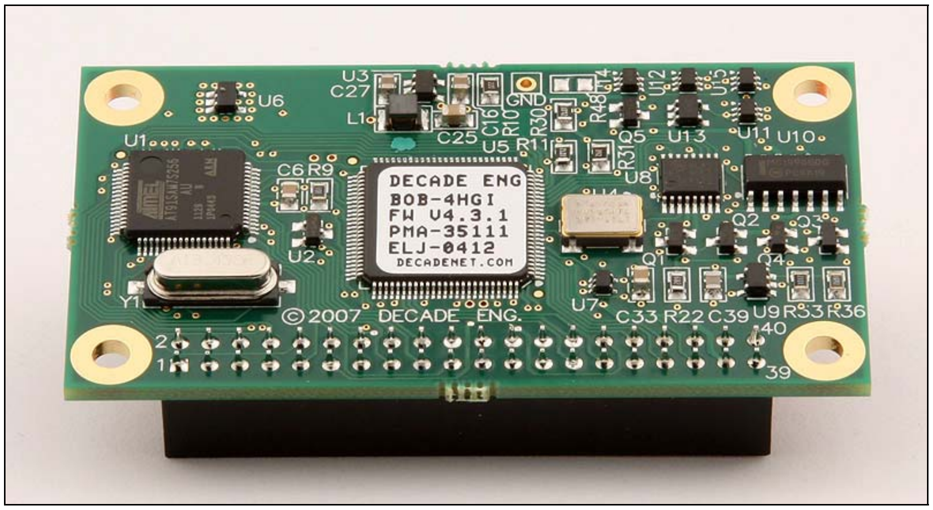
DECADE ENGINEERING

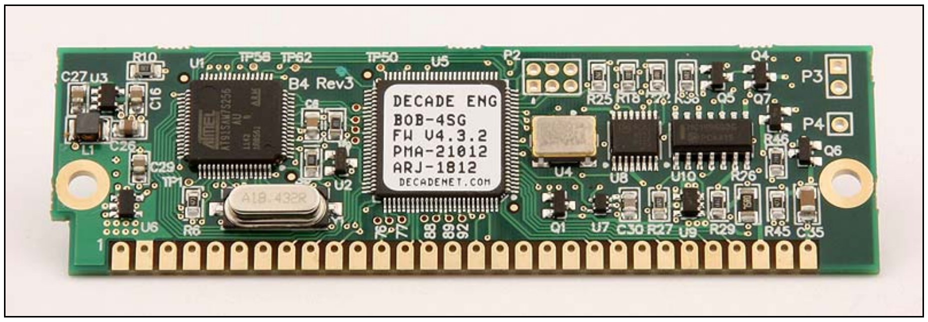
BOB-4H & BOB-4S Application Guide

Firmware V4.3.3 ~ 10 July 2012

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



BOB-4H (2x20 header format)



BOB-4S (30-pin SIMM 'legacy' format)

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 large 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 drive BOB-4 through SPI or RS-232 style data links, much like a serial terminal or printer. BOB-4 links directly to most microcontroller chips and modules, including the BASIC Stamp^{tm}. NTSC and PAL video standards are customer-configurable. Field firmware upgrades are supported via PC connection.

BOB-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, BOB-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.

Cautions!

BOB-4 is not designed to tolerate power supply shortcomings. Be certain that you have a stable 5VDC power supply, regulated to ±5% or better, BEFORE proceeding with BOB-4 application development. Applied voltage must never exceed +5.500V under any circumstances, even for a microsecond. Higher voltages or poor regulation can cause permanent damage to the BOB-4 module, and such damage is not covered by the warranty. Power supply voltage overshoots must not occur at turn-on or turn-off time, or when power supply load conditions change. Application circuits should include an electrolytic bypass capacitor of 100~470uF near the +5V power supply input pin to BOB-4. Switch-mode power supplies should include at least one extra L/C filter stage beyond the basic L/C components required for operation. Residual switching spikes should be suppressed to no more than a few millivolts at BOB-4's power supply input pin. A BOB-4 Power Supply Application Note is published at www.decadenet.com.

Decade Engineering has confirmed that Elenco model XP-660 and XP-760 benchtop power supplies can generate severe output voltage transients at turn-off time, apparently due to arcing at power switch contacts and lack of inter-winding shields in the transformer. It's almost certain that other low-cost power supply units exhibit similar defects. These transients easily blast through downstream voltage regulator chips and destroy modern high-density ICs such as those in BOB-4. IC damage may be slight at first, but it's cumulative, and total failure soon develops. An arc suppression network added across PSU switch contacts might solve the problem, but full confirmation is required. Don't attempt modifications of this nature without a complete understanding of the safety issues.

Total series resistance external to the RXD input pin must stay below 200 ohms. This includes any series resistance that may be present in the TXD output line of your controller, logic level shifter, filter or protection networks, etc. BOB-4 can fail to receive data if this caution is not observed.

Total series resistance external to any logic input pin *other than RXD* must not exceed 500 ohms. Logic level translator chips can include as much as 6K ohms in series with outputs (e.g. MAX3001) so read the data sheet carefully!

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.

External circuitry connected to BOB-4 must prevent ESD strikes from reaching BOB-4 on any pin. Treat all pins of the BOB-4 module with the same ESD preventive measures as typical CMOS logic chips.

Use the correct mating connector! Soldering directly to BOB-4 (except at J1 in the SIMM version) voids the warranty.

To increase awareness of problems that can occur in BOB-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 BOB-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**.

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.

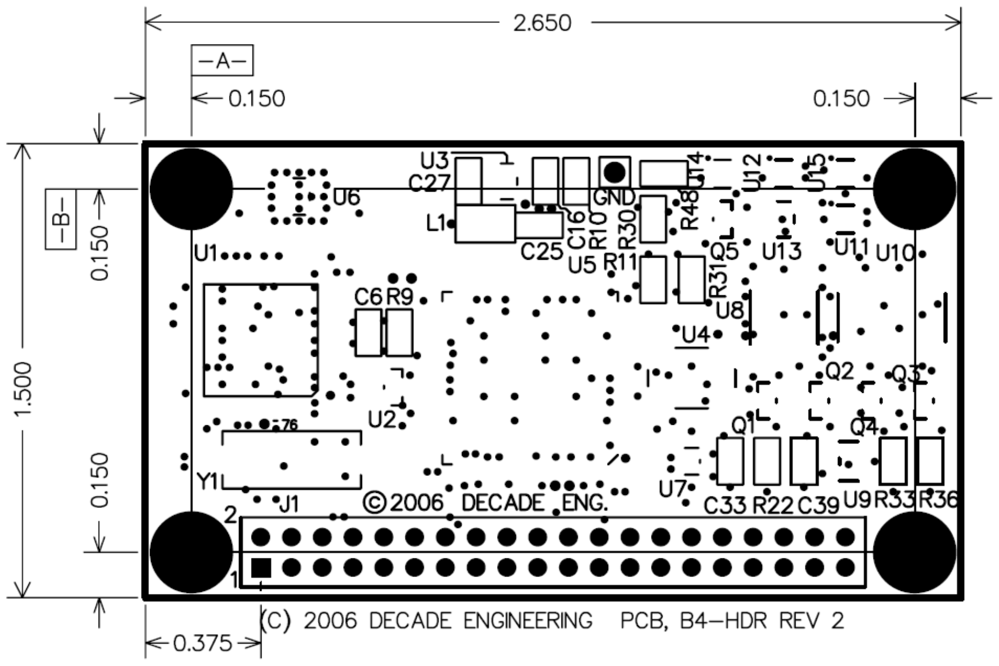
Specifications

Physical	BOB-4 is available in two hardware form factors. Both are nominally RoHS compliant (lead-free). **[1]** BOB-4HGI: Header connector format, 2.65 x 1.50 inches. Modules are provided with a bottom-side .100” pitch 2x20 pin header receptacle. Weight is about 0.55oz/15.6g. Industrial operating temperature range: –40 to +85°C. **[2]** BOB-4SG: 30-pin SIMM format, 3.50 x 1.05 x 0.35 inches. Weight is about 0.35oz/9.8g. Commercial operating temperature range: zero to 50°C.
Power Supply	BOB-4 requires +5VDC regulated within ±5% at 100~105mA typical. **Absolute maximum input voltage is +5.500V regardless of time scale**. External bulk capacitance of 100~470uF or more must be provided near the power supply input pin. The power supply should be sized to accommodate 150mA or more, and protected against fault currents above 250~300mA. Start-up time must be less than 25mS. A low-current (10mA) auxiliary +3.3VDC regulated output is available. Auxiliary power supply output current adds to main supply current drain. Please see the BOB-4 Power Supply Application Note for important additional discussion.

Data I/O	The primary data path is 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. 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 provided. This port is hard-coded to run at 115.2kbps, 8N1, without handshake. An SPI communication interface is provided. The SPI port may be used for BOB-4 control (slave mode) or font memory expansion (master mode), but it cannot simultaneously serve both purposes. Logic output specs conform to 3.3V CMOS standards, except pin 20 in the SIMM version. Logic output current is 8mA maximum except VMIS\ (16mA). Logic inputs are TTL-compatible (max logic low: 0.8V, min logic high: 2.0V) and 5V-tolerant (except ACR0~2), but hard 5V drive is not allowed during reset. The main RXD pin (and TXD) offers full 5V input tolerance, even during reset. Two analog (ADC) inputs with zero to +3.3VDC maximum range are provided on BOB-4H only.
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 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 (logic output available) and display position should be near screen bottom. To assist in achieving smooth animation, BOB-4 can also 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 (±10%) 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 255x255, limited only by font storage space. 62kB of on-board flash memory is available for custom fonts (shared with bitmap graphics). 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. English fonts of assorted size and appearance reside in permanent 'system' memory. Only the BOB-3 look-alike font, which is the default 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. In BOB-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. 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. Custom font installation is not supported in BOB-3 compatibility mode.

