2699 Skokie Valley Road • Highland Park, IL 60035 • (312) 433-7550

TABLE OF CONTENTS

MICRO LAB PRESENTS
The RELOCATING LINKING LOADER

by

M. J. Hatlak

INTRODUCTION

A Relocating Linking Loader is a powerful utility program that allows the machine language programmer to maintain a library of files which can be selectively loaded, moved to a particular place in memory, and linked into a larger program or set of programs. Machine code programs are position dependent, i.e., they will not normally function if moved from the location at which they were originally designed to work, as specified in the ORG statement from within an assembler source file. Therefore, it is necessary to use a Relocating Linking Loader to achieve successful transporting of machine code modules to any place in memory other than the originally intended location. The Micro Lab Relocating Linking Loader was originally designed for use with Language Plus+. This enables owners of Language Plus+ to select from their library of amper routines only the modules that they desire, and combine them into a single BINARY file on the disk suitable for use with their own APPLESOFT programs. This will provide extra power while incurring the minimum memory penalty.

There are actually two separate functions being performed by the Relocating Linking Loader in the overall process. They are:

RELOCATION

This process involves constructing a memory map based on the number and size of all the modules to be included in the output file. Then comes the loading of the modules followed by a scanning process through each module to find all addresses and internal pointers that must be changed. Then all addresses and pointers are recalculated and rewritten into the file.

LINKING

This process involves only modules that make references to modules other than themselves. For example, a MATH program module may call a subroutine in a SINE/COSINE module. The Relocating Linking Loader program will isolate the call (JSR) and plug in the newly computed address of the SINE/COSINE module based upon where the module now resides in memory. All of the modules used to construct one large program might need to share some variables contained in yet another module. The Linker can resolve all the addressing problems involved with this kind of operation.

WHICH ASSEMBLER TO USE?

In order for this process to work, it is necessary to have certain information available about the modules to be relocated and linked. This is the job of the Assembler program. The Micro Lab Relocating

Linking Loader was designed to be used in conjunction with Rel files generated by the Apple DOS Toolkit Editor/Assembler. This Assembler is capable of generating files in a relocatable format usable by the Relocating Linking Loader with a few limitations. Apple's Editor/Assembler has a few problems that must be avoided in order to insure correct operation of the Relocating Linking Loader. Follow these guidelines carefully when writing source code to be used later in relocation.

1) The Assembler contains a bug that precludes proper address calculations. To fix this you must modify the ASSM program and place it back on the disk. Place your DOS Toolkit Disk in the drive and type:

```
]BLOAD ASSM
]CALL -151
*18F5:03  <--This will be 04 and must be changed!
*3DOG
]BSAVE ASSM ,A$1200,L$22FB
```

2) The first line in your source code should be the REL psuedo-op.

3) You must use the ORG psuedo-op before defining ANY labels. The value you ORG at is inconsequential but required.

4) Block all ENTRY and EXTRN labels up front before any other code or declarations. Place all EXTRN labels first, followed by any ENTRY labels. If you do not declare the Link labels before the Assembler performs an address calculation, the symbol number count will get trashed and results will be unpredictable!

5) You may not use more than 255 EXTRN and ENTRY labels in any one program as the Assembler assigns symbol numbers to them as a one byte entry.

6) After specifying a label as an ENTRY type label, you may not establish its location utilizing the EQU psuedo-op. For some reason the Assembler will not handle this correctly, and the Relocating Linking Loader will not be able to relocate it properly.

7) The DDB psuedo-op may not relocate properly. Beware of using it.

8) You MUST NOT perform math upon or split labels that have been designated as ENTRY or EXTRN as the Assembler has no way to handle this and will not flag it as an error.

SAMPLE SOURCE FILE

```
 1            REL              <- Should be first
 2            ORG $1000        <- Done before any labels
 3            EXTRN OUTPUT     <- Extrn labels  first
 4            EXTRN INPUT
 5            ENTRY START      <- Entry labels next
 6 COUT       EQU $FDED        <- Then regulars
 7 *
 8 * TEST FILE
 9 *
10 START      LDA INPUT
11            STA OUTPUT
12            STA HOLD
13            RTS
14 HOLD       DFB 0
```

This short program uses both kinds of LINK labels. Line 11 references OUTPUT, and line 10 references INPUT. Both exist in some other module. Also, other modules may call this program by referencing START, which will point at line 10. Line 14 creates a one byte variable that is internal to the program. It will need to be relocated but not linked. The ORG statement on line 2 tells the Assembler to start the program at $1000, however this value is really meaningless.

