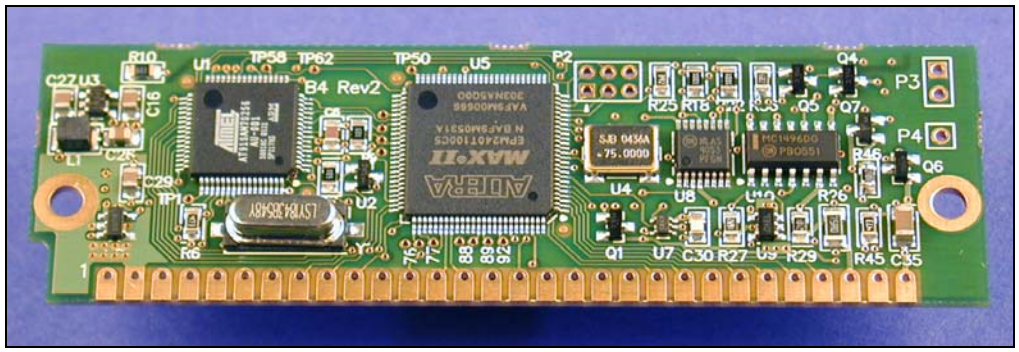
BOB-4 Application Guide ~ Firmware V4.1.1 ~ September 22, 2006

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



Introduction

BOB-4 is Decade's fourth-generation low-cost video information overlay module. BOB-4 lets your microcontroller or PC display text and vector graphics on standard TV monitors. With user-definable character sets*, BOB-4 also supports bitmap graphics and multiple languages. BOB-4 generates background video on-board, or automatically genlocks to your video source and superimposes characters over the image. Printable characters and commands control BOB-4 through SPI or RS-232 style data links, much like a serial printer. BOB-4 links directly to most microcontroller chips and modules, including the BASIC Stamp[tm]. NTSC and PAL video standards are supported in a single product version. Field firmware upgrades are supported via PC connection.

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

* This feature is not present in early production firmware.

Cautions

ESD (electro-static discharge) safety precautions must be followed at all times when handling BOB-4 modules. Use a grounded wrist strap and grounded work surface. BOB-4 modules must be stored and shipped in static-shield (metallic, not pink poly) packaging. Use a SIMM socket! Soldering directly to BOB-4 (except at J1) voids the warranty.

TVS diodes protecting the main serial port TXD/RXD lines against ESD hazards may not be installed in BOB-4 engineering samples. Please take extra care to avoid zapping these lines!

Table of Contents

Video Modes

This document makes references to the video operating modes offered by BOB-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 BOB-4’s ability to switch between the basic modes by detecting video input.

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

Application programmers can force BOB-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.

Specifications

Physical	BOB-4 is initially designed in the 30-pin SIMM form factor, 3.50 x 1.05 x 0.35 inches. Another form factor is expected later in 2006. Module weight is about 0.35oz/9.8g. Ambient operating temperature range is 0~50C. BOB-4 will become nominally RoHS-compliant in 4Q 2006.
Power Supply	BOB-4 requires +5VDC regulated to ±5% at 100mA typical. The power supply should be sized to accommodate 150mA or more, and protected against fault currents above 250~300mA. A low-current auxiliary +3.3VDC regulated output is available. Auxiliary supply output current adds to main supply current drain. Maximum +3.3V load current has not been characterized at this writing.
Data I/O	The data path is synchronous or asynchronous serial with BOB-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. BOB-3 standard rates are selectable via pin strapping—see Pin Descriptions. 115.2kbps and synchronous (not yet tested) rates are also available. Arbitrary rates are selectable via software from 46bps up to 998.4kbps. Software (<XON>/<XOFF>) and hardware (RTS/CTS) flow control schemes are implemented, but use is not mandatory. An SPI interface option will be available. The SPI port may be used for BOB-4 control or font memory expansion, but cannot simultaneously serve both purposes. Logic outputs are 3.3V CMOS except pin 20. Logic output current is 8mA max except pin 15 (16mA). Logic inputs are TTL-compatible (max logic low: 0.8V, min logic high: 2.0V) and 5V-tolerant (except pins 12, 13, 14), but hard 5V drive is not allowed during reset. Only pin 9 (RXD) offers full 5V input tolerance.
Print Speed	Small printable characters from internal 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). BOB-4 can print more than 7,000 chars/S continuously, but large characters and graphic objects consume 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 (logic output available) and display position should be near screen bottom. BOB-4 can message the host when vertical blanking begins.
Video I/O	BOB-4’s video environment is RS-170A (NTSC) or PAL-B composite baseband, 1Vpp 75 ohms unbalanced. Y/C video (S-Video) can be accommodated with external circuitry. 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. Video control and timing signals are available by software command at logic-level outputs.

Character Format	Character bitmaps can be of arbitrary dimensions up to 256x256, limited only by font storage space. 62kB internal font memory is provided. External font space is defined by the size of available SPI memory chips. 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. Fonts of assorted size and appearance are pre-loaded in font memory. Only the BOB-3 look-alike font currently includes European language support. New BOB-4 fonts can be created or imported and edited with the BOB-4 Conscriptor, a PC program supplied without charge by Decade Engineering. (Custom font loading is not possible with early release firmware.) In BOB-3 compatibility mode, 34 columns and 15 (NTSC) or 19 (PAL) rows of characters may be displayed on overscanning monitors. 40 columns are available if the full raster is used. 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 BOB-3 prior to firmware version 4, including the default RAM font (but now it's in flash memory).
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 backgrounds are optional, along with many other character rendering variations. In local video mode, a full-screen black matte background is automatically supplied. Blinking is selectable by character. Manual adjustment of overlay transparency is optional, with external circuits. 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 1024 characters sequentially without disturbing other display elements. The entire overlay 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 512 characters that may be used to configure BOB-4 and automatically generate a display at power-up time.

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.

System Hookup

Basic Hookup:

Important Notes

1. An outboard RS-232 hardware interface is **required** for connection to a PC COM port or any other 'genuine' RS-232 host port. See next section for a recommended circuit.

2. BOB-4's data input and output lines may be connected directly to a microcontroller IC. In this case, an RS-232 hardware interface circuit must **not** be used! There are a few exceptions to this rule, e.g. microcontroller modules or developer boards with integrated RS-232 hardware interfaces. See note #1 in this case.

3. Pin 2 and pin 29 must be linked through a PCB ground plane or other low-impedance connection. Use 22AWG or larger wire (or multiple wires) in hand-wired installations or development fixtures.

