

BLU

A Basic Lisa Utility

Documentation for V0.90

Updated: February 25, 2013

BLU is a Basic Lisa Utility. BLU runs on Apple Lisa base hardware without an operating system, and implements a few operations in support of the various operating systems that run on Lisa.

* denotes “new in BLU V0.90”

Updates require feedback

The users of BLU will be a very small group, so it is especially important that you report your experiences with BLU so that issues can be documented and improvements made.

Please report bugs and send suggestions to blubugs@SigmaSevenSystems.com - include the word Lisa in the subject line to get through the spam filter.

Or join the LisaList Google Group at <https://groups.google.com/forum/?!forum/lisalist>

For general help with Lisa, consult the LisaList and view the Lisa FAQ at <http://lisafaq.sunder.net/>

BLU web site (and bug list): <http://SigmaSevenSystems.com/BLU>

Some uses for BLU

- create a Lisa floppy disk from a disk image transferred from another computer
- create an image of a Lisa floppy disk or parallel port hard disk or Priam DataTower hard disk and transfer it to another computer
- “low level format” an Apple parallel port hard disk

License

BLU is licensed for your use, free of charge by Sigma Seven Systems Ltd., with no warrantee of any kind. BLU may not work as described. Using BLU may cause loss of data and have undesirable and undocumented effects. Use of BLU is at your own risk.

BLU is copyright 2013 by James MacPhail, portions copyright by Ray Arachelian. All rights are reserved.

System Requirements

Disk Drives Supported

- Twiggy Floppy
- 3.5" Single Sided / 400K Floppy
- 3.5" Double Sided / 800K Floppy
- ProFile 5MB and 10MB Hard Disk (and other sizes supported by X/ProFile™, IDEfile, etc.)
- Widget 10MB Hard Disk
- * Priam DataTower 80MB (approximately) Hard Disk

RAM

The floppy disk functions in BLU store an entire disk image in memory. As a result, at least 1MB of RAM is required to perform operations with Twiggy or 800k floppies.

Serial Communications

Except for those functions that BLU might perform “stand-alone” on Lisa, many BLU functions transfer data to/from another computer via RS-232 serial. The other computer is expected to be running terminal software that supports x-modem transfers.

Therefore, besides Lisa, you'll need a computer with a serial port (a USB-Serial converter may work), and terminal software that supports x-modem transfers (most terminal software does), and a suitable serial cable to connect the two computers. If you need to make a serial cable, see APPENDIX B Serial Cables.

BLU is configured to use the Serial B port, with a default setting of 57600 baud, 8 bits per character, no parity, 1 stop bit. Once BLU is running, the baud rate of the serial port can be changed via the Miscellaneous Functions menu. BLU can't keep up at 230k baud (so far, at least), but 115k works.

CRC, and 1K blocks are enabled for x-modem transfers. X-modem transfer progress is counted in 128 byte blocks, so if 1K blocks are in use, the progress counter will count by 8. When the data to be sent does not fill the last x-modem block, it is padded with \$1A.

Some BLU functions transmit status or error information via the serial port (when it isn't being used for x-modem transfers), so it can be useful to have your terminal program “record lines off top” to have a transcript of what happened. As BLU reads from the serial port only for x-modem transfers, you may turn on local echo and type in your terminal program to add comments to the transcript.

Handshaking

BLU is configured to use hardware handshaking; see APPENDIX B Serial Cables for information regarding handshaking connections. BLU does not use XOn/XOff handshaking.

Handshaking may be unnecessary for most functions and baud rates. You may need hardware handshaking for the higher baud rates, particularly for ProFile transfers from another computer to Lisa, but this depends on your terminal software: Some terminal software will begin sending another x-modem block before the previous one is acknowledged, but as BLU is not designed to buffer data before it is expected, this technique causes many re-sent blocks, making the data transfer very slow. This problem may be avoided using a hardware handshaking cable and configuring the terminal software to use hardware handshaking (eg. RTS/CTS or DTR&CTS).

If a transfer problem persists with hardware handshaking configured, try adjusting the terminal software to use plain x-modem rather than x-modem 1K or x-modem CRC.

When handshaking is active during transfers, annunciators are displayed in the menu bar. TX indicates Lisa is free to transmit, /T indicates Lisa is told it is not free to transmit. RX indicates Lisa is ready to receive, /R indicates Lisa is not ready to receive.

Macintosh Terminal Software

Regarding AppleWorks, ClarisWorks, and other terminal software that uses the “communications toolbox”:

The “serial tool” doesn't support a baud rate above 56k, whereas the “modem tool” does. However, the modem tool wants to “open” a connection by chatting with the modem before talking to the other end, so to satisfy it, BLU sends out CONNECT OK when configuring the serial port... this satisfies the modem tool and it considers the connection open.

To configure the modem tool, select “Direct Connection”, “DTR & CTS”, the desired baud rate, and set the phone number to “1”. To establish the connection, tell AppleWorks (or other terminal software) to open the connection, then use Configure Serial Port in the Miscellaneous Functions sub-menu to tell BLU to set the same baud rate (which causes it to send the connect string), and the modem tool should report the connection was established. Once the connection is open you can alter the baud rate etc. as desired, but if you quit, you'll need to repeat the procedure to open the connection again the next time.

Note: Some versions of the Macintosh application “ZTerm” will send MacBinary even when plain text is selected, so it cannot be used to send a disk image to BLU.

Getting BLU running the first time

Loading BLU on Lisa for the first time is easiest if a BLU floppy disk can be created on another computer, but BLU can be bootstrapped from “bare-metal” using the Service Mode built-in to Lisa’s ROM.

Once BLU is running, it can install itself onto a floppy disk (Twiggy or 3.5") or a parallel port hard disk; see the Miscellaneous Functions section for details. Once BLU is installed on a disk, it is bootable on any Lisa.

Floppy Disk

If you have a BLU floppy disk: turn on Lisa, press the space bar during the self test, and when you get to the "startup from" menu, insert the floppy disk and type Apple-2 (or Apple-1 if you are using an upper Twiggy drive) to boot from it.

