# quikLoader™

# USERS AND PROGRAMMERS MANUAL

**NOTE-** The information on programming contained herein is for advanced uses. For most programming needs, use the **PROGRAMMERS AID** program, packed with the quikLoader.

**SOUTHERN CALIFORNIA RESEARCH GROUP**
Post Office Box 593
Moorpark, CA 93020

Telephone (805) 529-2082

The **quikLoader**™ is covered under Southern California Research Group's ten day trial period. If you have purchased this unit directly from S.C.R.G., you may return it within 10 days if not satisfied with its performance. Contact your dealer if this unit was purchased from a dealer.

This unit is also covered under our six month warranty. If the unit fails to function within that time, return it, together with dated proof of purchase, to us. We will, at our option, repair or replace the unit.

The disclaimer below is required as part of our license agreement with APPLE COMPUTER, INC.

**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.



The quikLoader was designed by Jim Sather. The quikLoader operating system was written by Jim Sather.

# quikLoader INSTRUCTIONS

The quikLoader is a firmware peripheral card designed for use in the Apple ][, Apple ][ Plus, and Apple //e computers. It will hold up to 512K (524288) bytes of data and programs. Because the data and programs are stored in EPROM, they are instantly available to the computer operator at any time.

The quikLoader provides the user an unprecedented 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.

## 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), 27256 (32K), or 27512 (64K) EPROM or equivalent masked ROM. There are four solder pad jumpers associated with each socket, two split circle pads, a "bow tie", and a straight conducter. These pads must be changed for a socket if you elect to install a 24 pin chip (2716 or 2732) in it, or a 28 pin 27512. When configuring these jumpers for a socket, there is always electrical contact across one jumper or the other. Here are the possible configurations:

| BOW TIE | LOWER SPLIT CIRCLE | IC TYPES |
|---|---|---|
| MADE | DESOLDERED | 2764, 128, 256, and 512 |
| CUT | SOLDERED | 2716, 2732, 2764 |

In addition, to use the 27512, it is necessary to cut the trace (inside the white square) and bridge the upper split circle. Directions for doing all these operations are on the back of the quikLoader.

The quikLoader is delivered with all sockets (except zero) 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.

For Chip zero only, you are limited to using the 27256 or 27512 EPROMS. The card is configured for using the 27256. Make

the above mentioned changes for using the 27512.

## 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 is 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

## (quikLoader ON/OFF switch)

Some cards and software are incompatible with the presence of the quikLoader. We have included an method of turning off the quikLoader via an external switch. If you look on the front of the quikLoader, between the 74S74 and the 74LS174, you will see two unmarked solder pads. Turning the card on its back, you will see a "bow-tie" to the left of these pads. To install the switch, cut the bow tie, and solder two wires to the pads. These wires should go to a SPST (Single Pole, Single throw) switch. If your software or hardware does not appear to work with the quikLoader plugged in, you may turn off the card for as long as necessary. The card can be turned on or off without turning off the computer, but you will have to remember, in most cases, to do a Z-RESET after turning the card on.

## 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 interpreting 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.

### 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 absence of a chip in socket 6, the QLOS supplied with the quikLoader wil do its power-up routine. This 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 inserting a chip in socket 6 with a special power-up routing (see the PROGRAMERS AID). 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.

### 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 keypress 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 at (805) 529-2082 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 supplied with the quikLoader has the program COPYA as its n-reset function.

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 catalogging, 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.

If you use the PROGRAMMERS AID to program your EPROMS, all programs will show as P files.

```
A B S1 C0 FID
B P S1 C0 COPYA
C A S1 C0 QUIK LOADER/QLOS HELP
D
                .
                .
                .
V
W
```

The Katalog Display

Performing a Q-reset with a quikLoader installed in your Apple results in a screen display showing the quikLoader resident files. The above figure shows the katalog display of a standard

quikLoader. 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, but not run.

Z Toggle the parameter display. This enables the display of source, length, and destination information as an aid for persons programming QLOS 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 forward, the display will scroll to the last file then stop. When scrolling backward, 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 QLOS 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

NOTE - FOR MOST PURPOSES, PROGRAMMING SHOULD BE DONE BY USING THE PROGRAMMERS AID. THIS SECTION IS INCLUDED FOR VERY SPECIALIZED USES, AND REQUIRES SOME PROGRAMMING ABILITY.

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 short-wave 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.

CAUTION - Short-wave ultraviolet radiation can be harmful. Be sure to use erasers with adaquate shielding.

