing. <BS> is ignored if the cursor is at the start of a line. When the cursor reaches the end of the screen, printed text automatically scrolls upward. Form Feed <FF> (0x0C, Ctrl-L) clears the screen and returns the cursor to top left home position (row 0, column 0).

The ***String*** is a buffer memory that stores up to 4096 characters for use by a subsequent command. The *String* buffer is **not** involved in normal printing activity, but it's used in BOB-4 application programming to select fonts by name and set up crawl displays. Because font selection and crawl commands are allowed in boot scripts, and boot script storage also requires use of the *String* buffer, two forms of the *String* start/end command pair are provided: *String* input may be initiated with <ESC>X and terminated with <ESC>\, or with <ESC>Q and <ESC>R respectively. **The two forms must be nested in order to store a boot script that selects a font by name or invokes a text crawl.** The boot script editor in BOB-4 Conscriptor handles this issue automatically when the *String Start/End* buttons are used. <CAN> (0x18, Ctrl-X) may be issued anytime during *String* data input to abort this operation and leave the *String* empty. Use it if you're working at a terminal and mistype something, because control codes such as BackSpace will be loaded into the *String* buffer if you don't.

Command Example 1: Move the character cursor to the fourth row and 22nd column: <ESC>[3;21H

Command Example 2: Store a simple boot script: <ESC>XHello, World!<ESC>\<ESC>[8v<ESC>[1v

XBOB-4 boots with a big "BOB-4" splash screen display by default. This display is cleared when the first incoming character is detected. To eliminate the splash screen entirely, just clear the default boot script. This can be accomplished by sending <ESC>X<ESC>\ (empty string) <CSI>8v (capture boot script) <CSI>1v (store new configuration).

Complex boot scripts may be created and revised with any text editor that allows control codes such as <ESC> to be inserted in the text. Programmer's Notepad www.pnotepad.org is one example. With Programmer's Notepad, <ESC> is inserted by holding down the Alt key while typing 027 on the numeric keypad. Boot scripts created in this way can be downloaded to XBOB-4 via the file transfer capability of a terminal program.

Bitmap graphics are treated like custom fonts with only one character. To display a bitmap graphic object, just select its font and print a space character (0x20). <CSI>8z or <CSI>18m always selects a *single* custom font installed by the BOB-4 Conscriptor. See **z** and **m** commands if more than one custom font has been installed. Don't forget to select a language font before printing text again.

Pixels and character cells (character row/column locations) in XBOB-4 are numbered from the top left corner, where X=0 and Y=0. On the vertical axis, Y values increase downward. On the horizontal axis, X values increase rightward. In pixel mode (graphics commands and pixel cursor addressing), the screen can be visualized as a window onto a huge canvas. You can draw anywhere on the canvas, but only the part that is inside the window will appear in the display. The canvas extends to -16k and +16k pixels in both axes. See **x** and **q** commands for additional discussion.

Character cursor positioning on a terminal assumes constant character size. XBOB-4 allows different size fonts. Put simply, if you change the font to one of a different size, cursor address units change and confusion sets in. To get the cursor accurately to a given position for a given font size; first select the desired font, then position the cursor. Character cursor positioning is done in the size units of the current font. If a proportional font is selected, then its space character width is used to calculate horizontal character cursor position.

Fonts are selectable by name or index number. Each font can include up to 64k glyphs, memory space permitting. You can easily calculate the number of displayable characters per line for monospaced fonts: Just divide character width in pixels into current scan line length in pixels. The number of overlay pixels in a scan line is a function of pixel clock rate (**v** command, **n**=23) and unblanked horizontal scan time (51.2uS), minus about 15% if the display area constraint is enabled (**v** command, **n**=21). If you're using an LCD monitor, you can get 40 characters per line with the default font by simply defeating the area constraint. XBOB-4 is more compatible with CRT monitors if the display area constraint is enabled, because television CRTs normally overscan, masking a portion of the display perimeter. Full-custom fonts in any desired size may be created by using the free BOB-4 Conscriptor program on a PC.

A special note about the “target” system font: This font includes 13 glyphs intended for drawing target reticles with ‘pseudo-graphics’ technology, plus a space character <SP> (0x20). Transmitting the 13 punctuation character codes immediately following <SP> in the ASCII sequence will print the reticle glyphs.

XBOB-4 uses UTF-8 (8-bit Unicode Transmission Format) character coding, which is compatible with ASCII up to 0x7F, but allows character values up to 65535 (0xFFFF). Character codes from 0x80 to 0x7FF require two bytes with three MSBs of the first byte set to 110 and two MSBs of the second byte set to 10. Character codes from 0x800 to 0xFFFF require three bytes with four MSBs of the first byte set to 1110 and two MSBs of the following bytes set to 10.

Two-byte UTF-8 codes must be used to display non-ASCII characters in the default font. See character code value table in the section titled **System Fonts**. Check the BOB-4 page at www.decadenet.com for a UTF-8 code conversion utility program.

Character Coding Examples:

128 (0x80) (10000000b)
binary: (110)00010 (10)000000
hex: C2 80

300 (0x12C) (100101100b)
binary: (110)00100 (10)101100
hex: C4 AC

1000 (0x3E8) (1111101000b)
binary: (110)01111 (10)101000
hex: CF A8

5000 (0x1388) (1001110001000b)
binary: (1110)0001 (10)001110 (10)001000
hex: E1 8E 88

65535 (0xFFFF) (1111111111111111b)
binary: (1110)1111 (10)111111 (10)111111
hex: EF BF BF

The ECMA-048 standard allows for 8-bit characters (bytes with MSB set). This conflicts with UTF-8; therefore bytes defined in ECMA-048 with the MSB set are not recognized. ECMA-048 also allows for terminal devices that accept only 7-bit characters, so no functionality is lost.

XBOB-4 Commands

A <CSI>n<opt-dot>A Moves the cursor up by **n** character rows.

Ignored if cursor is already at the top of the screen. If **n** is absent, 1 is used. <opt-dot> is an optional "." that converts the argument to pixels if present. For instance, <CSI>30.A moves the cursor up by 30 pixels.

B <CSI>n<opt-dot>B Moves the cursor down **n** rows.

Ignored if the cursor is already at the bottom of the screen. If **n** is absent, 1 is used. <opt-dot> is an optional "." that converts the argument to pixels if present.

C <CSI>n<opt-dot>C Moves the cursor right **n** columns.

Ignored if the cursor is already at the right edge of the screen. If **n** is absent, 1 is used. <opt-dot> is an optional "." that converts the argument to pixels if present.

D <CSI>n<opt-dot>D Moves the cursor left n columns.

Ignored if the cursor is already at the left edge of the screen. If **n** is absent, 1 is used. <opt-dot> is an optional "." that converts the argument to pixels if present.

f <CSI>n;mf Equivalent to **H** command.

H <CSI>n;mH Moves the cursor to row **n**, column **m**. If an argument is missing, zero is used.

For example, <CSI>5;10H positions the cursor on the sixth row and eleventh column; <CSI>8H positions the cursor at the start of the ninth row; <CSI>H positions the cursor at the top left of the screen. Don't use this command prior to selecting a different size font. Use it **after** the new font is selected, based on the new column and row counts. This command sequence returns the current screen dimensions in characters (minus one): <CSI>999;999H<CSI>6n See **n** command for more detail.

J <CSI>nJ Clears part of the screen.

n=0: Clear from cursor to end of screen
n=1: Clear from cursor to beginning of the screen
n=2: Clear entire screen and move cursor to upper left

K <CSI>nK Erases part of the line.

n=0: Clear from cursor to the end of the line
n=1: Clear from cursor to beginning of the line
n=2: Clear entire line

L <CSI>nL Insert **n** lines starting at the current line. If **n** is absent, 1 is used.

The current line and lines below are pushed down. Cursor position is not changed. See comments at **M** command.

M <CSI>nM Delete **n** lines downward starting with the current line. If **n** is absent, 1 is used.

The insert-line and delete-line commands can be used to implement a scrolling region, by deleting a line at the top of the region and inserting a new blank line at the bottom of the region. For example: “<ESC>[5H <ESC>[1M<ESC>[10H<ESC>[1L” scrolls a region between lines 5 and 10.

One little benefit of delete-line in XBOB-4: There are often leftover scan lines at the bottom of the screen because font height is not an even multiple of scan lines. These lines can get partial text on them with the likes of the insert-line sequence. Try filling the screen, then insert one line with <ESC>[5H<ESC>[1L. The bottom line will be pushed into these leftovers and probably truncated along the bottom. To clear the leftovers, you can delete zero lines: “<ESC>[0M”.

m <CSI>nm Set display attributes. Convenient for use in-line with printable characters.