Graphics	A set of vector graphics drawing commands is provided, including a 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 BOB-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 BOB-4 user font memory space (flash). Limitations apply; see BOB-4 Conscriptor Help system for additional information. These objects are treated internally as a font with one character.
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. 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. Manual adjustment of overlay transparency (contrast) 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) message line can display up to 4096 characters sequentially without disturbing other display elements. The entire information display may be toggled on/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 BOB-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.



BOB-4H Mounting Dimensions (inches); top view; not to scale

The female header connector is located on opposite (bottom) side of BOB-4H. Mating pin headers should have .025” square pins, ‘short’ (.230”) mating pin length, and .100” pin spacing (pitch). Standoff height should be 7/16” (.4375”). Mounting hole ID is .125” (clears #4 machine screw). Mounting hardware OD must not exceed .270" at PCB surface. Allow .120” clearance for components on the bottom side. Avoid placing hostile EMI sources on host board directly below the right-hand 30% portion of BOB-4H, under the analog video processing circuits.

BOB-4H and BOB-4S Pin Description Table

Hdr Pin	SIMM Pin	Pin Name	Dir	Description
1		Reserved		Should be grounded
2		Reserved		Should connect to +3.3V
3		Reserved		Do not connect
4		Reserved		Do not connect
5	1	+3.3V	Out*	Auxiliary low-current +3.3V output (10mA max until further notice). *In the SIMM version, this pin is open-circuit as shipped. Add a solder blob at J1 (bottom side) in BOB-4S to bring out +3.3V here.
6	3	+5V	In	+5V ±5% 100mA (typical) regulated power supply input. **+5.500V absolute maximum regardless of time**. Bypass with 100~470uF electrolytic cap nearby.
7	2	GND	X	Ground; must connect to host PCB ground plane
8		Reserved		Do not connect
9	19	TEST	In	Factory test; should be grounded in application circuit
10		Reserved		Do not connect
11	10	RST\	I/O	System reset input (low true). No connection is required, but pullup is suggested if RST\ is deployed in noisy systems. See notes.
12	16	SCK/CPHA	In	Main serial port clock input (synchronous mode only) or SPI mode control. See *SPI Slave Mode* for SPI usage.
13	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 Slave Mode* for SPI usage.
14	5	RTSO	Out	Main serial port RTS output; hardware handshake signal
15	8	TXD	Out	Main serial port data output; protected by TVS diode and series resistor
16	9	RXD	In	Main serial port data input; protected by TVS diode and series resistor. **Do not add more than 200 ohms of series resistance external to this pin**.
17	17	DTX	Out	Debug serial port data output; 115.2kbps only, no added ESD protection
18	18	DRX	In	Debug serial port data input; 115.2kbps only, no added ESD protection
19		AREF	In	ADC reference input. 250uA maximum average current. **Only present in BOB-4H, this pin must connect to pin 5 or a precision reference of +2.6~3.3V**. The +3.3V source at BOB-4H pin 5 is an NCP551 regulator IC with 2% basic accuracy.
20		ADC6	In	Analog input; Vmax = AREF; signal source impedance <21k ohms; see notes
21		GND	X	Ground; must connect to host PCB ground plane
22		ADC7	In	Analog input; Vmax = AREF; signal source impedance <21k ohms; see notes
23	7	SS2\	Out	SPI master select 2 output (low true)
24	4	SS0\	I/O	SPI slave select input, SPI master select 0 output (low true)
25	21	MISO	I/O	SPI data I/O (Master Input – Slave Output)
26	22	MOSI	I/O	SPI data I/O (Master Output – Slave Input)
27	12	ACR0/SPCK	I/O	Baud rate select input, SPI clock I/O; not 5V-tolerant
28	13	ACR1/SPM0	In	Baud rate select, SPI mode select 0; not 5V-tolerant
29	14	ACR2/SPM1	In	Baud rate select, SPI mode select 1; not 5V-tolerant
30	11	ACR3/SS1\	I/O	SPI master select 1 output (low true), SPI mode select primary input (strap low for BOB-3 compatibility)
31	20	L0	Out*	Auxiliary logic output, controlled by application. *In the SIMM version, this pin is an open-collector output with built-in 2.1k pullup to +5V.
32	24	L1	Out	Auxiliary logic output, controlled by application
33	25	L2	Out	Auxiliary logic output, controlled by application
34	26	L4/FBLKG	Out	Auxiliary logic output, controlled by application (can be fast blanking output, for external S-Video chroma channel processor)
35	27	L3	Out	Auxiliary logic output, controlled by application
36	15	VMIS\	Out	Logic output; low if video not supplied to CVIN; 16mA max. May be unstable during initialization. Add series diode(s) with LED if returning to +5V. See notes below for additional pin functionality.
37	29	GND	X	Ground; must connect to host PCB ground plane
38	23	CLVID	Out	1Vpp video output; unbuffered, for external transparency control. Sync tips are clamped to zero volts.
39	30	CVIN	In	Video input; internal 75 ohm termination (1/4W max); 1Vpp required
40	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.

Hookup Notes:

All ground pins should go directly to a low-impedance ground system in the host PCB. The +5V power supply input pin must be decoupled with a 100~470uF or greater electrolytic capacitor close by.

Logic outputs are standard 3.3V CMOS type except L0 in BOB-4S (SIMM version). Logic output current is 8mA maximum except VMIS\ (16mA). Logic inputs are TTL-compatible (max logic low: 0.8V, min logic high: 2.0V) and 5V-tolerant (except ACR0~2), but **hard 5V drive is not allowed during reset.** RXD is the only input with full 5V tolerance. Series resistance to RXD must be less than 200 ohms. 500 ohms is the upper limit for all other inputs.

L0~L4 are programmable to output various video timing signals as well as fixed logic states. They all default to logic low. In the SIMM version only (BOB-4S), 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. L4/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.

ADC pins may be reassigned in future products with BOB-4H form factor. See **}** command for ADC usage.

RST\ must go low for at least 80uS if it's placed in service. 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 after an external reset to extend reset time. Add a 10K pullup to +3.3V if RST\ is used in the application circuit, to minimize EMI sensitivity.

VMIS\ may be held low during initialization to clear a rogue boot script and restore default configuration — useful when main port communication is dysfunctional. The debug port offers an alternate solution if BOB-4 gets hopelessly confused because it's hard-coded to run at 115.2kbps (8N1).

Pin 6 was used for RX Enable in BOB-3, 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 SS0\ serving as the SPI slave enable input. An external AND function in the RXD line could emulate BOB-3's RX Enable feature if necessary.

Decade Engineering recommends 10K pullup (to +3.3V) resistors at ACR0~ACR2 to reduce EMI sensitivity in noisy systems, unless they are tied permanently high or low for baud rate selection. If ACR3 is not grounded, a 10K pullup should be added there as well. Weak internal pullups are provided, but wrong configuration is possible in electrically noisy systems if pullup current is not augmented.

Header Connectors:

Here are a few example part numbers for 'BergStik' style male pin headers to mate with BOB-4H. These are all conventional THT (thru-hole technology) 40x2 pin headers that may be divided into a pair of 20x2 headers:

NSH-80DB-S1-TG (Robinson-Nugent)
2380-6121TG (3M)
517-6121TG (Mouser)
PH2-80-SGA (Adam Technologies)

SIMM Sockets:

Molex part numbers 15-46-3053 (Tin), and 15-46-3043 (Gold) are suitable for use with BOB-4S. Amp 822056-2 (Tin) and 822061-2 (Gold), and Adam Technologies SIMM-130-VTL (Tin) and SIMM-130-VGL (Gold) are equivalent. Although Amp and Molex have ceased production, their part numbers may still be useful for cross-reference purposes. Gold contacts are preferred for best reliability, but have apparently been phased out by the remaining manufacturers. Adam Tech distributors are a good source at this writing. Jameco catalog number 100302 has been tested with good results. Also see Methode's M43 series (e.g. M43E-20-130-02).

Basic System 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. BOB-4's primary ground pins 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.

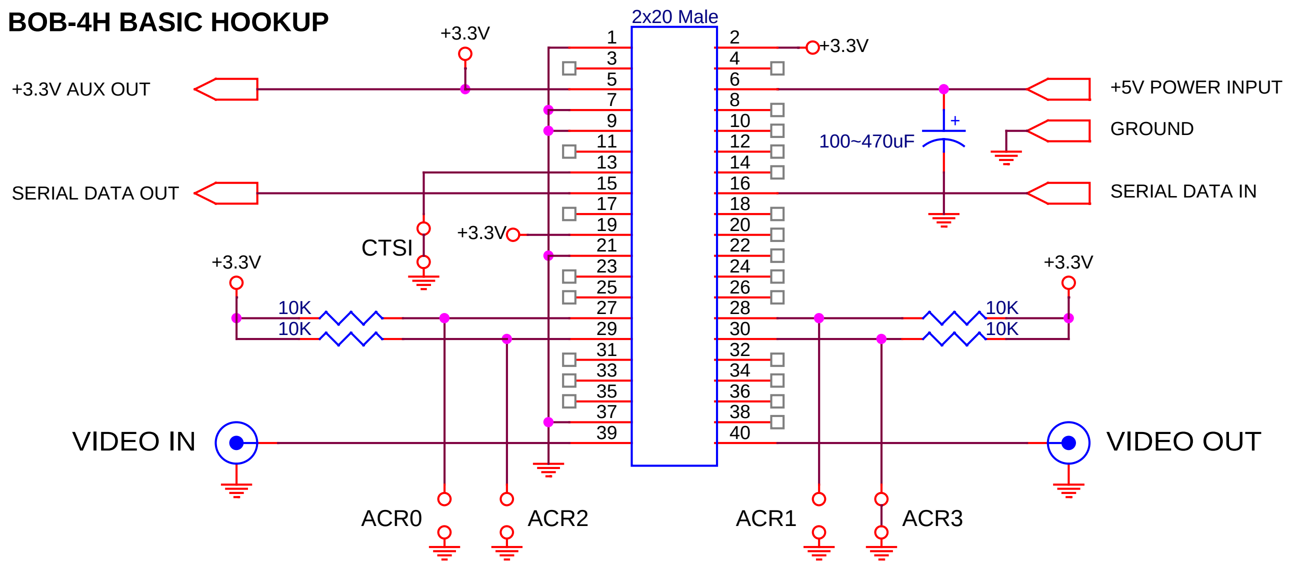
4. With the exception of RXD (and TXD), BOB-4's logic inputs can be permanently damaged by direct application of 5V from a low-impedance source. Please observe all notes on this topic carefully if BOB-4 is being integrated into a 5V system. Don't add more than 500 ohms in series to any BOB-4 I/O pin, and stay below 200 ohms in series to the RXD pin.