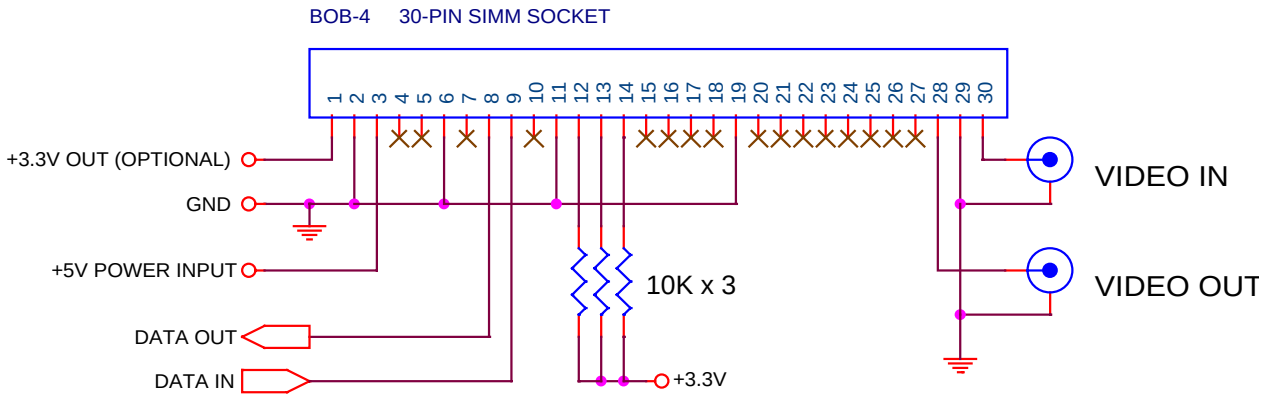
4. With the exception of pin 9 (RXD), BOB-4's logic inputs can be permanently damaged by direct application of 5V from a low-impedance source. Please observe all notes on this topic carefully if BOB-4 is being integrated into a 5V system.

The drawing below shows a typical BOB-4 installation using the asynchronous serial 'TTL-232' control port. The data rate defaults to 9600bps with <XON>/<XOFF> software flow control. 10K pullup resistors (to +3.3V) are recommended at pins 12, 13, and 14 unless no conductors are attached.

Pin 11 must be lifted to enable software bit rate selection or the SPI port. Add a 10K pullup on pin 11 if it's not grounded. Note that main port bit rate defaults to 115.2kbps if pin 11 is floated or pulled up. Pin 6 must not be grounded if RTS/CTS flow control (the hardware handshake) will be implemented.

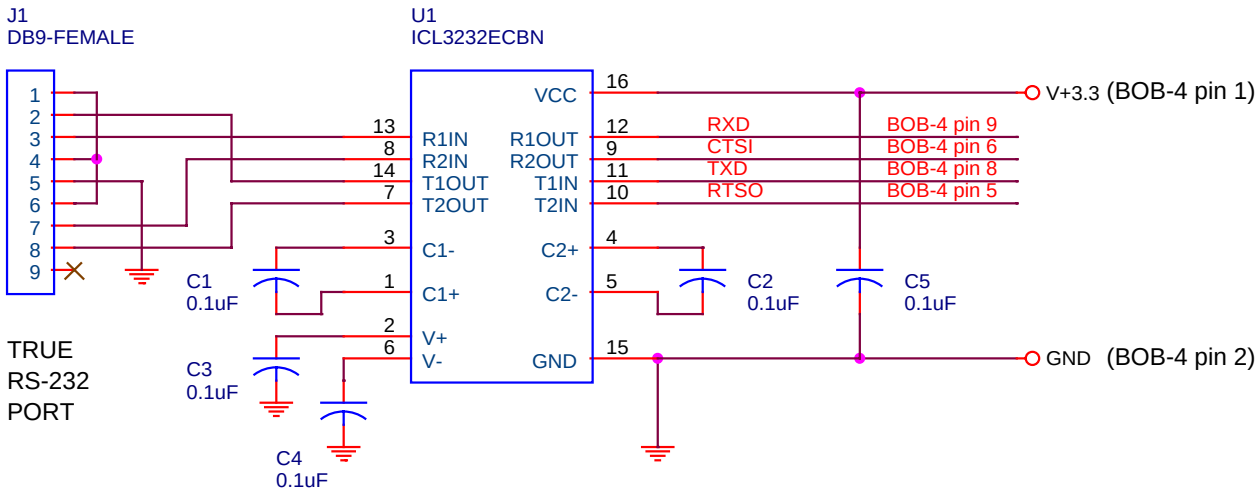
An alternate and recommended hookup: To boot at 115.2kbps with RTS/CTS flow control, ground pin 12 and pull up pins 11, 13, and 14. Pins 5 & 6 are used for RTS and CTS.

Some optional BOB-4 features are not implemented here, but are discussed in other parts of this document.



RS-232 Hardware Interface:

BOB-4's RXD and TXD signals can connect directly to the UART TXD/RXD pins of most microcontrollers, but you need an inverting RS-232 hardware interface if you want to connect to a PC serial port or any other 'true' RS-232 host system. Here's a PC COM port interface circuit example:



In the RS-232 interface example above, the PC's DCD, DTR and DSR signals are looped back, while RTS and CTS are routed through to BOB-4. The only host connections actually *required* are serial data in/out and ground. If you don't use RTS/CTS flow control, then it's usually best to link J1 pins 7 & 8 so you don't have to deal with PC handshake setup issues. Be sure to ground U1 pin 10 if it becomes an unused CMOS logic input. J1 pin connections are for a standard male/female 9-pin modem cable with all pins wired straight through. This hookup will not work with null-modem cables or any other serial data cables with internal cross-connections.

BOB-4 does not output 3.3V power on pin 1 by default. See Pin Description Table for this option.

The suggested Intersil ICL3232ECBN (SOIC package) chip incorporates 15kV ESD protection, but you should add protective networks in the RS-232 signal lines if your system environment isn't benign. EMI filtering may also be required.

RS-232 interface chips don't necessarily support the highest communication bit rates offered by BOB-4. The ICL3232ECBN, for instance, goes to 250kbps with 1000pF loads or 900kbps with 250pF loads. If you're pushing the bit rate, be sure to check this specification as part of your component selection process. Confirm that cable capacitance does not exceed interface chip drive capability at your system's maximum speed.

Many other circuit variations are possible, including the simple receive-only transistor circuit described in the BOB-II literature, but most BOB-4 applications will benefit from bi-directional communication and flow control. This is especially true at the highest baud rates.

Logic Interface Issues:

Logic level mismatching must be considered when BOB-4 is installed in 5V systems. This reference is a good summary of simple logic interfacing techniques: www.edn.com/article/CA6335309.html?ref=nbra

Pin Description Table:

Pin	Name	Dir	Description
1	Optional +3.3V output	X	Not connected by default; add solder blob at J1 for optional low-current +3.3V regulated DC output (5mA max until further notice).
2	GND	X	Ground (power & digital)
3	V+5	In	+5V ±5% 100mA (typical) regulated power supply input
4	SS0	I/O	SPI slave select input, SPI master select 0 output
5	RTSO	Out	Main serial port RTS output; hardware handshake signal
6	CTSI/CPOL	In	Main serial port CTS input or SPI mode control; suggest pulling down or grounding if RS-232 hardware handshake is not used. See *SPI Details* for SPI usage. See *Hookup Notes* for BOB-3 compatibility.
7	SS2	Out	SPI master select 2 output
8	TXD	Out	Main serial port output; protected by TVS diode and series resistor
9	RXD	In	Main serial port input; protected by TVS diode and series resistor
10	RST\	I/O	System reset input (low true). No connection is required, but pullup is suggested if RST\ is implemented in noisy systems. See notes.
11	SS1	I/O	SPI master select 1 output, SPI mode select primary input (strap low for BOB-3 compatibility)
12	ACR0/SPCK	I/O	Baud rate select input, SPI clock I/O; not 5V-tolerant
13	ACR1/SPM0	In	Baud rate select, SPI mode select 0; not 5V-tolerant
14	ACR2/SPM1	In	Baud rate select, SPI mode select 1; not 5V-tolerant
15	VMIS\	Out	Logic output; low if video not supplied to pin 28; 16mA max. May be unstable during initialization. Add series diode with LED if returning to V+5.
16	SCK/CPHA	In	Main serial port clock input (synchronous mode only) or SPI mode control. See *SPI Details* for SPI usage.
17	DTX	Out	Debug serial port output; 115.2kbps only, no added ESD protection
18	DRX	In	Debug serial port input; 115.2kbps only, no added ESD protection
19	TEST	In	Factory test; suggest grounding in application circuit
20	L0	Out	Auxiliary logic output; open collector with 2.1K pullup to V+5
21	MISO	I/O	SPI data I/O
22	MOSI	I/O	SPI data I/O
23	CLVID	Out	Clamped 1Vpp video output; unbuffered, for external transparency control. Sync tips are clamped to zero volts.
24	L1	Out	Auxiliary logic output, controlled by application
25	L2	Out	Auxiliary logic output, controlled by application
26	FBLKG	Out	Auxiliary logic output (or fast blanking output, for external S-Video chroma channel processor)
27	L3	Out	Auxiliary logic output, controlled by application
28	CVOUT	Out	Video output; drives 1Vpp into 75 ohms. Included DC bias brings sync tips 0.3V~0.5V above ground with 75 ohm termination.
29	GND	X	Ground (analog)
30	CVIN	In	Video input; internal 75 ohm termination (1/4W max), 1Vpp required