Bold, reverse, and blink are mutually exclusive. For example, if one attempts to set bold and reverse at the same time, reverse takes precedence. The bold setting apparently goes away, but it’s merely suspended. When the reverse attribute is turned off, bold returns.

n=0: Clear attributes (use defaults)

n=1: Start bold (same as **n**=71)

n=2: Start faint (same as **n**=79)

n=5: Start blinking (contingent on global blink enable; see **v** command, **n**=32)

n=6: Start rapid blinking (same as **n**=5)

Note: <CSI>6m is defined by DOS ANSI as rapid blink. In XBOB-4, it’s the same as normal blink. Only one blink rate is possible in XBOB-4, but it can be changed with the **v** command (**n**=33).

n=7: Start reverse video (same as **n**=75)

n=10~19: Select font by index (0~9)

XBOB-4 has eight permanent ‘system’ fonts. If **n**=18 or **n**=19, no change in font selection occurs unless custom fonts have been installed. See **z** command for additional font selection flexibility.

n=22: Stop bold/faint

n=25: Stop blink

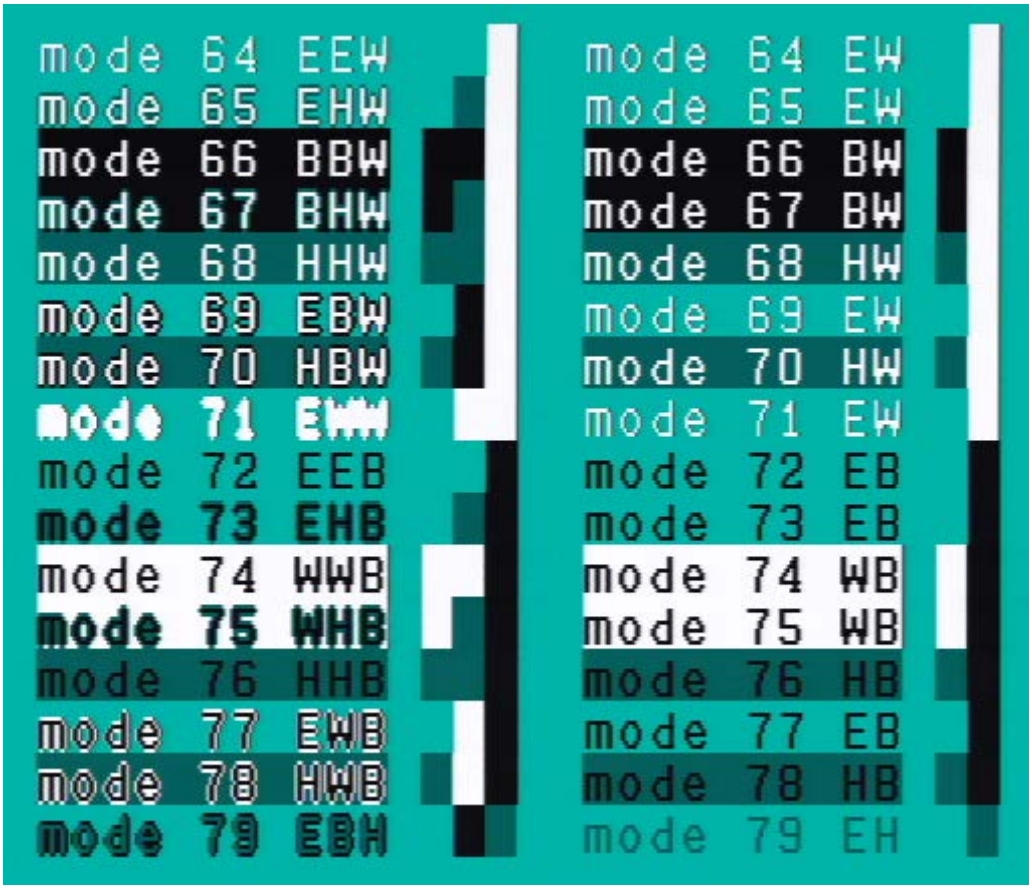
n=27: Stop reverse

n=64~79: Character rendering

In 2bpp (two bits per pixel) fonts, a pixel can be 00, 01 or 10. Halftone pixels are rendered gray in local video mode. See table and illustration below:

E: External video (transparent pixel)
W: White
B: Black
H: Halftone video (darkened pixel)

Render mode (n)	Two bits per pixel			One bit per pixel		Info
	00	01	10	0	1	
64	E	E	W	E	W	
65	E	H	W	E	W	Default
66	B	B	W	B	W	
67	B	H	W	B	W	
68	H	H	W	H	W	
69	E	B	W	E	W	
70	H	B	W	H	W	
71	E	W	W	E	W	
72	E	E	B	E	B	
73	E	H	B	E	B	
74	W	W	B	W	B	
75	W	H	B	W	B	
76	H	H	B	H	B	
77	E	W	B	E	B	
78	H	W	B	H	B	
79	E	B	H	E	H	



n <CSI>6n Report cursor position. Returns “<CSI>r;cR” where **r** = row number, **c** = column number.

When experimenting with this command, note that terminal programs can misbehave due to the <CSI> prefix in the returned data stream. Some terminal programs offer a “decode” mode to handle such non-printing codes correctly.

q <CSI><top>;<bottom>;<left>;<right><opt-dot>q Set up a restricted drawing area (window).

The window normally includes the entire frame buffer, but with the **q** command, window size may be reduced to a minimum of 16x16 pixels. Pixels outside the window are unaffected by subsequent drawing activities. Almost all XBOB-4 (and XBOB-3) commands are modified to work within the window, except text crawl, which always spans the whole frame buffer. This command can implement a scroll block much like XBOB-3, but using entirely different syntax. See additional comments with the **x** command.

<top> and <left> are the number of rows or columns from the top or left of the frame buffer to the top/left of the window (must be positive). <bottom> and <right> are the size in rows and columns of the window if positive, or the number of rows/columns from the right/bottom edge of the frame buffer to the right/bottom of the window if negative. <opt-dot> is an optional "." that converts all four dimensions into pixel units. The cursor is positioned at top left in the new window. For instance, this command sequence makes a tiny window near the center of the screen (in pixel units): <CSI>100;140;200;260.q

<CSI>q or <CSI>0;0;0;0q restores the window to full screen.

s <CSI>s Saves the cursor position. See **u** command to restore cursor position.

t <CSI>n;m;l;k;jt Text crawl (smooth horizontal scroll) control

Crawl text must be loaded into the *String* buffer prior to invoking this command. *String* contents are captured in a crawl buffer and used to repeat the crawl. Arguments not given use defaults; e.g. “<CSI>t” starts an infinite crawl if there’s text in the *String*. A crawl command with an empty *String* buffer disables the crawl.

n=0: Crawl Off (crawl text is not automatically erased)
n=1: Crawl On (default)

m: Crawl line location, in pixels from top if positive, or from bottom if negative (default = -22)

l: Crawl rate in pixels per video field; NTSC = 60 fields/S, PAL = 50 fields/S (default = 2)

k: Gap in percent of screen width between crawl cycles; range is 0~100 (default = 70)

j: Number of times to repeat the crawl message (default = infinity)

Crawl uses the font and render mode active at crawl start time. Font and render mode can subsequently be changed without affecting an active crawl. Display features outside the crawl line are not generally altered, but the use of blinking with crawl can produce undesirable visual effects. Characters may be printed elsewhere on the screen while the crawl is running. You can clear the screen, change screen resolution, and even switch between PAL and NTSC without destroying the crawl. If you happen to draw inside the crawl area, the crawl just overwrites it. There is some delay before the crawl updates, depending on internal timing relationships, so it’s important to test any display strategies that depend on crawl overwriting activity.

Display time for each character is simply its width in pixels times 1/**l** times 1/60. For instance: 12 x 1/2 x 1/60 = 100mS gives the crawl time per character cell for the default font (12 pixels wide), default crawl rate

(2 pixels/field), and NTSC video (60 fields/S). With the default screen width of 480 pixels, or 40 characters, it takes four seconds for this size of character to crawl from edge to edge.

When you're experimenting with the crawl command, avoid loading control codes (e.g. BackSpace) into the *String* buffer. They often cause undesirable artifacts in the resulting display.

u <CSI>u Restores the cursor position. See **s** command to save cursor position.

U <CSI>U Invokes Form Feed (screen clear) upon receipt of the next character

The trigger character is processed as usual, after clearing the screen. This is the last command in XBOB-4's default boot script. Note that <NUL> (0x00) is ignored, so it cannot serve as a trigger character.

v <CSI>n;mv Set/save configuration

Configuration changes (including the boot script) are **not** saved in flash memory until <CSI>1v is received.

For parameter n:

1 = Save the current configuration in flash memory.

Don't put this in a boot script or use it thousands of times. Flash memory wears out.

2 = Restore factory default configuration, including boot script.

8 = Capture boot script from *String* buffer. Boot scripts are limited to 900 characters (was 512).

9 = XBOB-3 compatibility lock. Also see "**{**" command.

m=0: Switching to XBOB-3 mode allowed
m=1: XBOB-3 compatibility mode locked out (default)

10 = Allow <STX> and <ETX> for *String* buffer data entry; for compatibility with older firmware.

m=0: <STX>/<ETX> codes ignored (default)
m=1: <STX>/<ETX> accepted as in firmware V4.1.1 and earlier

11 = <CR> and <LF> interpretation

m=0: <CR> and <LF> are both treated literally (default)
m=1: <CR> does both <CR> *and* <LF>; <LF> is treated literally
m=2: <LF> does both <CR> *and* <LF>; <CR> is treated literally
m=3: <CR> *or* <LF> does both <CR> *and* <LF>

16 = Video standards compatibility

Controls the type of sync generated in local mode. Change the default if XBOB-4 will be used in a PAL environment, so the video standard doesn't change when incoming video drops out.

m=0: NTSC (default)
m=1: PAL

XBOB-4 has automation to help customers in PAL countries. If it's operating in local video mode (no incoming video) while NTSC is the current default, and then external PAL video is applied, XBOB-4 reconfigures itself to make PAL the new default video standard. This procedure works in reverse as well. Note that ALL current configuration settings will be stored! Also see **n**=19.

17 = Scanning mode for locally generated video

m=0: Progressive / non-interlaced (default)
m=1: Interlaced

18 = Video input detector action

m=1: Always generate video – ignore incoming video
m=2: Lock to genlock/overlay – assume that remote video input is always present
m=3: Switch between local/remote video sources automatically (default)

19 = Response to external video standard (NTSC/PAL)

m=0: Do not select video standards automatically
m=1: Use detected standard to select XBOB-4 video standard (default)

20 = HighRate. Sets default pixel clock rates and horizontal size. PixRate (**n**=23) overrides HighRate.

m=0: Off; defaults to 'square pixel' rates of 6.25Mhz for NTSC or 7.5MHz for PAL.
m=1: On; default pixel rate is maximum; 9.375 MHz.