The drawings below show typical BOB-4 installations using the asynchronous serial 'TTL-232' main port. In these hookup schemes, the data rate defaults to 9600bps with XON/XOFF software flow control. 10K pullup resistors (to +3.3V) are recommended at ACR0~ACR3 unless no conductors are attached to these pins. Internal pullups are provided, but they're not strong enough to assure reliability in noisy systems.

ACR3 must be lifted to enable software bit rate selection or the SPI port. See the table under **Hardware Configuration**. Add a 10K pullup on ACR3 if it's not grounded. CTSI must not be grounded if RTS/CTS flow control (hardware handshake) will be implemented.

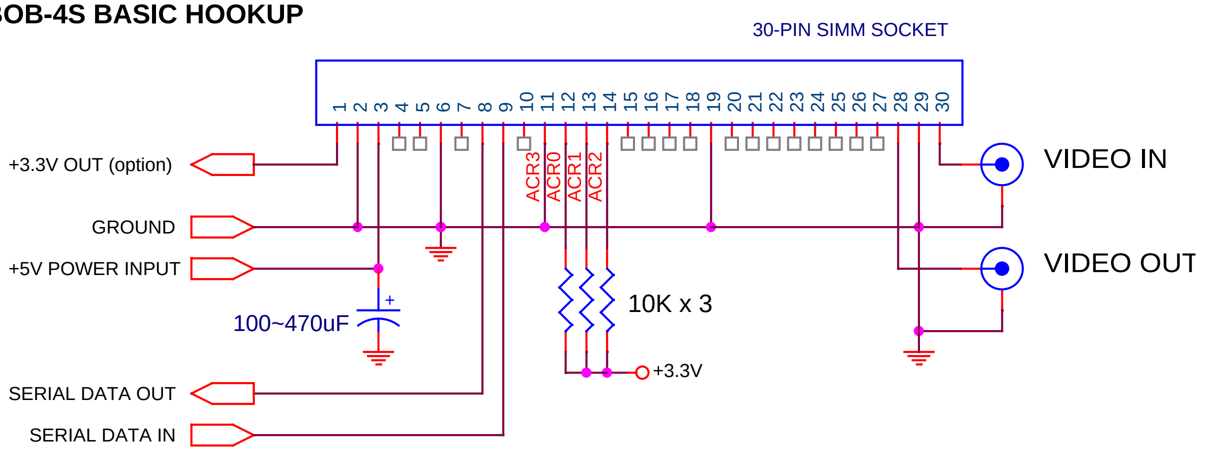
An alternate and recommended hookup: To boot at 115.2kbps with RTS/CTS hardware flow control, ground ACR0 and pull up ACR1, ACR2, and ACR3. The RTS and CTS pins will now be active.

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



Note: +3.3V connection bubbles must be linked together.

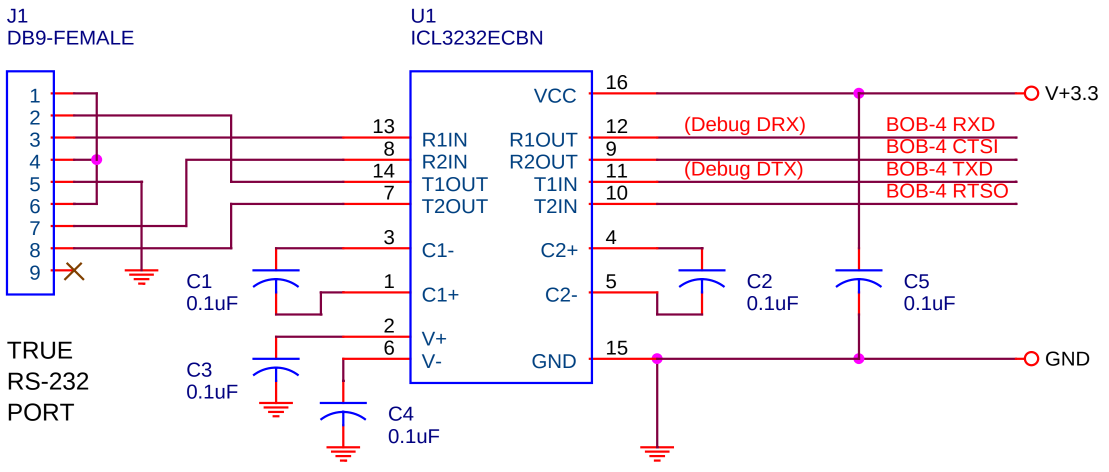
BOB-4S BASIC HOOKUP



BOB-4S (30-pin SIMM version) does not output 3.3V power on pin 1 by default. See **Pin Description Table**. Pins 1~3 are arranged to facilitate kludging 7805 style voltage regulators on BOB-3 host boards.

RS-232 Hardware Interface:

BOB-4 applications often benefit from bi-directional communication and data flow control, especially at the highest baud rates. 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 (as when this circuit is used for a Debug Port interface). 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 that have internal cross-connections.

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. Analog Devices' ADM3202ANZ is a good choice if you prefer the conventional DIP-16 package. 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.

Hardware Configuration:

ACR3 must **not** be grounded if anything other than standard (BOB-3 compatible) RS-232 style asynchronous serial communication and baud rates from the table below are desired. ACR0~2 assume new roles when ACR3 is not grounded. Use 10K pullup resistors on all four pins; **do not drive them active-high** (except SPI clock input may be driven to +3.3V). These pins are tested during BOB-4 initialization and used to select SPI configuration as well as main serial port handshake mode. SPI master mode is dedicated for use with supplementary font memory chips such as Atmel's AT45xxx DataFlash series. If this option is implemented, then the main serial port **must** be used to control BOB-4. See table:

Option	ACR0	ACR1	ACR2	ACR3	Bit Rate & Configuration
1	Lo	Lo	Lo	Lo	1200bps XON/XOFF
2	Lo	Lo	Hi	Lo	2400bps XON/XOFF
3	Hi	Lo	Hi	Lo	4800bps XON/XOFF
4	Hi	Hi	Hi	Lo	9600bps XON/XOFF
5	Lo	Hi	Hi	Lo	19,200bps XON/XOFF
6	Hi	Hi	Lo	Lo	38,400bps XON/XOFF
7	Lo	Hi	Lo	Lo	76,800bps XON/XOFF
8	Hi	Lo	Lo	Lo	153,600bps XON/XOFF
9	SPI Clock Out	Lo	Lo	Hi	SPI Master. Main serial port is async with software setup* defaulting to 9600bps.
10	SPI Clock Out	Lo	Hi	Hi	SPI Master. Main serial port is in synchronous mode. Clock input is on SCK/CPHA pin.
11	SPI Clock In	Hi	Lo	Hi	SPI Slave. Main serial port is disabled.
12	Lo	Hi	Hi	Hi	SPI disabled. Main serial port is async at 115.2kbps with RTS/CTS flow control.
13	Hi	Hi	Hi	Hi	SPI disabled. Main serial port is async at 115.2kbps with XON/XOFF flow control.

* For main serial port software setup, see **v** command (n=40~42). Save the new configuration and reboot.

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: http://www.edn.com/contents/images/6335309.pdf But don't use a resistive divider with more than 200 ohms of equivalent series resistance to BOB-4's RXD input pin! Series resistance to all other BOB-4 logic inputs should stay below 500R.

Firmware Upgrades & Custom Font Loading:

BOB-4 must be interfaced to a PC serial COM port in order to load new firmware or custom fonts, using the BOB-4 Conscriptor program. It's often helpful to attach a PC to BOB-4's debug port during new product development cycles, and the debug port is a bit more reliable than the main port for firmware installation. It's not necessary to include RS-232 interface hardware in production if it's not needed in your application — this interface can be part of an adapter cable that mates with a low-cost debug port connector in production boards. The BOB-4 Conscriptor Help system discusses firmware upgrade procedures in detail.

SPI Slave Mode

Except for buffer overflow prevention, BOB-4 uses the same format for commands and printable data on the Main Serial port and SPI slave port. Here's a quick summary of the hookups to use BOB-4 as an SPI slave device:

> ACR1 and ACR3 must be pulled up (10K to +3.3V), and ACR2 must be grounded.
>
> Host MOSI → BOB-4 MOSI (3.3V CMOS logic)
> Host MISO ← BOB-4 MISO (3.3V CMOS logic)
> Host SPI Clock Out → BOB-4 ACR0/SPCK (3.3V CMOS logic)
> Host Slave Select Out → BOB-4 SS0\ (3.3V CMOS logic)
>
> CTSI/CPOL and SCK/CPHA must be pulled up to 3.3V or grounded to select an SPI clock mode.

BOB-4 can operate in all four SPI clocking modes. SPI clock mode is configured using the CTSI/CPOL (clock polarity) and SCK/CPHA (clock phase) input pins. These pins are interrogated once after reset, just after the ACRx pins. Both pins have (weak) internal pullups, but as usual, Decade Engineering recommends 10K external pullups to insure good EMI rejection. See table:

CTSI/CPOL	SCK/CPHA	SPI Mode
Lo	Lo	0
Lo	Hi	1
Hi	Lo	2
Hi	Hi	3

BOB-4's basic SPI receive speed limit is 640kHz, but the SPI clock can go as high as 25MHz if sufficient inter-byte delay is added to stay under 80k bytes per second. BOB-4 is able to *process* received characters (from the receive data buffer) at a maximum rate of about 7,000 characters per second, but big fonts and some commands take extra execution time. Application programs should check BOB-4 receive buffer space as necessary to prevent overflow (see below).