USING THE RELOCATING LINKING LOADER

The process requires that certain information be available to the Relocating Linking Loader before the operation can begin. The system will need to know the following:

1) NAMES of all INPUT files and their locations. (R files)
2) NAME of the OUTPUT files and its location.
3) Where the output file will reside in memory.

Using a Relocating Linking Loader is not difficult, but the user should think out ahead of time just what it is he wants his output file to contain, and where he wants the file to be located. It would be best to have on hand a list of the files to be linked prior to starting the process. You might also wish to transfer the INPUT modules onto a single disk if they exist on different disks. You may use the FID program that accompanies your System Master. Note that all the INPUT files must be on line at LINK time, that is; they must all be resident on a disk in a drive so that the system can access them. After this initial preparation, boot the Relocating Linking Loader disk. It will display the hires logo and the copyright screen will be presented. When you are ready, you may press any key to continue. The system will then load the system driver files and you are ready to proceed. At this point you may remove the Relocating Linking Loader program disk from the drive.

TARGETTING MODE

The Micro Lab Relocating Linking Loader provides for two targetting modes for relocation. They work as follows:

1) TARGET UP - You will specify the target address and the Relocating Linking Loader will build the output file starting at that address and go up towards the top of memory.

2) TARGET DOWN - You will specify the target address, and the Relocating Linking Loader will build the output file so that it starts at an internally calculated address and continues up to but does not include the target address. The output file will sit just below the target address. This works well in a situation where you wish to set your code up against an upper limit. It is useful if you wish to place a module just below the DOS buffers at $9600 for use with APPLESOFT.

After you specify the target address, the Relocating Linking Loader will then prompt you to enter the target address in hexadecimal. Afterwards the program will ask you to acknowledge the target address. It will print it out for you in both hexadecimal and decimal for you to see. Press the <SPACE BAR> to accept it or <ESC> to reject it.

OUTPUT FILE

You will be prompted for the OUTPUT file name. This is the name to be assigned to the BINARY file that will be built from the INPUT files. You will then be prompted to enter SLOT & DRIVE. This is the location where you wish to have the OUTPUT file placed.

INPUT FILES

Next you must tell the Relocating Linking Loader the names of the INPUT files you will be using. These files must be of the R type when viewed in the catalog. You may LINK up to 50 files. You will then need to type in the filenames of all the INPUT files that you wish to include in the output file. Note that the files will be linked in the same order that you enter them. The method is semi-automatic and quite easy to use. The program will prompt you for a slot & drive location. Enter the location of the INPUT files and the program will scan the disk and list out any R type files found on that drive. Then you simply ask for them by number. When you have selected all of the files from that particular disk, enter 0 to select the next drive or press <RETURN> to review the files on the presently selected disk. If you enter 0 the system will again prompt you for the next slot & drive. Respond with RETURN twice to indicate no more INPUT files. The program will now proceed. Pressing RESET will restart the program at any time, but it should NOT be used while the disk drives are on!

EXECUTION

PASS ONE

When all this has been completed, the program will ask that you place your disks into the applicable drives and press the <SPACE BAR> to begin Pass One of the procedure. Remember that all INPUT files must be

on line at this time in order for the Relocating Linking Loader to complete its task. If you should mismount a disk in any of the drives, the program will halt and ask you to remount the errant disk. If you no longer wish to have that disk scanned, you may place another disk in the drive, or you may press <ESC> to tell the Relocating Linking Loader to skip this drive and proceed to the next slot/drive in the list.

As the Relocating Linking Loader program identifies INPUT files, it will scan them for LINK labels contained in the External Symbol Directory. It will then display them on the bottom of the screen as they are located and placed into the symbol table. If the Relocating Linking Loader is unable to locate all of the INPUT files, it will halt and tell you so. You will also be asked for permission to restart the scanning process, thus giving you a chance to place the correct disks into the drives. If you select not to restart, then the program will go back to the beginning.

COMPUTER MEMORY MAP

This phase of the program will lay out the INPUT files' location in memory for correct relocation and linking.

COMPUTE ENTRY POINTS

This phase computes and places the new values for all the LINK labels based on the memory map. After all these operations have been brought to a successful conclusion, Pass Two will begin.