See **n**=21 for display area constraint settings. If HighRate is set, pixel clock rate is 9.375MHz and unconstrained horizontal size is 480 pixels. At 480x240 resolution, the aspect ratio of a pixel is about 1**:**1.5. The 12x13 default font appeals to the eye in this case, while the 8x13 font looks too skinny. 6.25MHz gives a resolution of 320x240 (NTSC), which yields square pixels. For PAL, 7.5MHz yields square pixel resolution of 384x288. The default font looks too fat in these cases, while the 8x13 font appearance is better.

21 = Display area constraint. Insures that overscanning monitors don't mask part of the display.

m=0: Use entire raster; usually safe with LCD monitors
m=1: Constrain display to safe area for CRT monitors (default)

The display area constraint reduces display dimensions by about 15% in both axes. For example, the factory default NTSC display area is 416x208; reduced from 480x240. In the 320x240 NTSC square pixel mode, constrained display area becomes 272x208.

22 = Frame buffer size (reboot required)

Allocates enough RAM at boot time for a frame buffer of given size. Users may ignore this command until further notice.

m=0: 320x240
m=1: 384x288
m=2: 480x240
m=3: 480x288 (default)

23 = **<CSI>23;mv** PixRate (pixel clock rate). Overrides HighRate if **m** is not zero (see **n**=20).

Divide PixRate by 19,531.25 to get display pixels per line (without area constraint).

m=0: Default; HighRate assumes pixel rate control
m=1: 5 MHz
m=2: 5.375 MHz
m=3: 5.769 MHz
m=4: 6.25 MHz
m=5: 6.818 MHz
m=6: 7.5 MHz
m=7: 8.33 MHz
m=8: 9.375 MHz

24 = **<CSI>24;*m*v** Horizontal start position (depends on video mode and pixel rate)

Parameter *m* is the number of unused pixels from left raster edge. If m=0, horizontal start position is set to default value (116). Requested value may be limited depending on pixel rate, video standard, and display area constraint. Actual value in use is shown in configuration report under “Running Parameters” section as “first pixel position.” Negative *m* arguments must be prefixed with “**<**” rather than a minus sign.

25 = **<CSI>25;*m*v** Horizontal size (depends on video mode and pixel rate)

Parameter *m* is display width in pixels, i.e. the number of overlay pixels per horizontal line to be used for display. If m=0, horizontal size is set to default value (416). Requested value may be limited depending on pixel rate, video standard, and display area constraint. Actual value in use is shown in configuration report under "Running Parameters" section as "pixels per line." Granularity is 16 pixels.

26 = **<CSI>26;*m*v** Vertical start position (depends on video mode)

Parameter *m* is the number of unused pixels from raster top. If m=0, vertical start position is set to default value (39). Requested value may be limited depending on video standard and display area constraint. Actual value in use is shown in configuration report under "Running Parameters" section as "first line". Negative *m* arguments must be prefixed with “**<**” rather than a minus sign.

27 = **<CSI>27;*m*v** Vertical size (depends on video mode)

Parameter *m* is display height in pixels, i.e. the number of horizontal lines to be used for display. If *m*=0, vertical size is set to default value (208 for NTSC, 248 for PAL). Requested value may be limited depending on video standard and display area constraint. Actual value in use is shown in configuration report under "Running Parameters" section as "lines per frame."

28 = **<CSI>28;mv** Overlay enable (printing/drawing is allowed even when display is disabled)

m=0: Overlay is blanked, but present in frame buffer RAM
m=1: Overlay is displayed (default)

29 = **<CSI>29;mv** Set local video mode horizontal display offset

m=0~128, with 54 (720nS) being the default.

Local mode display offset relative to genlock/overlay mode is primarily a function of hardware design, but it varies slightly from unit to unit. Use this command only if you need the most precise calibration. Adjustment steps are 13.47nS, or 1/8 of a pixel width at maximum pixel clock rate.

30 = **<CSI>30;mv** Adjust overlay insertion edge rates for NTSC standard

m=0~100000 (defaults to 100000, i.e. 100%, or fastest edge rate)

Slowing the background/foreground transition rates can reduce chroma crawl artifacts. This command is useful primarily with displays that contain only larger characters. Small characters smear excessively when the artifact is eliminated. The control range is nonlinear. Most of its visible effect occurs at settings below 30000.

31 = **<CSI>31;mv** Adjust overlay insertion edge rates for PAL standard

m=0~100000 (defaults to 100000, i.e. 100%, or fastest edge rate)

See notes above, for **n**=30.

32 = Blink enable (global)

m=0: Disable character blinking (default)
m=1: Enable character blinking

XBOB-4 does not implement on/off blinking, but rather white pixels are alternately rendered black. Actual appearance depends on the character-rendering mode (see **m** command with **n**=64~79). Blinking requires global enabling before use, which also clears the screen to prevent odd-looking results. All fonts are then rendered as 1bpp (one bit per pixel). In XBOB-4 mode the global blink enable command is "<CSI>32;1v". The equivalent command in XBOB-3 compatibility mode is "{G1". Globally disabling blink mode clears the screen and restores normal font rendering. This is done with XBOB-4 command "<CSI>32;0v" or XBOB-3 command "{G0". Once blinking is globally enabled, then individual character blinking is switched on with "<CSI>5m" (XBOB-3 "{GE"), and switched off with "<CSI>25m" (XBOB-3 "{GD"). Printable characters transmitted subsequent to these switch commands obey the most recent setting. Blink enable, blink period, and blink duty cycle settings are included in the system configuration report (<CSI>5}).

Blink-enable globally shrinks font pixel depth to one bit (assuming that the current font has two bits per pixel), which limits rendering options. If you must have results more like XBOB-3, then you could implement blinking on the host side. It's a bit tedious to code, but works well as long as the serial com rate is adequate. An alternate attention-getting strategy is to render a portion of text differently instead of blinking it (see **m** command). Another solution: Use vector commands to draw an underline or box around a text field, and flash it by redrawing alternately in transparent and visible 'colors.'

33 = Blink period; **m**=2~100 (tenths of a second). Default is 10 (blink cycle is 1S).

34 = Blink duty cycle; **m**=1~99 (percent). Default is 50%.

35 = Display reversal, or "mirror image" control. Note that cursor control is also reversed as follows:

m=0: Default
m=1: Flip overlay pixels horizontally
m=2: Flip overlay pixels vertically
m=3: Flip both X and Y

36 = Allow printing on scan lines inside the VBI (vertical blanking interval).

m=0~8; default is zero.

Permits printing on scan lines up to 8 lines before the 'legal limit' (i.e. line 21 for NTSC; line 23 for PAL). Parameter **m** is the desired number of usable VBI scan lines before image frame start. The

display area constraint must be defeated (**v** command; n=21), and vertical start position set to zero (**v** command; n=26).

Video monitors do not normally display VBI scan lines, so this command may be used to hide supplementary data within a video signal. You could, for instance, paint line 20 with binary data that is subsequently recovered from video by downstream equipment (custom-built), thus eliminating the need for a parallel data path in your system. The DrawData command (in the vector graphics command set; **%r**) is convenient for this purpose. VBI data isn't guaranteed to survive passage through all possible video processing equipment. Be aware of the potential for trampling video test signals (VITS) and data (Closed Caption, XDS, etc.) on VBI lines that are already in service.

40 = Communication bit rate and sync/async mode control. **m** = bit rate; default is 9600bps

Correct ACRx shunt installation is required; see table in **Hardware Configuration** section. This command has no effect until the new configuration is saved and XBOB-4 is rebooted. For asynchronous serial (RS-232 style) communication, **m** can be any value from 46 to 1000000 (one million), but the RS-232 interface chip used in XBOB-4 has an upper limit of 460kbps. High rates are more granular. To compute the rate, 23.9616MHz is internally divided by **m** and the result is rounded to the nearest integer. Rounding does not compromise the accuracy of industry-standard baud rates. Internal result is in configuration report (see **}** command) after saving configuration.

41 = Communication flow control selection. Reset is required.

m=0: None
m=1: <XON>/<XOFF> software handshake (default)
m=2: RTS/CTS hardware handshake

42 = Communication parameter selection (UART setup).

m is a control byte with bit fields assigned as follows:

Bit Field of m	Value and Resulting UART Setup
Bit 7	Reserved; must be zero
Bit 6	0: No echo (default) 1: Echo on
Bits 5~4	0: One stop bit (default) 1: 1.5 stop bits 2: Two stop bits 3: Two stop bits
Bit 3	0: Eight data bits (default) 1: Seven data bits
Bits 2~0	0: Even Parity 1: Odd parity 2: Space (0) parity 3: Mark (1) parity 4: No parity (default) 5: Reserved 6: Reserved 7: Reserved

Echo mode (bit 6) configures XBOB-4's UART hardware to retransmit every character it receives, which could cause confusion when XBOB-4 is attempting to transmit data in response to some commands. Use this feature with caution.

