quikLoader

PRELIMINARY

quikLoader Reference Manual

by Jim Sather

12/12/83

Revision 1 ... 1/7/84

NOTE:

This is a PRELIMINARY manual, in draft form. Although it is not complete, we provide this so that you may start getting use out of your quikLoader. We will send you a more complete manual as soon as it is available.

THIS MANUAL MENTIONS THAT THE QUICKLOADER CONTAINS THREE EPROMS. WE HAVE UPGRADED ONE OF THEM TO A 27128. THUS, ONLY TWO EPROMS ARE NECESSARY.

SOUTHERN CALIFORNIA RESEARCH GROUP
P.O. Box 2231, Goleta, CA 93118 • (805) 685-1931

APPLE COMPUTER, INC. makes no warranties, either express or implied, regarding the enclosed computer software package, its merchantability or its fitness for any particular purpose. The exclusion of implied warranties is not permitted by some states. The above exclusion may not apply to you. This warranty provides you with specific legal rights. There may be other rights that you may have which vary from state to state.

DOS, INTEGER BASIC, FID, and COPYA, are copyrighted programs of APPLE COMPUTER, INC., licensed to Southern California Research Group to distribute for use only in combination with quikLoader.

Table of Contents

Figures

Tables

What is a quikLoader?

A quikLoader is a firmware peripheral card designed for use in the Apple][, Apple][Plus, and Apple //e computers. It will hold up to 256K (262144) bytes of data and programs. Because the data and programs are stored in EPROM or ROM, they are instantly available to the computer operator at any time.

The quikLoader provides the user an unprecidented level of speed and convenience in accessing the programs he uses most often. Many valuable programs are rarely used by their owners because of the inconvenience of locating the correct diskette and loading the programs. Whatever the value of a program to you, it will become more valuable if you install it on the quikLoader.

The quikLoader is supplied with several programs resident: QLOS (the QuikLoader Operating System), DOS 3.3, Integer BASIC, FID, COPYA, and a QLOS help screen generator. DOS, Integer, FID, and COPYA are programs from the Apple DOS master disk and are distributed under license from Apple Computer, Inc. QLOS is a program written for the quikLoader which supports a variety of operational features including immediate availability of DOS and Integer at power-up, loading RAM and running of Applesoft, Integer, or Binary files, execution of quikLoader resident primary routines, Katalogging (as opposed to disk catalogging) of QLOS files, and execution of a number of reset functions selected via the keyboard.

Installation

The QuikLoader can be installed in any slot of an Apple II except slot 0. Installation in an Apple //e is very easy, but installation in an Apple][or Apple][Plus requires some extra steps. Also, if you install more than one QuikLoader in any Apple II, you should read "QuikLoader Chains: DMA IN / DMA OUT Jumpers". Also read the same section if you wish to install a QuikLoader in the same Apple as an Applesoft or Integer 12K firmware card.

Apple //e

Install the QuikLoader in any slot of the Apple //e. It will operate in slot 3, even if an auxiliary memory card is installed in the auxiliary slot. The USR 1 jumper on the QuikLoader must not be soldered when the QuikLoader is installed in an Apple //e. The QuikLoader is manufactured with the USR 1 jumper desoldered so you don't have to do anything to the USR 1 jumper when installing a new QuikLoader in an Apple //e.

Apple][and Apple][Plus

The Quick Loader can be installed in any slot except slot 0 in an Apple][or][Plus. 48K of motherboard RAM is required. A 16K RAM card in slot 0 is not absolutely required, but if you have a 16K RAM card, the QuikLoader will automatically transfer Integer BASIC and the motherboard monitor to the RAM card at power up and other times you select it. However, to operate with the QuikLoader, the 16K RAM card must be modified so that it is disabled by system reset, just as the Apple //e built in 16K RAM card is.

A second requirement in the Apple][or Apple][Plus installation is that the USER 1 jumper on the motherboard and the USR 1 jumper on the QuikLoader must be made. This will enable the QuikLoader to respond to $C100-$C7FF addressing as is required in some 27128/27256 data formats. The RAM card modification and USER 1 jumper installation require some minor soldering. If you have no soldering experience, you should take your Apple][or][Plus, QuikLoader, "QuikLoader Reference Manual", and RAM card to an Apple dealer who is willing to perform the work. Apple][/][Plus related installation procedures follow.

QuikLoader USR 1 Jumper

The USR 1 jumper is located near the edge connector of the QuikLoader. Bridge the gap on the USR 1 jumper with a small amount of solder.

** Figures 1 and 2 making the motherboard USER 1 jumper

Motherboard USER 1 Jumper.

This is an unmarked jumper on the motherboard of Apple][s and Apple][Pluses. It is located to the right/front of slot 7 as shown in Figure 1 in older motherboards and Figure 2 in RFI Revision motherboards. The pair of holes can be mated by a short jumper wire if you remove the motherbaord and work from the bottom, but Figures 1 and 2 show alternate solder holes which you can jumper from the top without motherboard removal.

To install the jumper, remove power from your Apple and disconnect the power cord. See which figure matches your motherboard, and desolder the indicated solder holes. Cut and strip a short insulated wire jumper and solder between the two solder holes. Reconnect the power cord.

Modifying the 16K RAM card.

NOTE OF CAUTION:
Modifying your RAM card may result in voidance of the RAM card warranty if it is still in effect. It may also result in higher out of warranty repair costs should your RAM card fail. The decision to make such modification rests soley with the owner of the RAM card and he or she is soley responsible for any negative consequences of the modification. Neither the Southern California Research Group nor its associated engineers accept any responsibility for your decision to make this modification.

** Figure 3 RAM card modification

FIGURE 3 CAN BE FOUND IN THE BACK

Procedure:

Perform the following steps Regardless of RAM card manufacturer. Refer to Figure 3.

1. Locate the 74LS175 IC on the RAM card. Trace the conductor from pin 1 of the LS175 to the junction of a 10 microfarad capacitor and a resistor.
2. Disconnect one lead of the capacitor and one lead of the resistor from the PC board. Insulate these open leads with tape or heat shrink.
3. Connect an insulated wire between pin 1 of the LS175 and pin 31 of the edge connector. Solder the wire to the top of the edge connector so the solder does not make contact with pin 31 of motherboard slot when the RAM card is installed.

Perform the following step if you have an R.H. Electronics Super RAM II card.

1. Set switch 1-2 on the DIP switch to down. This disables the ROM or PROM on the 16K card.

**Figure 4 very small figure shows R.H. elex RAM card switch

FIGURE 4 IS IN THE BACK

Perform the following steps if you have an Apple Language System card.

1. Remove the ROM or PROM from its socket.
2. Remove the 74LS20 from its socket and bend pin 6 so it will not make contact in its socket. Reinstall the 74LS20.
3. Solder a jumper wire between pins 4 and 5 of the 74LS09.
4. If the ROM that was removed from the RAM card contains the system monitor you desire, replace the motherboard F8 ROM with this ROM.

ROM Select Jumpers

Before installing your QuikLoader in your Apple, take a moment to examine it visually. There are eight ROM/EPROM sockets, each of which can hold a 2716 (2K byte), 2732 (4K), 2764 (8K), 27128 (16K), or 27256 (16K) EPROM or equivalent masked ROM. There are two solder pad jumpers associated with each socket, a split circle pad on the component side and a bow tie pad on the solder side (see Figure 5). These pads must be changed for a socket if you elect to install a 24 pin chip (2716 or 2732) in it. When configuring these jumpers for a socket, there is always electrical contact across one jumper or the other. Here are the two possible configurations:

BOW TIE	SPLIT CIRCLE	IC TYPES
MADE	DESOLDERED	2764, 27128, 27256
CUT	SOLDERED	2716, 2732, 2764

The quik loader is delivered with all sockets configured for 2764/27128/27256. Please note that 2764 will work with either jumper configuration. Also notice that when 24 pin ICs are installed in the 28 pin QuikLoader sockets, they are installed in the lower 24 pins as shown in Figure 5.

** Figure 5 photographs of both sides of QuikLoader.
must show jumpers clearly and point out
24 pin chip installation in 28 pin socket.

Chip 0 256K Jumpers

There is an additional pair of solder pad jumpers for chip 0 (labeled) 256K. These are made one way if 27256 is installed at socket 0 and the other way for the smaller ROMs. The QuikLoader comes with the jumpers made correctly for the IC type installed. If you replace a 256K chip 0 with a non-256K chip 0 or vice versa, you must change the 256K jumpers.

QuikLoader Chains: DMA IN / DMA OUT Jumpers

The DMA IN and DMA OUT jumpers are located near the edge connector of the QuikLoader. These need to be soldered when more than one QuikLoader are used in a given Apple. The QuikLoader hardware and QLOS support katalogging, running, and loading of files in such multiple configurations as long as the QuikLoaders are part of a continuous chain of cards. For example, you can have one QuikLoader in slot 2, two unassociated cards in slots 3 and 4, and QuikLoaders in slots 5 and 6. This would give the user instaneous access to up to 768K bytes of data and programs. The unassociated cards in the chain must have pins 27 (DMA IN) and 24 (DMA OUT) jumpered together. This is because the DMA IN/OUT priority chain is used prioritize the QuikLoaders with the highest priority QuikLoader in the lowest numbered slot. The following rules should be observed when installing more than one QuikLoader in an Apple:

1. no empty slots between QuikLoaders.
2. All unassociated cards between QuikLoaders have pin 27 jumpered to 24.
3. DMA IN must be soldered on all but highest priority (lowest slot) QuikLoader.
4. DMA OUT must be soldered on all but lowest priority (highest slot) QuikLoader.
5. No DMA cards can be in the QuikLoader chain. This includes alternate MPU cards (e.g. Z80 cards), DMA based manual controllers like SCRG's D MAnual Controller, and DMA based I/O controllers. DMA cards should be outside of the QuikLoader chain and isolated from it by an empty slot or by desoldering the highest priority DMA IN jumper or the lowest priority DMA OUT jumper, whichever is appropriate. Note that DMA cards can be installed in an Apple with a QuikLoader. They just cannot be installed within a chain of two or more QuikLoaders.
6. The Applesoft and Integer 12K firmware cards also use the DMA IN/OUT chain to prioritize multiple firmware card configurations. QLOS does not support a mixed chain of 12K firmware cards and QuikLoaders, so 12K firmware cards should be isolated from QuikLoader chains just as DMA cards (described above in rule 5).

Operation

Some Basic Hardware Features

There are sockets for eight ROM chips on the QuikLoader referred to as chip 0 through chip 7. When the QuikLoader is enabled, one of these eight chips is selected for response to addressing in the $C100-$FFFF range and motherboard response to this range is inhibited. Any time an Apple reset occurs -- when RESET is pressed, at power up, or when a peripheral card pulls RESET' low -- the QuikLoader is enabled with chip 0 selected. The result is that an Apple reset will always cause the Apple's 6502 MPU (Micro Processing Unit) to start executing a program stored in chip 0 of the highest priority QuikLoader. The program that the 6502 executes is QLOS.

The action QLOS takes at reset depends on the key which the operator last pressed or is pressing, and the state of the power-up byte ($3F4) as compared to the high byte of the Autostart soft reset vector ($3F3). By pressing the desired key concurrently with CTRL-RESET the operator can select from a variety of resets such as normal reset, forced power up, forced disk boot, QuikLoader katalog, executing programs on chips, etc. After QLOS processing, the QuikLoader is disabled and motherboard processing is renewed.

Besides interpretting the keyboard at system reset, QLOS performs many functions which make the QuikLoader a usable device. Knowledge of these functions is of no particular use to an operator but is very useful to programmers. If you are interested, please read the sections beginning with "Programming Primary Routines".

The Power-up Reset.

QLOS interprets the power-up byte exactly as the Autostart ROM does. If $3F4 contains the exclusive OR of $A5 and the contents of $3F3, the Apple is considered to have previously powered up. If $3F4 is not correctly set, a power-up reset is performed. But the QLOS power-up reset is different from the Autostart power up. Instead of booting the disk, QLOS performs the power-up routine on chip 6 of the highest priority QuikLoader. In the chip 6 supplied with the QuikLoader, this power-up routine transfers Integer BASIC to $E000-$F7FF of high RAM, transfers the motherboard monitor to $F800-$FFFF of high RAM, transfers DOS 3.3 to its normal operating location in RAM, initializes DOS, and enters Applesoft. This is all performed so fast that, for all purposes, the Apple powers up with DOS and Integer instantly available.

The power-up routine may be changed by changing chip 6 (see "Example 2, The Primary Routine"). This means that the QuikLoader can power up with any application operating, and only data disks, not program disks, need be resident in disk drives.

NOTE:
A chip with a power up routine must be installed in socket 6 of the highest priority QuikLoader.

NOTE:
When QLOS transfers and initializes DOS, the "last accessed drive" is always set to slot 6, drive 1.

Resetting the Apple //e

In the Apple //e it is possible for a program to tell if a key is being held down. QLOS uses this feature to force a normal reset unless a key is held down while CTRL-RESET is pressed and released. If no key is held or an undefined key is held while RESET is pressed, the motherboard reset is performed (or the QLOS power-up reset is performed if the power-up byte is bad). Therefore, if you never hold a key down while you press and release CTRL-RESET, the operation of the Apple //e will remain unchanged with a QuikLoader installed, except when the power-up byte is not right.

To select one of the special resets, you must press the desired key while pressing and releasing control reset. For example, to Katalog the QuikLoader files, press CTRL and "Q" with the left hand, press and release RESET, then release the CTRL and "Q" keys. QLOS will wait until you have released "Q" before performing the Katalog.

Holding a key down overrides a bad power-up byte in the Apple //e. for example, if you perform a Q-reset and the power-up byte is bad, QLOS will fix the power-up byte and perform a Katalog, not a power-up reset.

The //e open Apple and solid Apple keys force disk boot and diagnostic execution just as they do when QuikLoader is not installed. QLOS looks for these keys and immediately exits to the motherboard reset handler if one of them is being held down.

Resetting the Apple][and Apple][Plus

In the Apple][and][Plus, programs cannot test for "any key down". As a result, QLOS interprets the last key pressed before a reset to determine which selectable reset to perform. This has advantages and disadvantages over //e operation. The advantage is that you don't have to hold a key while you press CTRL and RESET. To perform a Q-reset, you press "Q" and release it, then you press and release RESET or CTRL-RESET as you normally do with your Apple. The disadvantage is that you must get into the habit of pressing "A" before RESET when you want a normal motherboard reset. Otherwise, QLOS will possibly interpret the last key press as a reset selection you really didn't intend to make. Quite often, this accidental selection will put you in the monitor, simply because the ASCII for RETURN is identical to CTRL-M, the monitor select key, and RETURN is often the last previous operational keypress. The A-reset forces a motherboard reset if the power-up byte is good and a QLOS power up if the power-up byte is bad.

A second feature of][/][Plus operation is that a keypress will not override a power-up reset if the power-up byte is bad. This is necessary in the][and][Plus so the reset which occurs at power up is not random. The one exception to this rule is an M (no CTRL); CTRL-RESET. This is the sole reset which overrides the power-up byte and causes an entry to the system monitor. Because of the M-reset override, some Apple][s or][Pluses may occasionally power up in the monitor. If this happens, you may then perform a Z-reset (move Integer and monitor; move and initilize DOS). If you experience the occasional monitor power up and your application will not tolerate it, contact SCRG at (- -) for consultation. The M-reset override is a compromise which should give most owners maximum system usefulness. But the feature can be deleted from QLOS if necessary.

The Selectable Resets

A list of selectable QLOS resets follows. Where possible, the keys were selected to help you remember the selected function (such as D for disk boot). In other instances, the keys were selected for proximity to the CTRL key. This minimizes the dexterity required to select a reset. Following this short list is a more detailed description of each reset type.

KEYBOARD RESET COMMANDS

Z - Move Integer, monitor, and DOS to RAM; initialize DOS; enter FP.
n - (number 0-7) Do routine on chip n.
Q - quikLoader katalog.
H - "Z-reset" then execute HELLO.
B - (boot only) Move DOS to RAM; initialize DOS; enter Applesoft.
D - Disk boot.
C - Catalog the disk.
M - Enter monitor (ditto RETURN).
S - Soft reset (slot 0 16K RAM card reset).
X - Go to mini assembler.

A-reset
Vector to contents of $FFFC/$FFFD on motherboard if power-up byte is good. Perform power-up routine on chip 6 if power-up byte is bad. Undefined key causes A-reset.