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 developement, 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 |
| 27512 | 64K bytes / 512K 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. At this writing (March, '85) 27256s are the least cost per bit. Using anything smaller then this is wasteful of the quikLoader. Prices of the 27512 are expected to further decrease. 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, 27256 and 27512 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. Southern California Research Group has developed an inexpensive PROM burner that will burn 27nn series EPROMS up to 27512s. This PROM burner is available throug dealers, or direct from SCRG.

## quikLoader Addressing Ranges

The quikLoader addressing range is from $C100 to $FFFF. This is a range of 128K bytes minus 256. 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.

## The Katalog Record

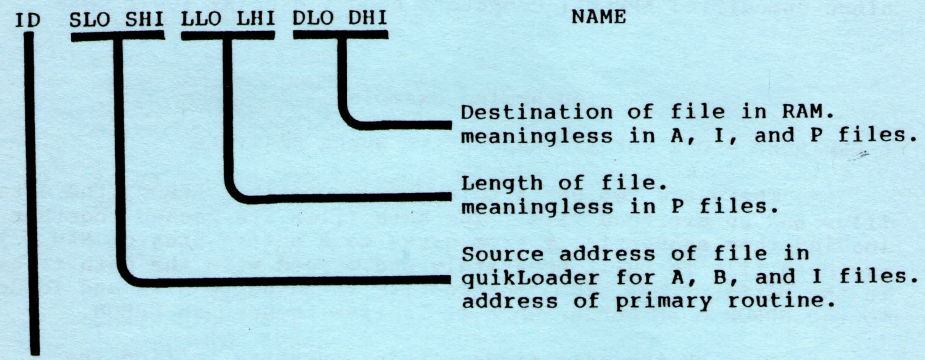
The katalog record of a QLOS EPROM is made up of individual katalog entries, stored sequentially starting at some base address. The base address of the katalog record of a QLOS EPROM must be stored at $FFF8 and $FFF9 of the EPROM. If there is no katalog record on a QLOS EPROM, this must be identified 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 2 shows the format of the katalog record and Figure 3 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 2 - The Katalog Record Format




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 3 - Katalog Entry 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. The following codes are used in the "NAME" portion of the katalog file. For example, to have the letter "A" appear in the katalog name, use a code of C1:

| char | code | char | code | char | code | char | code | char | code |
|---|---|---|---|---|---|---|---|---|---|
| A | C1 | B | C2 | C | C3 | D | C4 | E | C5 |
| F | C6 | G | C7 | H | C8 | I | C9 | J | CA |
| K | CB | L | CC | M | CD | N | CE | O | CF |
| P | D0 | Q | D1 | R | D2 | S | D3 | T | D4 |
| U | D5 | V | D6 | W | D7 | X | D8 | Y | D9 |
| Z | DA | 0 | B0 | 1 | B1 | 2 | B2 | 3 | B3 |
| 4 | B4 | 5 | B5 | 6 | B6 | 7 | B7 | 8 | B8 |
| 9 | B9 | : | BA | ; | BB | < | BC | = | BD |
| > | BE | ? | BF | ! | A1 | " | A2 | # | A3 |
| $ | A4 | % | A5 | & | A6 | ' | A7 | ( | A8 |
| ) | A9 | * | AA | + | AB | , | AC | - | AD |
| . | AE | / | AF | (space) | A0 | | | | |

Note - you may also use lower case, but it is not recommended since unmodified APPLE][ computers cannot display lower case.

## Programming Examples

A REMINDER - USE THE PROGAMMERS AID WHERE POSSIBLE

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 $FFF8
            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.

## 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. Since the PROGRAMMERS AID make learning all this unnecessary except in unusual cases, we have not decided not to include this information in the manual. However, if you need it, this information is available from us at no charge. Please contact us. Our address is at the end of the manual.

## 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:
Primary routines are always entered with USR high, and return from chip 0 routines is always with USR high. Also notice 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. A table of these routines, and further information, are available from us at no charge.

NOTE - 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.

## 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, directly inhibiting I/O SELECT' and I/O STROBE' so the quikLoader can respond to $C100-$CFFF 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.

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.

## Commercial Developement 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 300K 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".

Any company interested is making their programs available on quikLoader compatible EPROMS are encouraged to contact us.

## 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.

On the PROGRAMMERS AID disk is a utility program which will expunge the DOS image from disks. (Program name is EXPUNGE). 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). Southern California Research Group cannot accept responsibility of loss of data that might occur when using this program.

To operate the EXPUNGE program, just BRUN the 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.