56 = **<CSI>56;mv** Enable SPI slave device 0 (chip select is SS0\)
57 = **<CSI>57;mv** Enable SPI slave device 1 (not present in XBOB-4)
58 = **<CSI>58;mv** Enable SPI slave device 2 (distance encoder interface IC in XBOB-4E only)
59 = **<CSI>59;mv** Enable SPI slave device 3 (not present in XBOB-4)

m=0: Disable this chip select line; no SPI device present (default)
m=1: Use installed SPI flash memory device on this chip select line
m=10: Use installed XEI encoder interface chip (valid only for **n**=58 in XBOB-4E hardware)

60 = **<CSI>60;mv** Select bit rate for SPI slave device 0; configure to 12MHz (**m**=4) in XBOB-4
61 = **<CSI>61;mv** Select bit rate for SPI slave device 1 (not present in XBOB-4)
62 = **<CSI>62;mv** Select bit rate for SPI slave device 2 (use default for XBOB-4E)
63 = **<CSI>63;mv** Select bit rate for SPI slave device 3 (not present in XBOB-4)

m=2~255 inclusive. XBOB-4 sets the SPI slave clock to 47.9232MHz/**m**. Default is 6MHz (**m**=8), but the supplemental font memory in standard XBOB-4 boards operates reliably at 12MHz (**m**=4). Use the default SPI rate setting for the XEI encoder interface chip in XBOB-4E.

w <CSI>nw Wait for sync. **n** (1~60) is the number of video fields to wait.

Command processing is completely suspended while waiting. Times out after one second even if there's no video. This command is primarily useful in boot scripts that force screen geometry changes, to prevent unwanted display erasure. <CSI>10w should be sufficient in such cases.

X <CSI>nX Erase **n** characters to the right of the print position in the current line. If **n** is absent, 1 is used.

x <CSI>n;mx Positions the cursor in pixel units to column **n**, row **m**. Arguments may be negative.

Remember to use "<" instead of "**-**" for negative values. Commands **A**, **B**, **C**, **D**, and **6n** are switched to operate in pixel units instead of character units after **x** command execution. To switch back, clear the screen, or use the **f** or **H** command.

If you imagine a huge canvas with X and Y ranges of -16k to +16k pixels, the screen is normally a window on the canvas with its upper left corner at 0,0, and extending to the screen size. If the window has been redefined (**q** command), the window on the canvas is the '**q**' window. It is permissible, for instance, to position the cursor left of the window and start printing text, but the first characters would not be displayed. Text may proceed across the canvas, with only the portion inside the window appearing in the display, and continue off to right infinity (actually +16k). Unfortunately the "huge canvas" doesn't really exist in XBOB-4, so it's not possible to just reposition the window and see other parts of the text.

z <CSI>nz Select font by name or index; **n**=0~7 unless custom fonts have been installed (**n**<128).

If **n** is absent, font is selected by name from *String* contents. Example: "<ESC>X20x40<ESC>\<CSI>z" selects the 20x40 font by its *name*. "<CSI>6z" selects the 20x40 font by its index. Font names are case-sensitive. XBOB-4 firmware includes these eight permanent 'system' fonts:

n	Name	X/Y Size	Pixel Depth	Chars	Description
0	bob3	12x13	2	304	XBOB-3 look-alike font (default)
1	8x13	8x13	2	96	English ASCII
2	target	13x13	2	14	Target font; for reticle overlay applications
3	misc	8x14	1	96	Miscellaneous; 1bpp
4	6x10	6x10	2	96	English ASCII
5	13x34	13x34	2	96	English ASCII
6	20x40	20x40	2	95	English ASCII
7	bob4	136x33	2	1	"XBOB-4" bitmap graphic for splash screen

A *single* custom font installed by the BOB-4 Conscriptor can always be selected with "<CSI>8z", but multiple fonts may not occur in the expected sequence. Custom fonts should be selected with the name that was entered in the font name field (not the font file name) of the BOB-4 Conscriptor. Font files may be re-opened in the Conscriptor to find the font name if it has been forgotten. Currently available XBOB-4 font numbers and names are listed in the "<CSI>5}" configuration report (or use the debug port "config" command; its report is identical). This command sequence returns the current screen dimensions in characters (minus one): <CSI>999;999H<CSI>6n See **H** and **n** commands for more detail.

In XBOB-3 compatibility mode, the European language and pseudo-graphics characters in the XBOB-3 font are accessed as they were with XBOB-3, using the **{Tn** command.

{ <CSI>{ Escape to XBOB-3 compatibility mode. Must be repeated after using any XBOB-4 command.

This command is ignored if XBOB-3 mode is locked out (see **v** command with **n**=9). While in XBOB-3 command mode, any occurrence of <CSI> recalls XBOB-4 native mode. The associated XBOB-4 command will be executed. For instance, the XBOB-4 command sequence "<ESC>[2J" reverts to XBOB-4 mode and immediately clears the screen.

| <CSI>n;m| Miscellaneous commands. "|" is the vertical bar, or 'pipe' symbol, not a lower-case "L".

n=1: Clear screen after **m** seconds without incoming data; waits for the next character.

m=0: Cancel this command (default)
m>0: Clear the screen after **m** seconds *and* receipt of any character except <NUL>

n=2: Clear screen after **m** seconds without incoming data. Does not wait for a character after timeout.

m=0: Cancel this command (default)
m>0: Clear the screen immediately after **m** seconds

n=9: Transmit video timing signal.

XBOB-4 can emit an ASCII <FF> Form Feed code (0x0C) at the start of vertical blanking in every field; 60Hz for NTSC, 50Hz for PAL. Helps achieve flicker-free animation.

m=0: Off (default)
m=1: Transmit <FF> at start of V blanking interval

n=1234: Toot our horn.

m=0: Display big “BOB-4” splash screen
m=1: Display designer credits

The XBOB-4 splash screen command does just enough to get the image onto the screen. The cursor remains just to the right of the image and it’s left in **pixel** mode, which means that subsequent text prints off into the ‘infinite’ space to the right of the screen. Clearing the screen or moving the cursor restores normality.

n=3210: Reboot; if **m**=1, start bootloader for firmware download.

} <CSI>n} Transmit system information

n=1: XBOB-4 transmits an XBOB-3 style status string: “VT M01N DE v4.0.4 NTSC”

n=2: Yields a single-byte video status report. The response value is 0x40 with LSBs as follows:

Bit 3: Set if incoming video standard is detected as PAL
Bit 2: Set if external video is detected
Bit 1: Set if PAL video standard selected
Bit 0: Set if video mode is genlock/overlay (external)

n=5: XBOB-4 transmits a complete configuration report

Reported “Running Parameters” are those values currently in use. This command also reads and reports the contents of flash memory, which may differ if the current XBOB-4 system configuration has not been saved. The running baud rate is not in this report, but it is available through the debug port with the debug ‘serial’ command.

n=9: XBOB-4 transmits a screen dump in PGM format

To boost confidence in system operation, the host could examine portions of the screen dump to confirm that XBOB-4 created an expected pattern of pixels in display RAM. Hardware flow control is required with Windows PCs at communication rates above 2400bps. Linux PCs apparently don’t need flow control at rates up to 115.2kbps.

Another way to use the screen dump: Save the output, trim it so "P2" is the first line, then run (on the Linux command line): **`pgmtopnm <your-file | pnmtogif >a-gif-file`**

Vector Graphics Commands

Graphics primitives in XBOB-4 are modeled on PostScript. A path is first created, and then the path is either stroked or filled. **StrokePath** and **FillPath** are the only graphics primitives that draw anything. The **MoveTo**, **LineTo**, **Arc** and **ClosePath** commands are used to construct paths. Note that the graphics cursor is not the same as the character cursor.

Graphics are drawn onto a canvas with X and Y dimensions within -16k to +16k. The part of the screen defined by the 'window' command (**q**) forms a window onto the canvas. Only pixels rendered inside the window will be displayed. For instance, you can draw a small box originating at 1000,1000, but you will never see it. If you draw a box that partly overlaps the window, only the part that overlaps will be visible.

Vector graphics are compressed horizontally if the default pixel rate is used (as are characters). To eliminate aspect ratio distortion, reconfigure for 'square pixels' (e.g. 320 pixels/line for NTSC) or prescale your X coordinates. At high pixel rates, vertical lines are thinner than horizontal lines, and vertical lines of single-pixel width may display poorly (especially on color monitors) due to TV bandwidth limitations.

MoveTo <CSI> x ; y ; z .r Move the graphics cursor to the given X/Y coordinate.

MoveTo and LineTo accept an optional **z** parameter, which makes X and Y relative to the current window. The Y argument is 'rel' / 5, and the X argument is 'rel' mod 5. If X and Y are zero, then **z** values 0~24 yield the screen locations shown:

0 (default)	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

z values 64~79 retrieve data from drawing position memory slots 0~15 respectively. See SavePos command.

LineTo <CSI> x ; y ; z -r ['**-r**' and '**+r**' are equivalent]

Add a line segment to the path, from the current graphics cursor to the given X/Y coordinate. See MoveTo for **z** parameter discussion.

This example draws a line from the upper left corner to the lower right corner of the screen:

```
<CSI>0;0.r
<CSI>0;0;24+r
<CSI>/r
```

Arc <CSI> r ; a1 ; a2 (r ['**(r**' and '**)r**' are equivalent]

Add an arc segment to the path, beginning at the current graphics cursor location and sweeping from '**a1**' to '**a2**' degrees of arc with radius '**r**'. Zero degrees is at three o'clock. Angle increases counterclockwise. Arcs are approximated by drawing small line segments.

For example, "<CSI> 10 ; 0 ; 90 (r" draws an arc from the current cursor up and to the left, forming the upper right quadrant of a circle, finishing at a point 10 pixels up and 10 pixels to the left.

MoveToArc <CSI> r ; a 'r

Moves the graphics cursor to a point on a circle centered at the current position, with radius **r** and angle **a**. Zero degrees is at three o'clock and angle increases counterclockwise.

LineToArc <CSI> r ; a "r

Creates a line from the current position to a point on a circle centered at the current position, with radius **r** and angle **a**. Zero degrees is at three o'clock and angle increases counterclockwise.

SavePos <CSI> <slot> $r Save the current graphics cursor (drawing position).