In SPI slave mode, the SPI slave select input pin SS0\ is used for data synchronization. The SPI master should set this line high (false) between data transmissions. The master may begin driving the SPI clock pin ACR0/SPCK at any time during initialization, but it must wait for RTSO low (true) or one second from the release of reset or from power-up before driving any other SPI input lines. RTSO serves no other function in BOB-4 SPI systems. Here's a bit of advice from BOB-4's firmware programmer:

> "You can leave SS0\ active for long periods of time, and it makes no difference to BOB-4. The only caveat is that SPI has no other way to synchronize bytes; i.e. no start or stop bits. In theory, SS0\ could be left permanently active, so long as everything stayed in sync. I would activate SS0\, take a slight pause, send everything I had buffered, take another slight pause, and then deactivate SS0\."

SPI is bidirectional; for every byte received, a byte is simultaneously 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 consists of two bytes: 0xFF, and <status-byte>. See table:

Bit Field of <status-byte>	Data Coding
Bit 7	0: Reserved for future use
Bit 6	0: NTSC standard selected 1: PAL standard selected
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	Space remaining in receive buffer (n); see text

Incoming data will be lost if the receive buffer overflows. SPI status can be used to determine how much space remains in the receiver queue, to allow data flow control. Bits 3~0 of <status-byte> indicate space remaining in the receive buffer. 2^n (two to the power of bits 3~0) gives the minimum number of bytes available. For instance, if 899 bytes remain, then n will be 9, indicating that at least 512 bytes remain. If n is zero, then no space remains in the receive buffer and the host must transmit no additional data. In this case, the host may continue to poll until buffer space is once again reported.

SPI status information is 'out-of-band' data, which can occur any time. Worst case, BOB-4 could be sending a UTF-8 code sequence, and a status byte pair could appear in the middle. For instance, if BOB-4 is sending 0xC2 0x37, then you could get 0xC2 0xFF 0x00 0x37. On the user side, whenever 0xFF is seen, the next incoming byte must be grabbed and treated as a status byte. Otherwise, the current byte should go into your host receive queue. On the BOB-4 side, if 0xFF is seen, it's grabbed immediately and status is transmitted. Once the status sequence has been sent, BOB-4 resumes sending its transmit queue.

Look up Atmel application note AVR151 for detailed information on SPI port operation in the AVR microcontroller family. Other microcontroller manufacturers may offer similar application assistance.

SPI Master Mode

In SPI master mode, BOB-4 accommodates a variety of Atmel DataFlash^{tm} SPI flash memory chips (SPI slave devices) for supplementary font/bitmap data storage. Here's a list of the ones that BOB-4 currently recognizes, including memory size and support status:

Base Model Number	Byte Capacity	Status
AT26F004	512kB	Tested
AT26DF041	512kB	Known
AT26DF081A	1MB	Tested
AT26DF161	2MB	Known
AT26DF161A	2MB	Coverage
AT26DF321	4MB	Known
AT45DB011B	128kB	Tested
AT45DB011D	128kB	Family
AT45DB021B	256kB	Tested
AT45DB021D	256kB	Family
AT45DB041B	512kB	Tested
AT45DB041D	512kB	Tested
AT45DB081B	1MB	Family
AT45DB081D	1MB	Tested
AT45DB161D	2MB	Tested
AT45DB321C	4MB	Family
AT45DB321D	4MB	Tested
AT45DB642D	8MB	Tested

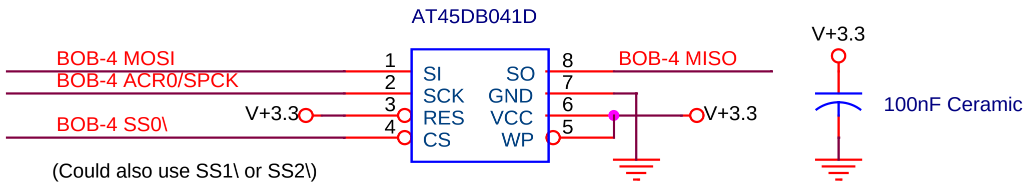
Status Key:

Tested: Samples have been tested and confirmed functional.
Family: Likely to work. At least two closely related types were tested.
Coverage: Likely to work, but only a single related type was tested.
Customer: A customer reported success with this device. Unconfirmed.
Known: Recognized by BOB-4, but never tested. It might work.

Poor PCB design can limit the usable SPI communication rate to something far below device capabilities. This becomes a critical issue when BOB-4 is expected to print characters quickly from custom fonts in external memory.

The older AT45 series (B suffix) devices are not specified for high SPI clock rates. Decade Engineering recommends against using them with BOB-4 due to multiple performance-limiting characteristics.

Here's a hookup example for an external SPI font memory chip. As with all high-performance logic ICs, it's important to keep power supply decoupling capacitor lead lengths near zero and do a good job routing the signal tracks:

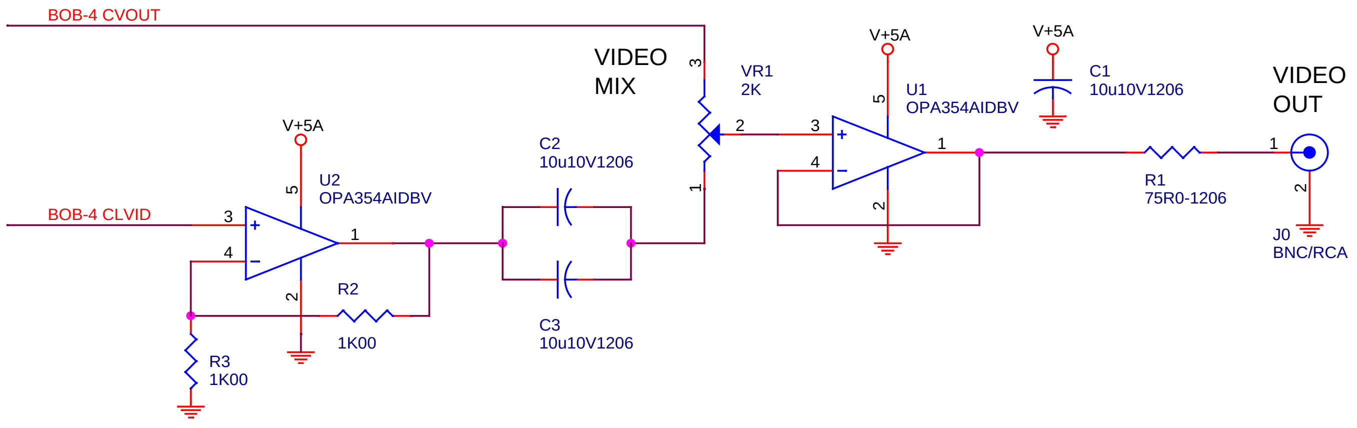


The BOB-4 Conscriptor loads font data into SPI memories or a reserved portion of the system processor's on-board flash memory according to user selections. See the **v** command with n=56~63 to enable one or more outboard SPI memories and configure their clock rates. Save the new configuration and reboot BOB-4 before attempting to load the memories with data. After they're loaded, BOB-4 discovers the new font files and makes them available for printing.

Before proceeding, the BOB-4 Conscriptor asks the user for a device number corresponding to the memory space that will receive custom font downloads. Devices 0~2 are external memories that may be attached as SPI slaves, with device zero corresponding to the slave selected by SS0\, etc. Device 3 is not implemented. Device 4 is the internal 62kB flash memory space, which is the only space available for custom fonts if no SPI memories are installed.

Variable Transparency

BOB-4 supports manually variable video overlay transparency, but this feature requires external application circuitry. The schematic below is an example of what's necessary. Basic circuit functions have been confirmed. Use the new video output at J0 instead of the BOB-4 CVOUT pin.



Using BOB-4 in a Conventional Cable TV System

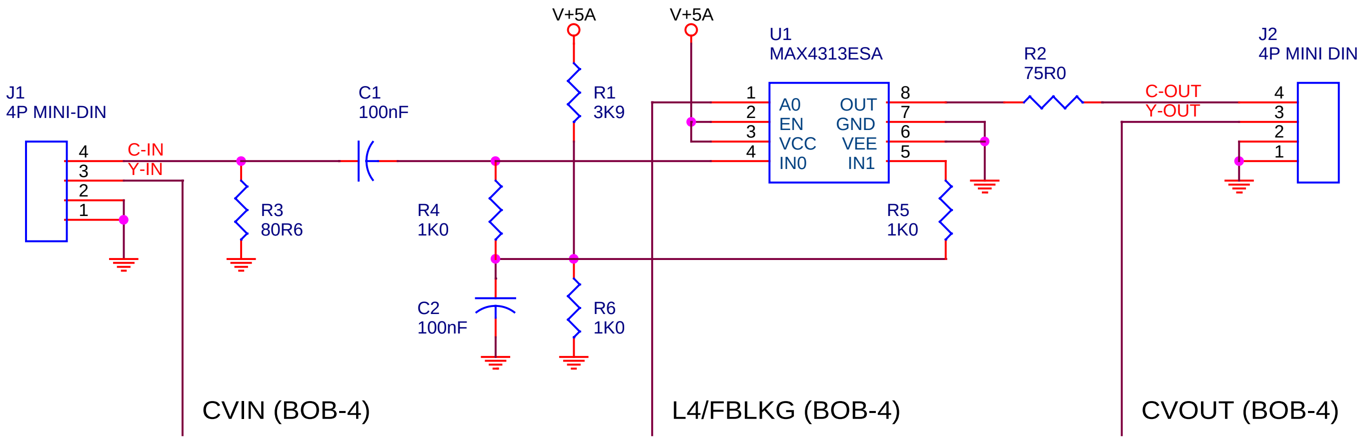
It's not possible to insert a single BOB-4 in a cable TV system and display the same text on all channels at once. There are a number of reasons for this. In a cable system, video signals are modulated onto RF carriers at different frequencies (that's how the TV tuner is able to pick out just one). The 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)

The L4/FBLKG pin 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.