Bootstrap

If you cannot obtain a floppy disk with BLU on it, you can load BLU from the serial port... see Appendix A Bootstrapping for further information. Once BLU is running, make a BLU floppy disk so that you don’t have to bootstrap next time.

Hard Disk

BLU can install itself on a parallel port hard disk, and the resulting disk is bootable on any Lisa from any parallel port: turn on Lisa, press the space bar during the self test, and when you get to the "startup from" menu, select the appropriate parallel port from the menu to boot from it.

Lisa 1 Tip

If you have a Lisa 1, a Lisa 2, a spare parallel port hard disk, and a way to make the 3.5" BLU floppy, you can get BLU running on a Lisa 1 without typing in the bootstrap code: connect the parallel port hard disk to the Lisa 2, run BLU on the Lisa 2 from a 3.5" floppy, install BLU on the hard disk, then shut down and move the hard disk to the Lisa 1.

Using BLU

BLU is entirely keyboard driven, the mouse is not used, and commands are not accepted from the serial port.

When BLU is waiting for keyboard input, an inverse question mark flashes at the right end of the menu bar.

Note the “Clear” key at the top left of the numeric keypad, which is used to terminate some functions.

The numbers displayed by BLU are hexadecimal (exceptions are baud rates and references to 400/800k disks).

When BLU is started, it loads completely and runs entirely in RAM, so you may switch disks freely once BLU is running. (Reminder: If you wish to change media connected to an X/ProFile, turn off the power to the X/ProFile first.)

Keyboard variations

✱ When BLU is waiting for a Y/N response, “Z” is considered to be the same as “Y”. (This is a result of German Lisa keyboards having the Y key in the location that sends the keycode associated with the Z key on other keyboards.)

If you are using a French keyboard, you may have to press “Q” to select a function labeled as “A”. (This is a result of French Lisa keyboards having the A key in the location that sends the keycode associated with the Q key on other keyboards.)

For functions that involve user interaction via the -/= keys on the keyboard: these are the two keys immediately to the left of the backspace key (the “=” key is beside the backspace key, and the “-” key is beside the 0 key). These keys may have different labeling depending on the language of your keyboard.

BLU Functions

Main Menu

- X - Exit to ROM

Exit BLU and return to Lisa ROM where you may restart or go to the Startup From menu or Service Mode.

- R - Read ROM/RAM to serial port
Transfer memory in binary format via x-modem from Lisa to another computer.
- H - Hard Disk Functions
Perform operations involving hard disks.
- F- Floppy Diskette Functions
Perform operations involving floppy disks.
- H - Miscellaneous Functions
Perform various operations, including those used for installing BLU, and changing the serial port settings.
- G - Debug Functions
Perform operations used primary for debugging BLU.
- A - About this software
Display version and credits.

Read ROM/RAM to serial port sub-menu

These functions use x-modem to send the selected binary data.

- M - Return to Main Menu
- 9 - CPU Board ROM
Sends the 16K Bytes of the CPU ROM, useful for emulators that require a ROM image.
- 8 - I/O Board ROM
Sends the 4K Bytes of the I/O Board's Floppy Disk Controller ROM.
- 3/2/1 - Slot 3/2/1 ROM
Sends the 2K Bytes of the ROM area of the interface board in the indicated slot.

The specification for Lisa peripheral slot cards allocates some space to a card specific ROM. Some cards do not use all (or any) of this space for ROM (eg. The Tecmar 4 port serial card has no ROM), so some of the data may be garbage. Similarly, some cards have larger ROMs than will fit in the allocated 2K space (eg. LSAC, QuickBoot™), so this data can be incomplete.
- 0 - Low 1K of RAM
Sends the first 1K Bytes of RAM which contain some run-time information such as RAM size and video page location.

Hard Disk Operations

Hard Disk Sector Checksums

For parallel port hard disks, the Lisa Operating System and MacWorks environments use the last byte of a block's tag bytes to store a checksum such that the xor of all the data and tags is 0.

In some operating systems, the ProFile and Widget drivers will not successfully read a block with a bad checksum (they will crash or return an error). However, if the block is never read, then a bad checksum isn't a problem, so there can be blocks with invalid checksums on a hard disk that operates with no errors.

BLU calculates the xor checksum for each block it writes to a parallel port hard disk, and if it appears invalid, it is counted and reported as a possible problem at the end of the operation.

BLU has a "Set Writes to Update Checksums" function; when writing to a hard disk with this mode selected, the tag byte that is commonly used for the checksum will be updated for each block. However, updating checksums is not

enabled by default in case there are some drivers that use a different checksum algorithm (in which case, BLU would break it by updating it incorrectly).

If you edit a BLU hard disk image (and don't update the checksum tag byte yourself), you may need to update the checksums when writing it back to a hard disk so the edited blocks will read as good. This is not a problem when editing floppy disk images as the sector checksums are handled transparently by the floppy disk controller, they are not in the tags.

The hard disk copy function allows copying from a hard disk to itself so you can check or fix bad checksums on a disk.

※ There does not appear to be a checksum byte in the tags of the Priam hard disk, so checksums are neither checked nor updated when writing to, or reading or copying from, a Priam hard disk.

Hard Disk Functions sub-menu

- M - Return to Main Menu

- I - Identify Hard Disk Device

Shows the type of disk and how many blocks it has. No information regarding what is on the disk is reported.

※ A delay while determining the number of spare blocks allocated on a Priam disk is normal.

- R - Read from Disk to Serial Port

Read Lisa's hard disk and transfer its binary image to your host computer via x-modem. The format of the image is described in Appendix C Hard Disk Image Format.

Note: preliminary versions of BLU use an interim hard disk image format, later versions of BLU may use a different format that supports additional Lisa information which is incompatible with early versions of BLU.

※ Pressing the clear key once now ends the file transfer rather than canceling it, so a portion of a disk image can be transferred.

- W - Write to Disk from Serial Port

Receive a binary image from your host computer via x-modem and write it to the hard disk. The format of the image is described in Appendix C Hard Disk Image Format.

Note: preliminary versions of BLU use an interim hard disk image format, later versions of BLU may use a different format that supports additional Lisa information which is incompatible with early versions of BLU.