<slot> = 0~15; corresponding with 'relative' argument 64~79

The current drawing position, resulting from the most recent MoveTo, LineTo, etc., is saved into one of 16 memories and made available to the 'relative' position mechanism. Saved positions are retrieved with arguments of 64 to 79 for the **z** parameter in MoveTo and LineTo, which allows drawing to proceed from a point relative to a saved <slot> value. Note that Bray's Terminal won't transmit # or $ unless duplicated, i.e. ## or $$.

ClosePath <CSI> !r

Add a line segment from the current cursor position to the position of the most recent MoveTo command. The resulting path is closed. For example, to draw a square:

<CSI> 100 ; 100 .r	MoveTo 100,100
<CSI> 150 ; 100 -r	LineTo 150,100
<CSI> 150 ; 150 -r	LineTo 150,150
<CSI> 100 ; 150 -r	LineTo 100,150
<CSI> !r	ClosePath (line to 100,100)
<CSI> /r	StrokePath (white)

StrokePath <CSI> c /r

Stroke the current path by rendering lines for each line segment in the path. The path is not automatically closed before stroking. The path is cleared once it is rendered.

'**c**' is the color to stroke in: **0=Transparent, 1=Halftone, 2=Black, 3=White**; default is White (3).

FillPath <CSI> c #r

Fill the current path by rendering pixels that lie inside the polygon that is represented by the current path. Filling uses the 'even-odd' rule (see wikipedia.org for an explanation). The path is automatically closed if necessary. Note that Bray's Terminal won't transmit # or $ unless duplicated, i.e. ## or $$.

'**c**' specifies color, as in StrokePath.

DrawData <CSI> <line> ; <X-pos> ; <bit-count> ; <bit-width> ; <data-0> ; <data-1> ; <data-2> ; <data-3> %r

Draws binary data as black/white regions on a video scan line, for use by downstream data recovery devices. Once written, data appears in every video field. Data may be written within the VBI (vertical blanking interval) if XBOB-4 is correctly configured; see **v** command with n=36. DrawData acts immediately, without a subsequent StrokePath or FillPath command.

This command takes from five to eight parameters. <Data-0>, <data-1>, <data-2> and <data-3> are 32-bit numbers (in decimal) considered as a string of up to 128 bits, with the least significant bit of <data-0> first and the most significant bit of <data-3> last. The bits are painted starting at horizontal position <X-pos> on video scan line number <line>. <Bit-count> bits from the data words are painted. <Bit-width> pixels are painted for each bit of data: <Bit-width> of 1 paints a single pixel for each data bit, <bit-width> of 2 paints two pixels for each data bit, and so on. Painting too many bits is not an error; the excess just gets lost.

Vector Graphics Examples:

Draw a circle of radius 50 at the center of the screen:

```
<CSI> 50 ; 0 ; 12 .r
<CSI> 50 ; 0 ; 360 )r
<CSI> /r
```

Draw a right triangle at bottom left:

```
<CSI> 0 ; 0 ; 20 .r
<CSI> 50 ; 0 ; 20 +r
<CSI> 0 ; <50 ; 20 +r
<CSI> !r
<CSI> /r
```

Draw three sides of a square:

```
<CSI> 10 ; 10 .r
<CSI> 50 ; 10 +r
<CSI> 50 ; 50 +r
<CSI> 10 ; 50 +r
<CSI> /r
```

A filled square:

```
<CSI> 60 ; 10 .r
<CSI> 100 ; 10 +r
<CSI> 100 ; 60 +r
<CSI> 60 ; 60 +r
<CSI> !r
<CSI> #r
```

Paint the screen white:

```
<CSI> 0 ; 0 .r
<CSI> 0 ; 0 ; 4 +r
<CSI> 0 ; 0 ; 24 +r
<CSI> 0 ; 0 ; 20 +r
<CSI> #r
```

A star (shows the even-odd rule):

```
<CSI> 100 ; 100 .r
<CSI> 100 ; 150 +r
<CSI> 70 ; 110 +r
<CSI> 120 ; 125 +r
<CSI> 70 ; 140 +r
<CSI> 100 ; 100 +r      (same as ClosePath)
<CSI> #r                (center not filled)
```

Here's something a bit more advanced. The following instructions draw a large 'pie' with a missing slice and a smaller pie, also with a missing slice, centered on the circumference of the large circle and located in its missing segment:

<CSI>200;100.r	Move to center of the screen
<CSI>0$r	save location in position 0
<CSI>100;45'r	move to degree 45 on a 100 radius arc
<CSI>100;45;315(r	draw arc from degree 45 to degree 315
<CSI>0;0;64.r	return to saved position 0
<CSI>95;45'r	move to degree 45 on a 95 radius arc
<CSI>1$r	save location in position 1
<CSI>0;0;64.r	move to saved position 0
<CSI>95;315'r	move to degree 315 on a 95 radius arc
<CSI>2$r	save location in position 2
<CSI>0;0;64.r	move to saved position 0
<CSI>5;45'r	move to degree 45 on a 5 radius arc
<CSI>3$r	save location in position 3
<CSI>0;0;64.r	move to saved position 0
<CSI>5;315'r	move to degree 315 on a 5 radius arc
<CSI>4$r	save location in position 4
<CSI>0;0;65.r	move to saved position 1
<CSI>0;0;67-r	draw line to saved position 3
<CSI>0;0;66.r	move to saved position 2
<CSI>0;0;68-r	draw line to saved position 4
<CSI>0;0;64.r	move to saved position 0
<CSI>10;180"r	draw line to degree 180 on a 10 radius arc
<CSI>0;0;64.r	move to saved position 0
<CSI>100;0'r	move to degree 0 on a 100 radius arc
<CSI>0$r	save location in position 0
<CSI>33;135"r	draw line to degree 135 to on a 33 radius arc
<CSI>33;135;405(r	draw arc from degree 135 to degree 405 on a 33 radius arc
<CSI>0;0;64-r	draw line to saved position 0
<CSI>3/r	stroke path

Debug Port Protocol

The debug port can help debug new XBOB-4 applications, load new firmware, and install custom fonts. It can be interfaced to a PC with the information given under **System Hookup** in this document. The debug port is always active. Note: BOB-4 Conscriptor uses the main port or debug port for firmware and font loading, but boot scripts and configuration files can be loaded through the main port only. The Conscriptor's one-click setup feature therefore depends on a main port interface.

Debug port communication settings are fixed as follows:

Bit rate: 115.2kbps
Data bits: 8
Parity: None
Stop bits: 1
Handshake: None
Echo: No

XBOB-4 can get confused if there's activity on the main port and debug port at the same time, particularly if you try to download a new font on the debug port while using the main port for something else. Font downloads use the *String* buffer, which is shared with the main serial port.

Debug Port Commands:

Debug commands require a <CR> termination. These commands are not case-sensitive.

config or **cf** Report configuration.

config n=m Sets configuration variables; n and m are identical to main port **v** command.

config default Restore factory default configuration settings.

config save Store current configuration in flash (make default).

draw fill [color] Paint the active display area with a single pixel value:

[color]	Description
E	Transparent (100% external video)
H	Halftone (50% external video)
W	White
B	Black

help Display available debug port commands.

level [lev] Query or set information reporting level for debug port:

[lev]	Description
0	Returns no data.
1	Transmits video input detection status (default).
2~8	Transmits increasingly detailed reports.
9	Transmits all possible detail, including every character received on the main serial port.

reset [d] Reset the system; if "d" is appended, execute bootloader for firmware download.

serial or **ser** Show main serial port settings.

serial [baud] [flow] Set main port baud rate and/or flow control. Either parameter may be omitted.

This command is effective immediately. Note that stored main port communication settings are read and used by the XBOB-4 system only once, at boot time.

[baud] argument may be any number from 46 to 1000000 (subject to setting granularity).

[flow] arguments:

[flow]	Description
X	XON/XOFF software flow control
R	RTS/CTS hardware flow control
H	Same as R
N	Flow control is disabled

spi Show SPI port settings.

string [data] Appends data to current contents of *String* buffer.

This command targets the same *String* buffer as the main serial port. It's used when loading new fonts through the debug port. Command functionality is limited because space characters and control codes in data are ignored.

stringclear or **strclr** Clears the *String* buffer.

XBOB-3 Compatibility Mode

The factory defaults for XBOB-4 lock it in XBOB-4 native mode. To use XBOB-3 compatibility mode, first unlock it by sending <CSI>9;0v, then escape to XBOB-3 mode with <CSI>{. While in XBOB-3 mode, any occurrence of <CSI> immediately recalls XBOB-4 mode (and the associated command will be executed). Send <CSI>{ again if you must return to XBOB-3 mode. Since the default boot script contains XBOB-4 commands to draw the XBOB-4 splash screen, it invokes XBOB-4 mode. To prevent this, the boot script must be cleared or replaced. Using XBOB-4 commands, the configuration procedure should be:

<CSI>9;0v	(unlock XBOB-3 mode; XBOB-4 will reboot in XBOB-3 mode)
<CSI>21;0v	(disable display area constraint)
<CSI>27;221v	(set vertical size to 221 lines)
<CSI>26;mv	(vertical centering; m=8 for NTSC, m=31 for PAL)
<ESC>X<ESC>\<CSI>8v	(clear the boot script)
<CSI>1v	(save configuration)

Only the XBOB-3 12x13 look-alike fonts are available in XBOB-3 mode. The original XBOB-3 screen layout of 40x17 character cells is not available with overscanning video monitors (most CRTs). It defaults to 34x15 instead. LCD monitors typically display the entire raster, so it's possible to get 40x17 characters by turning off the display area constraint (see XBOB-4 **v** command, n=21). Vertical size should be set to 221 lines (17 character rows), to prevent printing characters beyond the control range of certain XBOB-3 commands (see XBOB-4 **v** command, n=27).

