A.P.P.L.E.

PRESENTS:

Big Mac.LC

LANGUAGE CARD SUPPLEMENT

Apple PugetSound Program Library Exchange

DISCLAIMER

This manual and the accompanying diskette are available only to members of Apple Pugetsound Program Library Exchange.

A.P.P.L.E. PRODUCTS ARE SOLD "AS IS." A.P.P.L.E. DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR USE.

Big Mac.LC

LANGUAGE CARD SUPPLEMENT

Copyright © 1982

by Glen Bredon

Table of Contents

Language Card Version of Big Mac

(Special features and differences from the standard version)

EXEC MODE

[R]ead

This reads text files into Big Mac. It works as described under "READER," *except that it is always an "append."* Thus type "NEW" in the editor if you don't want an append. If no file is in memory, then the name given will become the default file name. Appended reads will not do this. If the file contains lines longer than 255 characters, these will be divided into two or more lines by the READ routine. The file will be read only until it reaches HIMEM and will produce a memory error if it goes beyond.

[W]rite

This writes a Big Mac file into a text file instead of a binary file. It works as described under "WRITER." The speed of the READ and WRITE routines is approximately that of a BLOAD or BSAVE. The WRITE routine does a VERIFY after the write. If you type a space, or any character less than "@"[less than ASCII $C0], before the file name, then that character and the "T." prefix will not be sent to DOS by the READ and WRITE commands.

Ctrl C

Typed after "COMMAND:" following a C (for CATALOG), this will output the Exec mode prompt "%" and then will accept any EXEC command such as L for load. This permits you to issue an Exec mode command while the catalog is still displayed on the screen. This feature is also present on the standard version of BIG MAC, but was added too late to be documented. In addition, if control C is typed at the "CATALOG pause" point, printing of the remainder of the catalog is aborted.

EDITOR

The edit mode control functions may now be used in both Insert and Add mode, except that control R will have no effect, and the exit from Add mode acts as described. For example, hitting return will accept the entire line as it appears on the screen, and control Q will truncate the line at the cursor position.

During an edit, if a line becomes greater than 255 characters, it will be truncated and a new blank line created following it.

Control L acts in all modes as described under Add mode, that is, as a case toggle. To change the case of a word during edit, you can type control L and then copy over the word using the right arrow.

Hex-dec conversion

In command mode, if you type a positive or negative decimal number, the hex equivalent is returned. If you type a hex number, prefixed by "$," then the decimal equivalent is returned. All commands accept hex numbers, which is mainly convenient for the HIMEM and SYM commands

Total key input

Control O, besides enabling the insertion of control characters, also allows you to type those characters not normally available on the Apple keyboard. Using the following conversion table, type a Ctrl O, followed by a character from the first column to produce the character shown opposite in the second column:

Type	to produce
<	**Ctrl** _
>	**Ctrl** \
K	[
L	\
M	]
N	∧
O	_
k	{
l	⋮
m	}
n	~
o	■ **(rubout)**

(Note that if you are using a shift key modification, then depending on which one you have, shift M may give upper case M and you will have to use [Ctrl O]M to get the right bracket.)

Commands

STRIP

This deletes all comments (initial "*" lines or following ";") from source. This was designed as a last resort for dealing with extremely large files, but with the great versatility of the language card version of Big Mac, you should never have to use it.

UPPER

This converts (permanently) all lower case text in comments to upper case. (See "UPCON.") As with the STRIP command, the other features of the language card version leave little utility for this command, except perhaps in converting a file for someone else who does not have the language card version, and does not have a lower case display chip.

TEXT

This converts *all* spaces in a source file to inverse spaces. The purpose of this is for use on "text" files, so that you do not have to remember to zero the tabs before printing such a file. This conversion has no effect on anything except tablulation.