※ The number of tag bytes per block in the image must match the number of tag bytes per block on the hard disk.

- C - Copy Disk to Disk

Copy the data from one Lisa hard disk to another Lisa hard disk. If the disks are different sizes, the smaller number of blocks are copied and the rest are ignored.

Note that the boot blocks are different for the different types of hard disk (ProFile, Widget, Priam), so copying to a different type of hard disk may not create a bootable disk. BLU is unique in having a boot block that works on both the Widget and ProFile.

Copying a parallel port hard disk to itself may be useful to update or validate block checksums. See U and N below.

※ Priam hard disks have \$18 tag bytes per block, whereas parallel port hard disks have \$14 tag bytes per block. BLU will copy from one to the other, using a tag byte compression/expansion algorithm that typically preserves the tag bytes when the copy is reversed. However, as the tag bytes are in a different format on the two kinds of hard disks, the data on the destination disk may not work as expected (until it is copied back to the original type of device). See Appendix F Tag Byte Compression/Expansion for further information.

- S - Spin Down Priam Hard Disk

* Commands a Priam Hard Disk to “sequence down” which is similar to “park” as it positions the heads over the landing zone, turns off the spindle motor, and applies the head and spindle lock. Some Priam documents suggest the drive might park automatically when the power is turned off, but it is unclear whether the heads are always moved to the landing zone.

- U - Set Writes to Update Checksums

Once this mode is selected, when writing to a Lisa parallel port hard disk (using W or C), the last byte of a block's tags will be updated so that the xor checksum of all the data and tags is 0. On completion of the operation, the number of checksums altered will be reported.

Note: Other operating systems may not use the last tag byte for a checksum, or may use a different checksum algorithm, so use this option with care. Make a backup image first if you are not sure.

When this mode is selected, it remains selected until you quit BLU or turn it off with “N - Set Writes to Leave Checksums alone”.

- N - Set Writes to Leave Checksums alone - Default

This mode is the BLU default: When writing to a Lisa parallel port hard disk (using W or C), the tags and data will be written verbatim with no update of a block's checksum tag byte. On completion of the operation, the number of checksums that appear to be incorrect will be reported, unless a Priam hard disk is involved.

- E - Exercise/Test Disk

Sequentially Write and Read every block on a hard disk with varying data followed by semi-random off-track seeks to test the disk. This will run perpetually until interrupted by pressing the Clear key. Any data on the disk will be destroyed.

- L - Low Level Format

Low level format an Apple ProFile or Apple Widget parallel port hard disk.

Note: Before BLU, this operation was always performed using an Apple ///. The implementation of the procedure in BLU is essentially the same when low level formatting proceeds normally. However if an error occurs, the Apple /// implementation may provide more information, where BLU will simply report the operation failed.

Low level formatting of an Apple ProFile requires that special “Formatter” Z8 processor hardware is first installed inside the ProFile, which temporarily replaces the Z8 processor used for normal operation.

For Widget formatting, an offset (of the first sector on each track from the index) can be specified. This might be used to move the spare table if there is a problematic defect in exactly the wrong place, but tests have not yet confirmed this is effective. 0 is the default used by the Apple /// Widget formatter.

ProFile emulators such as the X/ProFile™ and IDEfile don't need or use this kind of formatting.

Floppy Disk Operations

BLU is a stand-alone utility that runs on Lisa hardware directly, it does not use any other software/operating system. As a result, disks are not mounted by an OS before reading/writing, so a bad sector doesn't interfere with reading good sectors. However, when a disk is inserted, the floppy disk controller may spend a few moments “clamping” it; during this time the keyboard is unresponsive, please be patient when typing immediately after inserting a disk; look for the flashing question mark to indicated that BLU is expecting keyboard input.

When a bad sector is encountered, BLU prints some Floppy Disk Controller (FDC) information to the serial port, which may be helpful in determining the type of problem. The Miscellaneous Functions menu has a Verbose command which causes the floppy disk controller information to be sent for every sector, good and bad. The floppy disk controller information is described in Appendix D Floppy Disk Controller Data.

Floppy disk images used by BLU are in the format used by Apple DiskCopy 4.2 (DC42), which stores the data and tags from the floppy disk in the data fork of a Macintosh file. The DC42 file format is described briefly in Appendix E DiskCopy 4.2 Floppy Disk Image Format.

For correct x-modem transmission of disk images on a Macintosh, the terminal software should be set to “Straight Binary” (whereby the data fork of a file is sent verbatim, and the resource fork is ignored), not “Text” nor “MacBinary”.

When a DC42 file image is received from Lisa, usually the terminal software will not set the type and creator of the file; that must be done separately for a 3.5" disk image to be used with DiskCopy 4.2 itself on a Macintosh. The creator should be "dCpy" and the type should be "dlmg". Note that the DC42 images of Twiggy disks created by BLU cannot be used on a Macintosh with the actual DiskCopy 4.2 application or disk image mounting software as they have no knowledge of Twiggy disks and their format.

Difficult to read disks

If there is difficulty performing a floppy disk operation, you are presented the options of failing the operation, retrying, skipping the sector, read ignoring-checksum, or read with varying speed. The latter two apply only to the Read operation, and read with varying speed is implemented for Twiggies only.

Failing the operation will tell BLU to stop trying and return to the main menu.

Skipping a sector may be useful for reading or writing (but may render a disk/image unusable if the sector contained critical information).

In the case of skipping reading a problem sector, the missing data in the DC42 file image is replaced with "This sector could not be read!!!" (repeated 16 times to fill 512 bytes), and the missing 12 tag bytes are replaced with "Read Failure".

Portions of some Macintosh disks (those used on a Mac II, for instance) cannot be read by the stock Lisa hardware. Almost all of these disks can be read when using MacWorks Plus II, as the PFG hardware included with MacWorks Plus II provides the means to overcome this limitation of the stock Lisa Floppy Disk Controller.

Currently, BLU does not utilize the PFG if it is installed, so a 3.5" Macintosh disk that cannot be read by BLU may work fine with MacWorks Plus II. This problem does not occur with Lisa disks.

Read ignoring checksum is likely to produce garbage data, but might be better than nothing if there is a way for the user to scavenge the data manually afterwards.