Hookup Notes:

- Grounds at pin 2 and pin 29 must be linked through a low-impedance ground plane in the host PCB.
- Logic outputs are 3.3V CMOS except pin 20. Logic output current is 8mA max except pin 15 (16mA). Logic inputs are TTL-compatible (max logic low: 0.8V, min logic high: 2.0V) and 5V-tolerant (except 12, 13, 14), but **hard 5V drive is not allowed during reset.** Only pin 9 (RXD) offers full 5V input tolerance.
- L0~L3 and FBLKG are programmable to output various video timing signals as well as fixed logic states. They all default to logic low. L0 has a special 5V open-collector-with-pullup output characteristic. The logic high level for L0 may be clamped near 3.3V if necessary, with an external schottky diode. FBLKG can be a

gating signal for external Y/C (S-Video) chroma channel processing circuits. See **v** command (n=48~52) for port configuration details.

- RST\ must go low for at least 80us. A much shorter pulse can reset the PLD, so noise on RST\ could reset the PLD but not the CPU. **Do not drive this open-drain node active high**. The CPU pulls RST\ low during reset to extend reset time. We suggest adding a 10K pullup to 3.3V if RST\ is used in the customer's application, to minimize EMI sensitivity.

- BOB-3 used pin 6 for RX Enable, defaulting high true via internal pullup. This feature is not duplicated in BOB-4. The SPI interface offers improved control of multiple BOB-4 modules by a single host, with pin 4 (SS0) serving as the SPI slave enable input. An external AND gate could emulate BOB-3's RX Enable feature if necessary.

- Decade Engineering recommends the addition of 10K pullup (to +3.3V) resistors at pins 12~14 to reduce EMI sensitivity in noisy systems, unless they are tied permanently high or low for baud rate selection. If pin 11 is not grounded, a 10K pullup should be added there as well. Weak internal pullups are provided, but wrong configuration is possible in noisy systems if pullup current is not augmented.

  If pin 11 is grounded, then strap pins 12~14 for 'standard' serial communication rates in this table:

Pin 11 Grounded			
Pin 12	**Pin 13**	**Pin 14**	**Baud Rate (bps)**
Lo	Lo	Lo	1200
Lo	Lo	Hi	2400
Hi	Lo	Hi	4800
Hi	Hi	Hi	9600
Lo	Hi	Hi	19,200
Hi	Hi	Lo	38,400
Lo	Hi	Lo	76,800
Hi	Lo	Lo	153,600

NOTE: A coding error in firmware version 4.0.4 caused pins 12~14 to be sampled wrong-endian, making half of the rate entries in the above table incorrect. This problem was fixed in firmware V4.1.1.

Pin 11 must not be grounded if anything other than standard RS-232 style asynchronous serial communication and baud rates from the table above are desired. Pins 12, 13, & 14 assume new roles when pin 11 is not grounded. These pins are tested during BOB-4 initialization and used to select SPI configuration as well as main serial port handshake mode. Pin 12 becomes an SPI clock pin if pin 13 or 14 is grounded, for instance. See table:

Pin 11 Not Grounded			
Pin 12	**Pin 13**	**Pin 14**	**Configuration**
SPI Clock Out	Lo	Lo	SPI Master; main serial port is async, software-selected bit rate.
SPI Clock Out	Lo	Hi	SPI Master; main serial port is in synchronous mode.
SPI Clock In	Hi	Lo	SPI Slave; main serial port is disabled.
Lo	Hi	Hi	SPI disabled; main serial port is async defaulting to 115.2kbps with RTS/CTS flow control.
Hi	Hi	Hi	SPI disabled; main serial port is async defaulting to 115.2kbps with <XON>/<XOFF> flow control.

SPI Details:

If BOB-4 is configured as SPI slave, the host must wait until ACR1 (pin 13) goes low, indicating that BOB-4 has activated SPI slave mode (----------this information must be confirmed----------). Then the host can drive pin 4 as the slave select and pin 12 as the SPI clock. Alternatively, the host must wait two seconds from the release of reset or from power-up before driving any SPI lines.

At this writing, only SPI slave mode is implemented (firmware V4.1.1). BOB-4's SPI receive speed limit is about 120k bytes per second. The SPI clock may go up to 25MHz if sufficient inter-byte delay is added to stay below this 120kB/S net data rate. The SPI subsystem may fail if it's exceeded, requiring a full BOB-4 reset.

SPI is bidirectional: for every byte received, a byte is transmitted. BOB-4 ignores incoming zero (null) bytes, and the host must ignore null bytes emitted by BOB-4. An incoming 0xFF byte causes BOB-4 SPI status to be returned. An SPI status report is two bytes: 0xFF, <status-byte>. SPI status can be used to determine how much space remains in the receiver queue, to allow flow control. SPI status is coded as follows:

Bit 7	zero (currently unused)
Bit 6	0: NTSC mode, 1: PAL mode
Bit 5	0: genlock/overlay video mode, 1: local video mode
Bit 4	0: external video signal not detected, 1: external video detected
Bits 3~0	remaining space in receive buffer (n)

Bits 3~0 indicate space remaining in the receive buffer. 2^n (two to the power of bits 3:0) gives the number of bytes guaranteed to be available. The actual value is usually higher. For instance, if 899 bytes remain, n will be 9, indicating that at least 512 bytes remain. Incoming data is lost if the receive buffer overflows.

BOB-4 can run in SPI modes 0, 1, 2, or 3. SPI mode is configured using pin 6 for CPOL (clock polarity) and pin 16 for CPHA (clock phase). Those pins are read once after reset, just after the ACR pins are read. Both pins have (weak) internal pullups. See table below: (----------Not Tested----------)

Pin 6 (CPOL)	Pin 16 (CPHA)	SPI Mode
Lo	Lo	0
Lo	Hi	1
Hi	Lo	2
Hi	Hi	3

Variable Transparency:

BOB-4 supports manually variable video overlay transparency, but this feature requires a bit of external application circuitry. The circuit below is an example of what's necessary:

(----------Circuit & discussion must be revised!--------):

Use the new video output at J1 instead of BOB-4 pin 28. (----------More discussion----------)

The Video Mix pot must be set for maximum contrast (wiper to pin 3) in local video mode.