FIX

This undoes the effect of TEXT. It is also a substitute for the "FIX" program (which cannot be used with the language card version). Thus it is recommended that the command FIX be used on all files from external sources, after which the file should be saved. This command is not quite the same as the standard version's FIX program, in that it does more, but its effect is essentially the same. (Note that the TEXT and FIX routines are written in SWEET 16 and thus are somewhat slow. Several minutes may be needed for their execution on large files.) FIX or an edit will truncate any lines longer than 255 characters.

SYM

The language card version places the symbol table on the language card itself (in bank 1 of $D000-$DFFF). This space is quite adequate for all but gigantic programs. In case this space is used up however, the SYM command gives you a means to direct the assembler to continue the symbol table in another area. If you type SYM $9000, for example, and assemble the program, then if and when the symbol table uses up its normal space, it will be continued from $9000 to BASIC HIMEM. It must be noted that the SYM command will be canceled by a HIMEM command or by exit to EXEC mode and re-entry, thus set HIMEM prior to establishing a SYM address. The SYM address must be above HIMEM and below BASIC HIMEM. If the symbol table grows beyond the alloted space, then you will get a memory error during the first pass of assembly.

VIDeo

This is designed to select or deselect an 80 column board. The default condition can be selected using the configuration program. This is similar to the use of PR#in BASIC. *Do not use PR#* to select an 80 column board in slot 3 (for example) can be selected by typing, from the editor, VIDEO 3. It is deselected by VIDEO 0 or VIDEO $10 possibly followed by hitting the RESET key. The latter two forms both select the standard Apple screen, but VIDEO 0 will cause all lower case output to the screen to be converted to upper case, except for lower case in the source file, which will be converted to flashing upper case. (Output to a printer is never converted.) Thus, if you have a lower case adaptor, you will want to use VIDEO $10 (or VIDEO 16) instead of VIDEO 0 when selecting the Apple screen. If your 80 column card supports a software screen switch via an escape sequence, then you can switch to 40 columns by using this in the editor (or PR#0 [RESET]). This may have to be followed by a control X. This software switch will do a "VIDEO $10" and thus must be followed by a VIDEO 0 if you do not have a lower case adaptor.

ASSEMBLER

Pseudo-Ops

TR

This has the same effect in the assembler as does the editor's TR command. Thus, TR or TR ON limits object code printout to 3 bytes per line and TR OFF resets it to print all object bytes.

CHK

This places a checksum byte into object code at the location of the CHK opcode (usually at the end of the program).

PUT

PUT FILENAME, (drive and slot parameters accepted in standard DOS syntax) will read the named file (with the "T." prefix appended)-and "insert" it at the location of the opcode. (Note: "insert" refers to the effect on assembly of the source. The file itself is actually placed just following the main source.) Text files are required by this facility in order to insure memory protection. You will get a memory error if a PUT file goes beyond HIMEM. These files are in memory only one at a time, so a very large program can be assembled using the PUT facility.

There are two restrictions on a PUT file. There cannot be marcro DEFINITIONS inside a put file (they must be in the main source). Also, a PUT file may not call another one by a PUT (that is, this is not linking). Of course, linking can be simulated by having the "main program" just contain the macro definitions and call in turn all the others by the PUT opcode. Any variables (e.g.,]LABEL) may be used as "local" variables. The usual local variables]1 through]8 may be set up for this purpose using the VAR opcode.

The PUT facility provides a simple way to incorporate much used subroutines, such as MSGOUT or PRDEC, in a program.

VAR

This is just a convenient way to equate the variables]1-]8. "VAR 3;$42;LABEL" will set]1 = 3,]2 = $42, and]3 = LABEL. This is designed for use just prior to a PUT. If a PUT file uses]1-]8 (except in >>> lines for calling macros), then there must be a previous declaration of these.

KBD

This allows a label to be equated from the keyboard during assembly. Its syntax is: LABEL KBD.