Z-reset
Move Integer BASIC, motherboard monitor, and DOS to their normal RAM locations. Initialize DOS and enter Applesoft.

n-reset
Press a number, n, between zero and seven in conjunction with RESET to perform the n-reset primary routine of chip n on the highest priority quikLoader. The primary routine which will be executed on a given chip is identified by an inverse "P" in its katalog display. An n-reset to a socket with no chip installed causes unpredictable results. If you accidently do this, simply reset the system again, selecting the reset type you desire. Chips with no n-reset routine can be programmed so n-reset will fall through to the same chip number on the second highest priority quikLoader. The chip 0 and chip 7 that are supplied with the quikLoader have no primary programs, but are programmed to fall through on n-reset.

An n-reset is not the only way to execute a program. It is a very easy way to run a single high priority program on each chip. Most programs will be run via the Q-reset and up to 256 programs may be katalogged, loaded, and/or run using the Q-reset.

Q-reset
quikLoader katalog, load, and run. Please see the following section, "quikLoader Katalog (Q-reset)".

H-reset
Perform the functions of the Z-reset then execute a program on a disk named HELLO (slot 6, drive 1). Note that this is not the same as executing a DOS hello program. The DOS hello program is a special program whose name is selected at disk initialization. Its name is usually HELLO, and the H-reset is a convenient way of executing this program when it is named HELLO.

B-reset

Boot only. Move and initialize DOS and enter Applesoft. This is like a Z-reset except Integer and the monitor are not transferred to RAM.

D-reset
Force a disk boot. The D-reset clobbers the power-up byte then enters the motherboard reset handler. This will force a disk boot if the Autostart ROM is resident on the motherboard. This gives the capability of overriding software which hangs the Apple via the soft reset vector. It is similar to the open Apple reset of the Apple //e, but it does not systematically modify RAM as the open Apple reset does.

C-reset
Catalog the last accessed disk. This is a convenient way to evade typing the most often entered command in the Apple repertoire. It does not transfer and initialize DOS so DOS must be resident if C-reset is to work. After cataloging, the last active BASIC is reentered without destroying any resident BASIC program.

M-reset
Enter monitor. RETURN-reset also forces monitor entry. M (no CTRL) reset is the only selectable reset that will override a bad power-up byte in the Apple][or Apple][Plus. The M-reset disconnects DOS and sets up the screen display and keyboard as primary output and input. For this reason, M-reset is undefeatable from software. Perform an A-reset to reconnect DOS.

S-reset
Soft reset causes execution of the slot 0 16K RAM card reset handler in the Apple][and Apple][Plus. This will enable Pascal and CP/M users to perform the soft reset required by these systems. This is necessary because the modification to the RAM card, required for operation with quikLoader, causes the RAM card to be disabled when RESET is pressed.

S-reset does not cause a vector to the reset handler of the built-in 16K RAM card of the Apple //e. This is because QLOS goes to the RAM card via a "JMP ($FFFC)" stored in page 1 of memory. It is a feature of the Apple //e that a "JMP ($FFFC)" executed from page 1 disables the built-in RAM card for reading. This secret feature was discovered during the debugging of QLOS. It doesn't particularly harm quikLoader operation, because the S-reset is only designed to regain a capability which was taken away from Apple][and][Plus users by the RAM card modification.

X-reset
Transfer Integer BASIC and the motherboartd monitor to high RAM and enter the mini assembler with high RAM enabled for reading and writing. The mini assembler is one of the nicer features of the Apple. X-reset will get you there in a hurry.

QuikLoader Katalog (Q-reset)

The Q-reset is the primary means by which various QLOS programs are selected and run. QLOS katalogging is similar to disk catalogging, but it is faster and more versatile. The katalog can be scrolled forwards and backwards if the number of entries exceeds the size of the screen. Programs are selected and run by pressing a single key. Coupled with virtually instantaneous data transfer, this leads to a level of convenience and utility unknown to Apple users before the developement of the QuikLoader and QLOS.

QLOS File Types

QLOS supports loading and/or running of four types of files. These are:

A files --- Applesoft files
B files --- Binary files
I files --- Integer files
P files --- Primary routines.

The A, B, and I files are just like A, B, and I DOS files. It is quite easy to take A, B, and I DOS files, store them consecutively in an EPROM buffer, add a QLOS format katalog record, and burn an EPROM with your own choice of A, B, and I files instantly available via the Q-reset.

The A, B, and I files are all transferred to RAM for running. A files are transferred to low memory, and I files are transferred to high memory just as if they were loaded from a disk. The B files are loaded to an address specified in the katalog entry for that file. There is no provision in QLOS for specifying alternate RAM destinations for B files.

The P files are unique to QLOS. They are files which are actually run while resident in the QuikLoader and will be referred as primary routines. The object of primary routines is to transfer combinations of programs and data to RAM for initialization and execution. For example, COPYA is an Applesoft program which normally "BLOADS" COPY.OBJ from a disk. It is implemented in the QuikLoader as a primary routine which transfers the COPY.OBJ program to RAM, transfers COPYA to RAM, and runs COPYA. Similarly, any programming application which requires operations more complex than running or loading of a single A, B, or I file would have to be implemented via a primary routine.


```
A B S1 C6 FID
B P S1 C6 COPYA
C A S1 C6 QUIK LOADER/QLOS HELP
D I S1 C4 BOAT
E B S1 C4 SPLIT SCREEN
F A S1 C4 GRID
G P S2 C3 APA
H B S2 C3 EXPUNGE DOS
I P S2 C2 VISICALC
J P S2 C0 EDITOR
K P S2 C0 ASSEMBLER
L
M
N
O
P
Q
R
S
T
U
V
W
```

Figure 6 - The Katalog Display

**Note to editor: The P next to COPYA, APA, VISICALC, and EDITOR must be inverted in final draft

The Katalog Display

Performing a Q-reset with a QuikLoader installed in your Apple results in a screen display showing the QuikLoader resident files. Figure 6 shows the katalog display of a QuikLoader with some programs added. In addition to the name of each file, the display shows a selection index, a file identifier, and the slot and chip number of where each file resides. Also, by pressing Z, you can display the source, length, and destination parameters for each file.

On the far left side of the katalog display are the letters "A" through "W" in a single column. These letters are the file selection indices for the katalogged files. For example, the name COPYA is next to the letter "B". If you press "B" on the keyboard, you will run the primary routine, COPYA. Similarly, any program the katalog display can be run by pressing its index letter.

In addition to the letters A-W, other keys perform special functions as follows:

ESC Escape from katalog to BASIC.

Y Cause loading instead of running. After pressing Y, pressing the index letter for an A, B, or I file will cause the program to be transferred from the QuikLoader to RAM, bun not run.

Z Toggle the parameter display. This enables the display of source, length, and destination information as an aid for persons programming OLOS compatible PROMS. When the parameter display is on, the file names are truncated to nine characters instead of their normal 29. The source parameter is the base address of an A, B, or I file or the primary routine in QuikLoader. The length parameter is the length of A, B, or I files. The destination parameter is the destination in RAM of B files. The length parameter is meaningless for primary routines, and the destination parameter is meaningless for A files, I files, and primary routines.

Scrolling (left arrow, right arrow, and space)

If the number of QuikLoader files exceeds 23, then only the first 23 files will be displayed after a Q-reset. In this instance, you must scroll the katalog display to gain access to files not on the current display. Press right arrow to scroll forward one file. Press left arrow to scroll backwards one file. Press space for continuous scrolling in the direction of the last arrow press. Press right or left arrow to stop the continous scrolling and set the direction.

When scrolling forwards, the display will scroll to the last file then stop. When scrolling backwards, the display wraps around from file 0 to file 255. If, as normal, there are less than 233 katalog entries, this will cause the screen to go blank. If you find yourself with a blank screen and wish to get rid of it, press right arrow then space. After a short while the display will reappear.

The maximum number of katalog entries which can be displayed and selected by OLOS is 256. After entry number 255 is printed to the screen, the display wraps around to entry number 0. If you exceed 256 entries, there will be no way to access the extra files.

Programming QLOS Compatible EPROMs

QLOS compatible EPROMs are filled primarily with programs and data, packed in tightly with no space between them. Additionally, QLOS EPROMs have a certain amount of chip overhead. Chip overhead is data which is not part of the usable QLOS files, but is present to support QLOS formats. This overhead includes a katalog record and some other data which is required for QLOS operation.

If a chip has no primary routines, its overhead beyond the katalog record is very minimal and straightforward. This means you can easily learn to program QLOS EPROMs with any number of Applesoft, Integer, or binary programs available for transfer and execution. Programming primary routines is more complex. There are several possible variations of overhead requirements, and more knowledge of QLOS and quikLoader structure is required. Primary routines aren't particularly difficult to write, but writing primary routines is more difficult than packing A, B, and I files together with a katalog record.

EPROM

EPROM stands for Erasable Programmable Read Only Memory. It is non-volatile, which means that, like masked ROM, the data in EPROM cannot be altered in the course of normal operation. When your data is in EPROM, it is always available, just as the BASIC and monitor in ROM are always available in the Apple. Unlike ROM, however, EPROM can be erased and reprogrammed.

The acronym EPROM is used to refer to ultraviolet light erasable PROM. This UVEPROM has been in use for years, and is the type of EPROM you will probably use in the quikLoader. UVEPROM has a little transparent window in the top. Shining light from a common ultraviolet lamp through the window will erase a UVEPROM in about 20 minutes. A small UVEPROM eraser with a timer can be purchased for approximately $50.

A second type of EPROM is EEPROM (Electrically Erasable PROM). EEPROM is pin compatible with UVEPROM and can be used in the quikLoader. EEPROM is a much more recent development, though, and UVEPROM is less expensive and in more common usage than EEPROM.

EPROMs and ROM which can be used in the quikLoader are compatible with the Intel 27nn series. These include

2716	2K bytes / 16K bits
2732	4K bytes / 32K bits
2764	8K bytes / 64K bits
27128	16K bytes / 128K bits
27256	32K bytes / 256K bits.

You can program any of these sizes of chips for the quikLoader, and programs you puchase for the quikLoader may come on any of these sizes of chips. Size, cost, and availability will enter into consideration when deciding which type of chip to enter your programs on. 2764s give you a reasonable size of chip

for about $10 retail. 27128s are a good deal more expensive and hard to come by. 27256s are priced out of this world and nearly impossible to obtain as of December 1983. The price and availability of 27128s and 27256s are expected to become much more reasonable in 1984. As this occurs, you will be able store larger amounts of data on the quikLoader while remaining solvent.

To program EPROM, you need an EPROM burner. This is a device which will program EPROMs from a source file in some sort of computer. PROM burners exist which can be plugged into an Apple, and this is probably the type you will want to use for quikLoader EPROMs since you will burning Apple files into EPROM. It will be desireable for your PROM burner to be capable of burning 2764, 27128, and 27256 EPROMs. The quikLoader can utilize the smaller 2716 and 2732 EPROMS, but these EPROMs reduce the capacity of the quikLoader. Remember that one 27128 holds the same amount of data as eight 2716s or four 2732s.

PROM burners of various sorts can be purchased from electronics stores and computers. SCRG is in the process of developing a inexpensive PROM burner that will burn 27nn series EPROMS up to 27256s. This PROM burner is scheduled for release in 1984.

quikLoader Addressing Ranges

The quikLoader addressing range is from $C100 to $FFFF. This is a range of 128K bytes minus 256. Figure 7 shows the range at which each type of EPROM is addressed. As Figure 7 illustrates, the addressing ranges used for the smaller chips are the high portions of the quikLoader range. This is strictly a QLOS convention since the quikLoader hardware will let you address the smaller chips at more than one range. For example, a 6502 program can access the data in a 2732 at $F000-$FFFF, $E000-$EFFF, $D000-$DFFF, or $C100-$CFFF. QLOS, however, will only access 2732 data at $F000-$FFFF.

You may have noticed that the $C100-$FFFF range does not allow access to the bottom 256 bytes of data in a 27128 or 27256. This 256 bytes is not usable for storage of A, B, or I files or the katalog record. It is usable for storage of data accessed by primary routines using techniques described under "27128/27256 Programming Considerations". For EPROMs with no primary routines, the maximum amount of accessible data at a single socket is less than 16K bytes (16384 - 256 = 16128 usable bytes). For 27256s, only one bank is available for A, B, and I files and the katalog record. Therefore, 27256s are only practical when they contain at least one primary routine.

** Figure 7 address ranges of various size EPROMs in quikLoader

FIGURE 7 CAN BE FOUND IN THE BACK

The Katalog Record

The katalog record of a QLOS EPROM is made up of individual katalog entries, stored sequentially starting at some base address. The base address of the katalog record of a QLOS EPROM must be stored at $FFF8 and $FFF9 of the EPROM. If there is no katalog record on a QLOS EPROM, this must be identified by the value $FFFF or a value less than $C100 at $FFF8 and $FFF9. (It is possible that an EPROM will have no katalog record if it contains only data which is accessed by a primary routine on a different chip.)

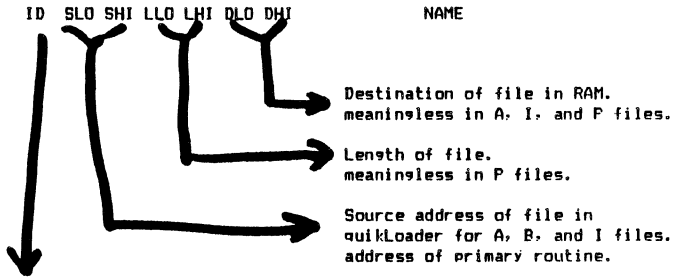
Figure 8 shows the format of the katalog record and Figure 9 shows the format of a single entry in a katalog record. The entry contains a file identifier, source/length/destination parameters, and ASCII for the name of the file as it appears on the katalog screen display. A typical katalog entry takes up about 20 bytes, so an EPROM with 10 QLOS files would require about 200 bytes for the katalog record, depending on the length of the file names.

```
ID  SLO SHI LLO LHI DLO DHI          NAME
ID  SLO SHI LLO LHI DLO DHI          NAME
ID  SLO SHI LLO LHI DLO DHI          NAME
              .
              .
              .
ID  SLO SHI LLO LHI DLO DHI          NAME
$86

        $86 (CONTROL-F) TERMINATES KATALOG RECORD
```

Figure 8 - The Katalog Record Format

ASCII name of QLOS file:
All ASCII above $8F is valid. This includes numbers, upper case, lower case, and special characters. It excludes control, inverse, and flashing characters. Maximum name length is 29 characters.



File ID:
$81 (CTRL-A) = Applesoft program
$82 (CTRL-B) = Binary program
$89 (CTRL-I) = Integer program
$90 (CTRL-P) = Primary routine
$86 (CTRL-F) = Finish - terminate katalog record

Figure 9 - Katalog Entry Format

Programming Examples

There are two parts to a QLOS compatible EPROM, the data files and an overhead file. The data files are packed together by loading them sequentially from disks to a buffer area of RAM. The overhead file is built separately and merged with the data files in the buffer area. This merged file is saved to disk and loaded to the EPROM burn buffer when it is time to burn an EPROM.

To pack the data files, it is necessary to find the lengths and compute the starting address of the various files. A "Texas Instruments Programmer" calculator is very helpful in computing the starting addresses. Programming Example 1 demonstrates how to find the lengths of A, B, and I files.

Building the overhead file should be done using a 6502 editor/assembler. It can be done by hand, but an assembler will make the task much easier. Examples shown here were assembled using Apple's "DOS TOOL KIT" editor/assembler. You do not have to know 6502 assembly language programming to grasp these examples or to use an assembler for similar purposes. Except for the primary routines, no 6502 programming is required.

The general method of the examples is to pack the data files starting at the low address of the EPROM being used ($C100 for 27128/27256, $E000 for 2764, $F000 for 2732, $F800 for 2716). Overhead files start at $FF00, an arbitrary but convenient starting place. This works out pretty well because parts of QLOS overhead must be at high addresses.

Example 1

making a QLOS compatible chip with no primary routines.

Objective
Place BOAT (I), SPLIT SCREEN (B), and GRID (A) on a QLOS formatted 2764 EPROM.