PASS TWO

At this point the Relocating Linking Loader has computed the memory map, resolved all the Link symbols, and must now create the OUTPUT file. It will transfer each INPUT file one by one into the OUTPUT file and perform all relocating and linking at this time. The program will display on the screen the name of the INPUT files as they are relocated, and it will flag each relocatable field as it is encountered. The screen will show the following information:

1) The field orientation as
<NORMAL> or <REVERSE>
(This identifies the field
as either low/high order
form or high/low order.)

2) The field type as
<8 BIT> or <16 BIT> or <EXTRN>
(This indicates whether or
not the field is 2 bytes
long and whether it was
an external reference.)

3) The offset as #12345
(This is a decimal number
equal to the byte offset
into the output file, and is
relative to its new
origin that points to
this relocatable field.)

The SUMMARY

At the end of Pass Two the program will print a summary of the operation either to the screen or to the printer. This summary is a representation of the memory map filled in with all the filenames, addresses, and labels used by each module. If you choose to go to the printer, you will be asked for its slot and then for a control string. If you need to send control characters to your printer or interface card to properly initialize it, enter them at this point. Most of the information in the summary of a numerical nature is printed first in decimal and then printed again, inside parentheses, in hexadecimal for those who may need it. The summary should be saved as it may prove invaluable if problems develop later that need to be diagnosed. The program will now print the summary containing the following information:

```
          OUTPUT FILE:filename ,A$----,L$----
           TARGET ADD:----- ($----)
                  MODE:TARGET UP or DOWN
          # INP FILES:-

           FILE:filename ,Sslot,Ddrive     <-One for each INPUT file.
----------------------------------------
          START:----- ($----)
            LEN:----- ($----)
            END:----- ($----)

                LABEL:label name

                 TYPE:ENTRY or EXTRN
                  ADD:----- ($----) if ENTRY type
                SYM #:-             if EXTRN type

         FILE:filename ,Sslot,Ddrive


       --------------------------------
       etc...
```

ERRORS that may occur

NOT RELOCATABLE TYPE FILE - Indicates that during pass one the Relocating Linking Loader located an INPUT file that was not of type R.

REMOUNT DISK - The Relocating Linking Loader encountered an I/O error while attempting to read a disk.

UNABLE TO LOCATE FILES - Pass one was unable to locate one or more of the files specified as INPUT. The program will list for you all the files that could not be found.

LABEL FORMAT ERROR - During pass two while attempting to decode an ESD entry, the Relocating Linking Loader encountered a symbol whose format was incorrect. Thus, indicating that the Assembler may have failed to create a proper symbol flag byte.

DUPLICATE LABEL - The Relocating Linking Loader has found an ENTRY label that is not unique. The label names assigned to ENTRY type labels must not be duplicated across INPUT modules designated for linking. EXTRN labels need not be unique.

SYMBOL TABLE OVERFLOW - The process has exceeded the number of labels that can be held in the symbol table.

TARGETTING MODE ERROR - The target mode specification is illegal and the program must be re-started.

UNIDENTIFIED LABEL ERROR - A label was encountered in the RLD that was not specified in the ESD.

LINK ERROR - An EXTRN reference was made to a label that does not exist in any of the modules as an ENTRY type label.

ILLEGAL VALUE ERROR - A numeric value was requested and an illegal response was entered, such as alpha data or negative numbers, etc.

The ERROR PAGE

If the Relocating Linking Loader program encounters an unidentifiable error, it will present the ERROR PAGE. This is a listing of the important internal variables. If you intend to request help, it would be wise to copy down the information you see on the error page prior to contacting Micro Lab, as it will be vital to correctly diagnose your problem.

LANGUAGE PLUS+ OPERATIONS

This section is designed for Language Plus+ owners who wish to create custom amper modules for their use with BASIC programs.

First you must decide which amper routines you would like to include and on which disks they will reside. Then it would be beneficial to transfer them over to a single blank disk using FID or some other like program. This will greatly simplify the process. Please read

carefully! After selecting the amper routines you will be linking, you MUST also include the two files called HEADER and TRAILER! The HEADER module must be the FIRST file linked, followed by all the amper routine modules, and then TRAILER must be the last file linked. This order is IMPERATIVE and must be followed exactly. The number of amper routine modules that you place between HEADER and TRAILER is up to you, but they must sit between these two modules for proper operation. At this point you should now have all your INPUT files (amper modules, HEADER, and TRAILER) on a disk or disks that the system can read. Follow these steps:

1) Boot the Relocating Linking Loader program disk. When the copyright text screen appears, press any key to continue.