SAV

SAV FILENAME,(drive and slot parameters accepted) will save the current object code under the specified name. This acts exactly as does the EXEC mode object saving command, but it can be done several times during assembly.

This pseudo-opcode gives you a means of saving portions of a program having more than one ORG. It also enables the assembly of extremely large files. After a save, the object address is reset to the last specification of OBJ, or to HIMEM by default.

The save command sets the address of the saved file to its correct value. For example, if your program contains three SAV commands, then it will be saved in three pieces. When BLOADed later, they will go to the correct locations, the third following the second and that following the first.

Together, the PUT and SAV opcodes make it possible to assemble extremely large files.

A minor incompatibility

In the language card version, the multiple operands of the PMC (or >>>) opcode must be separated by semicolons instead of commas as in the standard version. The program CONVERT will convert old source files to the new form. (With your source in memory, just type BRUN CONVERT from the EXEC mode’s command after catalog.) This was changed because the macro facility now accepts literal data. Thus the assembler now accepts the following type of macro call:

```
        DO 0
MUV     MAC
        LDA ]1
        STA ]2
        <<<
        FIN
        >>> MUV.(PNTR),YDEST
        >>> MUV.#3FLAG,X
```

It will also accept:

```
        DO 0
PRINT   MAC
        JSR SENDMSG
        ASC ]1
        BRK
        <<<
        FIN
        >>> PRINT.!“quote”!
        >>> PRINT.‘This is an example’
        >>> PRINT.“So’s this, understand?”
```

LIMITATION: If such strings contain spaces or semicolons then they *must* be delimited by quotes single or double). Also literals such as >>> WHAT.“A” must have the final delimiter (this is only true in macro calls or VAR statements, but it is good practice in all cases).

Nestable Macros
In the language card version, macros may be nested to a depth of 15. For nesting, macros *must* be defined with DO condition off.

Here is an example of a nested macro in which the definition itself is nested. (This can only be done when both definitions end at the same place.)

```
TRDB  MAC
      >>> TR.]1 + 1]2 + 1
 TR   MAC
      LDA ]1
      STA ]2
      <<<
```

In this example >>> TR.LOC;DEST will assemble as:

```
      LDA LOC
      STA DEST
```

and >>> TRDB.LOC;DEST will assemble as:

```
      LDA LOC +
      STA DEST + 1
      LDA LOC
      STA DEST
```

A more common form of nesting is illustrated by these two macro definitions (where CH = $24):

```
POKE  MAC
      LDA #]2
      STA ]1
      <<<

HTAB  MAC
      >>> POKE.CH]1
      <<<
```

Non-zero page addressing:
Opcodes that can accept either zero page or non-zero operands, such as LDA ADRS, can now be forced to use the non-zero page form by simply appending any character (the colon for example) other than "D" to the opcode. Thus LDA: ADRS will assemble as non-zero page. This is occasionally useful.

Multiple ORGs:
Multiple ORGs no longer defeat the object code if saved, will load to the address of the FIRST ORG used. Thus, for example, if you want to have a program that will BLOAD to $1000 but is written to be BRUN at $9000, you can begin it with:

```
      ORG $1000
      ORG $9000
```

and it will do just that. This is useful for programs that must do some housekeeping before being moved to their final location, and there are a number of other uses for this ability.

GENERAL:
Reentry after exit to BASIC is made by the "ASSEM" command. A BRUN BIG MAC or a disk boot will also provide a warm reentry and will not reload BIG MAC if it is already there. This may be forced by BRUN BOOT ASM which would then be a cold entry, "destroying" any file in memory.

After exit to the monitor via the editor's MON command, reentry can be made by any of control Y, control C or control B (the latter two because the Apple thinks BIG MAC is BASIC). A direct reentry to the editor is possible by typing 0G [rtn]. This reentry, unlike the others, will use the zero page pointers instead of the ones saved upon exit, so you must be sure that they have not been altered.