Application Programming

Asynchronous Communication (RS-232) Basics:

Main serial port communication parameters default to **8N1** (eight data bits, no parity, one stop bit). Bit rates and other communication options are configured by strapping pins ACR0~3. See **Hardware Configuration** table. See **v** command (n=40~42) for main port configuration via software (when allowed by pin strapping). Configuration straps are sampled only at system initialization time. Debug port baud rate is fixed to 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 the **Troubleshooting** section for additional comments on baud rate issues.

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 enabled). BOB-4 may send a garbage character or two during initialization. BOB-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 or SPI port.

Your host controller must manage data flow control handshaking correctly if you transmit data to BOB-4 continuously at a high rate. BOB-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, 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}. 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 BOB-4, hardware flow control should be used. Since BOB-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. See Pin Description Tables for RTS and CTS hookup. 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 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 BOB-4 processor's own transmitter does meet this requirement when CTS is asserted at the input.

BOB-4 Native Command Mode (default):

Much like a conventional serial terminal or printer, BOB-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 commands 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 BOB-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. PC terminal programs normally generate <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.

BOB-4 commands are a subset of the ECMA-048 standard. BOB-4 recognizes most escape commands in the DOS ANSI command set, except those inappropriate for BOB-4. 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. BOB-4 allows negative arguments, but because the minus character (hyphen) 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. If some arguments 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 BOB-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, collect text for boot scripts, and set up crawl displays. *String* input is initiated with <ESC>X and terminated with <ESC>\. *String* input may also begin with <ESC>Q and end with <ESC>R. **Both forms of the *String* input start/end commands are necessary to store a boot script that selects a font by name or invokes a text crawl.** The boot script editor in the 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

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 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 BOB-4 via the file transfer capability of a terminal program.

Bitmap graphics are treated like fonts with only one character. To display a bitmap graphic object, just select its font and print the character designated for this purpose by the BOB-4 Conscriptor. The Space character (0x20) is used at this writing (16 April 2007), but it’s not guaranteed to stay that way. <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 BOB-4 are numbered from the top left corner, where X=0 and Y=0. In the vertical axis, Y values increase downward. In 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 -32767 and +32767 pixels in both axes. See **x** and **q** commands for additional discussion.

Character 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. 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. BOB-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.

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). 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.

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>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*;*m*f** Equivalent to **H** command.

H **<CSI>*n*;*m*H** 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.

J **<CSI>*n*J** Clears all or 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 all or part of the current 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 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>*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. The **q** command offers an alternate technique.

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.

Bold, reverse, and blink are mutually exclusive. For example, if one attempts to set bold and reverse at the same time, reverse takes precedence. Bold 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 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)

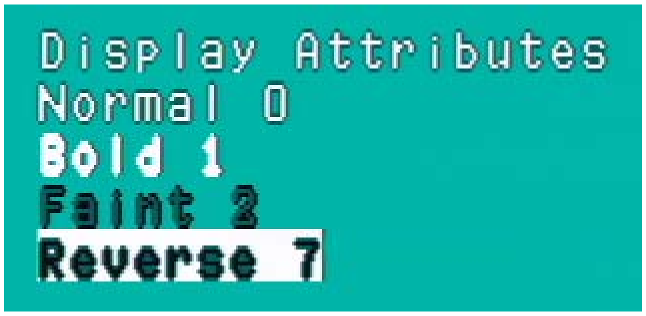
n=10~19: Select font by index (0~9)

BOB-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



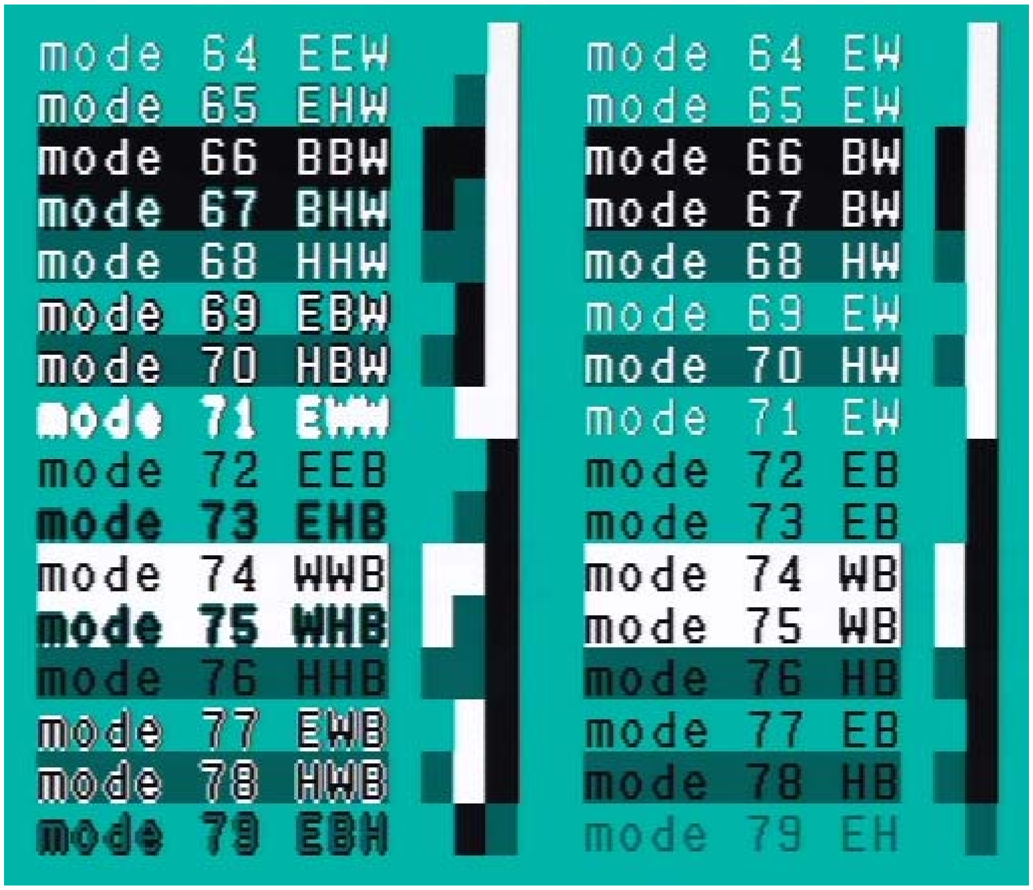
Display Attributes
Normal 0
Bold 1
Faint 2
Reverse 7

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 BOB-4 (and BOB-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 BOB-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 4.0S 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 BOB-4's default boot script. Note that <NUL> (0x00) is ignored, so it cannot serve as a trigger character.

v <CSI>*n*;*m*v Set/save configuration; configuration options listed by value of *n* parameter

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

m=0: Switching to BOB-3 mode allowed
m=1: BOB-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 (this command was added in V4.2.6 and expanded in V4.2.7)

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 BOB-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

BOB-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, BOB-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 BOB-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;*m*v** 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 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." Note that 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 (default = 416), i.e. the number of overlay pixels per horizontal line to be used for display. 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 offset in pixels from default value. If m=0, vertical start position is returned to default value (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". Note that negative *m* arguments must be prefixed with "<" rather than a minus sign.

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

Parameter *m* is display height in pixels (default = 208 for NTSC, 248 for PAL), i.e. the number of horizontal lines to be used for display. 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;*m*v** Overlay enable (printing/drawing in RAM is allowed even when display is disabled)

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

29 = **<CSI>29;*m*v** 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 (not available in BOB-4S)

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

Not available in the SIMM version of BOB-4. 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 (not available in BOB-4S)

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

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 **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 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, then 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 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 BOB-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 async.

Correct ACRx pin strapping is required; see table in **Hardware Configuration** section. This command has no effect until the new configuration is saved and BOB-4 is rebooted. For asynchronous serial (RS-232 style) communication, **m** can be any value from 46 to 1000000 (one million). 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.

Synchronous mode is selected if **m**=0. (-------------not tested--------------) Main port synchronous mode uses UART hardware, not SPI hardware. The synchronous clock input pin is SCK/CPHA. The data path remains on RXD and TXD.

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

44 = SPI mode; default is ‘hardware’ (controlled by pin hookup options)

m=0~3: SPI mode is set to the value of m
m>3: SPI mode is set by hardware (default)

48 = L0 output select* (In the SIMM version, 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 (best to help achieve flicker-free animation)

52 = L4/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.

56 = **<CSI>56;mv** Select SPI slave device 0 (chip select is SS0\)
57 = **<CSI>57;mv** Select SPI slave device 1 (chip select is SS1\)
58 = **<CSI>58;mv** Select SPI slave device 2 (chip select is SS2\)
59 = **<CSI>59;mv** Select SPI slave device 3 (not currently useful)

m=0: No SPI device present (default)
m=1: Use installed SPI device on this chip select line

Up to three SPI slave devices may be installed.

60 = **<CSI>60;mv** Select bit rate for SPI slave device 0
61 = **<CSI>61;mv** Select bit rate for SPI slave device 1
62 = **<CSI>62;mv** Select bit rate for SPI slave device 2
63 = **<CSI>63;mv** Select bit rate for SPI slave device 3 (not currently useful)

m=2~255 inclusive. BOB-4 sets the SPI slave clock to 47.9232MHz/m. Default is normally 6MHz. Some recognized device types (older AT45 series with B suffix) are not specified for high rates, so BOB-4 defaults to the maximum compatible rate in those cases. Decade Engineering recommends against customers using such devices due to multiple performance-limiting characteristics.