1. Calculate file lengths and arrange files together.

```
LOAD BOAT from disk.
]CALL -151
*$CA, $CB = $9449; length of BOAT = $9600-$9449 = $1B7
BSAVE BOATB,A$9449,L$1B7

BLOAD SPLIT SCREEN from disk.
*$AA72, $AA73 = $1F00 = SPLIT SCREEN destination
*$AA60, $AA61 = $003E = SPLIT SCREEN length

LOAD GRID from disk.
]CALL -151
*$69, $6A = $087F; length of GRID = $87F-$801 = $7E
BSAVE GRIDB,A$801,L$7E

start BOAT at address $E000 (arbitrary)
BOAT resides at $E000-$E1B6
SPLIT SCREEN resides at $E1B7-$E1F4
GRID resides at $E1F5-$E272

BLOAD BOATB,A$2000
BLOAD SPLIT SCREEN,A$21B7
BLOAD GRIDB,A$21F5
BSAVE BT.SPSCRN.GRD,A$2000,L$273
```

2. Bring up the DOS TOOL KIT editor.
   BRUN EDASM.OBJ

   Enter following text file:

```
        ORG $FF00       START KATALOG RECORD AT $FF00
BOATK   DFB $89         CONTROL-I (INTEGER)
        DW  $E000       SOURCE
        DW  $01B7        LENGTH
        DW  $0000       MEANINGLESS DESTINATION
        ASC 'BOAT'      NAME
SPLTK   DFB $82         CONTROL-B (BINARY)
        DW  $E1B7       SOURCE
        DW  $003E       LENGTH
        DW  $1F00       DESTINATION
        ASC 'SPLIT SCREEN'
GRIDK   DFB $81         CONTROL-A (APPLESOFT)
        DW  $E1F5       SOURCE
        DW  $007E       LENGTH
        DW  $0000       MEANINGLESS DESTINATION
        ASC 'GRID'      NAME
        DFB $86         CONTROL-F ENDS KATALOG RECORD
*
*
        DS  $FFEF-*     SKIP TO $FFEF
        LDA #$00        REQUIRED CODE MUST BEGIN AT $FFEF
        NOP
        STA $C081,X
        DS  3
        DW  BOATK       KATALOG POINTER AT $FFFB
        DW  $3FB        NMI POINTER AT $FFFA

:SAVE EXAMPLE 1 OVERHEAD
:ASM EXAMPLE 1 OVERHEAD
:END
```

3. Merge data blocks with chip overhead.

```
]CALL -151
*2000:FF
*2001<2000.3FFEM
*BLOAD BT.SPSCRN.GRD,A$2000
*BLOAD EXAMPLE 1 OVERHEAD.OBJ0,A$3F00
*BSAVE TEST,A$2000,L$2000
```

TEST is now a valid QLOS EPROM source file. BLOAD it to your EPROM burn buffer and burn it on 2764 EPROM.

The programs in Example 1 are very short and will not fill up an EPROM. This means that if you actually built this QLOS EPROM, you would have a lot of room for future expansion.

EXAMPLE 1 OVERHEAD.OBJ0 in Example 1 is the chip overhead. The portion beginning at $FFEF is top overhead, and the top overhead of Example 1 is that required when a katalog record is present but no primary routines are present. The "LDA #$00, NOP, STA $C081,X" sequence will allow fall through to the second priority quikLoader if an n-reset is performed to this chip.

** Figure 10 example 1 overhead source/object listing

FIGURE 10 CAN BE FOUND IN THE BACK

Primary Routines

Primary routines must be written when an application can't be implemented using straightforward A, B, or I files. Use primary routines when:

1. a program is made up of more than one contiguous block of data.
2. a program exceeds the capacity of a single EPROM.
3. you wish an important program to be activated by n-reset.
4. you wish to utilize the first 256 bytes of a 27128 or 27256 EPROM.
5. you wish to make a program the power-up program in the event the host chip is installed in chip socket 6.

The primary routine is actually a 6502 program, so you must be able to write 6502 assembly language programs if you want to design applications around primary routines.

The general idea of a primary routine is to transfer combinations of programs and data to RAM for execution. For example, COPYA is a combination of an Applesoft program and a 6502 machine language program. The primary routine for COPYA first transfers the machine language program, then transfers and executes the Applesoft program. The primary routines are part of the chip overhead, separate from the blocks of data which are transferred to RAM.

During the course of primary routine execution, control may be passed back and forth between the chip containing the primary routine and chip 0 on the highest priority quikLoader. Typically, chip 0 will initially call the primary routine, then the primary routine will make several calls to routines available on chip 0, then the primary routine will exit to some address on the motherboard via the chip 0 GO TO MOTHERBOARD routine.

The passing of control back and forth is done via bank switching instructions at fixed locations in the top overhead. "STA $C081,X" with slot number times $10 in the X-register accomplishes the switching, and strict protocol must be followed so program flow can proceed in an orderly way when one chip is switched off and another is switched on.

The quikLoader configuration register receives the bottom five bits of the accumulator when the "STA $C081,X" is executed. Additionally, QLOS protocol calls for the number of the exited chip to be in the top three bits of the accumulator. This results in the following chip switching control word:

ABC,O,U,DEF

where ABC is the exited chip number
O is the quikLoader ON/OFF flip-flop
U is the quikLoader USR flip-flop
DEF is the entered chip number.

This protocol makes it possible for QLOS to execute a subroutine for a chip then return to that chip.

A complete discourse on writing primary routines encompasses several related subjects. The following analysis of Example 2 is meant to introduce you to these subjects and give you a feel for developing EPROMs which include primary routines.

** Figure 11 example 2 overhead source/object listing

FIGURE 11 CAN BE FOUND IN THE BACK

Example 2

making a QLOS compatible chip with primary routines

Objective
Put FID (B), COPYA (P), and the quikLoader "HELP" screen program (A) on a QLOS formatted 2764 EPROM. Include GETSLOT overhead and a power-up routine.

This example shows the overhead portion of the chip supplied with the quikLoader in socket 6. It serves as a good example for several options you may wish to exercise when programming EPROMs for the quikLoader. Numbers in parenthesis in the following paragraphs refer to line numbers in the Example 2 overhead listing.

The data blocks of Example 2 are packed similarly to those in Example 1, and the packing for this example is shown in lines 10-13 of the overhead listing. For this example, only the overhead file will be discussed, but there are two points about the data block packing which should be mentioned. First, the statement in the Applesoft program which loads the COPY.OBJ file in the disk version is deleted. Second, the remarks are left in the Applesoft program. This was done to faithfully reproduce the program as it is released

by Apple Computer Inc. In normal practice, you should expunge all remarks from Applesoft programs stored on QuikLoader EPROMs becauses storage space is too valuable to waste on remarks.

The Top Overhead

The top overhead (221-230) for Example 2 is the top overhead which will normally be found in a QLOS chip. It supports primary routine entry from QLOS (225 and 227), calling of chip 0 subroutines (224), return from chip 0 subroutines (227), a katalog record pointer (229), and an NMI vector (230). An IRQ vector is normally not necessary because all QLOS processing is with interrupt requests disabled. You may set up an IRQ vector and enable interrupts during a primary routine, but you should disable interrupts before calling chip 0 routines. The reset vector is not necessary because only chip 0 of the highest priority QuikLoader is active just after a reset.

Lines 222 and 223 of the Example 2 top overhead are not mandantory. They are part of the scheme by which this particular primary routine calls chip 0 subroutines. You may find this method convenient in your routines, but it's the programmer's option.

The "JMP N.RESET" of line 225 will be executed after an n-reset to this chip. N.RESET will be the label of the high priority primary routine which you wish to be activated by this convenient method.

The Katalog Record

Line 229 shows that the first katalog entry begins at FIDK/$FF00. The katalog record shows three files FID, COPYA, and QUIK LOADER/QLOS HELP. COPYA is the only primary routine on this chip, but you can put more than one primary routine on a chip.

Even though there is only one primary routine in this katalog record, there may be any number of data blocks associated with the primary routine. In this particular example there are two data blocks -- an Applesoft program and a binary program -- associated with the primary routine. The katalog entry gives no information about these associated data blocks. They are handled by the primary routine, not QLOS.

Note that the beginning address of the primary routine, COPYA, is the same as the operand of the "JMP N.RESET" stored at $FFEF. When displaying the katalog, QLOS inverts the "P" file identifier on the screen for the file which has this equivalence. The inverted "P" identifies the primary routine which will be executed when an n-reset occurs for this chip.

The Primary Routine

The primary routine, COPYA, begins at line 180. This is also the n-reset routine of a chip 6 compatible EPROM, so the action to be taken at power up must be considered. The power-up status is passed in the 6502 carry flag. If the carry flag is clear at the beginning of an n-reset, then this is a power-up entry to chip 6, the autostart chip. If the carry flag is set, then this is an n-reset or Q-reset entry to the primary routine.

Besides the power-up status, the inputs from QLOS utilized by this primary routine are the control word in the accumulator and the slot number x $10 of the highest priority QuikLoader stored at PRISLOT/$26. The accumulator control word input to the primary routine will be 000,X,1,DEF meaning chip 000 is calling chip DEF with USR high. This should be inverted to DEF,0,0,000 (chip DEF calling chip 000 via PRISLOT turn on with USR low) to obtain the control word required to make calls to chip 0. INVERT (110) is a subroutine which will perform this inversion.

The COPYA routine saves the power-up status (180), inverts and saves the control word (181), then retrieves and interprets the power-up status (182-183). If the carry flag is clear, the power-up routine is executed (184-196).

This power-up routine is what happens when you turn your Apple on or a program sets the power-up byte for power up. It uses chip 0 subroutines to move Integer BASIC from the QuikLoader to high RAM, move the monitor from motherboard ROM to high RAM, move DOS from the QuikLoader to $9D00-$BFFF of RAM. It then sets the DOS command byte ($AA5F) to $1B. (Setting this value to $00 would result in the execution of a disk program named HELLO after the DOS initialization). Next the Applesoft ONERR status is set so disk I/O errors will be handled by DOS. Then the initialization address minus one ($B73A) is pushed to the stack, and the motherboard is entered and a DOS cold start is performed. This power up is not a fixed feature of the QuikLoader but a feature of the chip supplied with the QuikLoader in socket 6. You can change the power up by changing this chip, but you should study lines 170-196 of Example 2 for guidance. As an example, you can make the Apple enter the copy program at power-up by replacing lines 191-196 with a "JMP DOCOPY".

The chip 0 routines are called by placing the index of the desired subroutine in the Y-register and jumping to GOCHIP0 in the top overhead. The indexes of the available subroutines are shown in lines 25-34. Complete descriptions of the routines are given in the "Chip 0 Subroutines" section. For this discussion, the only important non-obvious point is that you must set up source, length, and destination values for MOVEBLK at $3A-$3F and you must set up source and length values for LOADFP, RUNFP, LOADINT, and RUNINT at $3A-$3D and identically at $204-$207. This is so QLOS will know where to move your data block, Applesoft program, or Integer program.

In Example 2, if a power up is not being executed, the DOCOPY routine at 205-219 will be run. This routine simply transfers the data block, COPY.OBJ, to RAM then uses RUNFP to run the Applesoft program, COPYA. The

source, length, and destination values are set up from the OBJPARM and CPYPARM tables (103-108). Using tables like these and comments like lines 10-13 in your overhead will help you keep the data blocks associated with primary routines organized.

GETSLOT Overhead

Lines 62-76 and 118-169 are GETSLOT overhead. This is optional overhead which is required only for a chip operating in socket 0 of quikLoaders other than the highest priority quikLoader. Chip 0 on the highest priority quikLoader is filled with DOS, Integer, and QLOS. But only a small portion of QLOS is required in chip 0 of the lower priority quikLoaders.

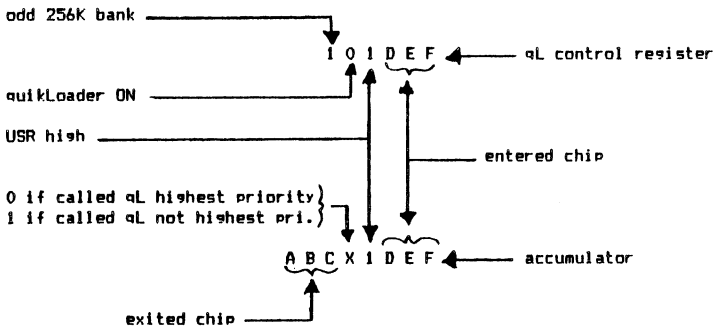
The required code is the GETSLOT portion of Example 2. For it to function, the "STA $C081,X" must be at $FF55, exactly as shown in line 122. This is quikLoader turn-off code which must be capable of flowing to the "RTS" located at $FF58 on the motherboard when a quikLoader is turned off.

The presence of GETSLOT code at chip 0 of every quikLoader allows each quikLoader, working down from highest priority to lowest priority, to locate its slot and turn itself off at go to motherboard time. At the end of this operation, a bit map of the slots with quikLoaders resides at QLMAP/$2D. This map is available for use by primary routines.

Why put the GETSLOT overhead in a chip 6 EPROM if it is only needed in chip 0? Because there is room for it. Someday you may have more than one quikLoader in your Apple and want to utilize chip 0 in the lower priority slots. Any chip with GETSLOT overhead can be used there, so it is a good idea to put GETSLOT into all of your EPROMs when there is room. Place the marking, "C0 OK", on chips which contain the GETSLOT overhead.

Inputs to Primary Routines.

Example 2 showed one method of utilizing a Primary routine, but many more are possible. For example You may move a screen menu driver to RAM and execute it. This driver will then perform functions selected by the operator including going back to the quikLoader for more data or programs. When writing such applications, the folowing list of inputs to primary routines should help.



carry flag	clear if power-up.
X-register	slot number x $10 of highest priority qL slot.
$26	slot number x $10 of highest priority qL slot.
$27	slot number x $10 of called qL slot.
$2C	MSB reset if Apple //e; set if Apple II.
$2D	bit map of qL slots 1,2,3,4,5,6,7,x

//e SLOTC3ROM soft switch set to INTERNAL.
//e INTCXROM soft switch set to INTERNAL.
called quikLoader ON.
all other quikLoaders OFF.

Top Overhead

The highest few addressed locations on a QuikLoader EPROM are referred to as the top overhead. The top overhead includes such things as bank switching locations, the katalog pointer, and 6502 NMI', RESET', and IRQ' vectors where appropriate. Examples 1 and 2 illustrate the two commonest types of top overhead. Table 1 shows how the various types of top overhead are tied together.

The QLOS top overhead is interlaced with that on the other chips. This is because switching is generally between QLOS and primary routines. For example, when a QLOS "STA $C081,X" is executed at $FFEC, a "JMP N.RESET" on the called chip picks up the flow at $FFEF.

The fourth column of Table 1 shows a type of top overhead required for an EPROM with blocks of data and programs on it but with no katalog record or primary routine to move the data blocks. The katalog record and primary routines are on an adjacent chip. This sort of handling is required if your data exceeds the capacity of the chips you are using.

A primary routine can use the chip 0 routines to move data on a separate chip by specifying that chip in the top three bits of the accumulator control word. After execution of the move routine, however, flow will return to the chip from which the data was transferred. The top overhead in column 4 of Table 1 will relay the flow to the next higher chip where the primary routine resides. The primary routine can then continue processing. If you change the "ADC #1" to "SBC #0", the return flow will be relayed to the next lower chip instead of the next higher. The column 4 top overhead also has fall through code to pass the n-reset to the second priority QuikLoader if there is one.

Table 1 - Top Overhead

	QLOS	FALL, NO KAT	NORMAL	MINUS CHIP
				GOPLUS LDX SECSLOT
				ADC #1
				AND #$EF
				BCC GOBACK
$FFEC	GOCHIP STA $C081,X		GOCHIP0 STA $C081,X	
$FFEF	JMP DOROUTS	NOP	JMP N.RESET	NOP
$FFF1		LDA #$00		LDA #$00
$FFF2	GOBACK STA $C081,X	STA $C081,X	DS 3	GOBACK STA $C081,X
$FFF5	JMP FALL	DS 3	RTS	CLC
$FFF6			DS 2	BCC GOPLUS
$FFF8	DW $0000	DW $0000	DW KATLOC	DW $0000
$FFFA	DW $3FB	DW $3FB	DW $3FB	DW $3FB
$FFFC	DW RESET			

27256 Programming Considerations

The natural addressing range of a 27256 would be $8000-$FFFF, so the 27256 must be banked switched. It behaves like two bank switched 27128s and all 27128 rules must be followed with 27256s. The odd bank is selected by storing the accumulator control word at $C081,X with slot number times $10 in the X-register. The even bank is selected by the same storage instruction with (slot number x $10) - 1 in the X-register.

QLOS only looks for A, .B, and I files and the katalog record in the odd bank. The even bank can be used only by primary routines. The chip 0 transfer routines will use the even bank as the transfer source if you perform a "DEC $26, DEC $27" before calling chip 0. This will result in (slot number x $10) - 1 in the X-register at transfer set up time. $26, $27, and the X-register will be restored by QLOS upon return to the calling chip. Note that only $26 and $27 were decremented before calling chip 0. The X-register should always contain slot number x $10 when calling chip 0.

```
Example:    LDX $26
            DEC $26
            DEC $27
            JSR GOCHIP0
```

There is no QLOS top overhead requirement for the even bank of 27256 chips. The even bank is never enabled unless a primary routine switches to the even bank or unless a transfer call to chip 0 is made after decrementing $26 and $27. Obviously, 27256s can only be used in connection with primary routines.