Memory organization is somewhat different in the language card version. While for ordinary sized files this is not of concern to the user, it is important to understand certain constraints for the handling of large files. HIMEM (which defaults to $8000) is an upper limit to the source file. It is also an upper limit for PUT files. If a memory error occurs during assembly, indicating a PUT line, this means the PUT file exceeded HIMEM and thus that HIMEM will have to be increased. The default ORG and OBJ addresses equal the present value of HIMEM (instead of $8000 as in the non-language card version). It is illegal, in the language card version, to specify an OBJ address that is less than HIMEM except that a page 3 address is allowed. (If you use a page 3 OBJ address, you must be careful that the file will not write over the DOS jumps at $3DO-$3FF as the assembler will NOT check this for you.) If, during assembly, the object code exceeds BASIC HIMEM (or the SYM address if one has been specified), then the code will not be written to memory, but assembly will appear to proceed and its output sent to the screen or printer. Your only clue that this has happened, if it is not intentional on your part, is that the object save command is disabled in this event. Thus, if you want a listing for a very long file without actually creating code, you can assemble over DOS and up.

Source is placed at $901 (partially because a disk boot writes over page 8). Both the editor and assembler use page 8 for various purposes.

On exit to BASIC or to the MONITOR, the pointers at $A-$F are saved at $E00A-$E00F.

USE WITH SHIFT KEY MODIFICATIONS

The language card version supports all hardware shift key modifications. The configuration program will establish the modification that you want supported. BIG MAC is smart enough to know if the modification actually exists in the Apple you are using and defeats the modification if it is not there. Thus it can be used on another machine without reconfiguration.

USE WITH 80 COLUMN BOARDS

Most, but not all, 80 column boards can be supported by the language card version. You may use the VIDEO command to select the 80 column board. To have the board selected upon boot, use the configuration program. (Then you may switch to the 40 column screen as described under the VIDEO command.)

If your board does not support inverse, then control characters in source will show as ordinary capital letters instead of inverse letters as with boards that support inverse. You can use the editor's Find command to search for particular control characters if you wish to verify their presence or absence, or simply switch over to the normal Apple screen.

If, for example, your copy of BIG MAC has been configured to support an 80 column card in slot 3 and there is no card in that slot, then BIG MAC will recognize this and will defeat the 80 column provision. Thus there is no need to reconfigure for use on another computer.

BIG MAC will *not* support any board that does not recognize the "POKE 36" method of tabbing. (As far as we are aware, this means simply that it will not support older versions of the FULL VIEW card.)

When in the editor, BIG MAC takes almost total control of input and output. Thus the effect of typing a control character will be as described in this manual and *not* as described in the manual for your 80 column card. For example, control L will not blank the screen, but is the case toggle. Control A, which acts as a case toggle on many 80 column cards, will not do this in BIG MAC's edit mode and simply produces a control A in the file line. (It may work in command mode but is not recommended.)

THE CONFIGURE ASM PROGRAM

This program allows you to make several minor modifications to BIG MAC's default conditions. It first allows you to change the "UPDATE SOURCE" character searched for at the entry to the assembler, the editor's wild card character, and the number of symbol fields printed per line in the symbol table printout. It also permits you to specify whether you want to have an 80 column board supported and, if so, which slot it is in.

It allows you to specify a hardware shift key modification. Any such modification can be supported. However, if your modification is the type that enables direct input of lower case (as with the VIDEX keyboard enhancer) instead of providing a memory location to be tested (as with the "game button 2" modification), then the default at the start of each line will be lower case rather than upper case and control L will function as a case lock toggle.

You may specify whether you have a lower case adapter. This will affect the condition on boot if you have not elected to have an 80 column board selected then. (It may always be defeated from the editor using the VIDEO command, however, so this is only to select the initial condition.)

Finally, you can save the configured version to another, or the same disk. There is no reason to keep the original version since you can always return to it by reconfiguration.