2) When asked for the targetting mode, select 1 Locate From Target Down.

3) When asked for the target address, enter 9600. Then the program will ask you to verify the target address to make certain that you have typed it correctly. Press the <SPACE BAR> if it appears correct, or press <ESC> to reject it and start over. NOTE: If you intend to alter the DOS MAXFILES from your BASIC program you will need to use a different target address than 9600. See the table below for the alternate values and substitute the appropriate ones.

4) Now enter the name that you would like to call the OUTPUT file. This will be the BINARY machine code program that can be BRUN from your BASIC program.

5) Now you must decide which of the amper routines you will be linking together. First tell the system which slot/drive to look at and then select the files by number. Enter 0 when you are done. Don't forget to add HEADER and TRAILER to the list. Enter HEADER first, then all the amper routines, and finally TRAILER. Make sure you follow this order!

6) Now remove the Relocating Linking Loader program disk and place the disk(s) into the correct slot/drive(s). When this is done press the <SPACE BAR>, and the program will now perform the relocation and linking process for you. The OUTPUT file will be on the disk that you specified when asked for the OUTPUT name.

7) When the process is completed, you will be asked where you would like the summary to be printed. It would be wise to use a printer if you have one. In the future this can aid in debugging the module you have created. If you choose the printer, you must specify its SLOT number. For those who must send out control characters, there is a place for that as well. If you do not need to send out control characters, just press <RETURN>. The summary will now print out.

8) You will now be asked if you wish to restart the program or end it. The choice is yours.

An example of the Linking ORDER

```
HEADER          preamble
AMPER ROUTINE
AMPER ROUTINE
AMPER ROUTINE
AMPER ROUTINE
AMPER ROUTINE

 ...
 ...
TRAILER         postamble
```

MAXFILES TABLE

MAXFILES 1	-	$9AA6
MAXFILES 2	-	$9853
MAXFILES 3	-	$9600
MAXFILES 4	-	$93AD
MAXFILES 5	-	$915A
MAXFILES 6	-	$8F07
MAXFILES 7	-	$8CB4

WRITING YOUR OWN AMPER ROUTINES

If you wish to write your own routines to be linked into some of the Language Plus+ routines, you must follow some simple protocol. Your routine should be set up to answer to a two letter integer variable call. It must expect and check the ASCII values for these two letters when it receives control from the HEADER routine. If the call variable is not the correct one, the routine must pass control to the routine immediately behind itself. The ASCII values are passed to your routine in the X and Y registers. For example, &AB% will leave the ASCII equivalent of 'A' in the X register and the ASCII equivalent of 'B' in the Y register. You will check these and execute only if they match your call letters. If they do not match, do a JMP to the first byte directly behind your routine. This allows the next amper routine to attempt to match its call letters. Do not alter X and Y.

If your routine is called, and you wish to pass back a value to BASIC in the calling variable, you must be able to find where this variable is. To do this you will specify two EXTRN labels from within your program. One will be RTNVARL and the other RTNVARH. Then you can access them from within your program to gain a pointer to the two byte integer calling variable in normal Applesoft storage format. The LINKER will fill in the correct locations at LINK time, so most of the work is done for you. There is also a location called STKSAV which you may access that contains the stack pointer when the ampersand jump was executed. If you wish to use it you may specify an EXTRN label called STKSAV and the LINKER will resolve this for you as well.

That is all there is to it! Be careful not to disturb any of Applesoft's pointers or variables. Co-existing with another program

requires a great deal of caution.

MISC NOTES

The Relocating Linking Loader will scan and copy from the INPUT files, but it does not modify them in any way. They remain intact on the disk and may be used again in subsequent OUTPUT files.

If the user wishes further information on REL type files, you may look in the back of the Editor/Assembler book that was provided with the DOS Toolkit. This will explain in detail what an R file looks like as well as the internals of the RLD (Relocation Dictionary) and ESD (External Symbol Directory).

The Relocating Linking Loader program is disk based. This means that you could conceivably link together a file that was bigger than the Apple's memory and place it on the disk. The only limitation on size is the number of LINK labels allowed in the Relocating Linking Loader symbol table.