XBOB-3 compatibility mode does not affect the serial data interface. XBOB-4 bit rates may be selected, including those set by the XBOB-4 **v** command (n=40).

Your host application program must manage the software handshake correctly if you transmit data continuously at a high rate. XBOB-4 transmits the <XOFF> character (0x13, Ctrl-S) if the receive data buffer is nearly full, and transmits <XON> (0x11, Ctrl-Q) when it's nearly empty. This is an industry-standard flow control technique that is fully compatible with common PC terminal emulation programs such as HyperTerminal[tm]. If the RTS/CTS hardware handshake is previously configured in XBOB-4 native mode, then it may also be used in XBOB-3 mode.

Initialization time is up to one second. A garbage character or two may be transmitted during this time. In XBOB-3 compatibility mode, XBOB-4 transmits {VT<CR> or {VF<CR>, indicating appearance or disappearance of incoming video.

Characters without the XBOB-3 command prefix (**{**) are interpreted as ASCII text and written to the screen at the current 'cursor' (print position) location. The cursor automatically advances to the next available character cell and wraps to the next line, or back up to the first line as required. Display rows (lines) are numbered from the top down starting with zero. Display columns are numbered from left to right starting with zero. Displayed characters are presented with white foreground and a thin halftone outline by default.

Non-ASCII characters and unsupported ASCII characters are ignored in character translation modes other than 3 and 4 (see **{T** command). In modes 3 and 4, transmit single-byte literal binary values (see character set illustrations in XBOB-3 V3.5 Application Guide) to specify each printable character. Do not send data containing the command prefix character (0x7B) while in translation mode 3 or 4 unless you intend to send a command. All ASCII character codes are supported in mode 0, the default translation mode, except for "**{**" and "**|**".

ASCII **<CR>** (carriage return) normally moves the print position to the left end of the next available line. The **{Z** and **{2** commands change this behavior in ways that are useful for some applications.

Commands sent in XBOB-3 mode must be prefixed by the left curly brace character: **{** All commands employ a fixed-length format, and do not require a command suffix. Command salvos require a **{** prefix to each command in the string. Command letters are not case-sensitive. Exceptions cause the command processor to abort without further action.

XBOB-3 Command	Description
{Ayy	Clears a single row of characters if "yy"=00~16. Clears the entire screen and sets the 'cursor' to top left home position if "yy"=17.
{BE & **{BD**	Display enable/disable. Enabled by default. Display RAM contents are not affected, and characters may be written to display RAM in either mode.
{Cxxyy	Moves print position (cursor). "xx" is the two-digit decimal ASCII column number (00~39) and "yy" is the row number (00~16). "yy" is ignored in scroll mode, but must be present.
{G0 & **{G1**	Blink mode global control. **{G0** (default) uses 2bpp (two bits per pixel) display rendering. **{G1** reconfigures XBOB-4 to permit character blinking, but rendering reverts to 1bpp. Both of these commands also clear the screen.
{GE & **{GD**	Blink enable/disable. Subsequent characters flash or don't flash in the display.
{GCn	Blink duty cycle. "n" = 0~3. 0: Off, 1: 25%, 2: 50%, 3: 75%. Defaults to 50%
{GTb	Blink rate. "b" = 0~1. 0: Default slow (1S), 1: Fast (0.5S)
{K	Returns current print position as **{X-hh Y-hh<CR>**, where "hh" are 2-digit hex numbers indicating column and row where the next printable character will appear.
{MF	Video mode locked to Local Generation.
{MI & **{MN**	Sets interlaced or non-interlaced video generation for local video mode. Defaults to non-interlace (progressive) scanning, which looks best on most video monitors.
{ML	Video mode locked to Genlock/Overlay.
{MM	Video mode selection is automatic (default).
{MP & **{MT**	Selects PAL or NTSC (default) video standards compatibility.
{R	Forces system re-initialization. Restores all defaults, clears display RAM, selects baud rate.
{S	System status query. Return message is: **{ST Vv Mmmi Dd v4.0.4 NTSC** where "v" is **T** or **F** (input video present or not), "mm" is 00~03 (video mode; 00: auto/local, 01: auto/genlock, 02: local, 03: genlock), "i" is **I** or **N** (local video is interlaced or non-interlaced), "d" is **E** or **D** (display enabled or disabled), **v4.0.4** (example) denotes firmware version, and **NTSC** or **PAL** denotes video compatibility mode.
{Tn	Character translation mode. "n" = 0~4. 0: standard ASCII (default), 1: italics, 2: same as mode 0, 3: direct access to ROM characters, 4: supplementary ROM characters. See character set illustrations in XBOB-3 V3.5 Application Guide. Note: Do not send data containing the command prefix character (hex 7B) while in translation mode 3 or 4 unless you intend to send a command.
{ZCn	**<CR>** clears to end of line if "n" = 1. "n" defaults to 0, for normal **<CR>** behavior.
{ZPnn	Sets new print position starting column (after **<CR>**). "nn" = 00~39. Default = 00.
{2C & **{2L**	Sets **<CR>** or **<LF>** to trigger the normal carriage return and line feed response. **<CR>** is default.

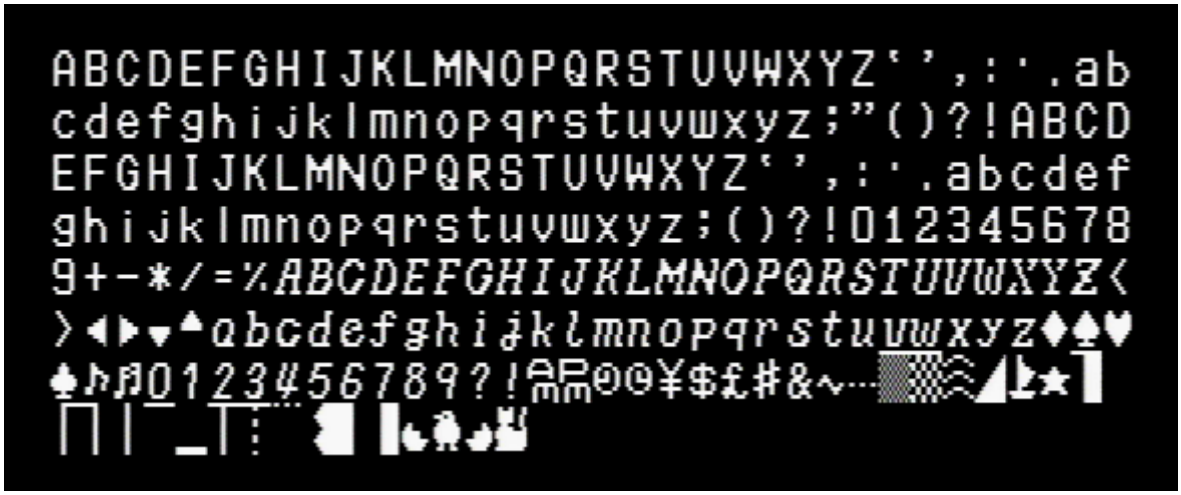
There are new blinking commands in XBOB-3 mode: **{G1** is the global blink enable. **{G0** turns it off. Both commands clear the screen and change the character drawing mode as needed. Note that the whole screen is cleared regardless of window setup (**q** in XBOB-4 command set).

Video standards configuration (NTSC/PAL) is done with the new **{MP** and **{MT** commands.

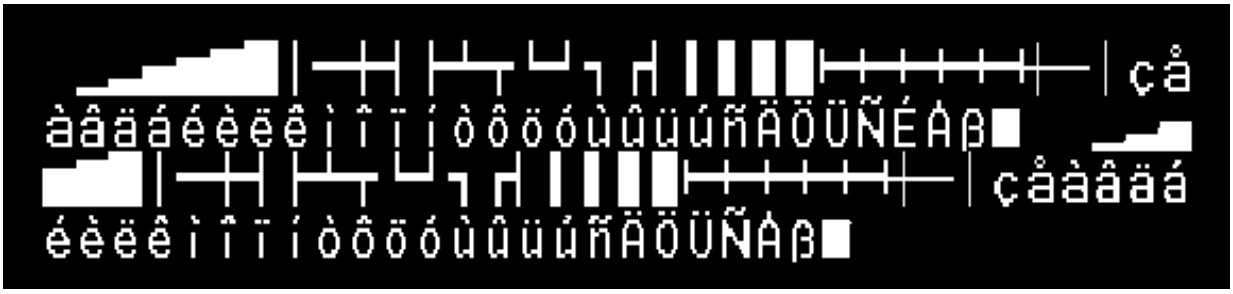
Many of the original XBOB-3 commands are not implemented. For instance, you cannot download a font in XBOB-3 mode or install a boot script or start a text crawl. There is enough functionality to support many but not all applications previously using XBOB-3. Switch to native XBOB-4 mode if more functionality is necessary.

In many cases where XBOB-3 would report an input error by emitting "**{?<CR>**", errors are not reported.