NOTE:
The 27256 even bank can also be enabled if the chip is in socket 0 of a lower priority quikLoader. See the discussion of GETSLOT overhead in the next section.

27128/27256 Programming Considerations

When programming 27128s and 27256s, contingencies arise which don't have to be taken into account with the smaller chips. You need to be aware of these when programming the big chips. In the following discussion, assume the natural addressing range of a 27128 or one bank of a 27256 is $C000-$FFFF.

The first problem is with the GETSLOT overhead (see programming Example 2). If you wish to use a 27128 or 27256 in chip 0 of a lower priority quikLoader, you need to include the GETSLOT overhead. But with a 27128 or 27256, the "STA $C081,X" must be at $DF55 instead of $FF55. In other words, at least part of the GETSLOT overhead must be in page $DF, even though the top overhead is at the top of page $FF. In a 27256, the "STA $C081,X" must be at $DF55 of the even bank. This is a bit of a nuisance, although it is not an insurmountable problem. Unless all of your quikLoaders are full of 27128s and 27256s, the easiest thing is not to use these big chips in socket 0. Just use a 2764 there. If you really need a 27128 or 27256 there, you will have to break your data blocks in page $DF.

The second thing you need to be aware of with 27128s and 27256s is the effects of the USR flip-flop on the quikLoader. When USR is high, all but 256 bytes of a 27128 or a bank of a 27256 can be addressed at $C100-$FFFF. The $C000-$CFFF area of the chip is unavailable in this mode. When USR is low, addresses in the $E000-$FFFF range access the bottom half of the 27128 or 27256 (the $C000-$DFFF area). This means that USR bank switches the $E000-$FFFF addressing range and that the bottom 256 bytes of the big chips is accessable at $E000-$E0FF when USR is low.

The QLOS katalog routine does not take advantage of the USR bank switching to access the bottom 256 bytes when loading A, B, or I files or reading katalog records. Therefore, A, B, and I files and katalog records must reside in the $C100-$FFFF area of 27128s and 27256 odd banks. Primary routines can easily access the bottom 256 bytes of the big chips via customized bank switching schemes, or via chip 0 move routines. In a call to chip 0, the carry flag is shifted to the USR flip-flop for the data transfer. This has no effect with the smaller chips, but it means that primary routines must specify USR low (carry clear) or USR high (carry set) when calling chip 0 routines. For example, to transfer the bottom half of a 27128 or 27256 to RAM, make source $E000, length $2000, clear the carry flag and call MOVEBLK (Y=0).

NOTE:
The Example 2 primary routine would require minor modification for 27128/27256 implementation because no cognizance was taken of the carry flag before calling chip 0 routines.

NOTE:
Primary routines are always entered with USR high, and return from chip 0 routines is always with USR high. Also notice in Example 2 that calls to chip 0 are made with USR low.

The Chip 0 Subroutines

There is a group of subroutines available on chip 0 of the highest priority quikLoader for use by primary routines on other chips. These routines can be called from any chip on any quikLoader by following some simple programming steps. The presence of these routines greatly reduces the amount of code necessary in primary routines.

The chip 0 subroutines are called via the bank switching "STA $C081,X" at $FFEC with the desired subroutine identified by the Y-register value. Table 2 is a general catagorization of the chip 0 subroutines. Some specific subroutine usage notes follow here.

The MOVEBLK, MOVEINT, MOVEDOS, and DOJSR routines are called via a "JSR" since a return is expected. The other routines are called via a "JMP". The accumulator control word must be set up and the X-register must contain the slot number x $10 of the highest priority quikLoader as shown in programming Example 2. Source, length, and destination values must be set up for transfer routines, and transfer configuration needs to be set up for 27128s and 27256s. The transfer configuration is set up as follows:

chip	action	result
27128, 27256	CLC	USR low during transfer
27128, 27256	SEC	USR high during transfer
27256	DEC $26, $27	even bank transfer source
27256	$26, $27, as is	odd bank transfer source

MOVEDOS moves DOS to RAM then initializes it. The initialization is intercepted at $9E41 in the midst of DOS first entry processing. You can then send a keyboard command to DOS by storing it in the GETLN buffer ($200-$2FF).

Example: Catalog the disk.

1. Store ASCII for "CATALOG" at $200-$206.
2. Store $8D (CR) at $AA56.
3. Store 7 (length of "CATALOG") at $AA5A.
4.
```
LDX $2E          SET STACK POINTER
INX
INX
TXS
```
5. Exit to $9FB3 via GOMRBRD chip 0 routine.

Primary routines can directly execute subroutines residing in $0000-$BFFF motherboard RAM. DOJSR allows you to execute subroutines in the $C100-$FFFF motherboard and peripheral slot address range. DOJSR performs the monitor routine, RESTORE ($FF3F), before subroutine execution and the routine, SAVE ($FF4A), after execution. This is a means of passing 6502 register values to and from the motherboard subroutine. These subroutines use locations $45-$49 for storage of the 6502 registers (see lines 45-49 of programming Example 2). You need to set up $45-$49 as appropriate if you want to specify 6502 register inputs to a motherboard subroutine.

LOADFP, RUNFP, LOADINT, and RUNINT all move DOS to RAM and initialize it as part of BASIC initialization. Running and loading of Applesoft and Integer programs without DOS is not supported by QLOS.

Table 2 - The Chip 0 Subroutines

Y-REG	NAME	FUNCTION	SET UP	MEMORY USAGE (4)
0	MOVEBLK	Move data block from QuikLoader to RAM.	(1, 2, 7)	$111-$151, (5)
2	MOVEINT	Move Integer from QuikLoader to high RAM. Move monitor from motherboard to high RAM.	none	$111-$151, (5)
4	MOVEDOS	Move DOS from QuikLoader to RAM and initialize.	none	(6)
6	DOJSR	Execute motherboard subroutine.	(7, 9)	$111-$128, $45-$49
8	GOMRBRD	Go to motherboard address	(7)	(8)
10	MBRDRST	Do motherboard reset	(7)	$111-$116
12	LOADFP	Move DOS to RAM and initialize. Move Applesoft program from qL to RAM. Enter Applesoft.	(1, 3)	$111-$114, $200-$202, FP page 0, (6, 8)
14	RUNFP	Do LOADFP. Run the Applesoft program.	(1, 3)	$111-$114, $200-$203, FP page 0, (6, 8)
16	LOADINT	Do MOVEINT. Move DOS to RAM and initialize. Move Integer program from qL to RAM. Enter Integer.	(1, 3)	$111-$151, $200-$202, Int page 0, (6, 8)
18	RUNINT	Do LOADINT. Run the Integer program.	(1, 3)	$111-$151, $200-$203, Int page 0, (6, 8)

(1) carry flag to USR if during transfer. DEC $26, $27 for even 256K bank (see "27256 Programming Considerations" and "27128/27256 Programming Considerations").
(2) source, length, destination to $3A-$3F.
(3) source, length to $3A-$3D $204-$207.
(4) all subroutines modify SAVCTRL/$30.
(5) $3A-$3F modified.
(6) $2E, $3A-$3F, $3D0-$3FF, $9D00-$BAFF, $BC00-$BFFF modified. Dos buffers initialized ($9600-$9CFF).
(7) high RAM control to $112 if needed (see "High RAM Control").
(8) restores INTCXROM and SLOTC3ROM soft switches (see "Apple //e INTCXROM and SLOTC3ROM Soft Switches").
(9) subroutine address to $129, ~~$150~~ 712A. 6502 register inputs to $45-$49.

High RAM Control

Primary routines are first entered from QLOS with high RAM (the 16K RAM card) enabled for writing and disabled for reading. High RAM will stay configured this way unless the primary routine changes the configuration. The primary routine can thus store data to high RAM at any time, and MOVEBLK calls to chip 0 can be used to transfer data to high RAM.

High RAM should not be configured for reading while the QuikLoader is enabled. In other words, don't do a "LDA $C080" or similar command from a primary routine. If you do enable high RAM for reading from a primary routine in the Apple][or][Plus, the QuikLoader will compete with the 16K RAM card for control of the data bus. If you enable high RAM from a primary routine in an Apple //e, high RAM will still be disabled for reading as long as the QuikLoader is enabled. This is because high RAM is disabled by the INHIBIT' line in the Apple //e.

The MOVEBLK, DOJSR, GOMRBRD, and MRBRDRST chip 0 subroutines respond to a special high RAM control byte. Before executing any of these routines a "STA $C0XX" (stored at $111-$113) is executed. Primary routines control this instruction by storing a value at $112. This byte is normally set to $81, and it will remain at $81 unless a primary routine changes it. This results in execution of "STA $C081" (disable high RAM read, leave write enable as is). You can change this byte to configure the high RAM for the chip 0 subroutine. For example, place $83 at $112 and call DOJSR to perform a high RAM subroutine. After MOVEBLK and DOJSR, high RAM is disabled for reading via a "STA $C081" before return to the primary routine. The $112 control of high RAM bank 2 is as follows:

$80	read on, write off.
$81	read off, write as is.
$82	read off, write off.
$83	read on, write as is.

Location $112 can be used to do the 16K RAM card reset via the MRBRDRST routine in an Apple][or][Plus. It cannot be used to do the high RAM reset in the Apple //e. This is because the "JMP ($FFFC)" from a page 1 memory address disables high RAM in the Apple //e.

The MOVEINT, RUNINT, and LOADINT routines assume that high RAM is configured for writing as it is when QLOS first passes control to the primary routine. Primary routines which disable high RAM writing must reenable it before calling these routines. When program flow goes to high RAM after RUNINT or LOADINT, high RAM will be disabled for writing.

Apple //e INTCXROM and SLOTC3ROM Soft Switches

The QuikLoader addressing range overlaps the I/O SELECT' ($C100-$C7FF) and I/O STROBE' ($C800-$CFFF) addressing ranges. I/O SELECT' must therefore be deactivated while the 6502 is addressing the $C100-$C7FF range of the QuikLoader. This will also eliminate I/O STROBE' conflicts because slot 1-7 peripheral card response to I/O STROBE' is initiated by I/O SELECT'.

I/O SELECT' in Apple][s and][Pluses is automatically inhibited by the USER 1' line of the Apple when the QuikLoader is enabled and its USR flip-flop is high. I/O SELECT' in the Apple //e must be inhibited by program control of the SLOTC3ROM and INTCXROM soft switches. Primary programs are always entered with these switches at INTERNAL so the QuikLoader can respond to $C100-$C7FF addressing while inhibiting motherboard response via the INHIBIT' line.

GOMRBRD, LOADFP, RUNFP, LOADINT, and RUNINT all restore the SLOTC3ROM and INTCXROM soft switches before entry to motherboard. INTC3ROM is set to SLOT response, and SLOTC3ROM is set to SLOT or INTERNAL depending on the presence of absence of an auxiliary RAM card.

Running Programs Resident in the QuikLoader

The operational philosophy of The QuikLoader and QLOS is to transfer programs to RAM for execution. However, programs can be run while they reside in the QuikLoader. There are some limitations on this capability, though.

Resident programs can exercise motherboard I/O features controlled by the $C000-$C07F address range without limitation. Slot I/O control via DEVICE SELECT' ($C080-$C0FF) can also be performed as long as it doesn't enable a device which will compete with the QuikLoader for control of the data bus. Peripheral card ROM programs in the I/O SELECT' range ($C100-$C7FF) can be called while QuikLoader is enabled with the USR flip-flop low. This means you could activate a printer driver as long as it doesn't utilize the I/O STROBE' gated expansion ROM. The QuikLoader has no provision for disabling response to the $C800-$CFFF range so resident programs can not activate I/O peripherals which respond to the I/O STROBE'.

A second problem with resident programs is that they have restricted access to motherboard monitor routines. You can execute motherboard subroutines in the $C100-$FFFF range via the DOJSR call to chip 0, but this becomes unwieldy if you need to make many calls. There are no limitations on calling motherboard subroutines in the $0000-$BFFF range.

QLOS Memory Usage

QLOS uses a certain amount of RAM, even though it runs in ROM. Like all programs, it requires pointers, counters, temporary storage, etc. Additionally, certain QLOS routines must be run from RAM. These routines are transferred from the quikLoader to RAM for execution.

Figure 12 shows the QLOS equate table. This information may be of use to primary routine programmers. Zero page locations are used for most temporary storage locations. This was done because there is a limited space available for QLOS in the quikLoader, and use of zero page locations makes a 6502 program more compact. An attempt was made to select the zero page locations so as to minimize likely interference with user programs.

The QLOS RAM routines run in page 1 and, in the case of the katalog routine, page 2. This memory area was chosen because it doesn't contain data critical to most programs. The idea here is to perform QLOS functions with a minimal chance of clobbering user data. For example you can do a B-reset to initialize DOS in the Apple and very few memory locations outside of the DOS area will be modified.

** Figure 12 QLOS equate table

FIGURE 12 IS IN THE BACK

Commercial Development of quikLoader Programs

It is quite easy for software publishers to publish their programs on quikLoader EPROM in addition to diskettes. SCRG encourages the publishing of such products and is anxious to consult with any persons or companies interested in doing so. SCRG is also willing to become a distributer of programs on EPROM or ROM for those companies hesitant to become involved with EPROM programming and adaptation to QLOS formats. It is SCRG's intention to keep all quikLoader puchasers advised of those programs that are available in EPROM.

The quikLoader is especially well suited to utilities, business, word processing, spreadsheet, and data base management applications and programs that generally put the Apple to work. People who work their Apple are very appreciative of the concept of instant and convenient access to applications.

Some programs are fairly massive and possibly inappropriate for quikLoader implementation. Programs which take up 100K bytes would have to be very valuable to a user to justify the cost in EPROM and quikLoader space. Of course, a valuable program coupled with quikLoader convenience can be a very markatable product in spite of substantial production cost.

Adapting a commerical program to quikLoader involves writing primary routines to handle the application and conversion of disk access to quikLoader

access when appropriate. It may be desireble to access the quikLoader occasionally to defeat unauthorized copying via NMI based RAM copying cards. This can be done with quikLoader without inconveniencing the user since access to quikLoader is so fast.

Published programs should normally have a power up routine as part of the n-reset routine. This will allow the user to put your chip in socket 6 and have the Apple power up in your application. Your documentation should inform the purchaser whether or not the chip is socket 6 compatible or not. This can be done by marking the chip "C6 OK". Also, if your chip contains the GETSLOT overhead, mark it "C0 OK".

SCRG has a developer's information package available to companies and persons interested in publishing programs on EPROM or ROM. This can be obtained by calling or writing

Southern California Research Group
attn: quikLoader Developers' Package
PO BOX 2231
Goleta, CA 93118

phone: (- -)

Hardware Description

The quikLoader is simply a firmware card that responds to read cycle addresses between $C100-$FFFF when it is on. It isolates motherboard ROM and Apple //e high RAM from the data bus by pulling the INHIBIT' line low. It isolates peripheral card ROM from the data bus by pulling USER 1 low in the Apple IC and IC Plus, and by manipulation of the SLOTC3ROM and INTCXROM soft switches of the Apple //e.

There are eight ROM sockets on the quikLoader, each of which is capable of response to the $C100-$FFFF range. Only one socket of the quikLoader is enabled at any one time, and chip 0 is the primary socket, enabled by Apple resets.

Control of the quikLoader is centered around the six bit configuration register (see Figures 13 and 14). This register determines whether the quikLoader is on or off, which chip is enabled, whether the odd or even 256K bank is enabled, and address response as determined by the USR flip-flop. The register is set to 000000 by an Apple reset, and it can be set under program control by storing values to the DEVICE SELECT' range of the slot in which a quikLoader resides.

The six bit register is divided up as follows:

B,0,U,DEF

where B = 256K even bank when low
0 = quikLoader on when low
U = USR flip-flop
DEF = the selected chip

From this you can see that an Apple reset forces the quikLoader to even 256K bank, on, USR low, chip 0.

When setting the quikLoader under program control, bits D4-D0 of the data bus go to bits 4-0 of the control register. These bits are set by placing the control value in the accumulator and performing a "STA $C081,X". A0 of the address bus is the input to control register bit 5, so odd addresses will select the odd 256K bank. The storing instruction is performed with slot number x $10 in the X-register to select the odd bank and (slot number x $10) - 1 in the X-register to select the even bank.

The USR flip-flop performs two functions. First, it bank switches the $E000-$FFFF addressing range for 27128 and 27256 EPROMs (see "27128/27256 Programming Considerations"). Second, it enables the chip 0 operating system to locate the quikLoader slot by polling the I/O SELECT' range of the Apple slots. Slot location is possible because when USR is low, the quikLoader will respond to addressing in the $C100-$C7FF range only when I/O SELECT' for its slot is low. Since USR is low after a reset, this enables a chip 0 routine to locate the quikLoader slot (see the GETSLOT overhead section of programming Example 2)\ The self find feature is what makes the quikLoader slot

independent.