At the end of the configuration program you are given the opportunity to transfer the boot program "BIG MAC" to another disk. This is just a convenient way of transferring that program. You can also use FID to copy BIG MAC or ASM.OBJ (the main program). DO NOT attempt to BLOAD or BSAVE BIG MAC from the keyboard; this will not work! When booted under normal DOS, COPYA or any standard copy program may be used to copy the entire disk.

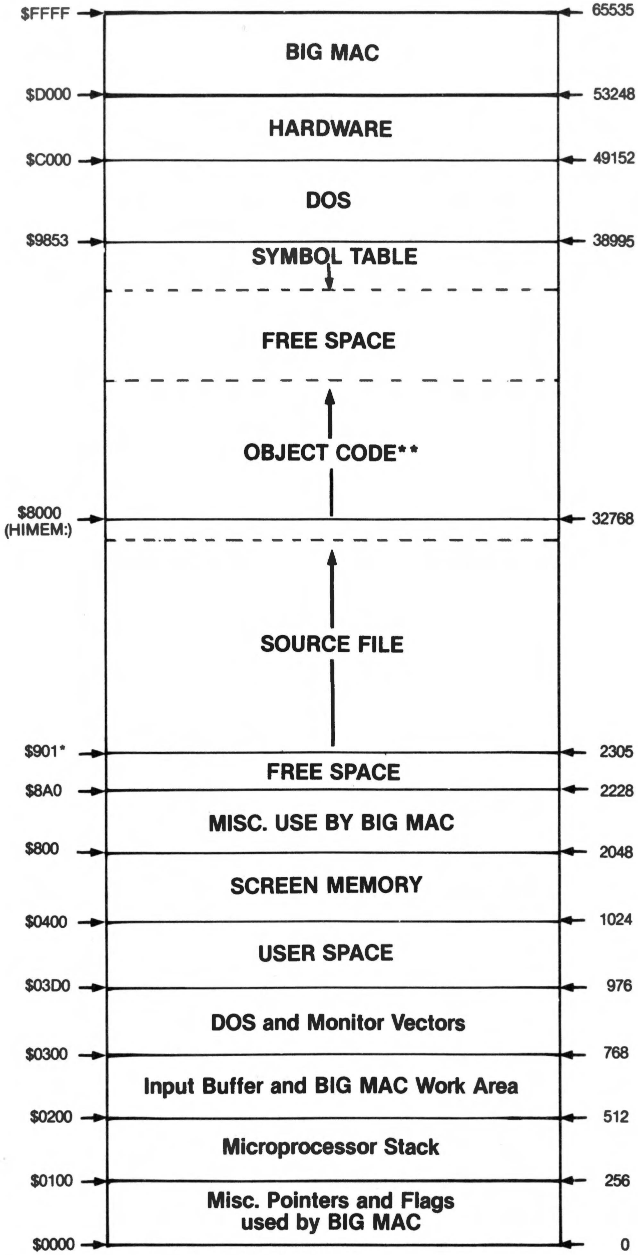
The configuration program should be BRUN only when the standard Apple screen is in use.

SPECIAL CONSIDERATIONS

The small program called BIG MAC on the disk may be BRUN. It is designed, however, to be a BOOT program. For this purpose, the DOS on the disk is modified to run a binary program upon boot. To put a copy of this DOS on another disk you may do the following: Rename BIG MAC,XXX, reboot the disk, insert a blank disk and type INIT BIG MAC, then DELETE BIG MAC, and finally rename XXX,BIG MAC on the original disk. Then use the configuration program, or FID, to transfer the files BIG MAC and ASM.OBJ to the new disk.