Poor PCB layouts can limit the useable SPI communication rate to something far below device capabilities. This becomes a critical issue when BOB-4 is expected to print characters quickly from custom fonts in external memory.

w <CSI>*n*w 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>*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 huge canvas with X and Y ranges of -32767 to +32767 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 printing may proceed across the canvas, with only the portion inside the window appearing in the display, and continue off to right infinity (actually +32767). Unfortunately the "huge canvas" doesn't really exist in BOB-4, so it's not possible to just reposition the window after printing and display other portions of the text.

z <CSI>*n*z 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. BOB-4 firmware includes these eight permanent 'system' fonts:

n	Name	X/Y Size	Pixel Depth	Chars	Description
0	bob3	12x13	2	304	BOB-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	"BOB-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 BOB-4 font numbers and names are listed in the "<CSI>5}" configuration report (or use the debug port "config" command; its report is identical).

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

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

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

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

The BOB-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;m*} Transmit system information

n=1: BOB-4 transmits a BOB-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: BOB-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 BOB-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=6: Read ADC6 and ADC7 inputs (useful in BOB-4H only). See table for use of **m** parameter:

Bit Field of m	Value	Resulting Configuration
Bits 7~2	0	Reserved
Bit 1	0	Data format: <data6><SP><data7><CR><LF>
	1	Data format: <CSI>6;<data6>;<data7>}
Bit 0	0	Return raw data, i.e. numbers ranging from zero to 1023
	1	Return voltage in millivolts (useful only if AREF = +3.3V)

Analog input channels ADC6 and ADC7 of the processor chip are connected to BOB-4H header pins 20 and 22 respectively. Input signal source impedance must stay below 21k ohms. AREF (pin 19) must be connected to an ADC reference voltage between +2.6VDC and +3.3VDC, and ADC input voltages must not exceed AREF. Reference voltage accuracy directly affects measurement accuracy, which is further degraded by the ADC error budget of Atmel's AT91SAM7S256 processor chip. Basic accuracy of the on-board +3.3V regulator is 2%. ADC6 and ADC7 are sampled every 20mS, but only the most recent samples are returned by the command. The CPU is not disabled during measurements, and there's no internal averaging.

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. 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 Linux): **`pgmtopnm <your-file | pnmtogif >a-gif-file`**

Vector Graphics Commands

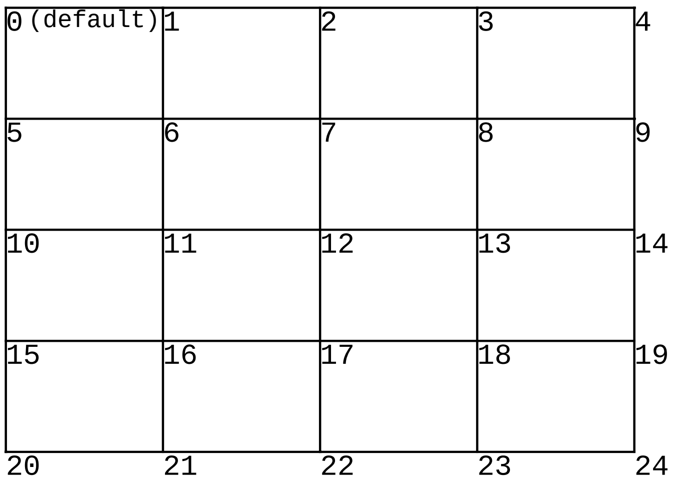
Graphics primitives in BOB-4 are modeled on PostScript. A path is first created, and then the path is either stroked or filled. **StrokePath** and **FillPath** are the only graphics primitives that draw anything. The **MoveTo**, **LineTo**, **Arc** and **ClosePath** commands are used to construct paths. 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 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:



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.

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.

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 BOB-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.

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> 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 BOB-4 applications, and load new firmware or custom fonts. 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. Note: The BOB-4 Conscriptor will use 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 BOB-4 setup feature therefore depends on a main port interface. Debug port communication settings are fixed as follows:

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

The debug port DTX (debug transmit data) and DRX (debug receive data) lines can be interfaced to a PC with the same circuit given for the main port TXD and RXD lines, except that RTS and CTS (hardware handshake) are not implemented. Loop these signals back as suggested, and observe the caution regarding open CMOS inputs on the RS-232 interface chip.

BOB-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 ports.

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 SPI or main serial ports.

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 BOB-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.

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 <CSI>9;0v, then escape to BOB-3 mode with <CSI>{. While in BOB-3 mode, any occurrence of <CSI> immediately recalls BOB-4 mode (and the associated command will be executed). Send <CSI>{ again if you must return to BOB-3 mode. Since the default boot script contains BOB-4 commands to draw the BOB-4 splash screen, it invokes BOB-4 mode. To prevent this, the boot script must be cleared or replaced. 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)
<ESC>X<ESC>\<CSI>8v	(clear the boot script)
<CSI>1v	(save configuration)

Only the BOB-3 12x13 look-alike fonts are available in BOB-3 mode. The original BOB-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 BOB-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 BOB-3 commands (see BOB-4 **v** command, n=27).

Serial communication parameters are **8N1** (8 data bits, no parity, 1 stop bit). Bit rates other than 9600bps are configured by grounding one or more of ACR0~2 on the BOB-4 socket at installation time. ACR3 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 ACR3 is not low at boot, then BOB-4 bit rates may be selected, including those set by the BOB-4 **v** command (n=40).

No RS-232 hardware interface is needed for use with a Parallax^{tm} BASIC Stamp^{tm} or most industry-standard microcontrollers, but some 5V MCUs may not be directly compatible with 3.3V CMOS logic input. If the BASIC Stamp^{tm} 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^{tm}. If the RTS/CTS 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 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 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 in BOB-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.

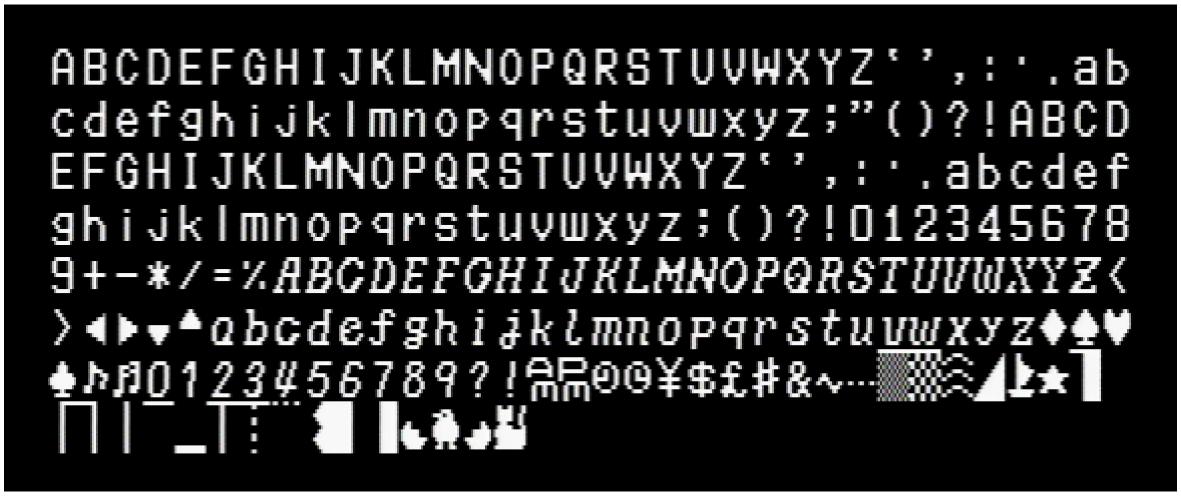
There are new blinking commands in BOB-3 mode: **{G1** is the global blink enable. **{G0** turns it off. Both commands clear the screen and change the character drawing mode as needed. Note that the whole screen is cleared regardless of window setup (**q** in BOB-4 command set).

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

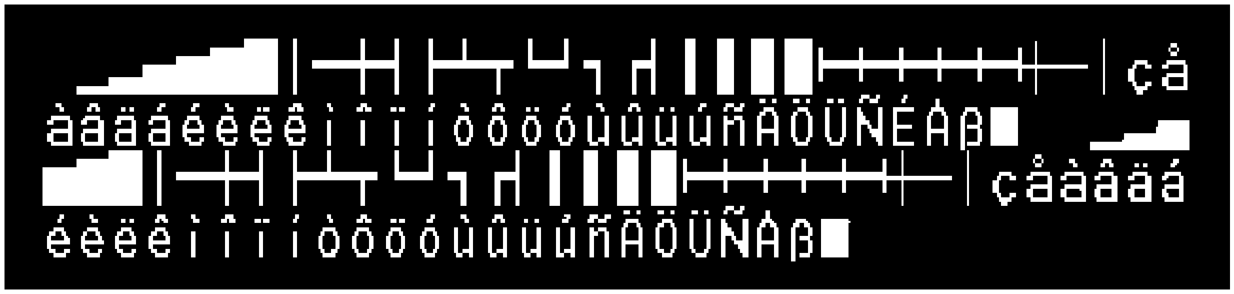
Many of the original BOB-3 commands are not implemented. For instance, you cannot download a font in BOB-3 mode or install a boot script or start a text crawl. There is enough functionality to support many but not all applications previously using the BOB-3 module. Switch to native BOB-4 mode if more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

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

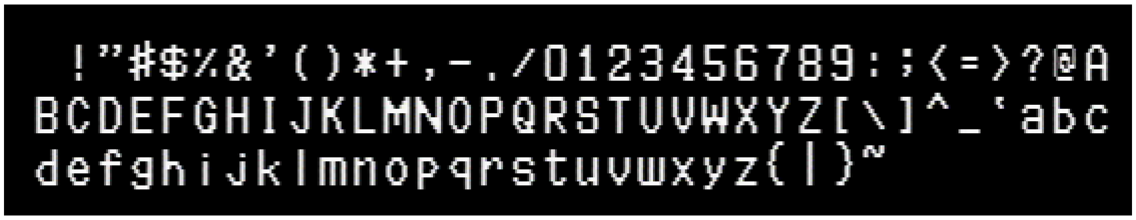
System Fonts



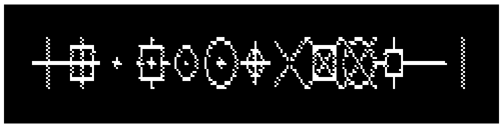
Default 12x13 Font Part 1 (shown in BOB-3 ROM sequence)