If the DMA IN line is low and the DMA IN jumper is made, the quikLoader will not respond to its address range, even if it is otherwise enabled. It will also echo the low DMA IN input to its DMA OUT output. Similarly, if a quikLoader is enabled and responding to its address range, it will bring DMA OUT low, thus disabling lower priority quikLoaders. When an Apple reset occurs, all quikLoaders are enabled, but only the highest priority one will be responding. This is because the DMA IN line of all but the highest priority quikLoader will be low. When the highest priority quikLoader first turns itself off after a reset, the second priority quikLoader will pick up the program flow, find itself, and turn itself off. This continues through the chain until all quikLoaders are off. At the completion of turn off, location $2D contains a bit map of slots which contain quikLoaders. This is used by the katalog routine and the n-reset fall through routine of QLOS. It is also available for possible use by primary routines.

** Figures 13 and 14 quikLoader block diagram and schematic

FIGURE 13 IN ON THE INSIDE BACK COVER
FIGURE 14 IN ON THE BACK COVER

Appendix A - QLOS Command Search Flowchart

Appendix B - Expunging DOS from Disks

Owners of QuikLoader do not normally need to boot DOS 3.3 from a disk. Therefore, with QuikLoader installed in your Apple, there is no reason to have DOS resident on all of your disks. This means you can remove the bootable DOS image from most of your disks and gain an additional 32 sectors per disk for data storage. You only need to keep DOS on a few disks for safekeeping in case the need arises to boot DOS from a disk.

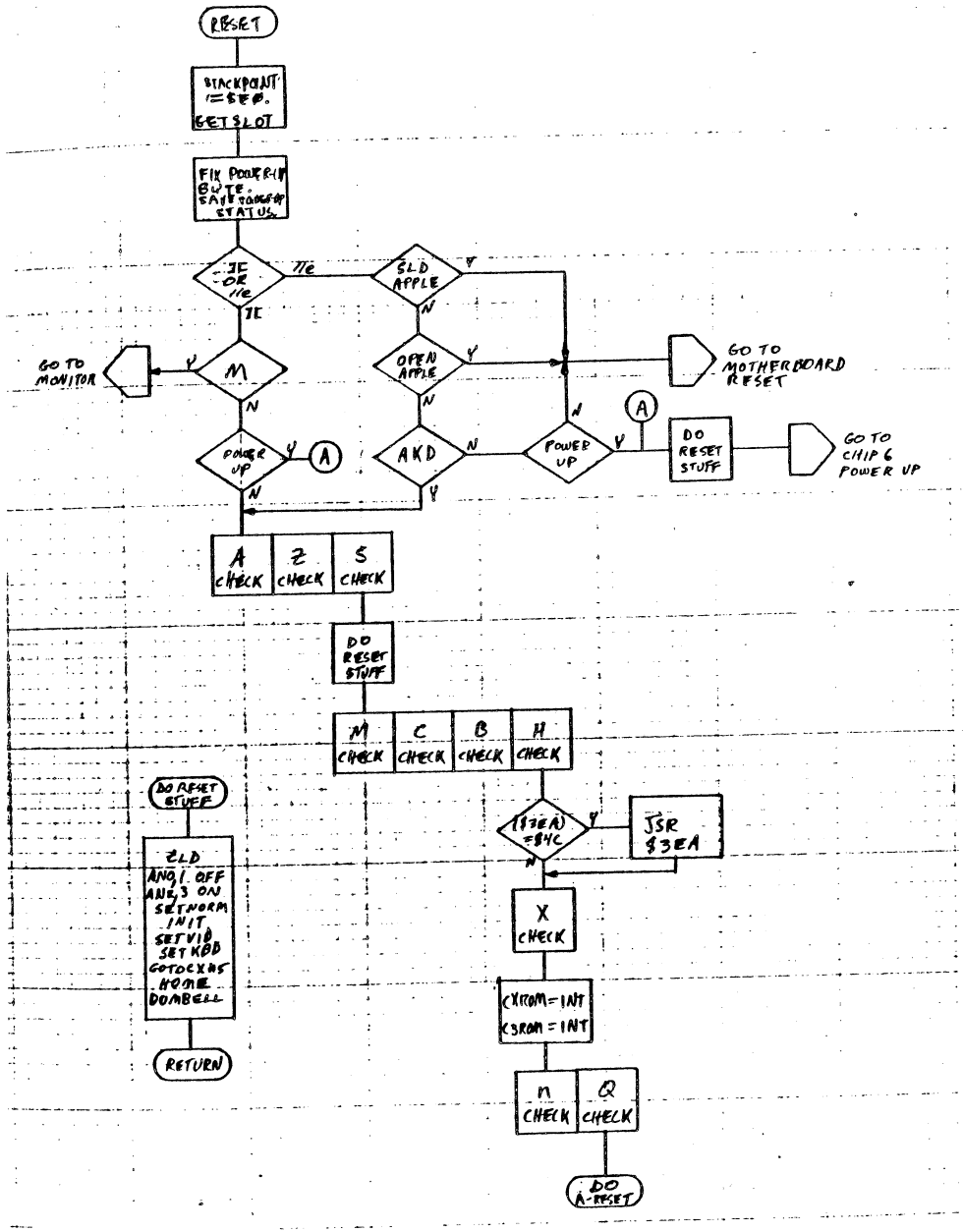
The accompanying program is a utility which will expunge the DOS image from disks. It does not actually overwrite DOS, but only frees tracks 1 and 2 in the VTOC (Volume Table Of Contents at track $11, sector $00). It does overwrite track $00, sector $00 with a short program to print a reminder that DOS has been expunged if you attempt to boot the disk. Since tracks 1 and 2 are free in the VTOC, DOS will eventually overwrite the tracks if you store enough data to the disk.

You should be cautious of a couple of pitfalls that you might encounter when expunging DOS from your disks. First, many commercial programs contain a modified version of DOS and won't run with the standard DOS 3.3. Expunging the modified DOS from a disk like this could cause the disk to become irretrievably clobbered. It is therefore recommended that you only expunge DOS from disks you have initialized yourself or from backups of commercial programs. Certainly you should never expunge DOS from disks which must be booted to bring up the resident application.

The second pitfall in expunging DOS is when you attempt to expunge DOS from a disk that doesn't contain DOS. When you free tracks 1 and 2 on a disk like this, you may well be enabling DOS to overwrite important data. In other words, don't run an expunge program more than once on a disk, and don't attempt to expunge DOS from a disk that never had DOS on it (e.g. disks formatted by spreadsheet programs or word processors).

To operate the listed EXPUNGE program, just BRUN the object program and do what the screen prompts say. This program will warn you if any sector on tracks 1 or 2 is free or if track $00, sector $00 contains a "not bootable" message. If this is the case, the DOS image is probably not present and you should probably not allow EXPUNGE to free tracks 1 and 2.

** source/object listing of EXPUNGE



APPENDIX A - Command Search Flowchart

```
SOURCE FILE: EXPUNGE
                    1 ********************************************************************
0000:               2 *
0000:               3 * EXPUNGE DOS UTILITY
0000:               4 *
0000:               5 * WRITTEN FOR QUIKLOADER BY JIM SATHER
0000:               6 *
0000:               7 * VERSION 1. 1/4/84
0000:               8 *
                    9 ********************************************************************
0000:              10 *
0006:              11 IOBL     EQU  $6
0007:              12 IOBH     EQU  $7
002B:              13 SLOTNUM  EQU  $2B        DOS SLOT NUMBER X $10
0025:              14 CV       EQU  $25
03F4:              15 PWRBYTE  EQU  $3F4
C088:              16 DRVOFF   EQU  $C088
FC24:              17 VTABZ    EQU  $FC24
FC58:              18 HOME     EQU  $FC58
FD0C:              19 RDKEY    EQU  $FD0C
FDED:              20 COUT     EQU  $FDED
FDF0:              21 COUT1    EQU  $FDF0
0000:              22 *
0000:              23 *
0002:              24 WRITE    EQU  2          IOB WRITE COMMAND
0000:              25 *
0004:              26 TRACK    EQU  $4         IOB INDEX
0008:              27 BUFL     EQU  $8         IOB INDEX
000C:              28 COMMAND  EQU  $C         IOB INDEX
0000:              29 *
0000:              30 *
----- NEXT OBJECT FILE NAME IS EXPUNGE.OBJ0
6000:              31          ORG  $6000
                   32 ********************************************************************
6000:A2 0D         33 BEGIN    LDX  #INSRT.M-ERR.M PRINT INSERT DISK MESSAGE:
6002:20 A1 60      34          JSR  PRINTM     WAIT FOR KEYPRESS.
6005:C9 9B         35          CMP  #$9B       ESCAPE?
6007:D0 03         36          BNE  *+5
6009:4C D0 03      37          JMP  $3D0       ESCAPE TO DOS WARM START.
600C:29 DF         38          AND  #$DF       MAKE LOWER CASE UPPER.
600E:C9 D8         39          CMP  #'X'
6010:D0 EE         40          BNE  BEGIN      EXPUNGE IF 'X' PRESSED.
                   41 ********************************************************************
6012:              42 *
6012:              43 * FIX IOB
6012:              44 *
                   45 ********************************************************************
6012:20 E3 03      46          JSR  $3E3       GET IOB ADDRESS.
6015:85 07         47          STA  IOBH
6017:84 06         48          STY  IOBL
6019:A0 00         49          LDY  #0         TRANSFER IOB.
601B:B9 6C 61      50 IOBLP    LDA  IOBSRC,Y
601E:91 06         51          STA  (IOBL),Y
6020:C8            52          INY
6021:C0 06         53          CPY  #6         SKIP DEVICE TABLE.
6023:D0 02         54          BNE  *+4
6025:A0 08         55          LDY  #8
6027:C0 0D         56          CPY  #$D
6029:90 F0         57          BCC  IOBLP
                   58 ********************************************************************
602B:              59 *
602B:              60 * NOW TRY TO MAKE SURE DOS NOT ALREADY EXPUNGED
602B:              61 *
                   62 ********************************************************************
602B:38            63          SEC
602C:08            64          PHP             SET NOT EXPUNGED FLAG.
602D:20 BE 60      65          JSR  GORWTS     READ SECTOR 0/0.
6030:AD F1 61      66          LDA  BUFFER+1
6033:C9 A6         67          CMP  #$A6
6035:D0 03         68          BNE  *+5
6037:28            69          PLP             SET EXPUNGED FLAG IF $A6.
6038:18            70          CLC
6039:08            71          PHP
603A:A9 11         72          LDA  #$11       READ VTOC (SECTOR 11/0).
603C:A0 04         73          LDY  #TRACK
603E:91 06         74          STA  (IOBL),Y
6040:20 BE 60      75          JSR  GORWTS
6043:A2 38         76          LDX  #$38       FIX VTOC FOR TRACKS 1 & 2 FREE:
6045:BD F0 61      77 VTOCLP   LDA  BUFFER,X
6048:F0 03         78          BEQ  *+5
604A:28            79          PLP             SET EXPUNGED FLAG IF BIT MAP NOT ALL Z
EROS.
604B:18            80          CLC
604C:08            81          PHP
604D:A9 FF         82          LDA  #$FF
604F:E0 3C         83          CPX  #$3C
6051:90 03         84          BCC  *+5        DON'T FREE TRACK 0.
6053:9D F0 61      85          STA  BUFFER,X
6056:E8            86 SKPLP    INX
6057:8A            87          TXA
6058:29 02         88          AND  #$02       FIX EVERY OTHER PAIR OF BYTES.
605A:D0 FA         89          BNE  SKPLP
605C:E0 42         90          CPX  #$42
605E:90 E5         91          BCC  VTOCLP
                   92 ********************************************************************
6060:              93 *
6060:              94 * READY TO WRITE VTOC NOW.
6060:              95 * IF SECTOR 0/0 OR VTOC LOOKS FISHY,
6060:              96 * LET OPERATER DECIDE WHETHER OR NOT TO PROCEED.
6060:              97 *
                   98 ********************************************************************
6060:28            99          PLP
6061:B0 0B        100          BCS  CONT       BRANCH IF OK TO EXPUNGE
6063:A2 5E        101          LDX  #EXPD.M-ERR.M
6065:20 A1 60     102          JSR  PRINTM     ALREADY EXPUNGED. CONTINUE?
6068:29 DF        103          AND  #$DF       CONVERT KEYPRESS LOWER TO UPPER CASE.
606A:C9 D9        104          CMP  #'Y'
606C:D0 1D        105          BNE  GOBEGIN    ESCAPE IF NOT 'Y'.
606E:A9 02        106 CONT     LDA  #WRITE     WRITE VTOC.
```

```
6070:A0 0C     107          LDY  #COMMAND
6072:91 06     108          STA  (IOBL),Y
6074:20 BE 60  109          JSR  GORWTS
6077:A9 00     110          LDA  #0         WRITE SECTOR 0/0.
6079:A0 04     111          LDY  #TRACK
607B:91 06     112          STA  (IOBL),Y
607D:A0 08     113          LDY  #BUFL      MAKE SRC0.0 THE RWTS BUFFER.
607F:A9 79     114          LDA  #>SRC0.0
6081:91 06     115          STA  (IOBL),Y
6083:C8        116          INY
6084:A9 61     117          LDA  #<SRC0.0
6086:91 06     118          STA  (IOBL),Y
6088:20 BE 60  119          JSR  GORWTS
608B:4C 00 60  120 GOBEGIN  JMP  BEGIN      GO BACK FOR MORE.
               121 ********************************************************************
               122 ********************************************************************
608E:          123 *
608E:          124 * GO TO RWTS SUBROUTINE
608E:          125 *
               126 ********************************************************************
608E:20 E3 03  127 GORWTS   JSR  $3E3       FIND IOB.
6091:20 D9 03  128          JSR  $3D9       DO RWTS.
6094:B0 01     129          BCS  IO.ERR
6096:60        130          RTS
6097:68        131 IO.ERR   PLA             POP.
6098:68        132          PLA
6099:A2 00     133          LDX  #0
609B:20 A1 60  134          JSR  PRINTM     PRINT "I/O ERROR"
609E:4C 00 60  135          JMP  BEGIN      GO BACK FOR MORE.
               136 ********************************************************************
               137 ********************************************************************
60A1:          138 *
60A1:          139 * PRINT MESSAGE AND WAIT SUBROUTINE
60A1:          140 *
               141 ********************************************************************
60A1:20 58 FC  142 PRINTM   JSR  HOME
60A4:A9 07     143          LDA  #7
60A6:85 25     144          STA  CV
60A8:20 24 FC  145          JSR  VTABZ
60AB:BD BA 60  146 PRTLP    LDA  ERR.M,X
60AE:F0 06     147          BEQ  ENDPRT     $00 ENDS MESSAGE
60B0:20 ED FD  148          JSR  COUT
60B3:E8        149          INX
60B4:D0 F5     150          BNE  PRTLP
60B6:20 0C FD  151 ENDPRT   JSR  RDKEY      WAIT FOR KEYPRESS.
60B9:60        152          RTS
               153 ********************************************************************
               154 ********************************************************************
60BA:          155 *
60BA:          156 * MESSAGES OUTPUT BY PRINTM SUBROUTINE
60BA:          157 *
               158 ********************************************************************
60BA:87 87 87  159 ERR.M    DFB  $87,$87,$87 THE BELL TOLLS.
60BD:C9 AF CF  160          ASC  'I/O       ERROR'
```



```
60C0:A0 C5 D2
60C3:D2 CF D2
               161          DFB  $00
60C6:00
60C7:C9 CE D3  162 INSRT.M  ASC  'INSERT   DISK IN SLOT 6, DRIVE 1.'
60CA:C5 D2 D4
60CD:A0 C4 C9
60D0:D3 CB A0
60D3:C9 CE A0
60D6:D3 CC CF
60D9:D4 A0 B6
60DC:AC A0 C4
60DF:D2 C9 D6
60E2:C5 A0 B1
60E5:AE
60E6:8D 8D     163          DFB  $8D,$8D
60E8:D0 D2 C5  164          ASC  'PRESS    "X" TO EXPUNGE DOS.'
60EB:D3 D3 A0
60EE:A2 D8 A2
60F1:A0 D4 CF
60F4:A0 C5 D8
60F7:D0 D5 CE
60FA:C7 C5 A0
60FD:C4 CF D3
6100:AE
6101:8D 8D     165          DFB  $8D,$8D
6103:D0 D2 C5  166          ASC  'PRESS    ESC TO ESCAPE.'
6106:D3 D3 A0
6109:C5 D3 C3
610C:A0 D4 CF
610F:A0 C5 D3
6112:C3 C1 D0
6115:C5 AE
6117:00        167          DFB  $00
6118:C4 C1 CE  168 EXPD.M   ASC  'DANGER'
611B:C7 C5 D2
611E:8D 8D     169          DFB  $8D,$8D
6120:87 87 87  170          DFB  $87,$87,$87 THE BELL TOLLS.
6123:C4 CF D3  171          ASC  'DOS      ALREADY EXPUNGED.'
6126:A0 C1 CC
6129:D2 C5 C1
612C:C4 D9 A0
612F:C5 D8 D0
6132:D5 CE C7
6135:C5 C4 AE
6138:8D 8D     172          DFB  $8D,$8D
613A:C4 CF A0  173          ASC  'DO       YOU STILL WISH TO FREE
613D:D9 CF D5
6140:A0 D3 D4
6143:C9 CC CC
6146:A0 D7 C9
6149:D3 C8 A0
614C:D4 CF A0
614F:C6 D2 C5
6152:C5
```

```
6153:8D 8D      174         DFB  $8D,$8D
6155:D4 D2 C1   175         ASC  'TRACKS   1 AND 2?    Y/N'
6158:C3 CB D3
615B:A0 B1 A0
615E:C1 CE C4
6161:A0 B2 BF
6164:A0 A0 A0
6167:A0 D9 AF
616A:CE
616B:00         176         DFB  $00
                177 ********************************************************************
                178 ********************************************************************
616C:           179 *
616C:           180 * IOB SOURCE DATA
616C:           181 *
616C:           182 * READ SECTOR 0/0 INITIALLY
616C:           183 *
                184 ********************************************************************
616C:01         185 IOBSRC  DFB  $01        (0)
616D:60         186         DFB  $60        (1) SLOT 6
616E:01         187         DFB  $01        (2) DRIVE 1
616F:00         188         DFB  $00        (3) ANY VOLUME
6170:00         189         DFB  $00        (4) TRACK 0
6171:00         190         DFB  $00        (5) SECTOR 0
6172:           191         DS   2          (6) DEVICE TABLE
6174:F0 61      192         DW   BUFFER     (8) BUFFER ADDRESS
6176:00         193         DFB  $00        (A) NOT USED
6177:00         194         DFB  $00        (B) 256 BYTE PARTIAL SECTOR
6178:01         195         DFB  $01        (C) COMMAND = READ
                196 ********************************************************************
                197 ********************************************************************
6179:           198 *
6179:           199 * SOURCE DATA FOR SECTOR 0/0 OF EXPUNGED DISK
6179:           200 *
6179:           201 * (AS SUGGESTED BY "BAG OF TRICKS",
6179:           202 * WORTH AND LECHNER, QUALITY SOFTWARE, 1982)
6179:           203 *
6179:01         204 SRC0.0  DFB  $01
617A:A6 2B      205         LDX  SLOTNUM
617C:BD 88 C0   206         LDA  DRVOFF,X
617F:EE F4 03   207         INC  PWRBYTE    FIX POWER-UP BYTE FOR REBOOT.
6182:20 58 FC   208         JSR  HOME
6185:A9 07      209         LDA  #7
6187:85 25      210         STA  CV
6189:20 24 FC   211         JSR  VTABZ
618C:A2 00      212         LDX  #0
618E:BD 20 08   213 MSGLP   LDA  MESSAGE-SRC0.0+$800,X
6191:F0 FE      214 HANG    BEQ  HANG
6193:20 F0 FD   215         JSR  COUT1
6196:E8         216         INX
6197:D0 F5      217         BNE  MSGLP
6199:D4 C8 C9   218 MESSAGE ASC  'THIS     DISK CANNOT BE BOOTED.'
619C:D3 A0 C4
619F:C9 D3 CB
```



```
61A2:A0 C3 C1
61A5:CE CE CF
61A8:D4 A0 C2
61AB:C5 A0 C2
61AE:CF CF D4
61B1:C5 C4 AE
61B4:8D 8D      219         DFB  $8D,$8D
61B6:C4 CF D3   220         ASC  'DOS       EXPUNGED BY QUIKLOADER.'
61B9:A0 C5 D8
61BC:D0 D5 CE
61BF:C7 C5 C4
61C2:A0 C2 D9
61C5:A0 D1 D5
61C8:C9 CB CC
61CB:CF C1 C4
61CE:C5 D2 AE
61D1:8D 8D      221         DFB  $8D,$8D
61D3:D0 D2 C5   222         ASC  'PRESS     CTRL-RESET FOR REBOOT.'
61D6:D3 D3 A0
61D9:C3 D4 D2
61DC:CC AD D2
61DF:C5 D3 C5
61E2:D4 A0 C6
61E5:CF D2 A0
61E8:D2 C5 C2
61EB:CF CF D4
61EE:AE
61EF:00         223         DFB  $00
                224 ********************************************************************
61F0:           225 BUFFER  EQU  *
                226 ********************************************************************

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