The boot program BIG MAC does several things. First it checks for a language card. Next it makes some minor changes to DOS that will be required by the main program (in particular, the INIT command will be disabled), and then it checks whether ASM.OBJ is already there. If not, then ASM.OBJ is loaded into the card and jumped to. A result of this is that a boot is always a warm boot, that is, it will NOT erase a source file that is in memory at the time. (For this it is imperative that you DO NOT convert the disk to a master disk! Leave it a slave!) If, for some reason, you wish to force reloading of the language card, you must either turn off the computer or BRUN BOOT ASM. The latter program is identical to BIG MAC except that it always loads ASM.OBJ and hence is a cold start, "erasing" any extant source file.

MEMORY MAP



* 900 also used by editor.
** OBJ and ORG default to value of HIMEN:

Symbol Cross-Reference

for BIG MAC

by Dale Waddell

One feature that is missing with the BIG MAC Macro-Assembler/TED is the ability to print a label cross reference listing. BIG MAC does have the ability to print a symbol table in both alphabetical and numerical order. The symbol tables are helpful to determine if a label exists in the program or if a label has been referenced at least once. If you have any need to determine how or where a label is used, the only choice is to use the FIND function in the editor.

Normally I place a "LST OFF" statement as the last statement of a program so the symbol tables will not be printed. I find the cross reference listing supplies everything in the symbol table except the numeric location of each label. For those of you that use macros, BIG MAC's symbol table does show macro names and labels defined within a macro. My cross reference treats macro names and variable names as if they are labels.

To run the cross reference program just follow these easy steps:

- have BIG MAC active
- have input source file loaded into memory
- if an object file is in memory, make sure that it is saved before running the cross reference program. The cross reference program uses all memory between the source file and DOS
- turn on the printer
- get to the EXEC MODE menu and press the "C" key
- after the catalog completes, enter BRUN BIG MAC.XREF
- after BIG MAC.XREF has collected all the information to be printed, it will ask the user to reply to the following message:

ENTER PRINTER SLOT # OR USER

The carriage return defaults to slot 1. A response that commences with "U" causes a JSR to $3F5, so that the printer driver routine can do its setup work. The default number of lines per page printed is 60. This may be modified by changing the contents of location $9BBA. If a formfeed at the end of each page is not desired, substitute $8D for the $8C found at locations $96FE and $9BC3, which will produce a carriage return instead. BSAVE parameters are:

BSAVE BIG MAC.XREF, A$93A0, L$6F7;

The cross reference program will print the entire cross reference listing and then return to BIG MAC's EXEC MODE menu. The cross reference program does not change the source file, therefore the source file in memory is still usable.

The cross reference program saves $D0-$FF at the beginning of execution and restores it at the end of execution. This is done so the cross reference can have some page zero work areas.

The BIG MAC diskette also contains a commented source file for those users who may be interested in studying and/or expanding BIG MAC.XREF. The following explanation about how information is stored in memory should assist the user in understanding more about how the program works.

LBLCHN - a zero page pointer to the first entry in the LABEL chain

Each entry in the label chain will contain the following information.

ADRNXT - pointer to the next entry in the LABEL chain

LLABEL - length of the label in this entry

CBYTE - Control byte where $80 means that the label is defined and $40 means that this is a duplicate label definition

WDEF - if $80 in CBYTE is ON this field will contain the line number where the label is defined

ADRREF - pointer to "where referenced" chain for this label. If this field is zero then no references for the label were found

LABEL - same length field as the content of CBYTE. If the first character is $FF this is a "dummy" entry that will always be the last entry in the chain. The reason for a "dummy" entry is to make it easier to merge new entries into the chain.

The "dummy" entry ensures that a new entry will never need merged onto the end of the label chain.

A separate "where referenced" chain exists for each LABEL chain entry that has non-zeros in WDEF. Each entry contains:

- pointer to next entry. If zero, this is the last entry.
- line number where the label is referenced.

F.P. CARD SLOT 4

A5B8: CØ

A5CØ: C1

Apple
PugetSound
Program
Library
Exchange

304 Main Ave. S.
Suite 300
Renton, Washington
98055
(206) 271-4514