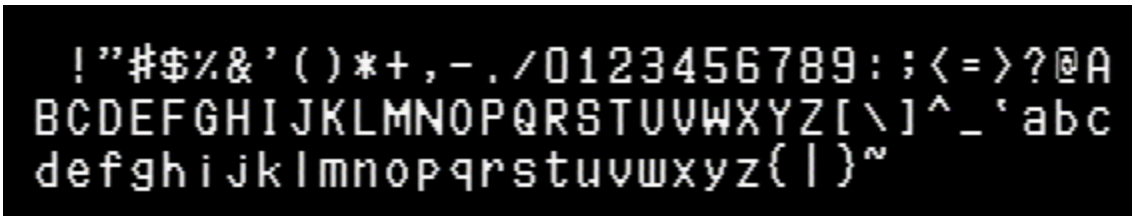
System Fonts



Default 12x13 Font Part 1 (shown in XBOB-3 ROM sequence)



Default 12x13 Font Part 2 (was XBOB-3 Default RAM Font)

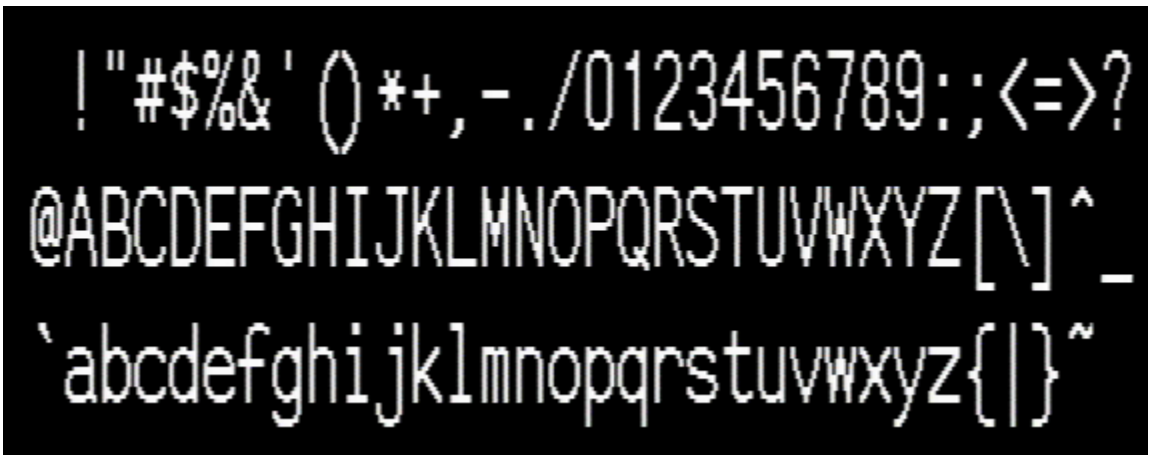


Default 12x13 Font ~ ASCII Sequence

Target Font (uses ASCII punctuation codes 0x20 to 0x2D)

6x10 Font

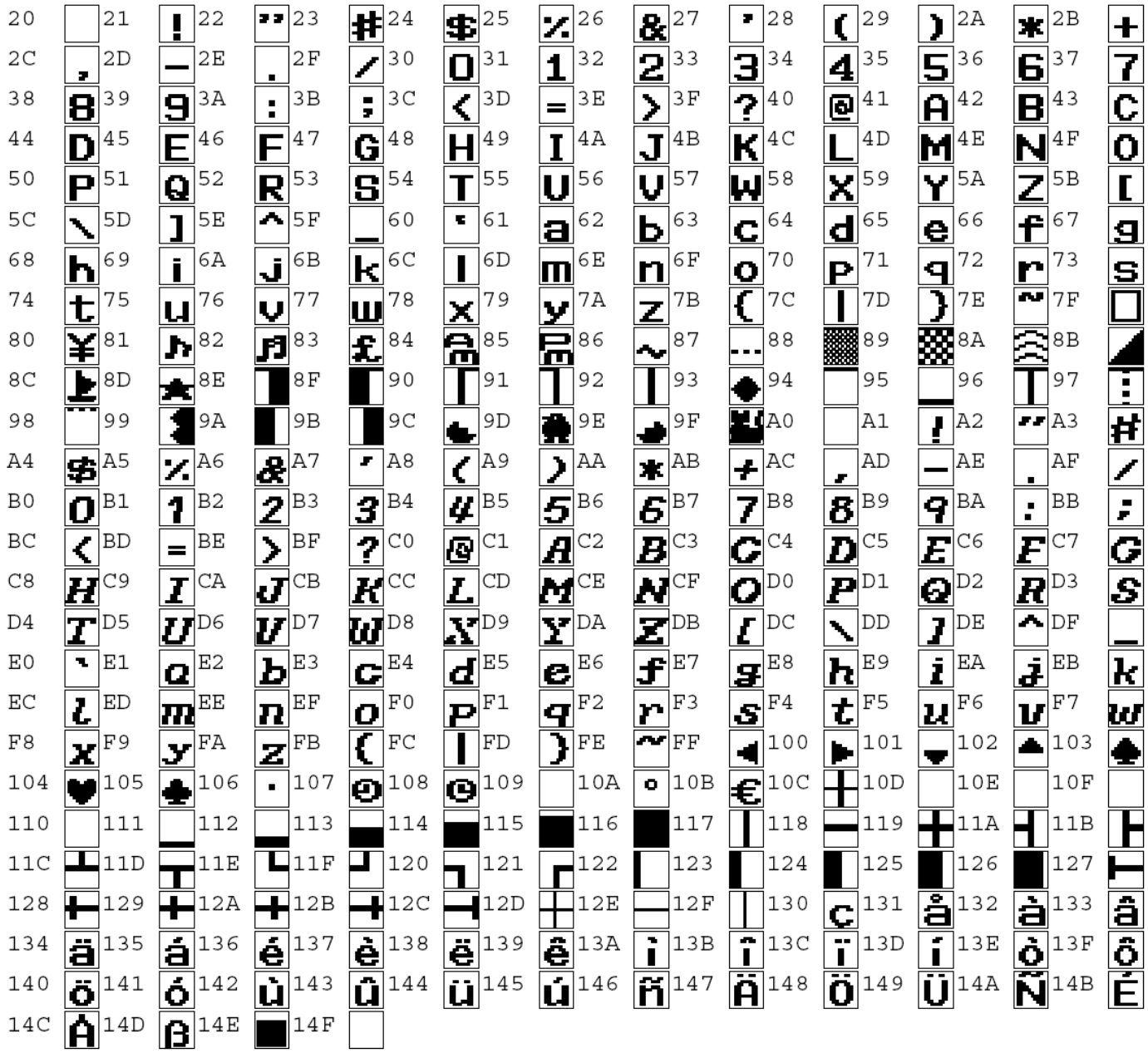
8x13 Font



13x34 Font

20x40 Font

Code assignments for 12x13 default font in XBOB-4 native mode (hex):



Troubleshooting

Firmware upgrade issues:

Don't upgrade firmware with anything other than **.enc** files obtained directly from Decade Engineering! Because these files are encrypted, BOB-4 Conscriptor cannot confirm the presence of valid XBOB-4 program code.

USB/RS-232 adapters can cause firmware upgrade problems even if normal communications with XBOB-4 have been confirmed. It's best to use a 'real' PC RS-232 serial port (COM port) for communicating with XBOB-4. If that's not possible, try more than one brand of USB adapter. IOGEAR model GUC232A has performed well in our shop.

Despite efforts to prevent this, we're suspicious that old XBOB-4 configuration data might sometimes be incompatible with new firmware. Correct default settings for new firmware can be installed by issuing "<ESC>[2v <ESC>[1v" at the main port, or "config default <CR> config save <CR>" at the debug port. The latter option ***must*** be used if main port UART setup (baud rate, etc.) becomes unknown due to mis-configuration. To help bring about a solution, please report trouble events of this type to Decade Engineering. Use the Feedback form at our website for this purpose.

Custom fonts, if any were previously installed, are not erased when firmware is upgraded. They will be discovered and made available for use by the new firmware. Also, custom fonts are not affected by restoring default configuration settings.

No characters will display, or displayed text is mangled:

Don't transmit the <CSI> prefix ahead of printable text. This is the most common explanation for only the first character of a print string to be consistently missing. Because <CSI> is the command prefix, XBOB-4 expects the next character to be a command. If you're especially unlucky, XBOB-4 executes an unwanted command that sends you into the weeds.

If printable characters are sometimes incorrect or missing, or if pacing delays must be inserted into the data stream to eliminate such data communication failures, then bit rate error must be suspected. This is especially likely if your host controller relies on a ceramic resonator instead of a quartz crystal for the master clock oscillator, or if your bit rate computation doesn't yield a correct integer result. All bit rate errors should total well within one percent. Note that commands are probably being clobbered as well, if displayed characters are missing or corrupt.

XBOB-4 acts confused or unresponsive:

Some video monitors (and other downstream equipment) can fail to detect video input when XBOB-4 operates in local video generation mode, resulting in display shutdown. There are at least two known causes for this phenomenon: [1] The monitor is looking for a color burst in the incoming video signal, which XBOB-4 does not supply in local mode. This problem is apparently quite rare. [2] The monitor wasn't designed to accommodate standard-definition (SD) video with progressive sync. XBOB-4 generates progressive sync in local mode by default, but this is easily changed. Use the **v** command (n=17) to select interlaced sync generation. Make this setting permanent, if desired, by saving the new configuration (**v** command, n=1).

Screen geometry changes (pixel rate, size, PAL/NTSC, etc.) always trigger a screen clear operation. These commands are handled asynchronously and sometimes delayed by more than 100mS, but printable character processing continues during this latent interval. To insure that no characters subsequent to a geometry change command are printed and then erased, you must delay for, say, 120ms. It is permissible to transmit <NUL> characters if data transmission cannot be suspended. One <NUL> delays 10 bit times, so at 9600 baud, which can transfer 960 characters per second, you could use 115 <NUL> characters. Also see **w** command (added in firmware V4.2.16).

Customers in PAL countries can encounter a subtle consequence of the foregoing: As shipped from the factory, XBOB-4's default video standard is NTSC. XBOB-4 switches to PAL when it detects incoming video and engages genlock mode. A screen clear operation is triggered because NTSC and PAL have different display geometry. Displays generated by boot scripts can therefore be erased immediately after they're written. This problem can be

fixed by simply re-configuring the default video standard to PAL. See **v** command with n=16, and save the new configuration with <CSI>1v. Also see **w** command (added in firmware V4.2.16).