```
6000 BEGIN
606E CONT
C088 DRVOFF
608B GOBEGIN
60C7 INSRT.M
601B IOBLP
60A1 PRINTM
6056 SKPLP
FC24 VTABZ
```

```
61F0 BUFFER
FDED COUT
60B6 ENDPRT
608E GORWTS
6097 ID.ERR
616C IOBSRC
60AB PRTLP
  2B SLOTNUM
6045 VTOCLP
```

```
  08 BUFL
FDF0 COUT1
60BA ERR.M
6191 HANG
  07 IOBH
6199 MESSAGE
03F4 PWRBYTE
6179 SRC0.0
  02 WRITE
```

```
  0C COMMAND
  25 CV
6118 EXPD.M
FC58 HOME
  06 IOBL
618E MSGLP
FD0C RDKEY
  04 TRACK
```

```
  02 WRITE
  08 BUFL
03F4 PWRBYTE
6056 SKPLP
6097 ID.ERR
60BA ERR.M
6179 SRC0.0
61F0 BUFFER
FD0C RDKEY
```

```
  04 TRACK
  0C COMMAND
6000 BEGIN
606E CONT
60A1 PRINTM
60C7 INSRT.M
618E MSGLP
C088 DRVOFF
FDED COUT
```

```
  06 IOBL
  25 CV
601B IOBLP
608B GOBEGIN
60AB PRTLP
6118 EXPD.M
6191 HANG
FC24 VTABZ
FDF0 COUT1
```

```
  07 IOBH
  2B SLOTNUM
6045 VTOCLP
608E GORWTS
60B6 ENDPRT
616C IOBSRC
6199 MESSAGE
FC58 HOME
```

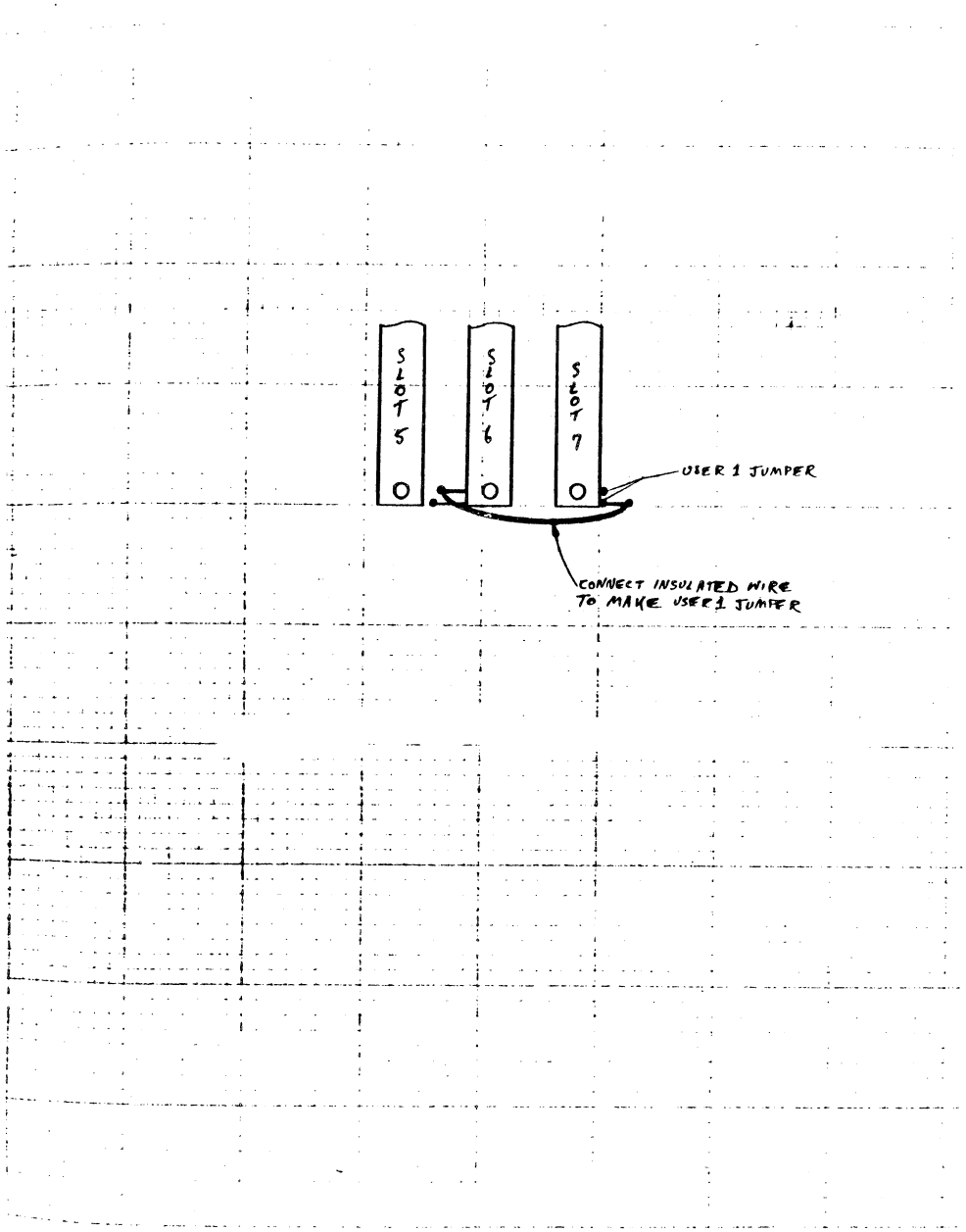



Figure 1 - USER1 Jumper prior to RFI revision

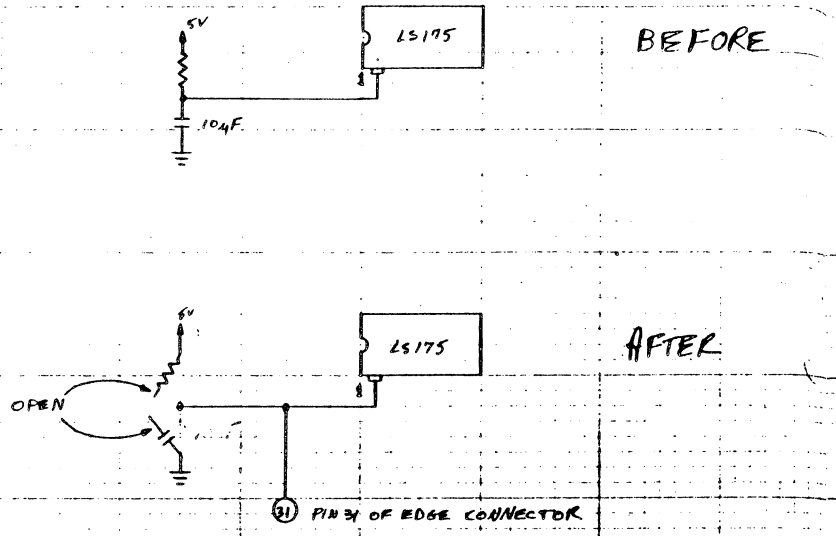


FIGURE 3 - Modifying the 16K RAM card

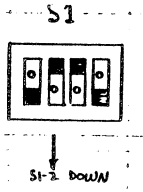


Figure 4 - Super RAM card switch setup

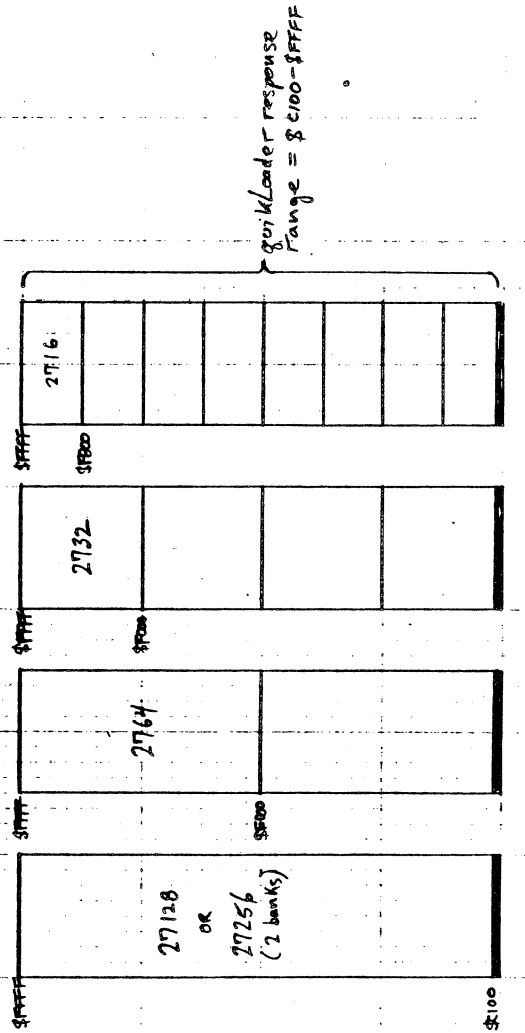


Figure 7 - EPROM addressing in the quikLoader

```
SOURCE FILE: EXAMPLE 1 OVERHEAD
----- NEXT OBJECT FILE NAME IS EXAMPLE 1 OVERHEAD.OBJ0
FF00:                1          ORG  $FF00     START KATALOG RECORD AT $FF00
FF00:89              2 BOATK    DFB  $89      CONTROL-I (INTEGER)
FF01:00 E0           3          DW   $E000     SOURCE
FF03:B7 01           4          DW   $1B7      LENGTH
FF05:00 00           5          DW   $0000     MEANINGLESS DESTINATION
FF07:C2 CF C1        6          ASC  'BOAT'    NAME
FF0A:D4
FF0B:82              7 SPLTK    DFB  $82      CONTROL-B (BINARY)
FF0C:B7 E1           8          DW   $E1B7     SOURCE
FF0E:3E 00           9          DW   $003E     LENGTH
FF10:00 1F          10          DW   $1F00     DESTINATION
FF12:D3 D0 CC       11          ASC  'SPLIT    SCREEN'
FF15:C9 D4 A0
FF18:D3 C3 D2
FF1B:C5 C5 CE
FF1E:81             12 GRIDK    DFB  $81      CONTROL-A (APPLESOFT)
FF1F:F5 E1          13          DW   $E1F5     SOURCE
FF21:7E 00          14          DW   $007E     LENGTH
FF23:00 00          15          DW   $0000     MEANINGLESS DESTINATION
FF25:C7 D2 C9       16          ASC  'GRID'    NAME
FF28:C4
FF29:86             17          DFB  $86      CONTROL-F ENDS KATALOG RECORD
FF2A:               18 *
FF2A:               19 *
FF2A:               20          DS   $FFEF-*  SKIP TO $FFEF
FFEF:A9 00          21          LDA  #$00     REQUIRED CODE MUST BEGIN AT $FFEF
FFF1:EA             22          NOP
FFF2:9D 81 C0       23          STA  $C081,X
FFF5:               24          DS   3
FFF8:00 FF          25          DW   BOATK    KATALOG POINTER AT $FFF8
FFFA:FB 03          26          DW   $3FB     NMI POINTER AT $FFFA