Read Varying Speed

Read Varying Speed provides fine adjustment of the rotation speed of the Twiggy. By observing the error counters (sent out the serial port), the effects of changing speed can be monitored. The speeds that gives the highest DCS counts are the ones most likely to be successful.

Use the + and - keys on the keypad to change the speed adjustment. A speed adjustment of 0 corresponds to the default speed for the track.

Attempts with varying speed are made without recalibrations. To recalibrate while using read varying speed, use the * key on the keypad; this may be helpful once the best speed is established.

Use the Backspace key to stop making further attempts to read the problem sector... you can then try read ignoring checksum to attempt again using the last speed set, or skip, etc.

256 attempts to read the sector will be made when Read Varying Speed is selected. Subsequently you are presented with the options of skipping, repeating Read Varying Speed for another 256 attempts, etc.

The Clear key cancels the operation entirely.

The speed adjustment is reset to 0 for each sector.

Twiggy Problems

If a Twiggy disk cannot be read at all, it may not have clamped properly; eject it and try again. This is sometimes indicated (in the status information sent to the serial port when a floppy disk error occurs) by ASB non-zero and the other counters zero, which indicates that the FDC did not find any Address fields on the track.

Experience shows that difficult to read Twiggy disk sectors can sometimes be read if one chooses Retry after waiting for the disk to stop spinning, or by retrying a few times before the disk stops.

A Twiggy disk can sometimes be read easily by one drive and not by another, so trying the other drive, or another Lisa 1, is useful when a problem persists.

For a particular disk, it can be that some sectors cannot be read on one drive, and different sectors cannot be read on another drive.... in this case, a complete image might be achieved by manually combining (using a hex

editor) the images obtained by reading the disk on both drives, skipping the problem sectors on each drive as they occur.

Using a hex editor, one can copy blocks of data from one image and paste into another image to replace missing data (assuming both images should be the same except for the bad blocks). Remember to copy the tag data as well as the sector data; in a DC42 file, all the tag data is at the end of the file, following the data from all of the sectors; see Appendix E DiskCopy 4.2 Floppy Disk Image Format.

Original/vintage Twiggy media ca. 1983 tend to have a coating of oxide dust, which quickly builds-up on the drive heads and impedes correct operation. It is important to frequently inspect (and clean when appropriate) both heads of a Twiggy drive when using old media. Cleaning the oxide dust off a disk before inserting it can help reduce head cleaning.

A head-cleaning disk is not always sufficiently effective, and the drive may need to be partially disassembled for cleaning. Don't assume that the rear head is clean just because the front one is.

When reading a Twiggy, the rear/upper head is used first, on the outer track, at the lowest speed. When formatting a Twiggy, the rear/upper head is used first, on the inner track, at the highest speed. Twiggies are configured such that the entire upper side is used before the lower side (other drives use both sides for each track).

When attempting to read an old Twiggy disk that may be particularly rare or valuable, always clean the heads first, as your chances of successfully reading an old disk may decline with every attempt.

The media in a vintage disk jacket may have a high resistance to rotation which can cause slipping or stalling, sometimes the disk can be seen to be barely turning. This might be alleviated by removing or otherwise disabling the foam pads that press on the top surface of the jacket of the disk when it is clamped, or removing the media from the jacket and putting it in a more modern jacket (with appropriate cut-outs).

Floppy Diskette Functions sub-menu

- M - Return to Main Menu
- E - Eject
Ejects a floppy disk.
- F - Format
Formats a floppy disk, destroying any data currently on the disk. If the drive is a double sided 3.5", the choice of a single side or double side format is offered.
On a Macintosh XL (aka Lisa 2/10) with an 800k drive, the disk is automatically zeroed (see Z below) before formatting (otherwise the format process would likely fail if the disk had been used before on a Macintosh or Lisa).
- C - Check Disk - Read Every Sector
Attempts to read every sector on the floppy disk to see if the disk is obviously bad.
- R - Read from Disk into Memory
Reads the data from a floppy disk and saves it in memory in DC42 format.
- W - Write from Memory Onto Disk
Writes the data from a DC42 file in memory to a floppy disk, replacing any existing data on the floppy disk. A DC42 file is placed in memory via the Read or Load command. A disk must be formatted (by BLU or an operating system) before writing can occur.
- V - Verify - Compare Disk and Memory
Reads the floppy disk and compares it to a DC42 file in memory. Differences are transmitted to the serial port. The DC42 file to compare is first placed in memory via the Read or Load command.
- L - Load DC42 File into Memory Via Serial Port
Receives via x-modem a DC42 file and stores it in memory. Normally this is in preparation to write the DC42 image to a floppy disk.
- D - Dump Memory to DC42 File via Serial port

Sends via x-modem the DC42 file in memory. Normally the DC42 file would be created in memory by reading a floppy disk with the Read command.

- I - Show information about DC42 file in memory

Shows the size and type of DC42 file that is currently in memory, and whether the checksums are correct.

- Z - Zero floppy - Erase without format

Writes null bytes over every track on the floppy disk, un-formatting it.

- N - Clean floppy - Shuffle heads across disk

For use with a head cleaning disk or other cleaning tool. Shuffles the heads forwards and backwards incrementally from one edge of the disk to the other. Repeats until cancelled with the Clear key.

Miscellaneous Functions sub-menu

- M - Return to Main Menu

- S - Read Lisa Serial Number

Decodes the manufacturing information particular to the Lisa and displays it as well as sending it out the serial port.

The manufacturing information is stored in the video state machine PROM on the CPU board. Lisa 2/Macintosh XL machines with the square pixel / 3A ROM modification do not have this information.

- I - Create BLU DC42 image to write to floppy

Creates a DC42 image, in memory, of a bootable BLU floppy disk (containing the version of BLU currently running). Once the DC42 image is in memory, it can then be used to create a bootable BLU floppy disk using the floppy disk Write command, or sent to another computer via the floppy disk Dump command. Although the DC42 image is of a 400K disk, the same image can be written to a Twiggy disk to make a bootable Twiggy BLU disk.