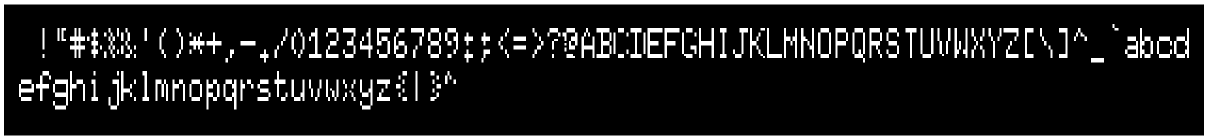
Default 12x13 Font Part 2 (was BOB-3 Default RAM Font)



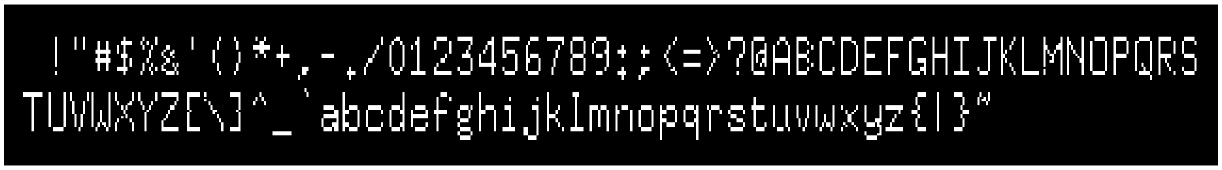
Default 12x13 Font ~ ASCII Sequence



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

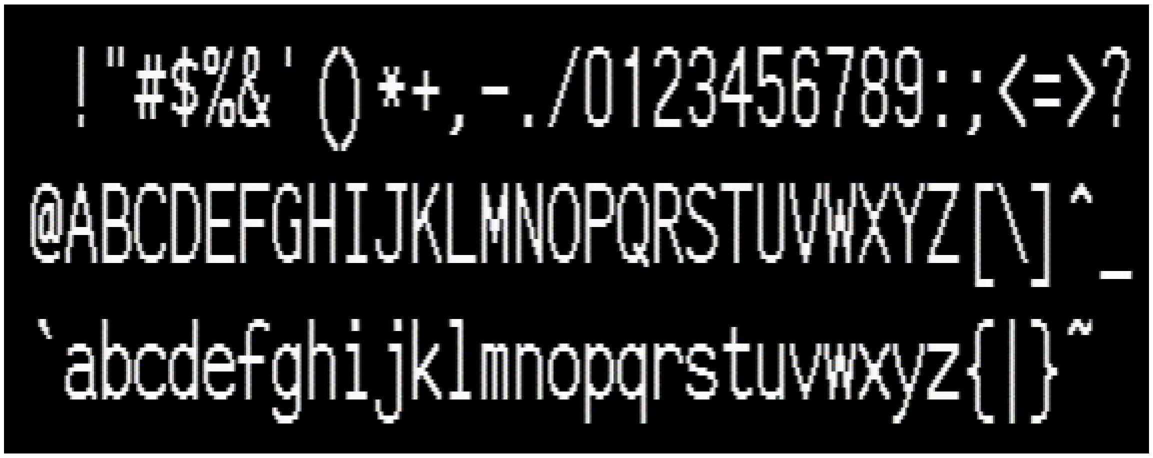


6x10 Font



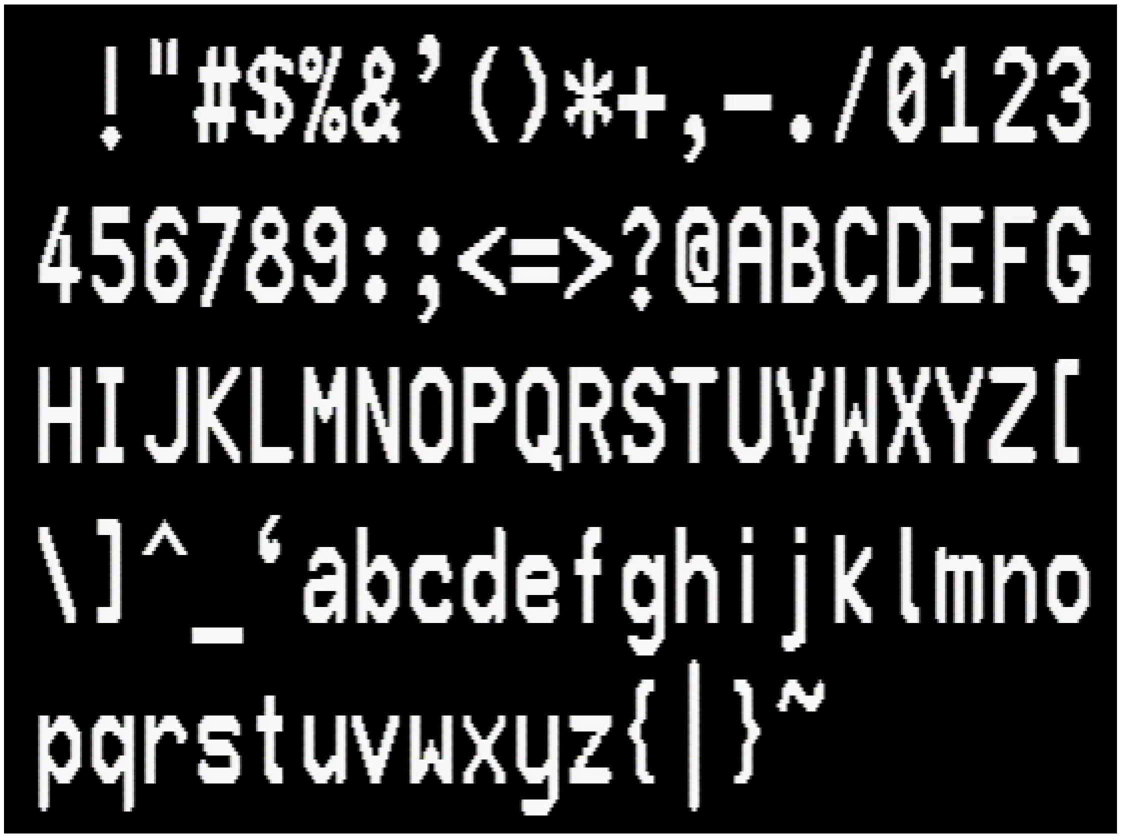
```
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRS
TUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
```

8x13 Font



```
 !"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
`abcdefghijklmnopqrstuvwxyz{|}~
```

13x34 Font



```
 !"#$%&'()*+,-./0123
456789:;<=>?@ABCDEFG
HIJKLMNOPQRSTUVWXYZ[
\]^_`abcdefghijklmno
pqrstuvwxyz{|}~
```

20x40 Font

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

20		21	!	22	"	23	#	24	$	25	%	26	&	27	'	28	(	29	)	2A	*	2B	+
2C	,	2D	-	2E	.	2F	/	30	0	31	1	32	2	33	3	34	4	35	5	36	6	37	7
38	8	39	9	3A	:	3B	;	3C	<	3D	=	3E	>	3F	?	40	@	41	A	42	B	43	C
44	D	45	E	46	F	47	G	48	H	49	I	4A	J	4B	K	4C	L	4D	M	4E	N	4F	O
50	P	51	Q	52	R	53	S	54	T	55	U	56	V	57	W	58	X	59	Y	5A	Z	5B	[
5C	\	5D	]	5E	^	5F	_	60	`	61	a	62	b	63	c	64	d	65	e	66	f	67	g
68	h	69	i	6A	j	6B	k	6C	l	6D	m	6E	n	6F	o	70	p	71	q	72	r	73	s
74	t	75	u	76	v	77	w	78	x	79	y	7A	z	7B	{	7C	|	7D	}	7E	~	7F	□
80	¥	81	♪	82	♫	83	£	84	[illegible]	85	[illegible]	86	~	87	…	88	▓	89	▒	8A	[illegible]	8B	◢
8C	[illegible]	8D	★	8E	▐	8F	▌	90	⎡	91	⎤	92	┃	93	◆	94		95	▁	96	┬	97	⋮
98	‥	99	[illegible]	9A	▌	9B	▐	9C	[illegible]	9D	[illegible]	9E	[illegible]	9F	[illegible]	A0		A1	*!*	A2	*"*	A3	*#*
A4	*$*	A5	*%*	A6	*&*	A7	*'*	A8	*(*	A9	*)*	AA	***	AB	*+*	AC	*,*	AD	*-*	AE	*.*	AF	*/*
B0	*0*	B1	*1*	B2	*2*	B3	*3*	B4	*4*	B5	*5*	B6	*6*	B7	*7*	B8	*8*	B9	*9*	BA	*:*	BB	*;*
BC	*<*	BD	*=*	BE	*>*	BF	*?*	C0	*@*	C1	*A*	C2	*B*	C3	*C*	C4	*D*	C5	*E*	C6	*F*	C7	*G*
C8	*H*	C9	*I*	CA	*J*	CB	*K*	CC	*L*	CD	*M*	CE	*N*	CF	*O*	D0	*P*	D1	*Q*	D2	*R*	D3	*S*
D4	*T*	D5	*U*	D6	*V*	D7	*W*	D8	*X*	D9	*Y*	DA	*Z*	DB	*[*	DC	**	DD	*]*	DE	*^*	DF	*_*
E0	*`*	E1	*a*	E2	*b*	E3	*c*	E4	*d*	E5	*e*	E6	*f*	E7	*g*	E8	*h*	E9	*i*	EA	*j*	EB	*k*
EC	*l*	ED	*m*	EE	*n*	EF	*o*	F0	*p*	F1	*q*	F2	*r*	F3	*s*	F4	*t*	F5	*u*	F6	*v*	F7	*w*
F8	*x*	F9	*y*	FA	*z*	FB	*{*	FC	*|*	FD	*}*	FE	*~*	FF	◀	100	▶	101	▼	102	▲	103	♠
104	♥	105	♣	106	·	107	©	108	®	109		10A	°	10B	€	10C	┼	10D		10E		10F	
110		111	▁	112	▂	113	▄	114	▆	115	▇	116	█	117	┃	118	━	119	╋	11A	┫	11B	┣
11C	┻	11D	┳	11E	┗	11F	┛	120	┓	121	┏	122	▎	123	▍	124	▌	125	▋	126	▉	127	┝
128	┝	129	┿	12A	┿	12B	┥	12C	┤	12D	┼	12E	─	12F	│	130	ç	131	å	132	à	133	â
134	ä	135	á	136	é	137	è	138	ë	139	ê	13A	ì	13B	î	13C	ï	13D	í	13E	ò	13F	ô
140	ö	141	ó	142	ù	143	û	144	ü	145	ú	146	ñ	147	Ä	148	Ö	149	Ü	14A	Ñ	14B	É
14C	Å	14D	ß	14E	█	14F																	

Troubleshooting

Interference with GPS receivers:

Beginning in October 2007, Decade Engineering introduced a hardware design revision to eliminate interference with GPS receivers. Older BOB-4 modules, when installed without shielding and filtering, emit incidental RF that may interfere with GPS receivers when physical separation is less than about three feet (1m). Contact Decade Engineering for assistance with this issue.