Terminal programs such as HyperTerminal™ have been implicated in multiple cases of bizarre trouble symptoms, e.g. failure of one or a few commands while others work as expected. If you're communicating with XBOB-4 via HyperTerminal, especially, then try another PC or another terminal program. Bray's Terminal is a free download, and seasoned programmers often recommend it.

USB/RS-232 adapters frequently misbehave. It's best to use a 'real' PC RS-232 serial port (COM port) for communicating with XBOB-4. If that's not possible, try more than one brand of USB adapter. IOGEAR model GUC232A has performed well in our shop.

Observe the one-second-delay requirement after power-up or transmitting a re-initialization command. Be sure to enter complete commands. Some mistakes can be especially confusing if they occur in a boot script. Clear the boot script memory by transmitting <ESC>X<ESC>\ (empty string) <CSI>8v (capture boot script) <CSI>1v (store configuration). Alternatively, just send <CSI>2v (restore default configuration) <CSI>1v (store configuration). Also see next paragraph.

Corrupted boot script or configuration memory can cause seemingly inexplicable problems. This is more likely if experimentation preceded (or might have prevented) operational status. Rogue boot scripts may be completely erased by holding VMIS\ to ground as power is applied, even if XBOB-4 is unable to communicate with the host computer. VMIS\ normally drives the 'missing video' indicator LED. During initialization (and re-initialization), it is an input with internal pullup. If VMIS\ is pulled low during those times, the boot script erase routine is invoked and default configuration is restored! This happens very quickly (in milliseconds), and no confirmation message is emitted. Note: VMIS\ is an internal node in XBOB-4. It's accessible at the end of R3 closest to U10, which is adjacent to the largest IC on the board. Return XBOB-4 to Decade Engineering for service if your technical skills are uncertain!

Flash memory corruption is a common sign of dirty power. In every case, but especially in automotive installations, make sure your power supply is stable and glitch-free under all operating conditions. Substitute a known-good power supply unit, or check the DC output line of your power supply circuit with a good scope. Ripple suppression capacitors can fail with age, and power supply regulator ICs can oscillate vigorously. Low-dropout regulators often exhibit a profound intolerance for certain ranges of load capacitance and ESR. This information is sometimes buried deep in the regulator's datasheet.

Video output display (or captured image) is distorted:

Try another video monitor, digitizer, etc. In rare cases, monitors and other downstream devices can react badly to DC bias in the video output from XBOB-4. If necessary, add a DC blocking capacitor of 470uF or 1000uF, rated at 6V or greater, in series with the video output line. The "+" side of the capacitor should connect to XBOB-4.

Display shifts or disappears when incoming video is cut:

This problem occurs primarily in PAL countries, because XBOB-4 ships from the factory configured for NTSC sync generation when there's no video input. To correct it, just reconfigure XBOB-4 to PAL video compatibility. See **v** command (n=16).

Display doesn't fit the monitor screen:

See **v** commands (n=24~27) to adjust horizontal and vertical start position and size. Also see **v** commands (n=21) to control the display area constraint. LCD monitors typically reveal most or all of the raster, so there's no need to constrain XBOB-4 display area. CRT monitors mask the image perimeter due to overscan. Overscan in low-cost CRTs is typically greater, and it varies with image brightness and power supply voltage. Display centering on CRT monitors can also be imprecise. It's best to adjust display centering with reference to a test pattern from a video signal generator or DVD player.

Dot crawl or sparkle on character edges:

When characters or graphics are superimposed on strongly colored regions of an image, XBOB-4 can exhibit 'chroma crawl' artifacts. This is a symptom of chroma-luma crosstalk in the composite video signal, which results from the simple overlay insertion technology used in XBOB-4. We couldn't eliminate this issue without pushing up the price, but there are several ways for customers to minimize it:

1. Position the data display over a region in the image with low color saturation, or reduce the overall color saturation of your incoming video. Color is commonly oversaturated in video from some sources; such as TV broadcast and cable signals.

2. Character outlines are halftone (50% video) by default. Try render mode 69 (see **m** command) to make outlines black, or mode 66 to fill the character cells with black background. Mode 66 is the most reliable solution because it completely suppresses chroma inside the character cell. Alternately, paint a region of the screen black (see **Vector Graphics Commands**) before drawing characters over it.

3. XBOB-4 offers hardware and commands to make overlay insertion edge rates programmable. Because reduced edge rates necessarily soften character edges, this feature works best with oversize fonts. See **v** command with n=30~31.

4. Reduce overlay contrast by setting the transparency (MIX) trimmer below 100% contrast.

Thin vertical lines are not as bright as horizontal lines:

XBOB-4 can generate pixel timing that is too fast for many video monitors (and other equipment), particularly color monitors. High-resolution monochrome TV monitors perform better in this respect, but that's not a useful option for most customers. The best fix is to use fonts with vertical character features at least two pixels wide.

The text overlay is unstable:

Display impairments such as vertical instability or 'flagging' can appear on LCD video monitors when XBOB-4 is operating in local mode. This problem is often corrected by configuring XBOB-4 to generate interlaced sync instead of progressive (non-interlaced) sync. XBOB-4 generates progressive sync in local mode by default, because it reduces flicker and looks best on most monitors. See **v** command (**n**=17).

Overlay jitters can be caused by weak and/or noisy video input. Typically, the video signal has been attenuated by passage through a long cable (or double termination). The best cure for long cable woes is a robust cable drive amplifier with pre-equalization for cable loss characteristics. Decade Engineering offers a Camera Adapter Board (CAB) with broad adjustment ranges and high drive capability for this purpose. A cable compensator or "Proc-Amp" (video processor) at the receiving end may also be suitable. Long cables are subject to noise injection from a variety of sources, including ground loops, so the cable receiving circuit may have to deal with several kinds of signal defect simultaneously. Coaxial cable losses in the baseband video spectrum are notoriously nonlinear as a function of frequency, making long cable compensation a distinctly non-trivial exercise.

XBOB-4 wasn't designed to work with VCR playback signals. It may perform as desired, but overlay stability can be poor with some VCRs and some (usually worn) cassettes. Performance is generally worse in special-effects modes such as freeze-frame. The new breed of low-cost frame synchronizers presents a possible solution.

Video received through RF transmission systems can become highly distorted due to multipath interference and fading. XBOB-4 does not tolerate distorted sync as well as a typical TV monitor, making data overlay stability uncertain in these system environments. If XBOB-4 (or BOB-4) cannot be located at the transmitting end of the system, consider using a diversity receiver. A proc-amp or frame synchronizer at the receiving end may also be necessary.

Failure to detect incoming video and switch to genlock/overlay mode:

The digital video detection algorithm in XBOB-4 with firmware prior to V4.2.16 can fail to recognize nonstandard or 'simplified' sync waveforms, such as the blue background video that is produced by default in certain video source devices. The **v** command (n=18) may be used if necessary to force genlock/overlay mode.

Firmware Revision History

V4.3.5 [29 July 2014] Added support for Adesto AT45DB041E DataFlash chip for font memory expansion.

V4.3.4 [18 December 2013] Fixed a bug that devoured much of the locally-generated vertical sync pulse, but only after altering the video standard or scanning mode and prior to saving those configuration changes in flash, and only in XBOB-4 boards using revision C or D processor chips (Atmel AT91SAM7S256).

V4.3.3 [27 June 2012] Increased text crawl length limit from 1024 to 4096 characters. Revised the hardware-triggered recovery procedure to include restoration of default configuration settings and boot script erasure.

V4.3.2 [13 April 2012] Fixed a bug in vector graphics processing that sometimes caused lines to be drawn in the wrong place. The vector path may now be saved (**SavePos**) and used later when drawing, and there is a new command for moving to a point on an arc (**MoveToArc**), allowing radial positioning.

V4.3.1 [19 January 2012] Improved SPI communications reliability and added a configuration command for SPI clock mode (**v** command; n=44). This revision affects BOB-4 modules, but has no practical effect in XBOB-4.

V4.2.18 [] Equivalent to V4.2.17; accommodates changes to factory test procedures only.

V4.2.17 [16 September 2008] Added VBI data drawing capability, including a new **v** command (n=36) and DrawData (**%r**) command. Fixed **u** command to allow repeated usage.

V4.2.16 [05 March 2008] Interim release. Added mirror image control and ADC input command (useful only in BOB-4H). Fixed bug in font loading that was present only in interim release 4.2.11. Fixed hardware flow control. Increased boot script size limit to 900 bytes. Added V2 font file format to accommodate fonts with thousands of glyphs, e.g. Chinese. Fixed minor display quality issues with 8x13 font. Added degree, Euro, and alternate crosshair glyphs at codes 266~268 (decimal) in default font. Optimized video timing for 74.25MHz and 75.00MHz master clocks. Fixed incorrect path closure in vector drawing system. Enabled synchronous serial clock input (still not tested). Modified video detector to accept 'simplified' sync. Added crawl message repeat control. Added SPI device type to accommodate the distance encoder interface in XBOB-4E. Added **w** (wait for sync) command.

V4.2.10 [08 May 2007] First XBOB-4 production release firmware.

Decade Engineering Contact Information

Please check our website for the most recent version of this document before concluding that a defect exists. Hardware warranty and service information is posted within the online ordering system. See below for software warranty statement.

Phone	503-743-3194
Fax	503-743-2095
Post	5504 Val View Dr. SE, Turner, OR 97392 (USA)
Email	Use Feedback/Contact form at website
Web	www.decadenet.com

Obligatory Boilerplate

Software Warranty Statement:

All software in XBOB-4 is provided “as is”, without warranty of any kind, express or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose and noninfringement. In no event shall Decade Engineering be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if Decade Engineering is advised of the possibility of such damage.

Appendix A ~ PCB Dimensions

All dimensions are given in inches.