Using BOB-4 in a cable TV system:

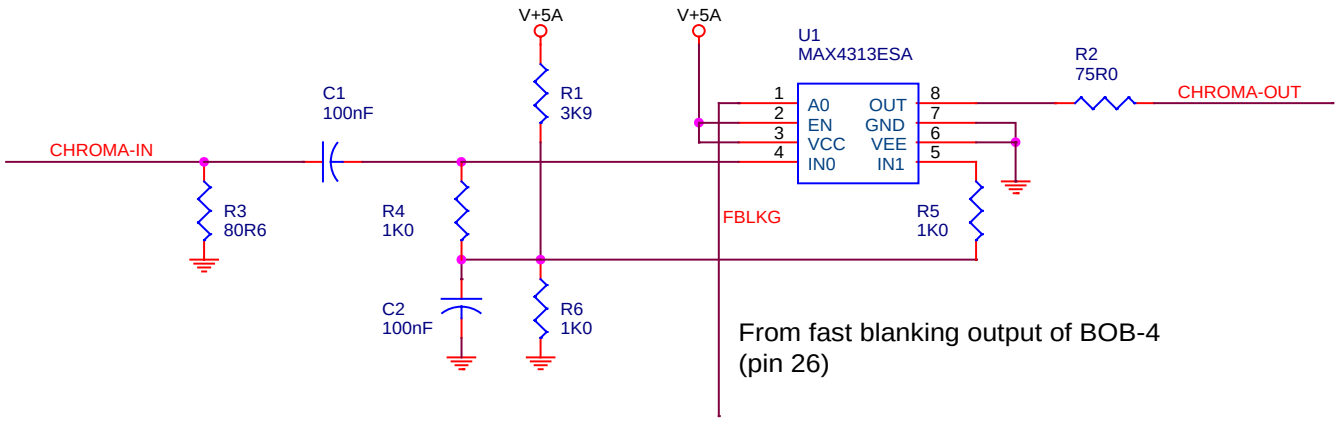
It's not possible to insert a single BOB-4 in a cable TV system and display the same text on all channels at once. There are a number of reasons for this. In a cable system, video signals are modulated onto RF carriers at different frequencies (that's how the TV tuner is able to pick out just one). The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

BOB-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 BOB-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 BOB-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.

Using BOB-4 with Y/C video (S-Video):

Pin 26 can output a fast blanking signal (FBLKG) for use in external video processing circuitry to suppress chroma during character insertion times. See **v** command (n=52) for more details. The luma (Y) channel is processed as usual by BOB-4. Here's a circuit example for the external chroma (C) processing channel:



Low-impedance power supply bypass capacitors must be added. 4.7~22uF tantalum in parallel with 100~220nF ceramic is suggested. Bypasses must connect to MAX4313 power and ground pins with minimum length. Due to analog signal processing delays in the luma channel in BOB-4 (~100nS), FBLKG slightly leads luma output. This chroma channel processor circuit helps equalize the delay differential, but closer scrutiny might be necessary in critical applications.

SIMM socket Information:

Molex part numbers 15-46-3053 (Tin), and 15-46-3043 (Gold) are suitable. Amp 822056-2 (Tin) and 822061-2 (Gold), and Adam Technologies SIMM-130-VTL (Tin) and SIMM-130-VGL (Gold) are equivalent. Amp and Molex have ceased production, but many component distributors cross-reference to other manufacturer's part numbers. Adam Tech distributors are a good source at this writing. Also see Methode's M43 series. Gold contacts are preferred for best reliability.

Application Programming

Communication Basics:

Main serial port communication parameters are: **8N1** (eight data bits, no parity, one stop bit). Bit rates other than 9600bps are configured by grounding pins 11~14 of the BOB-4 module. See pin descriptions. Also see **v** command (n=40) for alternate baud rate selection method. Rate configuration straps are sampled only at system initialization time. Debug port baud rate is fixed at 115.2kbps (8N1).

No RS-232 hardware interface is needed for use with a Parallax[tm] BASIC Stamp[tm] or most industry-standard microcontrollers. Some 5V microcontrollers do not offer TTL-compatible inputs, which can make BOB-4's 3.3V CMOS output level insufficient to guarantee a valid logic high. The cure is usually simple, such as a diode-and-resistor or active logic level shifter. If the BASIC Stamp is programmed for inverted data (relative to RS-232 spec), it may be connected directly to BOB-4. Note that BASIC Stamp SEROUT signal polarity conventions are ***opposite*** to those adopted here. Set BS2 Baudmode to 84 or 85 (for 9600bps). Set BS1 Baudmode to T1200 or T2400. See troubleshooting section for additional comments on baud rate issues.

Your host controller must manage data flow control handshaking correctly if you transmit data to BOB-4 continuously at a high rate. Two methods are provided:

1. Software: By default, BOB-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].

2. Hardware: BOB-4 offers RTS/CTS handshaking. See Pin Description Table for hookup. 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 BOB-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 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 BOB-4 processor's own transmitter does meet this requirement when CTS is asserted.

After a brief start-up delay (<1S), BOB-4 transmits <XON> to inform the host controller that it's on-line (even if RTS/CTS flow control or SPI is selected). (----------The state of RTS prior to complete initialization is unknown at this writing.----------) BOB-4 may send a garbage character or two during initialization.

BOB-4 Native Mode (default):

Much like a conventional serial printer, BOB-4 requires only 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 commands are used to alter the print position, clear the screen, etc.

Carriage Return <CR> (0x0D) and Line Feed <LF> (0x0A, Ctrl-J) codes are handled separately, so both are required to move the cursor to the beginning of the next line. 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' (print position) reaches the end of the screen, the entire screen automatically scrolls upward. Form Feed <FF> (0x0C, Ctrl-L) clears the screen and returns the cursor to the top left position (row 0, column 0).

BOB-4 commands are a subset of the ECMA-048 standard. BOB-4 recognizes most escape sequences handled by the DOS ANSI command set, except those inappropriate for BOB-4. All commands begin with an 'escape sequence' of two special characters: <ESC> (0x1B, Ctrl-[ ), and "[" (0x5B); otherwise known as the **<CSI>** (control sequence introducer). In this document, "<CSI>" is used interchangeably with "<ESC>[". Command identifiers are case-sensitive and postfix (subsequent to arguments). Numeric arguments are transmitted in an intuitive variable-length ASCII decimal format, separated by semicolons. If some arguments are not required, they are simply omitted. Defective commands are generally ignored.

The *String* is an arbitrary collection of up to 1024 bytes. *String* input begins with <STX> (0x02, Ctrl-B) and terminates with <ETX> (0x03, Ctrl-C). *String* input may be aborted by sending <CAN> (0x18, Ctrl-X) anytime after <STX>. The *String* has several applications in BOB-4 application programming, including selection of fonts by name, collecting text for boot scripts and crawl displays, etc, but it is **not** used for normal printing activity.

Command Example 1: An “H” command moving the cursor to row 3 and column 21 is: <ESC>[3;21H

Command Example 2: To store a simple boot script: <STX>Hello, World!<ETX><ESC>[8v<ESC>[1v

BOB-4 allows for negative arguments. Because the minus character ("-") is an "intermediate" character in ECMA-048, BOB-4 uses "<" for a sign character. For instance, <ESC>[<4;30x positions the cursor at pixel position -4, 30.