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

FIGURE 10

```
SOURCE FILE: FIDCOPY ROUTS
0000:                1 *
                     2 ********************************************************************************
0000:                3 *
0000:                4 *     EXAMPLE 2 ... OVERHEAD FILE
0000:                5 *
0000:                6 *     FID, COPY.OBJ0, COPYA
0000:                7 *
                     8 ********************************************************************************
0000:                9 *
0000:               10 *FID        E000-F24D
0000:               11 *COPY.OBJ0  F24E-F378
0000:               12 *COPYFP     F379-FAC9
0000:               13 *QL HELP    FACA-FE7F
0000:               14 *
0000:               15 *
                    16 ********************************************************************************
0000:               17 *
0000:               18 *      CHIP 0 ROUTINE EQUATES
0000:               19 *
0000:               20 * Y-REGISTER INDEXES OF THE CHIP 0 ROUTINES FOLLOW.  0-6 ARE
0000:               21 * INDEXES OF ROUTINES FROM WHICH A RETURN IS EXPECTED.
0000:               22 * INDEXES 8 AND UP ARE FOR ROUTINES
0000:               23 * FROM WHICH THERE IS NO RETURN.
0000:               24 *
0000:               25 MOVEBLK EQU  0          MOVE DATA BLOCK TO RAM.
0002:               26 MOVEINT EQU  2          MOVE INTEGER AND MONITOR TO RAM.
0004:               27 MOVEDOS EQU  4          MOVE DOS TO RAM AND INITIALIZE.
0006:               28 DOJSR   EQU  6          DO MOTHERBOARD SUBROUTINE.
0008:               29 GOMRBRD EQU  8          GO TO MOTHERBOARD.
000A:               30 MBRDRST EQU  10         DO THE MOTHERBOARD RESET.
000C:               31 LOADFP  EQU  12         LOAD APPLESOFT PROGRAM.
000E:               32 RUNFP   EQU  14         RUN APPLESOFT PROGRAM.
0010:               33 LOADINT EQU  16         LOAD INTEGER PROGRAM.
0012:               34 RUNINT  EQU  18         RUN AN INTEGER PROGRAM.
                    35 ********************************************************************************
0000:               36 *
0000:               37 *      GENERAL EQUATES
0000:               38 *
0026:               39 PRISLOT EQU  $26        STORAGE FOR PRIMARY SLOT.
0027:               40 SECSLOT EQU  $27        STORAGE FOR SECONDARY SLOT.
002C:               41 IIEFLG  EQU  $2C        MSB RESET IF IIE; SET IF II.
002D:               42 QLMAP   EQU  $2D        BITMAP OF QL SLOTS.
003A:               43 SRCL    EQU  $3A        INDIRECT SOURCE.
003B:               44 SRCH    EQU  $3B
003C:               45 LENL    EQU  $3C        INDIRECT LENGTH.
003D:               46 LENH    EQU  $3D
003E:               47 DSTL    EQU  $3E        INDIRECT DESTINATION.
003F:               48 DSTH    EQU  $3F
0045:               49 ACC     EQU  $45        SAVE/RESTORE EQUATES
0046:               50 XREG    EQU  $46
0047:               51 YREG    EQU  $47
0048:               52 STATUS  EQU  $48
0049:               53 SPNT    EQU  $49
```

FIGURE 11-1

```
00D8:           54 ONERR   EQU  $D8        APPLESOFT ONERR FLAG.
0112:           55 HRDSOFT EQU  $112       HIGH RAM CONTROL BYTE.
0204:           56 SAVPNT  EQU  $204       SAVE POINTER AREA.
020A:           57 SAVCTRL EQU  $20A
AA5F:           58 DOSCMDX EQU  $AA5F      DOS COMMAND INDEX.
B73A:           59 VCLDDOS EQU  $B73A      DOS COLD START VECTOR ($B73B).
C081:           60 QLCTRL  EQU  $C081      QL CONTROL REGISTER.
                61 ************************************************************
0000:           62 *
0000:           63 *      GET SLOT EQUATES
0000:           64 *
0000:           65 CHIP0   EQU  $00        00000000 CHIP 0 ON.
0018:           66 QLOFF   EQU  $18        00011000 QLOFF: CHIP0.
0020:           67 CHKNUM  EQU  $20        NUMBER OF FIND SLOT CHECKS.
0040:           68 GSCL    EQU  $40        GET SLOT C PARAMETER.
0041:           69 GSCH    EQU  $41
0042:           70 GSEL    EQU  $42        GET SLOT E PARAMETER.
0043:           71 GSEH    EQU  $43
C006:           72 SLTXROM EQU  $C006      IIE SOFT SWITCH.
C007:           73 INTXROM EQU  $C007      IIE SOFT SWITCH.
C00A:           74 INT3ROM EQU  $C00A      IIE SOFT SWITCH.
C00B:           75 SLT3ROM EQU  $C00B      IIE SOFT SWITCH.
CFFF:           76 CLRROM  EQU  $CFFF
                77 ************************************************************
0000:           78 *
----- NEXT OBJECT FILE NAME IS FIDCOPY ROUTS.OBJ0
FF00:           79         ORG  $FF00
FF00:           80 *
                81 ************************************************************
FF00:           82 *
FF00:           83 * KATALOG ENTRIES START HERE.
FF00:           84 *
FF00:82         85 FIDK    DFB  $82        CONTROL-B (BINARY).
FF01:00 E0      86         DW   $E000      SOURCE.
FF03:4E 12      87         DW   $124E      LENGTH.
FF05:03 08      88         DW   $0803      DESTINATION.
FF07:C6 C9 C4   89         ASC  'FID'
FF0A:90         90 COPYK   DFB  $90        CONTROL-P (PRIMARY).
FF0B:98 FF      91         DW   N.RESET    BASE ADDRESS OF MOVE COPY ROUT.
FF0D:00 00      92         DW   $0000      MEANINGLESS LENGTH
FF0F:00 00      93         DW   $0000      MEANINGLESS DESTINATION
FF11:C3 CF D0   94         ASC  'COPYA'
FF14:D9 C1
FF16:81         95 HELPK   DFB  $81        CONTROL-A (FP).
FF17:CA FA      96         DW   $FACA      SOURCE.
FF19:B5 03      97         DW   $03B5      LENGTH.
FF1B:00 00      98         DW   $0000      MEANINGLESS DESTINATION.
FF1D:D1 D5 C9   99         ASC  'QUIK      LOADER/QLOS HELP'
FF20:CB A0 CC
FF23:CF C1 C4
FF26:C5 D2 AF
FF29:D1 CC CF
FF2C:D3 A0 C8
FF2F:C5 CC D0
```

FIGURE 11-2

```
FF32:86
FF33:4E F2
FF35:2B 01
FF37:A0 02
FF39:79 F3
FF3B:51 07
FF3D:4A
FF3E:6A
FF3F:6A
FF40:6A
FF41:29 E0
FF43:8D 0A 02
FF46:60
FF47:80 40 20
FF4A:10 08 04
FF4D:02
FF4E:
FF53:A9 18
FF55:9D 81 C0
FF58:20 5D FF
FF5B:D0 F6
FF5D:
FF5D:
FF5D:
FF5D:
FF5D:
FF5D:8D 06 C0
FF60:8D 0B C0
FF63:A9 00
FF65:85 40
FF67:85 42
FF69:A9 C1
FF6B:85 41
FF6D:A9 E1
FF6F:85 43
FF71:A0 20
FF73:B1 40
FF75:D1 42
FF77:D0 19
FF79:88
FF7A:D0 F7
FF7C:A5 41
FF7E:AB
FF7F:0A
FF80:0A
FF81:0A
FF82:0A
```

```
100 ************************************************************
101         DFB  $B6        CONTROL-F TERMINATES KAT RECORD.
102 ************************************************************
103 OBJPARM DW   $F24E      SOURCE
104         DW   $012B      LENGTH
105         DW   $02A0      DESTINATION
106 ************************************************************
107 CPYPARM DW   $F379      SOURCE
108         DW   $0751      LENGTH
109 ************************************************************
110 INVERT  LSR  A          CHANGE 000ABXXX TO XXX00000.
111         ROR  A
112         ROR  A
113         ROR  A
114         AND  #$E0
115         STA  SAVCTRL
116         RTS
117 ************************************************************
118 MAP     DFB  $80,$40,$20,$10,$8,$4,$2

119 ************************************************************
120         DS   $FF53-*    SKIP TO $FF53.
121 OFFLP   LDA  #QLOFF
122         STA  QLCTRL,X   TURN OFF THE QL.
123 RTSLOC  JSR  GETSLOT    THIS INSTRUCTION AT $FF58 (RTS).
124         BNE  OFFLP
125 ************************************************************
126 *
127 *FIND SLOT NUMBER BY COMPARING CNXX TO ENXX FOR EACH
128 *SLOT.  START WITH SLOT 7.  USR MUST BE RESET FOR
129 *SEARCH TO BE EFFECTIVE IN II OR IIE.
130 *
131 GETSLOT STA  SLTXROM    ENABLE IIE I/O SELECTS.
132         STA  SLT3ROM
133         LDA  #0
134         STA  GSCL
135         STA  GSEL
136 TRYAGEN LDA  #$C1       START WITH SLOT 1.
137         STA  GSCH       SOURCE = $CN00.
138         LDA  #$E1
139         STA  GSEH       DESTINATION = $EN00.
140         LDY  #CHKNUM    GET NUMBER OF CHECKS TO VERIFY.
141 LOOKLP  LDA  (GSCL),Y
142         CMP  (GSEL),Y
143         BNE  NOTHERE    BRANCH IF QL NOT IN THIS SLOT.
144         DEY
145         BNE  LOOKLP
146         LDA  GSCH       THIS IS THE SLOT.
147         TAY
148         ASL  A          GET SLOTNUM TIMES $10 TO X.
149         ASL  A
150         ASL  A
151         ASL  A
```

FIGURE 11-3

```
FF83:AA          152          TAX
FF84:B9 86 FE    153          LDA  MAP-$C1,Y GET BIT MAP.
FF87:05 2D       154          ORA  QLMAP
FF89:85 2D       155          STA  QLMAP     SET BIT IN QLMAP.
FF8B:8D 0A C0    156          STA  INT3ROM   LEAVE INT3ROM AS NORMAL RESET DOES.
FF8E:            157 *
FF8E:            158 *NORMAL RESET FORCES SLTXROM.
FF8E:            159 *LEAVE 3ROM AND XROM AS WITH NORMAL RESET.
FF8E:            160 *
FF8E:AD FF CF    161          LDA  CLRROM    EXPANSION ROM OFF.
FF91:60          162          RTS
FF92:E6 41       163 NOTHERE  INC  GSCH
FF94:E6 43       164          INC  GSEH      CHECK IN NEXT SLOT.
FF96:D0 DB       165          BNE  LOOKLP    BRANCH ALWAYS TAKEN.
FF98:            166 *
FF98:            167 *EQU $C0 SHOULDN'T OCCUR; BOMB IF IT DOES.
FF98:            168 *
                 169 ************************************************************
FF98:            170 *
FF98:            171 * THIS IS N.RESET ROUTINE OF THIS CHIP.
FF98:            172 *
FF98:            173 * IF CARRY FLAG IS SET, MOVE COPY.OBJ
FF98:            174 * AND COPYA TO RAM AND EXECUTE COPYA.
FF98:            175 *
FF98:            176 * IF CARRY FLAG IS CLEAR, MOVE INTEGER
FF98:            177 * AND DOS THEN COLD START DOS.
FF98:            178 *
FF98:            179 *
FF98:08          180 N.RESET  PHP            SAVE POWER UP STATUS.
FF99:20 3D FF    181          JSR  INVERT
FF9C:28          182          PLP
FF9D:B0 1C       183          BCS  DOCOPY    BRANCH IF NOT POWER UP.
FF9F:A0 02       184          LDY  #MOVEINT  CARRY CLEAR SO DO POWER UP.
FFA1:20 E7 FF    185          JSR  GOCHIPO   MOVE INTEGER.
FFA4:A0 04       186          LDY  #MOVEDOS  MOVE DOS.
FFA6:20 E7 FF    187          JSR  GOCHIPO
FFA9:A9 1B       188          LDA  #$1B
FFAB:8D 5F AA    189          STA  DOSCMDX   SET DOS COMMAND INDEX FOR NO HELLO.
FFAE:85 D8       190          STA  ONERR     MSB RESET FIXES FP ONERR FLAG.
FFB0:A9 B7       191          LDA  #<VCLDDOS DOS COLD START IS $B73B.
FFB2:48          192          PHA
FFB3:A9 3A       193          LDA  #>VCLDDOS
FFB5:48          194          PHA
FFB6:A0 08       195          LDY  #GOMRBRD
FFB8:4C E7 FF    196          JMP  GOCHIPO   GO COLD START DOS.
                 197 ************************************************************
FFBB:            198 *
FFBB:            199 * DOCOPY
FFBB:            200 *
FFBB:            201 * ROUTINE TO MOVE COPY.OBJ0 AND COPYA AND EXECUTE.
FFBB:            202 *
FFBB:            203 * RUN COPY IF NOT POWER UP.
FFBB:            204 *
FFBB:A0 05       205 DOCOPY   LDY  #5
```

FIGURE 11-4

```
FFBD:B9 33 FF    206 OBJLP    LDA  OBJPARM,Y MOVE COPY.OBJ0 SRC/LEN/DST
FFC0:99 3A 00    207          STA  SRCL,Y    PARAMETERS TO PAGE 0 LOCATIONS.
FFC3:88          208          DEY
FFC4:10 F7       209          BPL  OBJLP
FFC6:A0 00       210          LDY  #MOVEBLK
FFC8:20 E7 FF    211          JSR  GOCHIPO   MOVE COPY.OBJ0 TO RAM.
FFCB:A0 03       212          LDY  #3
FFCD:B9 39 FF    213 COPYALP  LDA  CPYPARM,Y MOVE COPYA SRC/LEN
FFD0:99 3A 00    214          STA  SRCL,Y    PARAMETERS TO PAGE 0 LOCATIONS.
FFD3:99 04 02    215          STA  SAVPNT,Y  ALSO PLACE IN SAVE POINTER AREA.
FFD6:88          216          DEY
FFD7:10 F4       217          BPL  COPYALP
FFD9:A0 0E       218          LDY  #RUNFP
FFDB:4C E7 FF    219          JMP  GOCHIPO   GO RUN COPYA
                 220 ************************************************************
FFDE:            221          DS   $FFE7-*   SKIP TO $FFE7.
FFE7:A6 26       222 GOCHIPO  LDX  PRISLOT   GET QLOS SLOT NUMBER.
FFE9:AD 0A 02    223          LDA  SAVCTRL
FFEC:9D 81 C0    224          STA  QLCTRL,X  GO TO CHIP 0.
FFEF:4C 98 FF    225          JMP  N.RESET   DO N.RESET ROUTINE OF THIS CHIP.
FFF2:            226          DS   3
FFF5:60          227          RTS
FFF6:            228          DS   2
FFF8:00 FF       229          DW   FIDK      FIRST KATALOG LOCATION.
FFFA:FB 03       230          DW   $3FB      NMI VECTOR
FFFC:            231 *
FFFC:            232 * DON'T WORRY ABOUT RESET AND IRQ VECTORS.
FFFC:            233 *

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

FIGURE 11-5

```
?  45 ACC
 FFCD COPYALP
?  06 DOJSR
 FF00 FIDK
   41 GSCH
?FF16 HELPK
?C007 INTXROM
?  0C LOADFP
?  0A MBRDRST
 FF98 N.RESET
 FF53 OFFLP
   2D QLMAP
?  12 RUNINT
 C00B SLT3ROM
   3A SRCL
?  46 XREG
```

```
?  00 CHIPO
?FF0A COPYK
 AA5F DOSCMDX
 FF5D GETSLOT
   40 GSCL
?0112 HRDSOFT
 FF3D INVERT
?  10 LOADINT
   00 MOVEBLK
 FF92 NOTHERE
   D8 ONERR
   18 QLOFF
 020A SAVCTRL
 C006 SLTXROM
?  48 STATUS
?  47 YREG
```

```
   20 CHKNUM
 FF39 CPYPARM
?  3F DSTH
 FFE7 GOCHIPO
   43 GSEH
?  2C IIEFLG
?  3D LENH
 FF73 LOOKLP
   04 MOVEDOS
 FFBD OBJLP
   26 PRISLOT
?FF58 RTSLOC
 0204 SAVPNT
?  49 SPNT
?FF69 TRYAGEN
```

```
 CFFF CLRROM
 FFBB DOCOPY
?  3E DSTL
   08 GOMRBRD
   42 GSEL
 C00A INT3ROM
?  3C LENL
 FF47 MAP
   02 MOVEINT
 FF33 OBJPARM
 C081 QLCTRL
   0E RUNFP
?  27 SECSLOT
?  3B SRCH
 B73A VCLDDOS
```

FIGURE 11-6

```
   00 MOVEBLK
?  06 DOJSR
   0E RUNFP
   20 CHKNUM
   2D QLMAP
?  3D LENH
   41 GSCH
?  46 XREG
   D8 ONERR
 AA5F DOSCMDX
 C00A INT3ROM
 FF00 FIDK
 FF39 CPYPARM
?FF58 RTSLOC
 FF92 NOTHERE
 FFCD COPYALP
```

```
?  00 CHIPO
   08 GOMRBRD
?  10 LOADINT
   26 PRISLOT
   3A SRCL
?  3E DSTL
   42 GSEL
?  47 YREG
?0112 HRDSOFT
 B73A VCLDDOS
 C00B SLT3ROM
?FF0A COPYK
 FF3D INVERT
 FF5D GETSLOT
 FF98 N.RESET
 FFE7 GOCHIPO
```