- H - Install BLU onto hard disk

Installs the currently running version of BLU onto a parallel port hard disk which can be used to boot Lisa. When used with an otherwise spare hard disk image (eg. on an X/ProFile™ CF card) this provides an easy way to move BLU to a Lisa 1 (perhaps to make a bootable Twiggy disk with the Create BLU self image command). Use with care as this replaces/destroys any existing data on the hard disk.

BLU installed on an X/ProFile™ (or other parallel port hard disk) will boot on any Lisa, from any parallel port.

- C - Check/Validate DC42 file

Accepts a file via x-modem and validates the data and tag checksums. This function does not keep the DC42 file in memory, so the file can be larger than will fit in memory.

- T - Terse reporting - Default

This mode limits the Floppy Disk Controller status information sent to the serial port to errors only.

- V - Verbose reporting

This mode expands the Floppy Disk Controller status information sent to the serial port to include every sector.

- P - Configure serial port

Enter the serial port options sub-menu.

Configure Serial Port sub-menu

All configurations of Lisa's Serial Communications Controller (SCC) are 8 bits, no parity, 1 stop bit, with hardware handshaking enabled. A cable with hardware handshaking connections may be unnecessary for most functions and baud rates.

Note that Lisa's CPU is a limiting factor so that x-modem transmission at 230400 baud may be no faster than at 115200, and x-modem reception may fail at 230400 baud since Lisa's CPU cannot keep up. For this version, 115200 baud is recommended for maximum speed.

- M - Return to Main Menu
- A - Set SCC B to 9600
Changes the baud rate to 9600, then sends CONNECT OK
- B - Set SCC B to 19200
Changes the baud rate to 19200, then sends CONNECT OK
- C - Set SCC B to 38400
Changes the baud rate to 38400, then sends CONNECT OK
- D - Reset SCC B to 57600 Baud - Default
Resets the SCC, configures Serial B for 57600 baud, 8 bits, no parity, then sends CONNECT OK
- E - set SCC B to 115200
Changes the baud rate to 115200, then sends CONNECT OK
- F - Set SCC B to 230400
Changes the baud rate to 230400, then sends CONNECT OK
- T - Terminal Test
The Terminal Test provides a way to test the serial connection/cable. Characters typed are displayed when sent to the other computer, and characters received are displayed. Characters are limited to uppercase letters and digits due to the limited font built-in to Lisa's ROM. The Terminal Test transmits regardless of the hardware handshake signal received, but the TX/T status indicator is active. The Receive handshake signal can be toggled using the keypad +/- keys.

Debug Functions sub-menu

The debug functions are primarily for debugging BLU itself. These functions may cause strange behaviour, so do not use these functions unless directed.

- M - Return to Main Menu
- B - Buffer Info
Displays the location of the buffer that will be used for Download Code and Execute. The buffer can be accessed from Lisa's service mode (by choosing X from the main menu to leave BLU, then typing Apple-S to enter service mode).
- D - Download Code and Execute
Accepts binary data via x-modem for execution.
After downloading, the user is offered the choice of starting execution at the first byte transmitted (offset \$0), at the start of the first sector if a DC42 file was sent (offset \$54), or at the start of the second sector if a DC42 file was sent (offset \$254).
Execution at any offset can be achieved from Lisa's service mode (use Buffer Info in the Debug Functions menu and note the start of the buffer, choose X from the main menu to leave BLU, type Apple-S to enter service mode, and use the call program command to start execution wherever desired in the buffer).
If you have the DC42 floppy disk image for a new version of BLU, you can use Download Code and Execute to update your BLU boot disk... send the new BLU DC42 disk image as the download code, and once the transfer is complete, type S to start execution at the start of the second sector, which will run the new version of BLU. You can then use the new version's self-install commands to update your BLU boot disk to the new version.
- R - Reset Floppy Controller

Provides a way to reset the Floppy Disk Controller without turning off the computer.

- 0 - Do All Floppy Ops as usual (may not be present)

Resets the skip mode flags set by the following functions.

- 1 - Skip Floppy Reads (may not be present)

When this mode is set, floppy disk operations proceed as usual, except that sectors are not actually read from the floppy disk. The sector buffer is filled with dummy data instead.

- 2 - Skip Floppy Writes (may not be present)

When this mode is set, floppy disk operations proceed as usual, except that sectors are not actually written to disk.

- U - Run Misc Debug Code (may not be present)

Executes whatever extra debugging code was included when that particular version of BLU was built.

- I - Interleave timer (may not be present)

Outputs a signal on Serial A - DTR (pin 20) which represents the access delay time for reads between two blocks on a hard disk. The +/- and up&down keys on the keypad adjust which blocks are timed. The -/= keys on the main keyboard control whether the data is transferred to Lisa or whether the read is performed by the hard disk alone; data is always transferred if it is a Priam disk.

* Typing S or D enables hexadecimal input for the start sector or delta values.

- Tilde - Dump FDC Vars (may not be present)

After Verbose mode is selected from the Miscellaneous Functions, the tilde/back-quote (~`) key transmits the low memory variables of the Floppy Disk Controller to the serial port. This function is asynchronous... the tilde key can be pressed at any time, it does not have to be selected from the Debug Functions menu.

- (etc.)

Various other functions may be included in Debug Functions depending on the build version.

About this software

Displays version information and credits in a format similar to that below; if you agree to use BLU at your own risk, type the Y key to continue.

Basic Lisa Utility Vx.xx

Build Date - yyyy/mm/dd at hh.mm.ss

(Credits)

This software is offered for use without warranty of any kind

Do you agree to use this software at your own risk... Y/N?

APPENDIX A Bootstrapping

If you have a Lisa but no way to make a disk for it, you will need to bootstrap by loading from the serial port. This is achieved by typing in a small program such as the Simple Serial Loader described below.

Lisa Simple Serial Loader

Purpose: Loads something from serial port B at 57600,N,8,1

Usage:

Connect another computer to Lisa Serial B...

Lisa transmits on pin 2, receives on Pin 3, and pin 7 is ground.

No other pins are needed/used by this routine, but if making a cable, you may want to connect the handshake pins for use by whatever utility you download.

Pin 6 is an input (to throttle Lisa transmit), pin 20 is an output (to indicate Lisa ready-to-receive).