Fonts are selectable. Each font can include up to 64k glyphs, memory space permitting. The maximum number of displayable characters is determined by the size and type of the currently selected font, as well as the pixel clock rate and any constraints used to prevent display masking by overscanning monitors. Monospaced and proportional fonts are supported.

BOB-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). Characters 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. Characters 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. Non-ASCII characters in the default BOB-3 look-alike fonts may be displayed in BOB-4 mode by transmitting character codes formed according to these rules. [But UTF-8 decoding is not fully implemented in firmware versions up to 4.1.1]

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

50000 (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.

Cursor positioning on a terminal assumes constant character size. BOB-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. Cursor positioning is done in the size units of the current font. If a proportional font is selected, then space character width is used to calculate horizontal cursor position.

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

BOB-4 Commands:

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

H <CSI>*n*;*m*H Moves the cursor to row *n*, column *m*. If *n* is absent, 1 is used. **<CSI>*n*;*m*f** may be used instead.

J <CSI>*n*J 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>*n*K 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>*n*L Insert *n* lines above the current line. If *n* is absent, 1 is used.

Lines below are moved down. Cursor position is unchanged. See comments at M command.

M <CSI>*n*M 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 BOB-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>*n*m Set display attributes. Convenient for use in-line with printable characters.

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 BOB-4, it’s the same as normal blink. Only one blink rate is possible in BOB-4, but it can be changed with the **v** command (n=33).

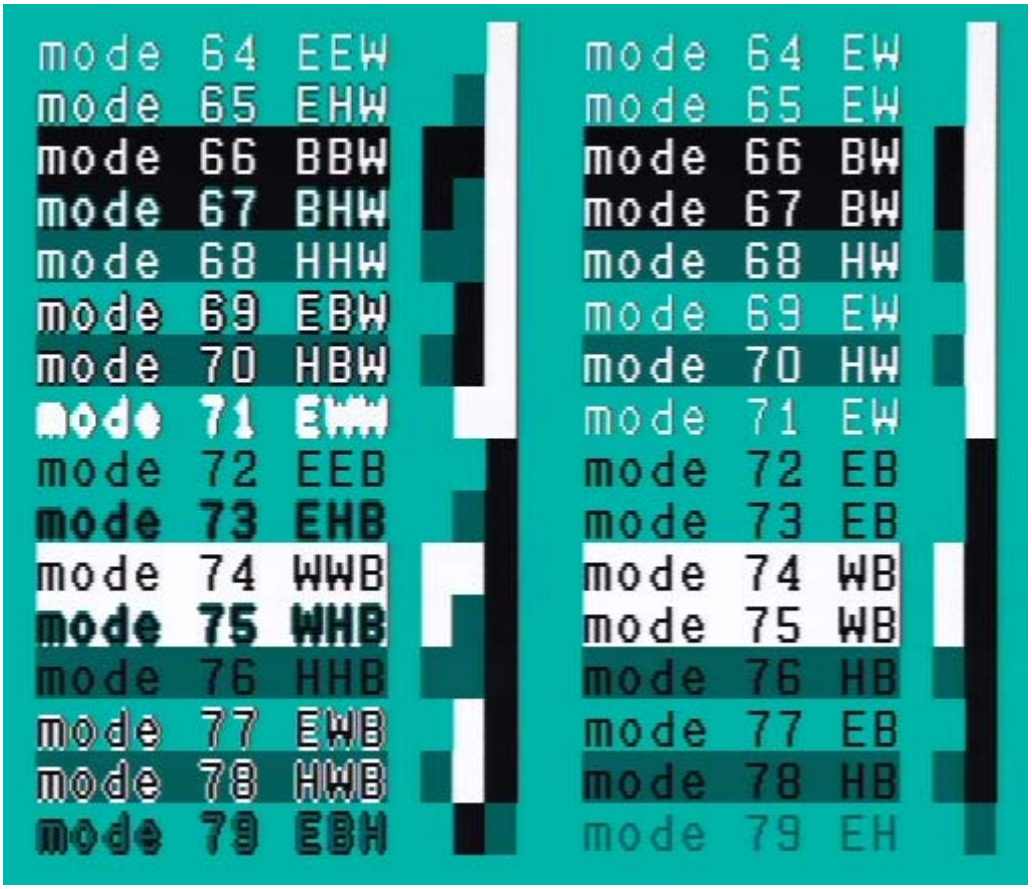
n=7: Start reverse video (same as n=75)
n=10~19: Select font by index (0~9)
n=22: Stop bold/faint
n=25: Stop blink
n=27: Stop reverse

n=64~79: Character rendering

In 2bpp (two bits per pixel) fonts, a pixel can be 00, 01 or 10. Halftone pixels render 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.

q <CSI>*n;m;l;k;*<opt-dot>q Set drawing window (a restricted drawing area of the screen).

The window normally includes the entire frame buffer, but with the **q** command, window size may be reduced. Almost all BOB-4 (and BOB-3) commands continue to operate within this window. Pixels outside the window are unaffected.

n = <top>
m = <bottom>
l = <left>
k = <right>

<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 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.

All commands are modified to work within the window except for pixel cursor addressing, which can draw anywhere, and text crawl, which always spans the whole frame buffer.

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

t <CSI>*n;m;l;k*t Text crawl control.

Crawl text must be loaded in the *String* prior to invoking this command. *String* contents are captured in a crawl buffer and used to repeat the crawl continuously until it’s turned off, at which time buffer memory is returned to the pool. Arguments not given use defaults; e.g. “<CSI>t” starts the crawl if there’s text in the *String*.

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 is -22)

l: Crawl rate in pixels per vertical scan cycle (default is 2)

k: Gap in percent of screen width between crawl cycles (default is 70%)

Crawl uses the current font and render mode. Font and render mode can be changed without affecting an active crawl. Display features outside the crawl line are not altered. You can draw characters elsewhere on the screen while the crawl is running. If you happen to draw in the crawl area, the crawl just overwrites what you did. You can clear the screen, change screen resolution and even change from PAL to NTSC without affecting the crawl.

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

v <CSI>*n*;*m*v 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

8 = Capture boot script from *String*. Boot scripts are limited to 512 characters.

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

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

16 = Video standards compatibility

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

17 = Scanning mode for locally generated video

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

18 = Video input detector action

m=1: Always use locally generated video
m=2: Lock to genlock/overlay – assume that video input is always present
m=3: Switch video sources automatically (default)

20 = HighRate mode. Sets default pixel rate and horizontal size. See n=23.

m=0: No; default pixel rate is 6.25 Mhz. Horizontal size is 272 pixels.
m=1: Yes; default pixel rate is 9.375 Mhz. Horizontal size is 416 pixels.

If HighRate is set, pixel clock rates can go as high as 9.375MHz and horizontal resolutions up to 480 pixels. At 480x240 resolution, the aspect ratio of a pixel is about 1.5:1. The BOB-3 (12x13 default) font has an aspect ratio to match, but the 8x13 font will appear tall and thin. 6.25MHz gives a resolution of 320x240 (NTSC), which yields square pixels. In this case, the default font appears "fat", and the 8x13 font appears normal.

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

m=0: Use entire raster
m=1: Constrain display to safe area (default)

22 = Frame buffer size (reboot required). See n=20.

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

23 = Pixel clock rate. Parameter m=0~8:

m=0: Default (depends on HighRate mode; see n=20)
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 = Horizontal start position (depends on video mode and pixel rate)

Parameter m is offset in pixels from the default value (116 pixels from left raster edge). If m=0, horizontal start position is returned to default value. 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".