A gentle reminder: Sensitive radio receiving equipment should never be operated in close proximity to unshielded digital electronics of any kind. There's a powerful incentive to reduce mass in airborne systems, but without effective shielding and filtering, you have a perfect recipe for disaster. It is simply unreasonable to expect anything more than a slight respite from this brutal reality.

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 BOB-4 program code.

USB/RS-232 adapters can cause firmware upgrade problems even if normal communications with BOB-4 have been confirmed. It's best to use a 'real' PC RS-232 serial port (COM port) for communicating with BOB-4. If that's not possible, try more than one brand of USB adapter. We've heard that IOGEAR makes a reliable one, and it has been found at Wal-Mart. Decade Engineering staff hasn't verified this report. EasySync dual RS-232 adapters have worked well in our shop.

Despite efforts to prevent this, we're suspicious that old BOB-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.

Early production BOB-4S (SIMM) 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 a green dot are Pb-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 proceed to 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 SIMM pin 10 to anything in your system, then a 10K pullup resistor (to +3.3V) should be added on this line to insure EMI resistance.

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, BOB-4 expects the next character to be a command. If you're especially unlucky, BOB-4 executes an unwanted command that sends you into the weeds.

Total resistance in series to BOB-4's on-board MCU inputs must not exceed 500 ohms. Watch out for logic level translators with internal series resistance (e.g. MAX3001). Current production Atmel MCUs may have excessive input pullup current, which can result in failure to reach the logic low input threshold when series resistance is added. Most of BOB-4's logic inputs are linked *directly* to the MCU chip, but **the RXD input is different**. It's protected with 300 ohms of series resistance (R9) on the board, so resistance external to BOB-4's RXD pin must stay below 200R. R9 was 680R (marked "681") in early production BOB-4 modules. It should therefore be replaced (or bridged) to bring total resistance below 500R if communication reliability is questionable. R9 has a similar location in both versions of BOB-4: Topside, between the two biggest ICs, adjacent to C6. In the SIMM version, R9's PCB marking is unfortunately squished into the tiny space between R9 and U2. Decade Engineering allows an exception to the standard hardware warranty policy for customer rework at R9, but maximum ESD precautions are mandatory! Please return the board to Decade Engineering if your technical skills are in doubt.

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.

Another likely cause for wrong characters in the display is power supply and/or ground noise. Switch-mode power supplies are notoriously noisy, and this problem can easily be compounded by poor ground layout. As logic voltage levels are reduced (BOB-4 is a 3.3V system), these issues grow more prominent. It may be necessary to employ a first-class oscilloscope and expert probing techniques to be certain such noise is eliminated. In particular, it's essential to use a very short probe ground connection. Watch out for RS-232 interface chips with built-in charge pumps. They will modulate the supply rail with noise if they're not properly decoupled!

When BOB-4 is used with BASIC Stamp^{tm} modules, it's important to confirm that correct serial data polarity is coded in the Stamp. Decade Engineering docs conflict with Parallax docs on this item, which can lead to confusion. Of course we feel that Parallax chose an incorrect point of reference! It boils down to this: Data polarity presented to BOB-4's RXD input must be inverted relative to a standard RS-232 transmission line. Data output from BOB-4's TXD pin is also inverted relative to an RS-232 transmission line. This is a natural result of the **direct** serial interface in BOB-4. If you add an industry-standard (inverting) RS-232 interface circuit in order to use BOB-4 on a PC serial COM port, then data polarity on the transmission line will be correct (non-inverted).

BOB-4 acts confused or unresponsive:

Some video monitors (and other downstream equipment) can fail to detect video input when BOB-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 BOB-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. BOB-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, BOB-4's default video standard is NTSC. BOB-4 switches to PAL when it detects incoming video and engages genlock mode. A screen clear operation is triggered because NTSC and PAL display geometry differs. 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 emulation programs such as HyperTerminal^{tm} 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

BOB-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 BOB-4. If that's not possible, try more than one brand of 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\ (BOB-4S pin 15, BOB-4H pin 36) low as power is applied, even if BOB-4 is unable to communicate with the host computer. VMIS\ normally drives a video input status indicator. 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.

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 with a good scope. Ripple suppression and decoupling 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/or capacitor equivalent series resistance (ESR). This information is sometimes buried deep in the regulator's data sheet. Check power supply start-up time, especially if you're using a switcher, because soft-start features are integrated into most modern switch-mode power control chips. It shouldn't be more than several milliseconds — 40mS is definitely too long. Also review the **Cautions** section near the beginning of this document, and the BOB-4 Power Supply Application Note.

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 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. The "+" side of the capacitor should connect to BOB-4.

Display shifts or disappears when incoming video is cut:

This problem occurs primarily in PAL countries, because BOB-4 modules ship from the factory configured for NTSC sync generation when there's no video input. To correct it, just reconfigure BOB-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 BOB-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, BOB-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 BOB-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 cable or TV broadcast 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 or halftone (see **Vector Graphics Commands**) before printing characters over it.

3. BOB-4H offers additional internal circuitry 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 implementing the variable transparency option and setting the transparency control below 100% contrast. See **Other Hookup Notes** in this document for a circuit suggestion.

5. Consider switching your system design to Y/C video (S-Video). With chroma and luma on separate channels, the chroma-luma crosstalk issue is eliminated. BOB-4 can handle Y/C video with a little assistance from external circuitry (also in **Other Hookup Notes**).

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.

The text overlay is unstable:

Display impairments such as vertical instability or ‘flagging’ can appear on LCD video monitors when BOB-4 is operating in local mode. This problem is often corrected by configuring BOB-4 to generate interlaced sync instead of progressive (non-interlaced) sync. BOB-4 generates progressive sync in local mode by default, because it reduces flicker and looks best on most monitors. See **v** command (n=17). Why just LCD monitors? We surmise that many video hardware design veterans have retired from the industry, leaving a less-seasoned crew on the job.

Overlay jitters can be caused by weak and/or noisy video applied to CVIN. 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 VCR playback signals. It may perform as desired, but overlay stability can be unacceptable with some VCRs and some (usually worn) cassettes. The symptoms are often worse in VCR 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. BOB-4 does not tolerate distorted sync as well as a typical TV monitor, making data overlay stability uncertain in these system environments. If 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.

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^{tm} and ProGold^{tm} (or PreservIT^{tm}), from Caig Laboratories. Apply the contact treatment to a cotton swab and polish all of the contact pads on your BOB-4S 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-4S module or SIMM socket contacts.

Also see **BOB-4 acts confused or unresponsive** (above), especially the paragraph on power supply issues.

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

The digital video detection algorithm in BOB-4 modules 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

V4.3.3 [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 vector graphics processing bug 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).

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 (cursor restore) to allow repeated usage.

V4.2.16 [06 May 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 alpha 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] Fixed a bug in text crawl using characters that don't exist in currently selected font.

V4.2.9 [20 March 2007] Fixed a screen erasure issue with scrolling. Increased number of display lines from 207 to 208 in NTSC mode with display area constraints on, to accommodate 16 complete lines of text with default font.

V4.2.7 [09 March 2007] Interim release. Expanded CR/LF interpretation options. Adjusted pixel insertion timing.

V4.2.6 [27 February 2007] Interim release. Added support for custom fonts and bitmap images, including UTF-8 decoding. Added SPI master mode and supplementary font/bitmap data storage. Changed *String* buffer input commands, to enhance reliability. Extended configuration report. Added UART setup command. Added <CR> and <LF> interpretation. Added configuration setting via debug port. Adjusted pixel insertion timing. Beginning with V4.2.6, <STX> (0x02, Ctrl-B) and <ETX> (0x03, Ctrl-C) no longer work for *String* buffer input control codes by default. A configuration command was added to allow re-activation if necessary. See **v** command with n=10.

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

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.

UTF-8 decoding is not implemented in firmware versions 4.1.1 and earlier. Non-ASCII characters in the BOB-3 look-alike fonts may be displayed only by switching to BOB-3 compatibility mode and using the **T** command in the

BOB-3 repertoire. See **v** command (n=9), and **{** command, both in the BOB-4 command set. Once written to the screen, displayed characters are not affected by command mode changes. It's perfectly reasonable to switch between BOB-3 and BOB-4 command modes as often as necessary to print the desired 12x13 characters, but this little dance should be unnecessary when all planned features are present in BOB-4 firmware.

V4.0.4 [06 June 2006] First production 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. A coding error in this version caused the ACR0~2 inputs to be sampled wrong-endian, making half of the entries in the baud rate selection table (with ACR3 low) incorrect. This problem was fixed in V4.1.1.

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:

---O---