Since this utility runs at 57600 baud, you should use a shielded or short cable.

Input the Loader:

Turn on Lisa, press the space bar during the self test

When you get to the "startup from" menu, enter service mode by typing Apple-2 Apple-S

In service mode, input the loader using these 3 lines of keystrokes:

```
2900 4DFA 0020 3456 227C 00FC D201 3C09 129E <return>
2910 66F4 0811 0000 67FA 14E9 0004 51CE FFF4 <return>
2920 4E75 09C0 0BD0 0444 0E01 03C1 05E8 0C00 <return>
```

Note: the first "2" selects the "set memory" function, so it won't appear on the screen, the "9x0" specifies the address, and the subsequent eight 4-letter words are the data/code include the spaces between the hexadecimal words, lowercase letters are ok

Display the data entered above (so you can confirm it was entered correctly) using

```
1900 30 <return>
```

Run the loader by calling the program at its address using

```
3900 <return>
```

Once the loader is running (there is no indication it is, but if you did the above correctly, it will be), start sending from your other computer (upload your data file as plain text/binary, this is not x-modem). If you are bootstrapping BLU, send the BLU DC42 floppy disk image.

Once enough data is received (53762 bytes is the amount pre-set in this version of the loader), the loader will return control to service mode.

If the transfer doesn't complete automatically, you can either:

- a. press the interrupt button and type Apple-S to re-enter Service Mode
(the interrupt button is beside serial B on a Lisa 2/10, the stock Lisa 1 and Lisa 2 don't have an interrupt button)
- b. upload anything of sufficient size to satisfy the byte counter (stop the upload once Lisa returns to Service Mode)

Note: Don't press reset, as Lisa's self-test will over-write the memory and you will need to start over with typing in the loader.

When the transfer is completed, you may view the data received in the buffer starting at memory address \$9C0 with

```
19C0 10 <return>
```

If you transferred a DC42 Lisa floppy disk image, the first bytes should be 162D 6E6F.

- If the first transferred bytes are 5374 7566, you have transferred a stuffit archive instead of its contents. You will need to unstuff the archive, re-run the loader, and send the unstuffed file.
- If the first transferred bytes are 504B, you have transferred a zip archive instead of its contents. You will need to unzip the archive, re-run the loader, and send the unzipped file.
- If the first transferred bytes are 2854, you have transferred a binhex HQX archive instead of its contents. You will need to un-binhex the archive, re-run the loader, and send the extracted file.

If the buffer doesn't look right (eg. sometimes there is a garbage byte at the beginning if you connected the cable or launched the terminal software after starting the loader), then you can re-run the loader to try again.

If all is well so far, examine BLU's code entry point with

```
1C14 10 <return>
```

The first bytes are likely to be (correctly):

```
6012 5354 0D01 0AFF
```

If the bytes are similar but not identical to these, it is possible your terminal program is intentionally altering the data sent. If it has an option to send line feeds after carriage returns, turn it off. If you have transfer options of binary and text, choose binary (but not MacBinary).

If you have problems transferring the data correctly, you can alter the baud rate with this addition, then re-run the loader:

```
292E 0Cxx 0000 <return>
```

where xx is the time constant for the desired baud rate as follows:

```
57600 38400 19200 9600 4800 2400 1200
00    01    04    0A    16    2E    5E
```

Once satisfied with the data loaded, run the utility (or whatever you loaded with the loader) from service mode according to the utility's instructions. If the utility is BLU, type:

```
3C14 <return>
```

(In the case of a utility such as BLU whose code starts at the second sector in a DiskCopy 4.2 disk image, the code is run from

the start of the buffer plus an offset for the DC42 header plus one sector: $\$9C0 + \$54 + \$200 = \$C14$)

If all went well, BLU will be running: make a BLU floppy disk so you won't need to go through the bootstrap procedure next time.

If Lisa beeps three times and shows an error code, return to service mode and double-check the data at the code entry point. It is likely that the loader is still in memory and you may run it again if needed.

Source Code

```
*****
* Lisa Simple Serial Loader
*****

LISASCC      EQU          $FCD201          ; Zilog 8530 Serial Controller

MAIN

* yes, this code is slightly bizarre. Since it has to be typed in,
* clarity and features have been compromised to minimize bytes
*
* in particular, these two items are re-used in an odd manner:
* a) the buffer pointer is derived from the first two bytes of SCC configuration data
* b) the byte counter is derived from the low word of the SCC address

InitSCC      EQU          *
             LEA           InitTable,A6      ; Point to InitTable containing SCC configuration data
             MOVEA.W       (A6),A2          ; use the first word of SCC data as the buffer pointer:

$0009C0
@in1         EQU          *
             MOVEA.L       #LISASCC,A1      ; point to SCC control register (in loop for scc delay,
                                           ; MOVEA is slower than LEA)
             MOVE.W        A1,D6            ; initialize receive byte counter (in loop for SCC delay)
             MOVE.B        (A6)+,(A1)       ; alternately select an SCC configuration register then
write
                                           ; data to the selected SCC register
             BNE.S         @in1             ; loop until zero terminator

ReceiveLoop  EQU          *                  ; on entry, D6 = D201, so 53762 bytes will be accepted
(more are ignored)
             BTST.B        #0,(A1)          ; RX register full?
             BEQ.S         ReceiveLoop
             MOVE.B        4(A1),(A2)+      ; copy byte from SCC to RAM [change A2 to A0 to use
                                           ; preloaded address from $1E0.L]
             DBRA          D6,ReceiveLoop   ; [change to D7 to use preloaded count from $1DE.W]

AnRTS        RTS

InitTable    EQU          *
```

```

*
* assumes Lisa has performed self test, which resets the SCC, but leaves a few useful settings, such
as
* baud rate divisor = 0,0, clock mode x16, 1 stop bit, no parity
*
        DC.B  $9,$C0                ; SCC reset to clear the receive queue NB: This pair is
                                        ; also the buffer address!!
        DC.B  $B,$D0                ; XTAL, BR gen is clock source
        DC.B  $4,$44                ; x16, 1 stop bit, no parity
        DC.B  $E,$01                ; BR gen from XTAL, enable
        DC.B  $3,$C1                ; 8 bits, Rx enable
        DC.B  $5,$E8                ; DTR, /RTS, 8 bits, Tx enable
        DC.B  $C,$00                ; TC: 0 = 57600 baud, #10 = 9600 baud
        DC.B  0                    ; extra terminator in case TC is changed

* for baud rates slower than 450, would need to configure register D (the high byte of a divisor >
255)
*        DC.B  $D,$00                ; TC high byte = 0 (for 450 baud and up)
*

* TimeConstant = (clockFreq / (baud rate * 2 * clock mode)) - 2
* clockFreq is 3.6864 MHz, clock mode is 16
* Baud Rate   TC
* 57600       00 00
* 38400       00 01
* 19200       00 04
* 9600        00 0A
* 4800        00 16
* 1200        00 5E
* 300         01 7E

*****
*
* Possible modifications:
*
* When called from service mode...
* on entry:
*         A6 is our entry point (called address)
*         D0-D7/A0-A5 are pre-loaded via MOVEM.L from $1C0
* on exit  D0-D7/A0-A5 are written back to $1C0
*
* in particular, in service mode, one could preload as follows:
*         D7.W - $1DE - preload with (byte count minus 1).W
*         A0.L - $1E0 - preload with load address
*
* By changing ReceiveLoop to use "DBRA D7" and/or "MOVE.B 4(A1),(A0)+", then
* the data could be loaded at any location, and
* any number of bytes up to 64k could be specified
*
* After returning to service mode, the preload address will be updated, and D7 will be FFFF,
* so a multipart utility could be received with successive calls
*