25 = Horizontal size (depends on video mode and pixel rate)

Parameter m is display width in pixels (default = 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 = Vertical start position (depends on video mode)

Parameter m is offset in pixels from default value. If m=0, vertical start position is returned to default value of 39 pixels from top of raster. 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".

Example: Default vertical start position is 39, so the command “<CSI>26;10v” sets first line reported in running parameters to 49 and vertical position in configuration report to 10.

27 = Vertical size (depends on video mode)

Parameter m is display height in pixels (default = 207). 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 = Overlay enable

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

32 = Blink enable (global)

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

BOB-4 does not implement on/off blinking, but rather white pixels are alternately rendered black. Actual appearance depends on the character-rendering mode (see "<CSI>nm" for n of 64 to 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 BOB-4 mode the global blink enable command is "<CSI>32;1v". The equivalent command in BOB-3 compatibility mode is "{G1". Globally disabling blink mode clears the screen and restores normal font rendering. This is done with BOB-4 command "<CSI>32;0v" or BOB-3 command "{G0". Once blinking is globally enabled,

individual character blinking is switched on with "<CSI>5m" (BOB-3 "{GE"), and switched off with "<CSI>25m" (BOB-3 "{GD"). Printable characters transmitted subsequent to switch commands obey the current setting.

Blink enable, blink period, and blink duty cycle settings are included in the configuration report ("<CSI>5}").

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

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

40 = Communication bit rate and sync/async mode control. Default is 9600bps asynchronous.

For asynchronous serial (RS-232 style) communication, m can be any value from 46 to one million. High rates are more 'granular'. To compute the rate, 23,961,600 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 configuration is saved.

Synchronous mode is selected if m=0. (----------not tested----------) Main port synchronous mode uses UART hardware, not SPI hardware. The clock input is on pin 16. Data remains on pins 8 and 9.

41 = Communication flow control. Reset is required.

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

48 = L0 output select* (This is a 'special' 5V open-collector-with-pullup output.)

m=0: Low (default)
m=1: High
m=2: Composite sync
m=3: Vertical sync

49 = L1 output select*

m=0: Low (default)
m=1: High
m=2: Horizontal sync
m=3: Vertical sync

50 = L2 output select*

m=0: Low (default)
m=1: High
m=2: Horizontal sync
m=3: Composite blanking

51 = L3 output select*

m=0: Low (default)
m=1: High
m=2: Composite sync
m=3: Vertical blanking (Can help achieve flicker-free animation)

52 = FBLKG output select*

m=0: Low (default)
m=1: High
m=2: Fast blanking
m=3: Fast blanking inverted

*** Video timing output details:**

The composite sync output follows video input sync with a small (1~2uS) analog signal processing delay. Horizontal sync and blanking outputs, including fast blanking, include this processing delay plus tiny logic delays of less than 10nS. The vertical sync output is further delayed by about 30uS due to the method used for Vsync detection in logic. This delay applies to both leading and trailing edges of Vsync.

The horizontal sync output has a few quirks around vertical sync time. It's correct during active video scanning, but when equalizing pulses before and after Vsync occur, Hsync goes low for only the period of each equalizing pulse. Within vertical sync, Hsync goes low at the right time and stays low for about 30us (logic takes this long to detect Vsync). Another way to look at it: Hsync always goes low at the right time, but around Vsync it may not go high when you expect.

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

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

If you imagine a canvas with x and y range -32767 to 32767, the screen is a window on the canvas starting at 0,0, and extending to the screen size, or if the window is set (**q** command), the window on the canvas is the 'q' window. This means you can do something like position the cursor left of the screen and start drawing text. The text will cross the screen being clipped by the screen, or window, and continue off to right infinity (actually 32767).

z <CSI>*n*z Select font; *n* =0~7; if *n* is absent, font is selected by name from *String* contents.

Example: "<STX>20x40<ETX><CSI>z" selects the 20x40 font by name. BOB-4 firmware includes eight default fonts:

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

Available font numbers and names are listed at the end of the "<CSI>5}" configuration report or "config" debug command (report is the same for either command).

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

{ <CSI>{ Escape to BOB-3 compatibility mode

Any native BOB-4 command recalls BOB-4 mode and is executed. For instance, the BOB-4 command sequence “<ESC>[2J” selects BOB-4 mode and immediately clears the screen. This command is ignored if BOB-3 mode is locked out (see **v** command with n=9).

| <CSI>*n*;*m*| Miscellaneous actions

n=1: Clear screen after m seconds without incoming data. Screen will be cleared upon receipt of the next character after timeout. Set m=0 to cancel.

n=2: Clear screen after m seconds without incoming data. Does not wait for a character after timeout. Set m=0 to cancel.

n=9: Transmit video timing signal. BOB-4 emits an ASCII 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: On

n=1234: Toot our horn

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

n=3210: Reboot. If m=1, start bootloader for firmware download.

} <CSI>*n*} Transmit system information

n=1: BOB-4 transmits a BOB-3 style status string: “VT M01N DE v4.0.4 NTSC”
n=5: BOB-4 transmits a complete configuration report
n=9: BOB-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 BOB-4 created an expected pattern of pixels in display RAM.

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

Vector Graphics Commands

Graphics primitives in BOB-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 -32767 to 32767. 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.

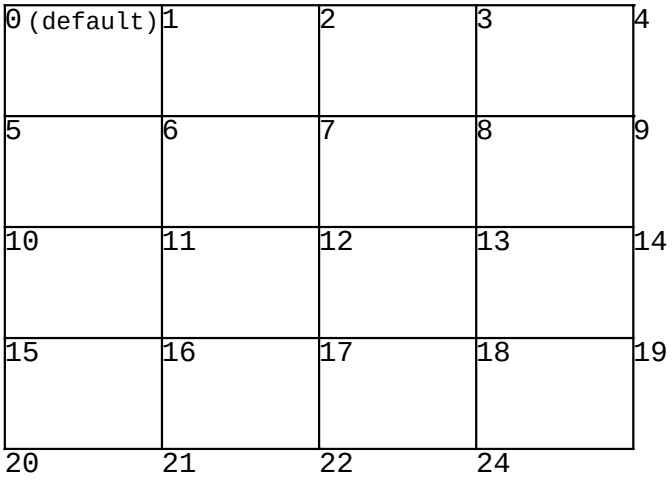
At higher pixel rates, vertical lines will be thinner than horizontal lines, and vertical lines of single-pixel width may display poorly (especially on color monitors) due to TV bandwidth limitations.

[Version 4.1.1 firmware draws to the frame buffer rather than the window, so the **q** command has no effect on graphics. This will be fixed in the next release.]

MoveTo <CSI> x ; y .r

Move the graphics cursor to the given XY coordinate.

Note: Both MoveTo and LineTo accept a third parameter, which makes X and Y relative to the current window. The Y portion is given as 'rel' / 5, and the X portion by 'rel' mod 5. Examples:



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

Add a line segment to the path from the current graphics cursor to the given XY coordinate.

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

Add an arc segment to the path from the current graphics cursor, sweeping from 'a1' to 'a2' degrees of arc with radius 'r'. Angles are measured in degrees, with 0 being in the positive X direction (to the right), and sweeping counterclockwise. Arcs are approximated by drawing line segments.

For example, "<CSI> 10 ; 0 ; 90 .(" 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.

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.