```
   02 MOVEINT
?  0A MBRDRST
?  12 RUNINT
?  27 SECSLOT
?  3B SRCH
?  3F DSTH
   43 GSEH
?  48 STATUS
 0204 SAVPNT
 C006 SLTXROM
 C081 QLCTRL
?FF16 HELPK
 FF47 MAP
?FF69 TRYAGEN
 FFBB DOCOPY
```

```
   04 MOVEDOS
?  0C LOADFP
   18 QLOFF
?  2C IIEFLG
?  3C LENL
   40 GSCL
?  45 ACC
?  49 SPNT
 020A SAVCTRL
?C007 INTXROM
 CFFF CLRROM
 FF33 OBJPARM
 FF53 OFFLP
 FF73 LOOKLP
 FFBD OBJLP
```

FIGURE 11-7

```
SOURCE FILE: QLOS EQUATES
0000:              1 ************************************************************
0000:              2 *                                                          *
0000:              3 *              quikLoader CHIP 0 ROUTINES                  *
0000:              4 *                                                          *
0000:              5 *                     BY JIM SATHER                        *
0000:              6 *                                                          *
0000:              7 *            COPYRIGHT SEPTEMBER 19, 1983                  *
0000:              8 *                                                          *
0000:              9 ************************************************************
0000:             10 *
0000:             11 *
0001:             12 K64      EQU  1          64K OR 128K ASSEMBLY.
0000:             13 *
0000:             14 *
0000:             15 HELLO    EQU  $0         RUN HELLO AFTER DOS INIT FLAG.
0000:             16 CHIP0    EQU  $00        00000000 CHIP 0 ON.
0008:             17 CHIP0.1  EQU  $08        USED IN 128K ASSEMBLY.
000E:             18 CHIP6    EQU  $0E        00001110 CHIP 6 ON.
000F:             19 CHIP7    EQU  $0F        00001111 CHIP 7 ON.
001B:             20 QLOFF    EQU  $1B        00011000 QLOFF: CHIP 0.
001B:             21 NOHELLO  EQU  $1B        NO RUN HELLO AFTER DOS INIT.
0020:             22 CHKNUM   EQU  $20        NUMBER OF II.IIE & FIND SLOT CHECKS.
0020:             23 WNDLFT   EQU  $20
0021:             24 WNDWDTH  EQU  $21
0024:             25 CH       EQU  $24
0025:             26 CV       EQU  $25
0026:             27 PRISLOT  EQU  $26        STORE PRIMARY SLOT NUMBER.
0027:             28 SECSLOT  EQU  $27        STORE SECONDARY SLOT NUMBER.
002C:             29 IIEFLG   EQU  $2C        MSB RESET IF IIE; SET IF II.
002D:             30 QLMAP    EQU  $2D        BITMAP OF SLOTS WITH QLS.
002E:             31 SAVSTAK  EQU  $2E        SAVE STACK LOCATION.
002F:             32 IO3FLAG  EQU  $2F        SAVE SLOT3ROM SWITCH STATUS.
0030:             33 SAVCTRL  EQU  $30        SAVE QL CONTROL WORD.
003A:             34 SRCL     EQU  $3A        INDIRECT SOURCE.
003B:             35 SRCH     EQU  $3B
003A:             36 TOPKAT   EQU  $3A        NUMBER OF TOP KATALOG ENTRY.
003B:             37 KATCNT   EQU  $3B        COUNTS KATALOG ENTRIES.
003C:             38 LENL     EQU  $3C        LENGTH OF MOVES.
003D:             39 LENH     EQU  $3D
003C:             40 SAVEMAP  EQU  $3C        SAVE SHIFTED QL MAP DURING KATALOG.
003D:             41 LINNUM   EQU  $3D        KATALOG LINE NUMBER BEING PRINTED.
0000:             42 *
0000:             43 *0 = NO PRINT YET
0000:             44 *1-24 = PRINTING
0000:             45 *30 = SCROLL FORWARD ONE LINE.
0000:             46 *$C0 = GET ROUTINE FOR EXECUTION.
0000:             47 *
003E:             48 DSTL     EQU  $3E        INDIRECT DESTINATION.
003F:             49 DSTH     EQU  $3F
003E:             50 DELTA    EQU  $3E        CHANGE IN TOPKAT DURING SCROLL.
003F:             51 NAMELEN  EQU  $3F        DISPLAY PARAMETERS FLAG.
0000:             52 *
0000:             53 *16 = DISPLAY PARAMETERS.
```

FIGURE 12, PAGE 1

```
0000:             54 *36 = DON'T DISPLAY PARAMETERS.
0000:             55 *
0040:             56 GSCL     EQU  $40        GET SLOT C PARAMETER.
0041:             57 GSCH     EQU  $41
0040:             58 MOD256   EQU  $40        KATALOG WRAP COUNTER.
0041:             59 R.LFLAG  EQU  $41        RUN/LOAD FLAG DURING KATALOG.
0042:             60 GSEL     EQU  $42        GET SLOT E PARAMETER.
0043:             61 GSEH     EQU  $43
0042:             62 KBSL     EQU  $42        KATALOG BASE ADDRESS.
0043:             63 KBSH     EQU  $43
0045:             64 BFFRFLG  EQU  $45        DOS BUFFER AVAILABLE FLAG.
004C:             65 HIMINTL  EQU  $4C        INTEGER HIMEM.
004D:             66 HIMINTH  EQU  $4D
0067:             67 FPBOPL   EQU  $67        FP BEGINNING OF PROGRAM.
0068:             68 FPBOPH   EQU  $68
0069:             69 VARL     EQU  $69        START OF FP SIMPLE VARIABLES.
006A:             70 VARH     EQU  $6A
0080:             71 RONWOFF  EQU  $80        HIGH RAM READ ON, WRITE OFF.
0081:             72 ROFFWOK  EQU  $81        HIGH RAM READ OFF, WRITE OK.
0083:             73 RONWOK   EQU  $83        HIGH RAM READ ON, WRITE OK.
00AF:             74 EOPL     EQU  $AF        FP END OF PROGRAM.
00B0:             75 EOPH     EQU  $B0
00CA:             76 INTBOPL  EQU  $CA        INTEGER BEGINNING OF PROGRAM.
00CB:             77 INTBOPH  EQU  $CB
00D8:             78 ONERR    EQU  $D8        FP ONERR FLAG.
0200:             79 GLNBFFR  EQU  $200       GET LINE BUFFER
0204:             80 SAVPNT   EQU  $204       SAVE SRC/LEN/DST AREA.
0208:             81 SAVDSTL  EQU  $208
0209:             82 SAVDSTH  EQU  $209
03CF:             83 DOSWARM  EQU  $3CF       USED FOR WARM START AFTER CATALOG.
03EA:             84 CNCTDOS  EQU  $3EA       CONNECT DOS ROUTINE.
03F4:             85 PWRBYTE  EQU  $3F4       POWER UP RESET BYTE.
03F8:             86 CTRLY    EQU  $3F8       CONTROL-Y VECTOR.
0111:             87 BUFFER   EQU  $111       RAM ROUTINE AREA.
0112:             88 HRDSOFT  EQU  BUFFER+1   HIGH RAM CONTROL LOCATION.
0115:             89 KATBFFR  EQU  BUFFER+4   KAT ROUTINE AREA.
AA5F:             90 DOSCMDX  EQU  $AA5F      DOS COMMAND BEING EXECUTED.
C000:             91 KEYBRD   EQU  $C000
C006:             92 SLTXROM  EQU  $C006      IIE SOFT SWITCH.
C007:             93 INTXROM  EQU  $C007      IIE SOFT SWITCH.
C00A:             94 INT3ROM  EQU  $C00A      IIE SOFT SWITCH.
C00B:             95 SLT3ROM  EQU  $C00B      IIE SOFT SWITCH.
C010:             96 AKD      EQU  $C010      ANY KEY DOWN?
C010:             97 KBDSTB   EQU  $C010
C015:             98 READCX   EQU  $C015      READ CXROM SOFT SWITCH.
C017:             99 READC3   EQU  $C017      READ C3ROM SOFT SWITCH.
C030:            100 SPEAKER  EQU  $C030
C058:            101 AN0OFF   EQU  $C058
C05A:            102 AN1OFF   EQU  $C05A
C05D:            103 AN2ON    EQU  $C05D
C05F:            104 AN3ON    EQU  $C05F
C061:            105 OPENAPL  EQU  $C061
C062:            106 SLDAPL   EQU  $C062
C081:            107 QLCTRL   EQU  $C081      QUICK LOADER CONFIGURATION REGISTER.
```

FIGURE 12, PAGE 2

```
C081:           108 WRTONLY EQU  $C081      HIGH RAM CONTROL.
CFFF:           109 CLRROM  EQU  $CFFF
FB2F:           110 INIT    EQU  $FB2F
FBB4:           111 GOTOCX  EQU  $FBB4
FC42:           112 CLREOP  EQU  $FC42      CLEAR END OF PAGE.
FC58:           113 HOME    EQU  $FC58
FD0C:           114 RDKEY   EQU  $FD0C      MONITOR READ KEY ROUTINE.
FD8B:           115 CROUT1  EQU  $FD8B      CLREOL AND CR.
FDDA:           116 PRBYTE  EQU  $FDDA
FDED:           117 COUT    EQU  $FDED
FE84:           118 SETNORM EQU  $FE84
FE89:           119 SETKBD  EQU  $FE89
FE93:           120 SETVID  EQU  $FE93
FF3F:           121 RESTORE EQU  $FF3F      RESTORE 6502 REGISTERS.
FF4A:           122 SAVE    EQU  $FF4A      SAVE 6502 REGISTERS.

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

FIGURE 12, PAGE 3

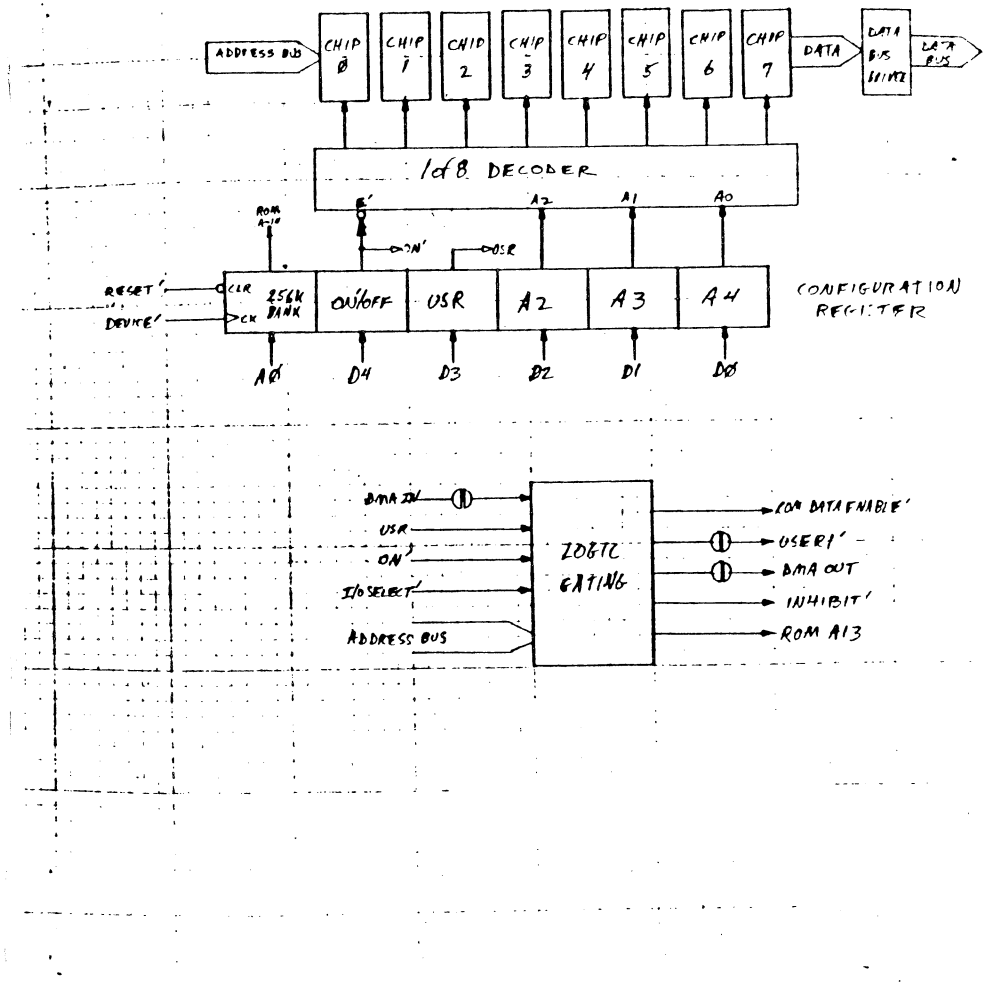


Figure 13 - quikLoader Block Diagram

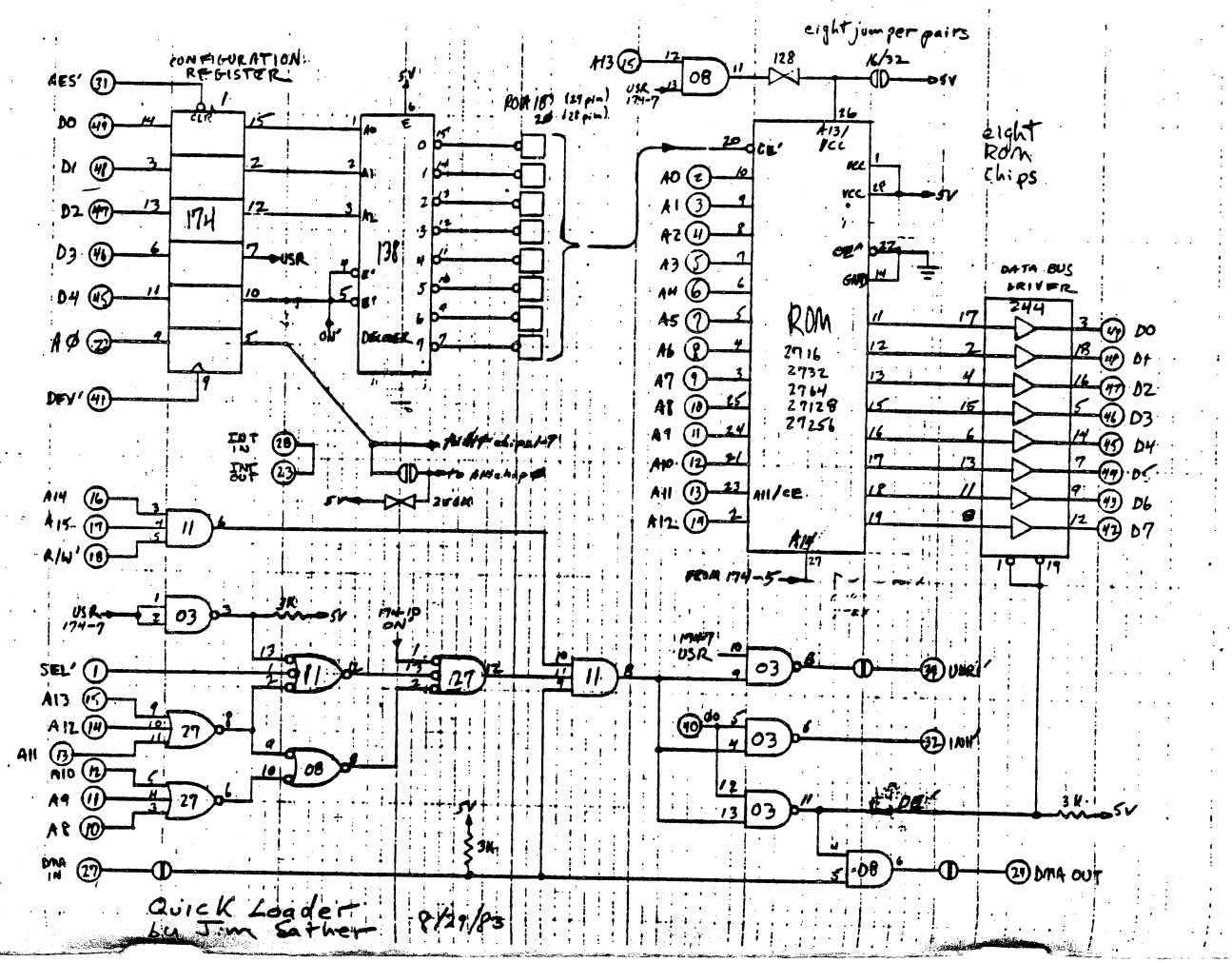


Figure 14 - quikLoader schematic