```

APPENDIX B Serial Cables

Lisa transmits on pin 2, receives on Pin 3, and pin 7 is ground.

For handshaking, pin 6 is an input (to throttle Lisa transmit), and pin 20 is an output (to indicate Lisa ready-to-receive).

If making a cable, you can wire just Tx, Rx, and Gnd, but that may limit the x-modem transfer speed. If you are able, also wire the handshake lines. If the cable has a shield wire, you can connect it to the shell of the connector(s) or to Gnd.

To make a cable to go from Lisa to a Mac mini-DIN, wire as follows:

Lisa	Signals	Mac
DB25M		MiniDIN8M
2	Tx > Rx-	5

3	Rx < Tx-	3
7	Gnd & Rx+	4 & 8
6	DSR < Hsko	1
20	DTR > Hski	2

(Lisa pin 7 goes to Mac pin 4 and Mac pin 8)

In case you were wondering... Pin 19 on the Lisa is not used when the serial port is configured for RS-232 mode, it is only active when it switches to LocalTalk mode. So, you can leave pin 19 disconnected.

To make a cable to go from Lisa to a PC DE9 com port, wire as follows:

Lisa	Signals	PC
DB25M		DE9F
2	Tx > Rx	2
3	Rx < Tx	3
7	Gnd	5
6	DSR < RTS	7
20	DTR > CTS	8

APPENDIX C Hard Disk Image Format

Note: this and earlier versions of BLU use a limited hard disk image format; an extended format that supports additional Lisa information is under consideration for later versions of BLU. An extended format will be incompatible with early versions of BLU.

The limited format BLU hard disk image is composed of records of \$200 bytes of Data followed by the corresponding block's Tag bytes (\$14 bytes for parallel port hard disks, \$18 for Priam hard disks).

The data in the first record of the image identifies the type and size of disk (ie. the information corresponding to sector \$FFFFFF of a ProFile/Widget), and the total number of bytes per block (data+tags, typically \$214 bytes, \$218 for Priam disks). The tags of the first record can be used to identify the source of the image (eg. "Lisa HD Img BLUV0.07"). The second record of the image (typically at offset \$214, or \$218 for Priam disks) corresponds to logical block \$000000 of the hard disk, the third record of the image (typically at offset \$428 or \$430) corresponds to logical block \$000001 of the hard disk, and so on, up to the number of blocks indicated in the first record of the image. (The logical blocks correspond to consecutive blocks at the operating system level, even though the physical blocks on a ProFile are interleaved 5:1.)

Multi-byte values are MSB first.

Note: This image format is "unnatural" in that BLU is aware of where the tags are in the byte stream (they precede the data for ProFiles and Priam hard disks, and follow the data for Widgets) and of the interleaving performed by ProFile drivers. Other utilities treat the data in an "as-is" manner, which places the tags before the data for ProFiles (after the data for Widgets), and do not de-interleave the blocks of a ProFile.

✱ **Note:** Versions of BLU prior to v0.90 would send an extra \$200 bytes at the end of the hard disk image when reading from the hard disk to the serial port. This extra block should be ignored.

Offset	Data	Size				
\$00	Name of Device	\$D bytes	PROFILE	PROFILE 10	WIDGET-10	PRIAMDTATOWER
\$0D	Type of Device	3 bytes	\$000000	\$000000	\$000100	\$00FF00
\$12	Blocks in Device*	3 bytes	\$002600	\$004C00	\$004C00	\$022C7C
\$15	Bytes per Block	2 bytes	\$0214	\$0214	\$0214	\$0218

*Note: A device named "PROFILE" may have an arbitrary number of blocks, as ProFile emulators (X/ProFile™, IDEfile, etc.) implement a variety of user selectable sizes.

APPENDIX D Floppy Disk Controller Data

When a floppy disk read/write problem occurs, status information is sent to the serial port as follows:

Err Trk Sct Sid Spd Try Rcl DSB DEB DCS ASB AEB ScX TrX ACS

Err = Error code returned by the Floppy Disk Controller

Trk = Track requested when the problem occurred

Sct = Sector requested when the problem occurred

Sid = Side requested when the problem occurred (0 or 1. Twiggy 0 = upper; 3.5" 0 = lower)

Spd = Speed adjustment byte when the problem occurred (00 = normal speed)

Try = Tries remaining. Usually an operation starts with \$64 tries and decrements from there. Tries are reset (eg. to \$64) upon recalibration. Some errors will terminate the operation before the tries remaining reaches 0 or before it starts counting.