'c' specifies color, as in StrokePath.

More 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> 100 ; 10 +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)
```

Debug Port Protocol

The debug port can help debug new BOB-4 applications and download new firmware. It's always active. With the debug port interfaced to a PC, users don't need to switch over the main port from an embedded controller or build a separate programming fixture for firmware upgrades. Communication settings are as follows:

Bit rate: 115,200bps
Data bits: 8
Parity: None
Stop bits: 1
Handshake: None

The debug port can be interfaced to a PC with the same circuit given for the main port, except that the RTS and CTS lines are not implemented. Loop these signals back as suggested, and observe the caution regarding an open CMOS input on the interface chip.

Debug Port Commands:

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

help Display available debug port commands

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

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

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 main serial port.

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

config Show configuration

config save Save current configuration

serial Show serial port settings

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

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

[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

BOB-3 Compatibility Mode

The factory defaults for BOB-4 lock it in BOB-4 native mode. To use BOB-3 compatibility mode, first unlock it by sending "<ESC>[9;0v", then reconfigure the screen dimensions to allow 17 lines of 40 columns. This involves turning "underscan" off, "highrate" on, and restricting vertical size to yield 17 lines in PAL mode. Additional details will be added to this section.

If BOB-3 configuration is going to be saved, then the default boot script must be changed. Using BOB-4 commands, the configuration procedure should be:

<CSI>9;0v	(unlock BOB-3 mode; BOB-4 will reboot in BOB-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)
<STX><CSI>{<ETX>	(string to invoke BOB-3 mode; may be omitted because reboot does the job)
<CSI>8v	(capture string in boot script; also may be omitted…)
<CSI>1v	(save configuration)

Serial communication parameters are **8N1** (8 data bits, no parity, 1 stop bit). Bit rates other than 9600bps are configured by grounding pins 12~14 of the BOB-4 socket at installation time. Pin 11 must be grounded for this to work correctly. See pin descriptions, but note that BOB-3 compatibility mode does not affect the electrical interface. For instance, if pin 11 is not low at boot, then BOB-4 bit rates may be selected, including those set by the <CSI>40v command.

No RS-232 hardware interface is needed for use with a Parallax™ BASIC Stamp™ or most industry-standard microcontrollers, but some 5V micros may not be directly compatible with 3.3V CMOS logic input. If the BASIC Stamp™ is programmed for inverted data, it may be connected directly. BASIC Stamp SEROUT signal polarity conventions are opposite to that adopted here. Set BS2 Baudmode to 84 or 85 (for 9600bps). Set BS1 Baudmode to T1200 or T2400.

Your application program must manage the software handshake correctly if you transmit data continuously at a high rate. BOB-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™. If the hardware handshake is previously configured in BOB-4 native mode, then it may also be used in BOB-3 mode.

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

Characters without the BOB-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 BOB-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 BOB-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.

BOB-3 Commands	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 BOB-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. 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.

Notes:

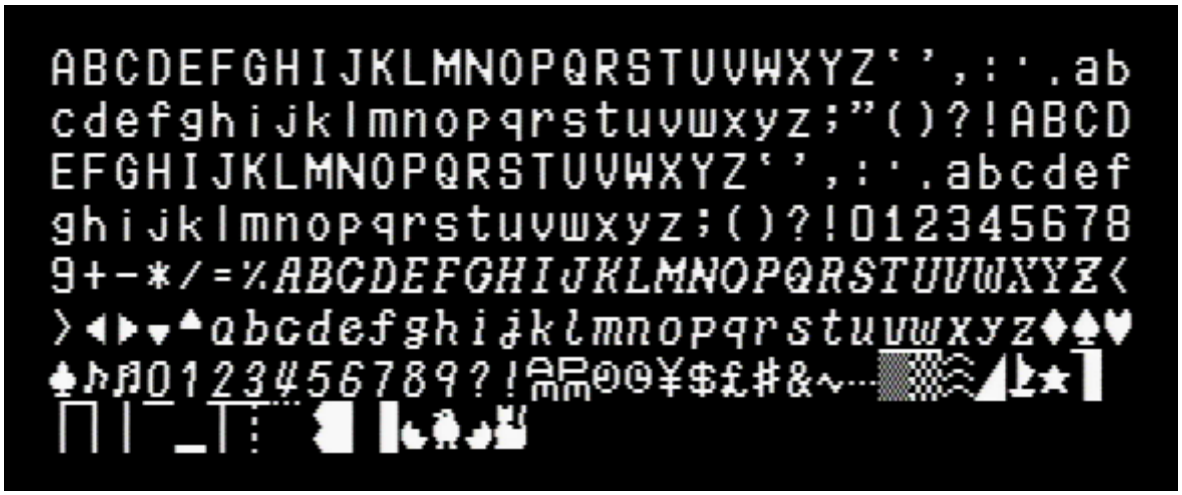
There are new blinking commands in BOB-3 mode: **{G1** is the global blink enable. **{G0** turns it off. Both commands clear the screen and change the character drawing mode as needed. Note that the whole screen is cleared regardless of window setup (in BOB-4 command set).

Configuration for PAL or NTSC video is done with the new **{MP** and **{MT** commands. You don't have to order a different version of the board anymore.

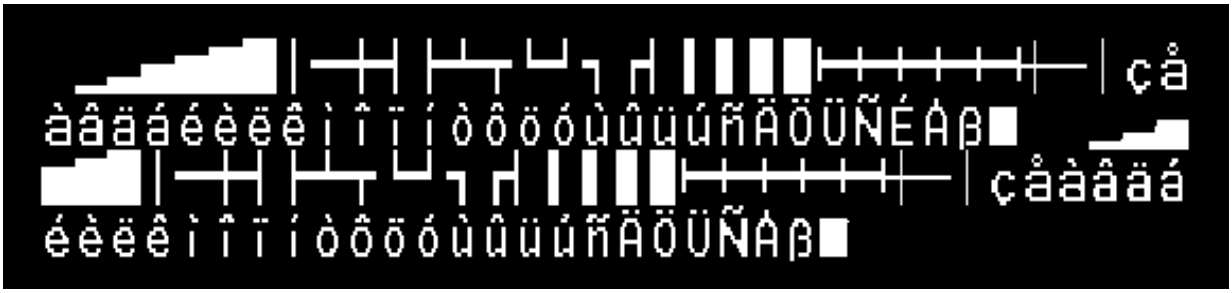
Many of the original BOB-3 commands are not implemented. For instance, you cannot download a font in BOB-3 mode or set the boot script or start a crawl. There is enough functionality to support many but not all applications previously using the BOB-3 module. If you need extended functionality, switch to BOB-4 native mode.

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

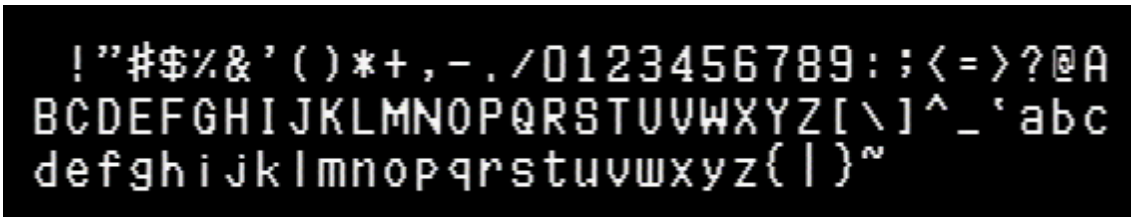
Default Character Sets



BOB-3 12x13 Look-Alike Font (was ROM Font)



BOB-3 12x13 Supplementary Characters (was Default RAM Font)

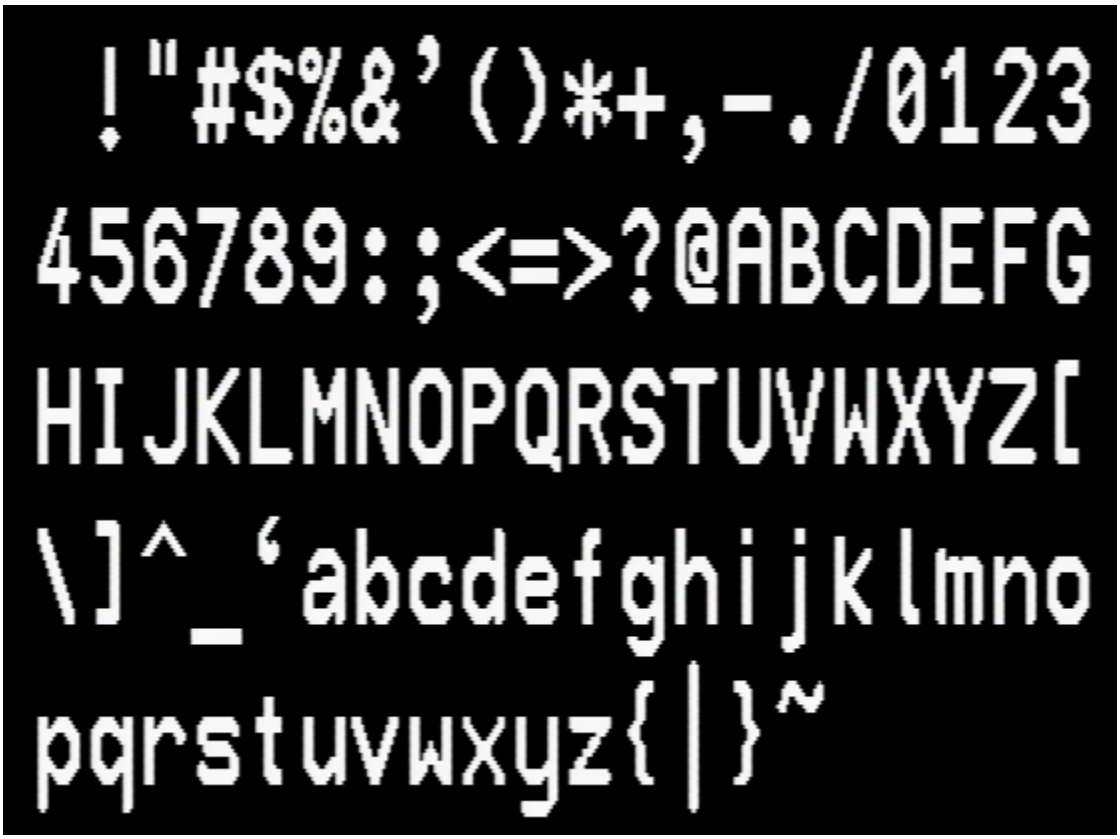


BOB-3 12x13 Font in BOB-4 Mode

6x10 Font

8x13 Font