Rcl = Recalibrations remaining. Usually an operation starts with 1 recalibration remaining. For twiggy's, 0 indicates 1 remaining.

DSB = Data start bitslip error count. Incremented when the 3 data field starting bytes (D5 AA AD) are not found within \$20 bytes after successfully reading the address field.

DEB = Data end bitslip error count. Incremented when the 2 data field ending bytes (DE AA) are not found at the end of the data field.

DCS = Data checksum error count. Incremented when the sector's data field appeared to be successfully read but the checksum was incorrect.

ASB = Address start bitslip error count. Incremented when the 3 address field starting bytes (D5 AA 96) are not found within \$800 tries. Normally this is a fatal error and retries are not performed since it appears the disk is not formatted.

AEB = Address end bitslip error count. Incremented when the 2 address field ending bytes (DE AA) are not found at the end of an address field.

ScX = Wrong sector count. The number of address fields read before the finding the correct one. It is normal for this to be non-zero as it counts the sectors that pass while waiting for the one requested.

TrX = Track error count. Incremented if the track in an address field is not the correct one. Normally this is a fatal error and retry will not occur unless recalibration is performed. This error may be due to reading an address field incorrectly rather than a seek error.

ACS = Address checksum error count. Incremented when the checksum of an address field appears to be incorrect.

Errors are encountered in the following order: ASB ACS AEB TrX ScX DSB DEB DCS

Once any counter is incremented, the next try (if any) begins again at the Address Start Bitslip stage.

APPENDIX E DiskCopy 4.2 Floppy Disk Image Format

There are documents available that describe the DiskCopy 4.2 format in more detail than that below; this is a general outline of a DC42 file and the values used by BLU:

	Offset	Size & Contents
diskName padded	(\$00)	\$40 Bytes; A Pascal string (length byte followed by \$3F ASCII characters, as necessary) containing the name of the disk. For a Lisa disk, this string is typically "-not a Macintosh disk-". BLU includes its version number in the padding after the string, and "Twiggy Image" if appropriate.
dataSize	(\$40)	Long Word (4 bytes, MSB first); The total number of bytes (not sectors) in data fields in the image. A data field is \$200 bytes per sector. For an image of an entire disk, the dataSize is \$200 times the number of sectors on the disk, ie.: Twiggy:\$000D4C00, 400k:\$00064000, 800k:\$000C8000
tagSize	(\$44)	Long Word (4 bytes, MSB first); The total number of bytes of tag fields in the image. Tags are an extra \$C bytes of "scavenger" information per sector on Twiggy, 400K and 800K Lisa and Macintosh disks. If there are no tag bytes in the image, this field will be zero. For an image of an entire disk, the tagSize is \$C times the number of sectors on the disk, ie. Twiggy:\$00004FC8, 400k:\$00002580, 800k:\$00004B00
dataChecksum image.	(\$48)	Long Word (4 bytes, MSB first); Checksum of all the data fields in the disk
tagChecksum image.	(\$4C)	Long Word (4 bytes, MSB first); Checksum of all the tag fields in the disk
diskType	(\$50)	One Byte; Indicates the type of disk: 400K:\$00, 800K:\$01, Twiggy:\$54
formatByte	(\$51)	One Byte; Indicates the format of the disk: 400K:\$02, 800K:\$22, Twiggy:\$01
fileFormat	(\$52)	Two Bytes; Always \$0100
userData	(\$54)	dataSize Bytes; The data fields for the disk. These are in order from sector zero through the end of the disk. Data fields are \$200 bytes each.
tagData	(\$54+dataSize)	tagSize Bytes; The tag fields for the disk. These are in order from sector zero through the end of the disk. Tag fields are \$C bytes each. Note that all of the tag fields follow all of the data fields in the image. For an image of an entire disk, the offset of the first tag field is thus: Twiggy:\$000D4C54, 400k:\$00064054, 800k:\$000C8054

The value \$1A is used as a trailing pad byte to fill the last x-modem block; this is not part of the DiskCopy image, but it does no harm to leave the padding at the end of the file.

The checksum algorithm is to add a word (two bytes, MSB first, not sign extended) to a 32 bit accumulator, then rotate the 32 bits right 1 bit. The tag bytes of sector 0 are skipped when calculating the tagChecksum.

For a DC42 image file to be recognized by the DiskCopy application running on a Macintosh, the file creator should be "dCpy" and the file type should be "dImg".

The DiskCopy application will not recognize a Twiggy image as valid as it has no knowledge of Twiggy disks. The values used for a Twiggy image have been invented for BLU.

APPENDIX F Tag Byte Compression/Expansion

✱ The Priam DataTower hard disk uses \$18 tag bytes per block, while parallel port hard disks use \$14 tag bytes per block.

When copying from a Priam hard disk to a parallel port hard disk, the tag bytes are compressed as follows:

\$18 Tag Bytes: aaaaaaaaa bbbbbbbb jjccccc kddddd mmeeee nnxyyz

\$14 Tag Bytes: aaaaaaaaa bbbbbbbb xxxccc yydddd zzzeee

To expand the tag bytes when copying from a parallel port hard disk to a Priam, the values in the tag bytes xx,yy,zz are used for the last 3 tag bytes, and then the "missing" values (jj,kk,mm,nn) are set to 0 unless the byte immediately to the right is \$ff, in which case the missing byte is also set to \$ff (eg. nn is 0 or \$ff depending on whether xx = \$ff).

Theory: The tag bytes on the Priam appear to be used only by the Lisa Office System. When using LOS on the Priam it appears that the last four quad-byte long-words of the tag bytes are block numbers, with a special value of \$fffffff indicating n/a. Since the Priam has \$00022C7C blocks, the high byte of these block numbers is zero unless the value is indicating n/a, so this compression/expansion process potentially results in the preservation of all \$18 tag bytes.

Unix operating environments do not use the tag bytes, but copying between devices does not alter the boot blocks, so the copy of a unix disk to a different kind of hard disk may be mountable and usable but may not be bootable.

Copyright 2013, Sigma Seven Systems Ltd. All rights reserved.
X/ProFile and QuickBoot are trademarks of Sigma Seven Systems Ltd.
Other trademarks are of their respective owners.