13x34 Font



20x40 Font

Troubleshooting

Cannot perform firmware upgrade:

Early production BOB-4 modules may not allow firmware upgrades to proceed normally. These boards can be identified by the lack of a green paint dot near the "Rev2" marking on the topside of the board. (Boards with the green dot are lead-free.) If you have tried using the BOB-4 Conscriptor to perform a firmware upgrade on one of these modules and failed, even though good communication with the PC has been confirmed, then we recommend that you desolder and remove C40 from the board. This component is located on the bottom side (without ICs) near top edge center. Please observe standard ESD safety precautions. The BOB-4 Conscriptor should now complete firmware upgrades as intended. Return the module to Decade Engineering for this operation if you have any doubts about your technical skills.

Note that C40 provided insurance against noise pickup on the reset line in electrically hostile environments. If you have connected pin 10 to anything in your system, then a 10K pullup resistor (to +3.3V) should be added on this line.

BOB-4 is confused or unresponsive:

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 <STX><ETX> (empty string) <CSI>8v (capture boot script) <CSI>1v (save new configuration), and see next paragraph.

For bizarre trouble symptoms, suspect boot script corruption. This is especially likely if experimentation preceded (or might have prevented) operational status. Rogue boot scripts may be completely erased by holding pin 15 low as power is applied, even if BOB-4 is unable to communicate with the host computer. Pin 15 normally drives a video input status indicator. During initialization (and re-initialization), it is an input with internal pullup. If pin 15 is pulled under the logic low input threshold during those times, the boot script erase routine is invoked! This happens very quickly (in milliseconds), and no confirmation message is emitted.

If printable characters are corrupted or fail to display, 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 1%. Note that commands are probably being clobbered as well, if displayed characters are missing or corrupt.

Make sure your power supply is healthy. Substitute a known-good power supply unit, or check the DC output line of your power supply circuit with a scope. Ripple suppression capacitors can fail (especially 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/or capacitor equivalent series resistance (ESR).

Try another video monitor. In rare cases, monitors and other downstream devices can react badly to DC bias in the video output from BOB-4. If necessary, add a DC blocking capacitor of 470uF or 1000uF, rated at 6V or greater, in series with the video output line (pin 28). The "+" side of the capacitor should connect to BOB-4 pin 28.

The overlay doesn't fit your monitor screen:

See **v** commands (n=24~27) to adjust horizontal and vertical start position and size. Some LCD monitors reveal more of the raster than CRT monitors. Low-cost CRTs usually mask large portions of the image perimeter due to overscan. The amount of overscan often varies with image brightness and power supply voltage. Display centering on monitors can also be imprecise. It's best to adjust display centering with reference to a test pattern from a video signal generator.

Thin vertical lines are not as bright as horizontal lines:

BOB-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. Contact Decade Engineering for technical assistance if you're unable to resolve this issue.

The text overlay is unstable:

Overlay jitters can be caused by weak and/or noisy video applied to pin 30. 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.

BOB-4 was not designed to work with tape playback signals from VCRs. It may perform as desired, but overlay stability can be unacceptable with some VCRs and some (usually worn) cassettes. Performance is generally worse in VCR special effects modes such as freeze-frame.

BOB-4 is dead or operates intermittently:

Tin-plated contacts can become unreliable if an electrical contact treatment has not been applied. We suggest DeoxIT Power Booster™ and ProGold™ (or PreservIT™), from Caig Laboratories. Apply the contact treatment to a cotton swab and polish all of the contact pads on your BOB-4 module, then reinstall it in the SIMM socket. This procedure should result in a permanent cure for intermittent operation due to poor contact, if tin or tin/lead plating is present on the BOB-4 module or SIMM socket contacts.

Firmware revision history:

[V4.1.1] [24 August 2006] Added vector graphics command set and SPI slave mode. Fixed wrong-endian sampling of baud rate select pins.

[V4.0.4] [06 June 2006] First release, without significant planned features: [1] Cannot download custom fonts or bitmap images. [2] No vector graphics support. [3] SPI port and supplementary font/bitmap memory is not operational.

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	See website for current email contact information
Web	www.decadenet.com

Obligatory Boilerplate

Software Warranty Statement: