(M)otorola (A)ssember and (C)o-resident (E)ditor for the MC6809

by Graham Trott

```
First  Release: 1 Mar  1982 with Version 1.00 Software
Second Release: 1 July 1983 with Version 2.30 Software
Third  Release: 15 Oct 1983 with Version 2.40 Software
Fourth Release: 10 Oct 1984 with Version 2.60 Software
```

(M)otorola (A)ssember and (C)o-resident (E)ditor for the MC6809

Windrush Micro Systems, Worstead Labs, N. Walsham, Norfolk, NR28 9SA, England

(M)otorola (A)ssember and (C)o-resident (E)ditor for the MC6809

by Graham Trott

TABLE OF CONTENTS

1.0 INTRODUCTION

This is the fourth major release of MACE for the MC6809. With this release we have removed the appendices on the MC6809 from the main manual and have printed them in a separate A5 sized booklet entitled 'MC6809 PROGRAMMING REFERENCE GUIDE FOR ASSEMBLY LANGUAGE PROGRAMMERS' which should serve as a handy reference.

We have also added many new features to the co-resident editor and have improved the disk file handling. The assembler portion of MACE may now be called from the FLEX command line. In addition the assembler listing output may now be directed to the system terminal, printer and an output disk file simultaneously whilst creating the object file!

A C K N O W L E D G E M E N T

Our thanks to Neil Jarman for his efforts in improving the file handling capabilities and editor features of this this product.

MACE is a combined editor and assembler designed to enable the user of a FLEX system to edit, assemble and test programs of any size with the minimum of effort.

It is designed primarily for writers of small to medium sized system programs, where an interactive approach is often more useful than macro and conditional features, and facilitates an rapid edit-assemble-test cycle that is very valuable when in the primary program debugging phase.

Most system monitors provide rudimentary debug facilities. The system monitor is easily entered from within MACE for debugging exercises.

In many cases it is possible to have MACE, the source and object code and the Windrush D-BUG (or TSC DEBUG) tracers co-resident in memory at the same time, greatly speeding program development.

Larger programs are handled by a technique that allows multiple files to be assembled directly from source to object, thereby dispensing with the need for a linking loader. If you require macro capability or prefer using a link loader the TSC ASSEMBLER and TSC RELOCATING ASSEMBLER are recommended.

Note to beginners

MACE is quite forgiving of mistakes so there is no need to understand the whole of this manual (nor all the 6809 instructions); just type in what you think is right and MACE will help you correct any errors. Try just using the editor first until you're familiar with that. Before you assemble your first program read the section on error handling; that way you'll understand the messages MACE gives when it doesn't like what it sees.

We also offer a very easy to use single-step tracer called D-BUG which is an ideal aid to those who are not yet familiar with assembly language programming. Whilst it is not as powerful as the TSC DEBUG package it is infinitely easier to use and will locate most bugs a lot quicker.

2.0 THE MACE EDITOR

One of the strongest features of MACE is its built-in editor, which cuts out much of the loading and saving that makes using other editors and assemblers a time consuming business. The editor is partly responsible for the minimal memory requirements, achieved by encoding each mnemonic into one byte as the line is entered, thus making the source file up to 20% more compact than it would otherwise be (see notes below).

The editor is broadly similar to, and compatible with, the TSC text editing system, although the commands are not identical. Anyone familiar with the latter should have little difficulty in using the MACE editor.

The editor prints line numbers while listing the source file; these numbers are not part of the file, however, and are not saved on disc. It also allows editing of a line as it is being entered, by means of the following control characters:

BACKSPACE...moves the cursor to the left destructively one place.

CANCEL......erases the entire line.

ESCAPE......in the left column terminates the insert session.

RETURN......generates a new line.

The default key values supplied may not suit your terminal. The SETMACE program, described in section six, provides a convenient method of altering the keycodes MACE recognizes to those available on your terminal.

MACE will accept ANY text file that is stored on disk in the TSC TEXT EDITOR FORMAT. Many cursor oriented text editors/word processors, e.g. SCREDITOR III, save text in this format.

If you use an editor other than the TSC TEXT EDITOR or SCREDITOR III to enter your programs and have problems with MACE's editor don't blame us! The fault lies with the file format produced by your editor. The STYLOGRAPH disk file format is typical example of a non-standard format that will be absolutely useless to MACE unless you are very careful when you enter the program to ensure that a <CR> is present at the end of each and every line.

2.1 DESCRIPTION OF EDITOR COMMANDS

Each of the editor commands is described fully in the following paragraphs. This section groups the various editor commands by function. A command summary can be found at the end of this document.

Generally speaking the MACE editor does not support multiple command entry. Edit commands must be entered singly i.e. the command followed by a carriage return. There are a few exceptions to this rule which will be described as they are encountered.

NOTE 1: It is not practical to use the editor in MACE to prepare non assembly language text files as the editor will encode the source line into the appropriate assembler field as each line is entered.

NOTE 2: It is not possible to enter assembly language programs in lower case due to the encoding process MACE uses with the source lines.

EDITOR COMMAND SYMBOLS

The following symbols will be used as part of the definition of the editing and assembler commands:

<CR>	represents a carriage return.
<>	symbols are used to enclose a variable.
<NUMBER>	represents a decimal number such as 36 or 192, and which defaults to one if it is omitted.
<TARGET>	represents the decimal number of linesspecified by the command, and defaults to one if none is given.
<#TARGET>	represents the decimalline numberspecified by the command.
[]	symbols indicate that the enclosed data is optional and may be omitted, in which case a default value is usually supplied by MACE.

The ASSEMBLE (A) command may be called from within the editor or from the FLEX command line this command is described in its own section.

All of the commands outlined in the following pages are only active when the (#) prompt is present.

M O D E C O N T R O L

I

INSERT lines into the file. The editor will change its prompt from (#) to (+) to remind you that you are now in the insert mode. Every line you type from now on will be added to the file immediately ABOVE the current line. This may seem strange at first if you are used to the TSC editor, but it has the advantage that having added a line to the file the current line is still the same one as before. It also enables you to insert a line above line number 1, something which is very awkward with the TSC editor. The current line and the rest of the file will be moved to make room for the new line, so the file will always be in sequence, i.e. it is automatically re-numbered each time lines are added.

When you have finished adding lines, ensure that the cursor is at the left margin and press the <ESCAPE> key. You will then be returned to the editor command level, with the prompt restored to a (#). Your current line will now have a new (larger) number because of the extra lines inserted above it.

X

EXIT to FLEX. You will be prompted 'IS TEXT SECURE?' which must be answered 'Y' to enter FLEX. Any other response will return you to the editor.

M

MONITOR. Enter the ROM System Monitor. A warning message will be posted along with instructions on how to re-enter MACE through the warm start address.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                       *
*  NEVER RE-ENTER MACE AT THE COLD START ADDRESS $0000. *
*  Doing so will cause the existing file to be erased!  *
*                                                       *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

See the section on file start and file end markers at the end of this section of the manual for techniques to use in the event that you accidentally loose a file.

/<COMMAND>

Execute a FLEX command. Warning: Only use commands that reside in the Utility Command Area, or you may risk bombing MACE, with possibly disastrous results.

The only commands we recommend are: CAT, DIR, LIST, DELETE, RENAME, ZAP, TTYSET, and ASN. Using COPY, for example, is a definite no-no!

*

Toggle from formatted to unformatted mode and vice-versa. The startup mode is normally formatted mode but this can be changed by the SETMACE command. This command is used to provide compatibility with files that have been created on another editor and already have the mnemonic fields tabbed out.

L I N E   P O S I T I O N I N G   C O M M A N D S

<NUMBER>[COMMAND]

Make line <NUMBER> the current line, then execute the command (optional) that follows the number.

1 or ^

Go to the first line in the file.

B or !

Go to the bottom (/EOF) of the file.

+<NUMBER>

Move down <NUMBER> lines from the current position.

-<NUMBER>

Move up <NUMBER> lines from the current position. This command may be followed by the print command, i.e. -23P23<CR> would back up 23 lines and print 23 lines.

<CR>

Display the current line.

<ESCAPE>

Hitting the escape key will cause the next line in the file to be displayed and to become the current line. This provides a convenient method of 'stepping' through your file a line at a time. If you are already at the bottom of the file then you will get /EOF printed every time.

F I L E O R I E N T E D C O M M A N D S

N

NEW file. This command erases the file currently in memory to make room for a new file. MACE will prompt with "ARE YOU SURE?" to prevent you from inadvertently erasing your file.

The file is not actually deleted from memory when 'N' is used. The file is 'erased' by setting the end of file marker to beginning of the text buffer. See the section on file start and file end markers at the end of this section of the manual for techniques to use in the event that you accidentally loose a file.

P<TARGET>

PRINT a number of lines of the file on the terminal, starting at the current line. The last line printed becomes the current line. Examples:

```
    #P50<CR>    Print 50 lines, starting at the current line.

    #142P8<CR>  Print 8 lines, starting with line 142.

    #P<CR>      Print some more lines.
```

In the last example, the number of lines printed will be the same as that specified by the last P command. When you start up MACE, this number will be preset to one less than that given by the TTYSET DP count (see your FLEX manual). This facility allows you to scan your file N lines at a time, by pressing P then <CR> repeatedly.

D<TARGET> or D<#TARGET>

DELETE line(s). The first form will delete the requested NUMBER of lines from the file starting with the current line. Lines above the current line are unaffected and it does not matter if you specify a target that is beyond the bottom of the file.

The second form deletes all of the lines starting with the current line TO AND INCLUDING, the line number followed by the #

If D is typed by itself then only the current line is deleted. After deletion, the editor displays the new current line. Examples:

```
    #D15      Delete 15 lines, starting at the current line.

    #81D3     Delete 3 lines, starting at line 81. The line that was previously
              line 84 will become the current line, which will still be numbered
              81.

    #43D#72   Delete from line 43 to 72 inclusive. The line that was previously
              line 73 will become the current line.
```

CAUTION: Be very careful when deleting multiple lines as the file will automatically be renumbered after EACH line is deleted. When you wish to delete several lines simply start at the highest line number and work toward the lowest.

L I N E E D I T I N G

O<CHAR>

OVERLAY the current line. This command is useful when a change has to be made near the start of a line. MACE displays the line in question, then prompts immediately underneath with a > symbol. You can then type in a new line containing only the characters you wish to change, in their correct positions under the displayed line. If <CHAR> is omitted, then spaces typed in the new line indicate characters that should be left alone; if <CHAR> is supplied then it becomes the character that you type to leave the corresponding one in the original as it was. The following example first shows the current line which is then overlayed twice:

```
0123    IF COUNT > 5 THEN CHAR = 'X      /* TEST CASE */
    #O
0123    IF COUNT > 5 THEN CHAR = 'X      /* TEST CASE */
    >         =                ;
0123    IF COUNT = 5 THEN CHAR = 'X;     /* TEST CASE */

    #O-
0123    IF COUNT = 5 THEN CHAR = 'X;     /* TEST CASE */
    >------------------------------------SPECIAL CASE */
0123    IF COUNT = 5 THEN CHAR = 'X;     /* SPECIAL CASE */
    #
```

E

Edit the current line. This command causes MACE to display the line and to leave the cursor at the end, as if you had just typed it in but had not yet pressed <CR>. The line can then be altered by backspacing or by adding more text.

=<TEXT>

Delete the current line and put in its place the remainder of the command line. Example:

```
    #52=   REPEAT COUNT=COUNT-1 UNTIL COUNT=0;
    #52
0052    REPEAT COUNT=COUNT-1 UNTIL COUNT=0;
    #
```

G L O B A L   E D I T I N G

F<NUMBER>/<STRING>

FIND the next <NUMBER> occurrences of <STRING>, starting with the line following the current line. If <NUMBER> is omitted it defaults to one. Any delimiter can be used in place of the / symbol. Examples:

#F20/IF	Find the next 20 occurrences of IF.
#F,/What?/	Find the next occurrence of /What?/.
#^F!/HELLO/	Find every occurrence of HELLO from the top of the file (^) but excluding the top line, to the bottom of the file (!).

C<NUMBER>/<STRING1>/<STRING2>

CHANGE the next <NUMBER> occurrences of <STRING1> into <STRING2>, starting with the current line. Only the first occurrence of <STRING1> will be changed on any one line. Again any delimiter is allowed. Examples:

#C/THIS/THAT	Change THIS into THAT in the current line.
#93C;WILE;WHILE	Change WILE in line 93 into WHILE.
#^C!/THESE/THOSE	Change every occurrence of THESE to THOSE.

NOTE 1: Find and Change operate by setting up two buffers, one for <STRING1> and the other for <STRING2>, every time either command is called. If an incomplete command line is typed, only the specified data will be updated. For example, F23 instructs MACE to find the next 23 occurrences of the previously defined <STRING1>, while C8;VAR (note the missing second delimiter) will change the next 8 occurrences of VAR into whatever <STRING2> had been previously set to.

This facility simplifies making global changes where the same string occurs more than once on a line.

NOTE 2: <STRING> (in FIND) and <STRING1> (in CHANGE) may contain one or more "?" as "don't care" characters. For example, ^F!/VAR? finds all occurrences of VAR1, VAR2, VAR3 etc.

DISK FILE HANDLING

?

Print the name of the file last specified by in a Load, Save or Write command.

Q

Query the default file names. MACE prints a table of file names, for example as follows, where the command L=TEST has been issued.

```
    Present defaults are:
    ====================

    L/S = 1.TEST    .ASM
    R/W = 1.SCRATCH .SCR
    A:O = 1.TEST    .BIN
    A:L = 1.TEST    .OUT
    A:G = 1.TEST    .LIB
```

Whenever you specify a filename you can give any combination of drive number, filename and extension, and MACE will take whatever you omit from the current default for the command you are using. Using Save as an example:

```
    #S<CR>                 Save to 1.TEST.ASM
    #S=0<CR>               Save to 0.TEST.ASM
    #S=.TXT.0<CR>          Save to 0.TEST.TXT <--- (note the order)
    #S=JUNK.TXT<CR>        Save to 1.JUNK.TXT
    #S=0.JUNK<CR>          Save to 0.JUNK.ASM
    #S=0.JUNK.TXT<CR>      Save to 0.JUNK.TXT
```

Using the L/S (load and save) command will set up the default file name for the (L/S), (A:O), (A:L) and (A:G) commands. It will also set up the default drive and extension for the (L/S) command.

The default drive and extension for the (A:O), (A:L) and (A:G) commands are not altered by the (L/S) command.

The (A:O) command can alter its default drive and extension but does not have any effect on the other command defaults. The only way the (A:L) and (A:G) defaults can be altered is with the SETMACE command.

The (A:L) and (A:G) command can temporarily override the the defaults to produce an output file on a specific drive, with a specific name and with a specific extension if desired. The default drive, file name and extension are not altered however.

The READ/WRITE (R/W) command may alter any of the defaults according to the information supplied. For example:

```
    #W23<>                 Write to 1.SCRATCH.SCR
    #W23=0<CR>             Write to 0.SCRATCH.SCR (default drive now 0)
    #W23=TEMP<CR>          Write to 0.TEMP.SCR (default file name now TEMP)
    #W23=.TMP<CR>          Write to 0.TEMP.TMP (default extension now .TMP)

    #R=1.TEST.TXT          Read file 1.TEST.TXT (the default drive, file name,
                           and file extension will be updated accordingly.
```

D I S K F I L E H A N D L I N G

L[=<FILENAME>]

LOAD a disc file. The default drive and file extension specified when configuring MACE with the SETMACE command need not be supplied. The filename supplied will become the default name to be used by further Load, Save, Assemble to Object or Assemble to Listing file commands.

S[=<FILENAME>]

SAVE the file on disc in TSC editor format. The editor will over-write any existing file of the same name. File names have the default extension ".ASM" but this can be changed using SETMACE.

P L E A S E N O T E

MACE does not make backup copies of files; if you require a backup you must create it explicitly (e.g. S=FILE.BAK). If the filename is omitted then the name of the file that was loaded will be used again, allowing files to be loaded, modified and re-saved without the name having to be typed more than once.

W<TARGET>[=<FILENAME>] or W<#TARGET>...

WRITE part of a file to disc. As for (S) except that MACE writes only the specified number of lines, starting at the current line in the first form, and writes from the current line to the specified line in the second form. The default filename in this case is 1.SCRATCH.SCR, but this may be changed using SETMACE.

R[=<FILENAME>]

READ in a file, inserting it into the buffer immediately above the current line. The default file name is 1.SCRATCH.SCR, as for (W). These two commands enable block moves to be made safely, by writing part of the file to disc and then re-loading it at the new position. This technique for block copy-move operations may be a bit inconvenient at times but it does away with the overhead of reserving a large chunk of memory for a seldom used text buffer.

RECOVERING A FILE IN MEMORY

If your System Monitor has a memory dump facility that also displays the contents of memory in ASCII on the VDU screen you stand a 50-50 chance of recovering a file that has been lost though an accidental use of the 'N' command or re-entering MACE through the cold start entry point at $0000.

This same technique can also save a file in memory when a system crash occurs, but this time the odds are about 1 in 10 that you will be successful.

The first case concerns an accidental use of the 'N' command or a cold start of MACE. In both of these cases you can be confident that the original file is still present in memory and intact. What you have to do is enter your system monitor and dump the memory contents out to your VDU starting at the memory location CONTAINED in $2800/1. This is the beginning of file marker. As you work your way through the file you should recognize the text of your source file. Keep searching until you find the last line of the file. The memory location that you are interested in is the location of the first byte past the carriage return ($0D) in the last line. Once you locate this position in the file make a of note the memory location. Use your system monitor memory examine and change facility to alter the contents of $2802/3 to the memory address just noted. Now warm start MACE by a JUMP to $0003. Your file should be back to normal.

The second case concerns recovering a file when a system crash has occurred. In these circumstances the following course of action should be followed to the letter.

(1) Hit hardware RESET.

(2) Examine the contents of memory location $2802/3 and make a note of the address pointed to.

(3) Re-boot FLEX using a disk that does not have a STARTUP file on it. This is very important unless you are absolutely 100% positive that your STARTUP file does not cause the memory below say $B800 to be altered.

(4) Use the 'GET' command to load MACE, i.e. +++GET,MACE.CMD<CR>

(5) Enter your system monitor. Use the system monitor dump memory command to display the contents of memory starting about 500 bytes or so before the address noted in step (2). Work your way up to the end of the file as described in the earlier recovery instructions and verify that the address noted does in fact point to the end of the text file. If it doesn't then go back to the beginning of the file and start working your way up it until the text becomes junk. Make a note of the address of the byte following the address of the last sensible line in the file. Insert this address into to memory location $2802/3.

(6) Open both disk drive doors, unless you like to live dangerously!

(7) Warm start MACE by jumping to $0003.

(8) With a bit of luck you will have recovered your file or at least a reasonable part of it. Save it out to disk with a full file specification, i.e. #S=1.CRASH.SAV<CR>

3.0 THE MACE ASSEMBLER

The assembler is invoked by the (A) command described later in this section.

The syntax used by MACE conforms, in general, to the Motorola standard, with several enhancements and a few restrictions. Most existing programs will therefore assemble with a minimum of changes. The syntax of MACE is the same as other M6800/09 assemblers, i.e. comment lines and labels start in column one, while unlabeled source lines start in column two.

Only a single space is required between any two fields of the source line, since both the editor and the assembler "pretty print" the text. Putting in extra spaces will not adversely affect the operation of the assembler but may on occasion produce strange output formats. See the editor (*) command (section 2.1) for further information.

3.1 COMMENTS

Any line starting with (*) or (+) is treated by the assembler as a comment line. The only difference is when the (G) option is invoked during assembly of a source file. The (G) option, in conjunction with the (+) symbol, provide a facility that allows assembly-language procedures to be generated for inclusion in PL/9 compiler programs. Unless you wish to make use of this facility then comment lines should start with an asterisk.

Comments may be placed on any line, immediately following the operand (if any).

NOTE: If the length of the comment is such that it causes the total assembled line length to exceed the figure specified by the FLEX 'TTYSET' WD value then it will be truncated in any assembly listing, i.e. A:T, A:P or A:L.

If you wish to have a full width listing the value of 'WD' should be set to equal that of your printer. For example if you have a printer capable of printing 132 columns you should type 'TTYSET,WD=132<CR>' before calling MACE.

3.2 LABELS

MACE allows two types of label, as follows:

GLOBAL LABELS

May be up to 8 characters long, must start with a letter or a period and may comprise any sequence of letters, numbers and periods. Examples:

```
            DELAY.50        COUNT        ADD.X.Y        .538
```

LOCAL LABELS

Are used, as their name implies, locally in a program instead of global labels such as LOOP1, LOOP2 etc. They consist of a colon followed by a decimal number between 0 and 127, e.g. :5, :74. They are only valid between the global label most recently defined and the next, which enables them to be re-used in another part of the program. For example:

```
CLEAR    LDX   #START        POINT TO MEMORY
:1       CLR   ,X+           SET TO ZERO AND MOVE ON
         CMPX  #FINISH       DONE YET?
         BNE   :1            NO: DO THE NEXT ONE
*
INCREM   LDX   #START        POINT AGAIN
:1       INC   ,X+           BUMP THE CONTENTS AND MOVE ON
         CMPX  #FINISH
         BNE   :1            UNTIL DONE
```

Although :1 is used twice, there is no confusion as to which is referred to in each case. Where a local label has to be referenced from outside the range of its global, its full specification must be given, e.g. CLEAR:1 or INCREM:1 in the above example.

Local labels speed assembly, save space in the symbol table (requiring only 3 bytes as against 10 for a global label) and result in a clearer source listing. They are not included in the symbol listing.

Local labels may NOT be used with EQU, SET or EXT mnemonics.

3.3 MNEMONICS

MACE accepts any 6800, 6801 or 6809 mnemonic, generating appropriate 6809 code. For the sake of convenience and completeness, the following additional opcodes are also recognized:

```
  INY     =       LEAY  1,Y

  DEY     =       LEAY  -1,Y

  SEZ     =       ORCC  #4

  CLZ     =       ANDCC #$FB

  SEN     =       ORCC  #8

  CLN     =       ANDCC #$F7

  ASLD    =       ASLB
                  ROLA

  ASRD    =       ASRA
                  RORB

  CLRD    =       CLRA
                  CLRB

  COMD    =       COMA
                  COMB

  DECD    =       TSTB
                  BNE   *+3
                  DECA
                  DECB

  INCD    =       INCB
                  BNE   *+3
                  INCA

  LSLD    =       LSLB
                  ROLA

  LSRD    =       LSRA
                  RORB

  NEGD    =       NEGA
                  NEGB
                  SBCA  #0

  ROLD    =       ROLB
                  ROLA

  RORD    =       RORA
                  RORB

  TSTD    =       SUBD  #0

  SKIP1   =       $21 (BRN)

  SKIP2   =       $8C (CMPX)  Don't use this one unless you understand the
                              effect it has on the condition code register.
```

3.4 OPERANDS

The assembler supports the following data types:-

1. Decimal Numbers e.g. 1, 9442, 0

2. Hexadecimal Numbers e.g. $A, $F12, $36

3. Binary Numbers, e.g. %101, %00011011

4. ASCII Values e.g. 'A, '?

5. Labels e.g. FRED, :25, ADD:1

6. Current PC value, indicated by *

Program Counter Relative (PCR) addressing requires the user to know whether the operand is more or less than 128 bytes away. MACE assumes the 8-bit mode unless instructed otherwise in the following way:

```
        LEAX  TABLE,PCR      8-bit offset
        LEAX  >TABLE,PCR     16-bit offset
```

The programmer can similarly "force" direct or extended addressing so as to generate the desired form irrespective of the value assigned by the SETDP directive. For example:

```
        LDA   <VALUE         Force Direct Addressing
        STA   >BUFFER        Force Extended Addressing
```

Arithmetic may be performed on operands. Execution of an expression is without arithmetic precedence, from left to right. The four operators + - * / may be used, and an expression may commence with a minus. For example:

```
        LDX   #-LABEL*5
        BRA   *-:73+$6B
        CMPA  #'G-'A/2
```

3.5 ASSEMBLER DIRECTIVES

Assembler directives are special kinds of mnemonic, giving instructions not to the microprocessor but to the assembler. Most of these are Motorola standard, but there are some differences:-

EQU Equate

Assigns the value of the operand to the label (global labels only).

SET

Performs the same function as EQU but allows a label to be re- defined as often as necessary without an error occurring.

EXT External

Defines a label that is in a module external to the program being assembled. A value of $FFFF will be assigned to the label. (see section four for information on spooling.)

END

There is usually no need to use an END statement since assembly will terminate at the end of the file or list of files. The END, if present, need not be the last statement, but when encountered it will cause assembly to cease. Any expression in the operand field will be evaluated, and if an object file is generated will be written last of all to disc as a transfer address.

CON Conditional

CON may be used in the sense of "conditional skip" or "conditional assembly". The former is usually required when spooling multiple files (see the section on spooling), while the latter is needed if subroutine libraries are to be used. In either case, the operand must be a global label (not an expression). CON FLAG will cause a skip until the next NOC if FLAG is non-zero. If FLAG is zero or undefined, assembly will continue at the next line. CON -FLAG will cause a skip if FLAG is zero or undefined. Note that it is not possible to "nest" conditional statements.

NOC No Conditional

Assemble all instructions (see CON).

NAM Name or TTL Title

The operand (up to 50 characters) will be printed at the top of each page when listing to a printer or a listing file.

SPC Space

Is not implemented. Use instead an empty line or a line containing a single asterisk.

3.5 ASSEMBLER DIRECTIVES (continued)

PAG Page

Is also un-implemented, and should be replaced by a double asterisk (in the label field) which will cause a new page to be started.

FCB Form Constant Bytes

Converts the operand(s) (separated by commas) into 8-bit values.

FDB Form Double Bytes

Converts the operand(s) into 16-bit values.

FRA Form Relative Address

In order to achieve position-independent code, dispatch tables (i.e. tables of internal routine addresses) must contain relative values. FRA LABEL is equivalent to FDB LABEL-*, and in the assembly listing the absolute value of LABEL will be printed in the same way as the destination of a branch, as an aid to finding one's way around the program.

FCC Form Constant Characters

This directive allows text to be included in a program. The operand may comprise any sequence of numbers (decimal or hexadecimal) or ASCII strings bracketed by matching delimiters (or by a delimiter at the start and a carriage return at the end). For example: FCC CR,LF,/BREAK/,CR,LF,4

FCS Form Constant String

This is identical to FCC except that the last character of the operand has bit 7 set high (as an end of string flag).

RMB Reserve Memory Bytes

The operand is added to the current program counter value. No code is generated. The instruction is used to reserve space for variables and data.

ORG Origin

The value of the operand defines where in memory the following code is to be located (originated).

SETDP Set Direct Page

SETDP N will cause direct addressing to be generated only for variables in page N, from that point in the program until the end or another SETDP directive. It is up to the programmer to ensure that the 6809's direct page register is set to the correct value; MACE has no way of knowing this.

3.6 INVOKING THE ASSEMBLER

The MACE assembler resides in memory with the editor and may, for all intents and purposes, be considered to be an integral part of the editor. In order to segregate the editor and assembler commands we are covering the assembler commands separately from those of the editor.

The assembler may be called from within the MACE editor or may be called from the FLEX command line. This latter facility speeds up assembly of large programs via the FLEX 'EXEC' command when desired.

A

Assemble the edit file without any listing, printout or object file. Generally used to perform a quick syntax/typographical error check.

A[:<options>]

Assemble the file resident in memory. Options are as follows:

A:T

Assemble with a listing on the terminal; no titles or page numbers will be printed.

A:P

Assemble with a printer listing. The page number and the date will be printed at the top of each page.

A:N

Assemble with a cross reference listing only, i.e. suppress main listing. This option only makes sense if used with the L, P, T or X options.

A:X

Generate a cross reference table. The symbol table listing contains the first value of the symbol, then the source line number in which it was defined, followed by the number of each line in which it was referenced. This option only makes sense if used with the L, P, or T options and may be used with 'N'.

A:<N1>-<N2>

If one of the T, P or L options is in force, the assembler can be requested to generate output for only the specified range of source line numbers. No symbol table will be output in this case. If the -<N2> is omitted then only one line will be generated.

3.6 INVOKING THE ASSEMBLER (continued)

A:M

Write object code directly into memory. MACE will not allow itself, its edit file or any of its tables to be over-written, and complains with the message "CAN'T WRITE TO $MMMM", where MMMM is the address of the attempted write. See the diagram of memory usage, in section six, for information on what areas of memory are used by MACE.

A:O[=<FILENAME>]

Write object code to disc, overwriting any existing file of that name. Use the Q command to see what default drive and extension will be used; if you don't like them then use SETMACE to change them.

A:$XXXX

When using either the M or O options it is frequently useful to be able to offset the program (for example when the object code is to be put into an EPROM and there is no RAM on the development system at the required address). The offset $XXXX is added to the normal program counter value.

A:L[=<FILENAME>]

Write the assemble listing to disc into the named file. Use the Q command to see what default drive and extension will be used; if you don't like them use SETMACE to change them. As for the (A:P) command titles and page numbers will be printed at the top of each page. The A:L option produces a file with CR-LF ($0D, $0A) sequences at the end of each line so the file will be ready for use with the FLEX print spooler.

A:G,L

Generate PL/9 source. Instead of a normal listing, the assembler will produce lines of output in the form "GEN $XX,$XX,$XX....etc;" which the PL/9 compiler can use as source files. Any line starting with a (+) will be copied intact (but with the '+' removed) to the output device or file. The assembly language labels and mnemonics will be passed to the output file as comments enclosed within the usual '/*.....*/' pair PROVIDED THAT 'PRETTY PRINTING' IS ENABLED. Any comments after the mnemonics will be ignored. This option will normally be used as A:G,L[=FILENAME], but may be used with 'T' or 'P'. The A:G option produces a file with CR ($0D) only at the end of each line so that the file will be ready for use by PL/9.

A:S=<FILENAME>

Spool from a named file. The file will be opened and read into the edit buffer, and is assumed to contain a list of file names (one on each line) comprising the segments of the program to be assembled. See section four for more information on spooling. The default extension on the spool file is '.ASM'

3.6 INVOKING THE ASSEMBLER (continued)

Assemble options may be strung together, as in the following examples:

```
A:P,100-200
```

Assemble to the printer, generating a listing only for lines 100-200.

```
A:T,281
```

Assemble only line 281.

```
A:M,$4000
```

Assemble to memory, loading the program at a location offset by $4000 from any origin specified.

```
A:O,L
```

Generate a binary file and a listing file, both files having the names given by the Q command.

```
A:O,P,T,L,X
```

Using the default file names generate a binary file, and direct a listing with a cross reference to a disk file, to the printer, and to the terminal.

```
A:O=1.MYFILE.BIN,P,T,L=MYFILE.OUT,X
```

As above but override the default file names.

NOTE: Any one, or all of the listing options (T, P or L) may be in force at any given time. i.e it is possible to specify 'A:T,P,L,O' and generate a listing on the terminal, a listing on the printer, a disk file (for spooling later) and the output object file if this is what you require.

CALLING THE ASSEMBLER FROM FLEX

The MACE assembler may also be called from FLEX with multiple options specified, as the following examples illustrate:

```
+++MACE,1.FILENAME.EXT<CR>
```

Assemble the file specified reporting any errors.

```
+++MACE,FILENAME<CR>
```

Assemble the file specified using the default drive number and file extension that PL/9 was configured for using the SETMACE command.

```
+++MACE,FILENAME+T<CR>
```

Assemble the file specified to the system console.

```
+++MACE,FILENAME+P<CR>
```

Assemble the file specified to the system printer.

```
+++MACE,FILENAME+O<CR>
```

Assemble the file specified to an output file (binary) with the same name as 'FILENAME' but using the default drive and file extension specified by the SETMACE defaults.

```
+++MACE,FILENAME+O=0.OBJECT.BIN<CR>
```

As above but override the default drive, filename and extension.

```
+++MACE,FILENAME+L,O<CR>
```

Assemble the file but direct the output listing to a disk file using the default drive number and file extension defined by SETMACE. Also produce an object file on disk in the same manner.

```
+++MACE,FILENAME+L,O,M<CR>
```

Assemble the file as above but also assemble the file into memory.

NOTE: Any one, or all of the listing options (T, P or L) may be in force at any given time. i.e it is possible to specify 'A:T,P,L,O' and generate a listing on the terminal, a listing on the printer, a disk file (for spooling later) and the output object file if this is what you require.

4.0 SPOOLING

Spooling is used when the source file is too large to assemble in one piece. Any number of files can be assembled together, but there must be no labels that are repeated from one file to another or multiply defined symbol errors will result in pass 1 unless a conditional structure is used as shown below. To spool multiple files it is necessary to create a file containing the names of each of the component files, one to each line, then to use the A:S=FNAME version of the assemble command (see section 3.6). For example, suppose that a program is split up into three parts, called INTRO.ASM, MAIN.ASM and IOSUBS.ASM. A file called ASM.ASM (for example) is created (using 'BUILD' or your normal editor) having the following contents:

```
1.INTRO.ASM
1.MAIN.ASM
1.IOSUBS.ASM
```

To assemble the program, use the command A:S=ASM, with any other options that may be required, e.g A:S=ASM,O=MYFILE.BIN,L=MYFILE.OUT,T,P.

Alternatively you can invoke the spool option from the FLEX command line thus:

```
+++MACE+S=ASM.ASM,O,T,P,L<CR>   (note the 'O,T,P,L' are optional)
```

NOTE: The last file must have an 'END' directive on the last line of the file otherwise a DOS error will occur between PASS 1 and PASS 2.

The main problem that is likely to arise is that (for example) MAIN and IOSUBS use the same variable storage space, and in order for each file to be assembled by itself, these variables are declared in the form:

```
TEMP     RMB  2
POINTER  RMB  2
BUFFER   RMB  80
```

etc. When the files are spooled, however, these declarations are seen by the assembler as multiply defined symbols unless a conditional structure is used to prevent them from appearing more than once. To do this, a variable (I always use SPOOL) is defined in the first module (i.e. SPOOL EQU 1), and in IOSUBS the following structure is used:

```
         CON  SPOOL
TEMP     RMB  2
POINTER  RMB  2
BUFFER   RMB  80
          .
          .
         NOC
```

4.0 SPOOLING (continued)

The use of the CON-NOC pair prevents the included source lines from being passed to the assembler as long as SPOOL is non-zero, but when IOSUBS is being assembled alone, since SPOOL has not been defined, all of the included lines are assembled normally. SPOOL may be re-defined (using SET) at any point, allowing a flexible method of handling large programs.

A similar problem exists if the souce program in one file needs to access a subroutine or a data segment/table that is in another file. This too is catered for with the CON ... NOC pair thus:

```
         CON  SPOOL

SUB1     EQU  EXT
SUB2     EQU  EXT
SUB3     EQU  EXT
TABLE1   EQU  EXT
TABLE2   EQU  EXT
          .
          .
         NOC
```

If variables, program and data segments exist in external modules they should all be declared within a single CON ... NOC pair

The objective of all of this is to make each source file module of a program capable of being assembled on its own. This means that any program segment, data segment or variable declaration that is made outside of the current file will have to be declared within the CON ... NOC pair to ensure that the program module can be assembled on its own AND to ensure that 'multiply defined symbol' errors do not occur during assembly of the composite program by spooling.

To make life easy it is recommended that you use the 'LONG BRANCH' instructions to access any program segments outside of the current module and 16-bit PCR addressing (LEAX >LABEL,PCR) when addressing any program, data or variable segments outside of the current module. Once the program is up and running you can produce a source listing and tidy up the address ranges if the code saving is justified.

It is not absolutely essential that you make each module capable of being assembled on its own. If you wish you can just break up a large source file into arbitrary size modules and then spool assemble them. The only difficulties that arise with this approach is that simple typographical errors and local branch out of range errors get rather tedious to correct as the entire program must be assembled to spot them.

The primary advantage of spool assembling files is that it is possible to build up large programs from a composite of small library modules thus dispensing with the need for link-loading programs. Since MACE can generally assemble files faster than most link-loaders can perform their jobs there is little advantage to link loading in most small system applications.

Link-loading comes into its own when you want to generate modules each with their own data storage areas and program segment areas and you want to produce an output file which uses many such modules where all program segments are in one block and all data segments are in another. There are advantages to each approach. Only you, the programmer, can decide which is best for you and your particular needs.

5.0 ERROR HANDLING

When an error is detected, one of the messages below is printed, followed by the offending line, under which will be a caret (up arrow) pointing to the point MACE had reached when it detected the error. It then waits for the operator to hit a key. If a carriage return is typed, assembly will cease and you will be returned to the editor at the line MACE stopped at (often enough the faulty line). Any other character will allow assembly to continue, but the faulty line will not be further processed, and instead assembly will continue at the next line. Error messages and their meanings are as follows:-

LABEL ERRORS

MULTIPLY DEFINED SYMBOL

The symbol has been defined twice (in the case of local labels, the label has been used twice in the range of the same global).

UNDEFINED SYMBOL

The symbol has not been defined anywhere in the program.

ILLEGAL SYMBOL

Label too long or contains illegal characters.

MISUSE OF LOCAL LABEL

Usually an attempt to EQUate a local label.

SYNTAX ERRORS

MISSING OR ILLEGAL MNEMONIC

Usually means that the mnemonic was not recognized as such. Have you forgotten the space before the mnemonic?

ILLEGAL REGISTER SPECIFICATION

Mnemonic not followed by the correct register designation, e.g. LDC #1 or LEAQ 1,X.

ILLEGAL ADDRESSING MODE

Usually a mis-use of immediate addressing, e.g. JMP #25.

5.0 ERROR HANDLING (continued)

SYNTAX ERROR

Anything not covered by another message.

ENVIRONMENT ERRORS

BRANCH OUT OF RANGE

Destination of branch is too far away. Convert to long branch.

8-BIT RANGE EXCEEDED

The operand of a PCR is too far away for the 8-bit mode. Convert to >LABEL,PCR (see section 3.4). This error is quite often encountered when assembling programs generated on other systems.

PHASING ERROR

Program counter in Pass 2 does not agree with its value in Pass 1. Usually caused by a forward reference to a direct variable (i.e. value < 256). This error can be quite difficult to track down, but the cause is always between the last declared label and the line the error was reported at. Try putting dummy labels in this region, to try to narrow down to where the error is.

OUT OF MEMORY

Not enough memory to assemble the file. Try not using a cross - reference table or split the file into segments and use spooling (q.v.).

NOTE: The assembler will print '>' just to the left of any LONG BRANCH instruction that can be converted to a short branch. This is not considered to be an error and is therefore not reported as one.

6.0 CONFIGURING MACE TO YOUR SYSTEM HARDWARE ENVIRONMENT

This section will provide details of how to install MACE in your system.

(1) The first thing you should do upon receipt of this software is to complete and return the registration form that accompanies it. If it is not present contact us and we will send you one. Completion of the registration form is very important as it is through registration forms (not sales records) that we keep you informed of any upgrades that are available for this product.

We will only supportone user per copy sold. If you fail to register your copy with us and then phone for technical support we have no way of knowing whether you are a legitimate user or someone who has come by this product by dishonest means. We will not answer any questions until we have a signed registration form in our files. When you send the form back to us ensure that it is sent RECORDED delivery. <u>DO IT NOW!</u>

(2) Remember that this product is licensed for use by a single user on a single computer system if used within any private or commercial environment (see note below). If any of your friends or associates within the same organization wants a copy contact the factory for details of our low cost 'KLONING' service. This service will create any specified number of copies of your disk and manual. Each copy may be registered to a different individual and <u>EACH</u> user will be eligible for the same upgrades, support service, etc, as the original purchaser.

<u>THIS IS A LOW COST SERVICE DESIGNED TO KEEP HONEST PEOPLE HONEST!</u>

(3) After completing the registration form you should format a fresh disk in the system you will be using MACE with. You should then copy ALL of the files on the disk we have supplied onto the freshly formatted disk. The original disk we supplied should then be put in a safe place as we will require that you return it to us if you want to upgrade or klone your copy of MACE at a later date.

(4) The next step is to copy the following files from the working copy of the MACE disk onto your ususal SYSTEM disk, i.e. the disk that normally resides in drive #0:

```
a. MACE    .CMD
b. SETMACE .CMD
```

(5) You should now run the 'SETMACE' program described on the following pages.

NOTE: Relaxed conditions of use and reproduction apply to bona-fide educational and technical training establishements. If you are intending to use MACE in a non productive educational environment contact the factory for terms of use.

6.0 CONFIGURING MACE TO YOUR SYSTEM HARDWARE ENVIRONMENT (continued)

A special program has been provided to greatly simplify the task of configuring MACE to your FLEX system and its terminal and printer. The program is called 'SETMACE.CMD' and it runs like this:

```
+++SETMACE<CR>


*****   MACE Configuration Program   *****
        =========================

       For use with MACE version X.XX.

This  program  allows you to configure MACE
to your  own  particular  requirements  and
those of your computer system.

Some  of  the questions do not need answers
unless you wish to change the data  already
supplied.  In these cases the existing data
will  be  displayed.  To  leave  the   data <------------- note!
unaltered, just hit <CR>.

       PUT YOUR MACE DISK IN DRIVE
       ZERO THEN HIT ANY KEY......          <------------- hit <CR>


            INPUT WITHOUT ECHO
            ==================

MACE requires your keyboard  input  routine
to  return  the  ASCII  value  of  the  key
pressed, in the A-accumulator, WITHOUT  the
character  being  echoed. Early versions of
FLEX9 did not support this feature, so  you
should  first  check  whether  you  have an
"INCHNE" vector (i.e. address of an input-
without-echo routine) at $D3E5. If you need
to exit SETMACE to examine your system just
type <CR>.

Does your system have INCHNE at $D3E5?      <------------- 'Y' for most systems.
```

If you answer this question 'Y' the prompt just after '***' on following page will appear.

If you have an early FLEX 9 system produced by SWTP (et al) the INCHNE vector at $D3E5 may not be present. If this is the case answer this question 'N' and the following options will be available.

6.0 CONFIGURING MACE TO YOUR SYSTEM HARDWARE ENVIRONMENT (continued)

```
There are three  ways  to  implement  input
without echo:

1) You can give me the address of a routine
   in  your  system  that  performs  input
   without echo;

2) You  can give me the address of a vector
   that points to your input  routine  (NOT
   the address of the routine itself);

3) You  can  give  me  the  address of your
   input device (6850 ACIA, 6821 PIA  etc.)
   at  which  the  ASCII  key  code  can be
   found. It is essential that the  act  of
   reading   the   character   clears   the
   "character waiting" flag;

4) You can exit and think about it.

Which do you want to do (1-4)? 2          <------------ If you have an  SBUG-E
                                                         compatible     system
                                                         monitor.

Address of your input vector: $F804       <------------ 'INCHNE' vector

                                          NOTE:  If you are not  sure  of  the
                                                 address   you  are  supplying
                                                OPEN THE DISK DRIVE DOORS as
                                                 an invalid address will cause
                                                 the system toCRASH!
***


Just to check, type the number "1":1      <------------ type '1'

I got a "1"!! Did it echo on the screen? N  <------------ it should be 'N'

                                          NOTE:  If  any  part  of  this check
                                                 fails SETMACE will return  to
                                                 FLEX.

That's the difficult part over. Now for the
easy bits!


                 KEYBOARD
                 ========

First lets set up MACE for  your  keyboard.
Each  question  should  be  answered with a
single keypress or control key combination,
e.g. (Control H) for backspace:

First press your  backspace  key.........  <------------ CTRL H if in doubt
Next your line cancel key................  <------------ CTRL X if in doubt
And lastly your escape key...............  <------------ CTRL [ if in doubt
```

6.0 CONFIGURING MACE TO YOUR SYSTEM HARDWARE ENVIRONMENT (continued)

```
                 PRINTER
                 =======

Now to set up MACE for your printer.

How many listing lines are to be printed on
each page? Leave  some  room  for  top  and
bottom margins.
...................... (currently 55)?     <------------ number then <CR>

Total  length  (in  lines)  of  each sheet?
...................... (currently 66)?     <------------ number then <CR>

Does your printer support form feed?       <------------ Y or N (<CR> = Y)
What HEX character is it?............$     <------------ $0C<CR> for most

Do you want MACE to pretty-print?          <------------ Y or N (<CR> = Y)


                DISK FILES
                ==========

Now  I  want  to  know the default filename
extensions and default drive numbers:

LOAD and SAVE file extension.(currently ASM): <---------- new extension <CR>
and its default drive number..(currently #1): <---------- new drive number

OBJECT (A:O) file extension..(currently BIN): <---------- new extension <CR>
and its default drive number..(currently #1): <---------- new drive number

LISTING (A:L) file extension.(currently OUT): <---------- new extension <CR>
and its default drive number..(currently #1): <---------- new drive number

PL/9 (A:G) file extension....(currently LIB): <---------- new extension <CR>
and its default drive number..(currently #1): <---------- new drive number

READ and Write use a file called: 1.SCRATCH.SCR".
...............................Is  this  OK? <---------- Y or N (<CR> = Y)

R/W scratch file name?...(currently SCRATCH): <---------- new name <CR>
R/W scratch file extension?..(currently SCR): <---------- new extension <CR>
and its default drive number..(currently #1): <---------- new drive number


Your copy of MACE is now configured!
====================================


+++
```

NOTE: SETMACE no longer asks you how many columns are supported by your printer as MACE now obeys the TTYSET 'WD' value. If you have problems with screen or printer truncation it is due to TTYSET either not being set up at all or set to too low a value. If you have a 132 column printer you should invoke TTYSET thus: '+++TTYSET WD=132<CR>'

6.1 MACE MEMORY MAP

MACE uses system memory as follows:

```
MEMEND  +------------------+  <------- FLEX MEMEND ($CC2B) POINTS HERE
  |     |                  |
  |     |       USER       |
  |     |      MEMORY      |
  |     |                  |
  v     +------------------+

  ^     +------------------+  <------- "MEMORY FREE ABOVE"
  |     |      SYMBOL      |
  |     |      TABLES      |
  |     |------------------|
  |     |      SOURCE      |
  |     |      PROGRAM     |
  |     |------------------|
  |     |       MACE       |
  |     | VARIABLES/STACK  |
$2611   |------------------|
  |     |       MACE       |
  |     |     ASSEMBLER    |
$0000   +------------------+
```

A good memory map to use when developing a program a program with MACE and D-BUG is to load the Windrush D-BUG package at $A800 and origin the test program at any address after 'MEMORY FREE ABOVE'. Obviously the object code must fit between this address and $A800.

7.0 COMPATIBILITY

Although MACE can handle programs written for other assemblers with little modifications required, the reverse is not necessarily true. In order to ensure that programs developed using MACE can be transferred to other systems, the following points should be noted:-

1. Do not use labels of more than six characters in length.
2. Do not use any periods in labels.
3. Do not use local labels at all, i.e. :1, :2, :3, etc.
4. Avoid using SKIP1, SKIP2, CON, NOC, FRA, FCS or EXT mnemonics, since their meanings may vary from one system to another.
5. The argument of the NAM directive is often restricted to a maximum of six characters.
6. Avoid using the expanded form of FCC:

```
FCC  CR,LF,/MESSAGE/,CR,LF,EOT   (only generate the '/MESSAGE/' section)
```

7. Avoid using the expanded form of FCB:

```
FCB  CR,LF,LF,NL,NL,EOT          (only generate one byte per line)
```

8. Avoid using the expanded form of FDB:

```
FDB  0,0,0,0                     (only generate one byte pair per line)
```

8.0 MOST OFTEN ASKED QUESTIONS

This section is to document the answers to the questions we are most often asked on the phone.

(Q) Can associates in my office legally use my copy of MACE on their own machines?

(A) NO! They will have to buy their own copies or get klones of yours.

-- o --

(Q) Can I buy additional copies of the MACE manual and/or the ASSEMBLY LANGUAGE PROGRAMMERS GUIDE?

(A) YES. Contact the factory or your dealer for prices.

-- o --

(Q) Do you have any other products for the Motorola family of processors?

(A) YES! Lots of them. Contact the factory or your dealer for prices.

-- o --

(Q) When I use the A:T, A:P or A:L options the source file is truncated on the right side. Why?

(A) Because TTYSET has not been set up to match the desired width of the output text line. If you have a 132 column printer and wish to make long comments in your program you should type 'TTYSET,WD=132' before using the A:P or A:L options otherwise the listing will be truncated on the right side.

-- o --

(Q) When I use the 'A:L' option and then try to use the file with SCREDITOR III I get a bunch of '^' (carrets) up the left side of the text. Why?

(A) SCREDITOR III is taking the inclusion of the line-feed character ($0A) in the output file as an imbedded command. The imbedded command can be deleted just as though it were any other character. If you own a system running Windrush FLEX or have a copy of the Windrush Utilties package you can use the 'S' command to strip out the line-feed characters before editing the file with SCREDITOR III.

-- o --

(Q) I get a DOS error between PASS 1 and PASS 2 when I am spooling files. Why?

(A) Because the 'END' directive is not present on the last line of the last file being spooled.

-- o --

(Q) How do I generate an output binary file with a transfer address so that I can use it as a FLEX command?

(A) Use the 'END' directive on the last line of the program to indicate the desired transfer address, e.g. 'END $C100' or 'END LABEL'.

8.0 MOST OFTEN ASKED QUESTIONS (continued)

(Q) How can I determine if my system has the INCHNE vector implemented at $D3E5?

(A) 1. Boot up FLEX and then enter your system monitor by typing 'MON<CR>'.

2. Enter the following code at $C000 with your system monitors memory examine and change facility ('M' in most monitors):

```
C000 AD
C001 9F
C002 D3
C003 E5  JSR [$D3E5]   (INPUT CHARACTER WITHOUT ECHO)
C004 BD
C005 CD
C006 0F  JSR $CD0F     (OUTPUT CHARACTER)
C007 7E
C008 C0
C009 00  JMP $C000     (LOOP FOREVER)
```

3. Open the disk drive doors and use your system monitors 'JUMP' command ('J' in most monitors) to pass control to $C000.

4. Now start typing at the terminal keyboard. All characters typed should appear only once on the screen if INCHNE is implemented correctly. If nothing happens the INCHNE vector is probably not implemented and the system has crashed. If the characters you type appear twice then INCHNE has not been implemented properly.

THIS PAGE IS INTENTIONALLY LEFT BLANK

MACE EDITOR COMMAND SUMMARY

S Y M B O L S

<CR>	represents a carriage return.
<>	symbols are used to enclose a variable.
[]	symbols indicate that the enclosed data is optional.
<NUMBER>	a decimal number such as 36 or 192. (defaults to one)
<TARGET>	represents the decimal number of lines speied by the command, and defaults to one if none is given.
<#TARGET>	represents the decimal line number specified by the command.

M O D E   C O N T R O L

I — INSERT lines mode. Prompt will change from (#) to (+) and the following commands are available:

BACKSPACE...moves the cursor to the left one place.
CANCEL......erases the entire line.
ESCAPE......(in left col) terminates the insert
session.
RETURN......generates a new line.

X	EXIT to FLEX.
M	MONITOR. Enter the ROM System Monitor.
/<COMMAND>	Execute a FLEX command.
*	Toggles between formatted and unformatted text.

L I N E   P O S I T I O N I N G   C O M M A N D S

<NUMBER>[COMMAND]	Make <NUMBER> the current line, then execute [command].
1 or ^	Go to the first line in the file.
B or !	Go to the bottom (/EOF) of the file.
+<NUMBER>	Move down <NUMBER> lines from the current position.
-<NUMBER>	Move up <NUMBER> lines from the current position.
<CR>	Display the current line.
<ESCAPE>	Display the next line.

MACE EDITOR COMMAND SUMMARY

F I L E   O R I E N T E D   C O M M A N D S

N	NEW file. Erase the current file.
P<TARGET>	PRINT <TARGET> number of lines on the terminal.
D<TARGET>	DELETE <TARGET> number of line(s).
D<#TARGET>	DELETE from current line to <#TARGET> line.

L I N E   E D I T I N G

O<CHAR>	OVERLAY the current line.
E	EDIT the current line. (leaves cursor at end of line).
=<TEXT>	REPLACE the current line with <TEXT>.

G L O B A L   E D I T I N G

F<NUMBER>/<STRING>	FIND the next <NUMBER> occurrences of <STRING>.
C<NUMBER>/<ST1>/<ST2>	CHANGE the next <NUMBER> occurrences of <ST1> to <ST2>.

D I S K   F I L E   H A N D L I N G

?	Display the last filename used by Load, Save or Write.
Q	Query the default filenames.
L[=<FILENAME>]	LOAD a disc file.
S[=<FILENAME>]	SAVE the file on disc.
W<TARGET>[=<FILENAME>]	WRITE <TARGET> number of lines to disk.
W<#TARGET>[=<FILENAME>]	WRITE from current line to <TARGET> line number to disk.
R[=<FILENAME>]	READ in a file above the current line.

MACE ASSEMBLER COMMAND SUMMARY

A	Assemble only showing errors.
A:N	Assemble with symbol table only (use with T,P,L).
A:X	Assemble with cross reference only (use with T,P,L).
A:T	Assemble with a listing on the terminal.
A:P	Assemble with a printer listing.
A:G,L	Assemble producing PL/9 'GEN' statements.
A:M	Write object code directly into memory.
A:S=FILENAME	Assemble from spool file.
A:O[=FILENAME]	Write object code to disc.
A:L[=FILENAME]	Write the assembly listing to disc into the named file.
A:$XXXX	Offset the object code. (used with the M or O options).
A:[P T L],<N1>-<N2>	Generate output for specified range of line numbers.

MULTIPLE COMMAND EXAMPLES

A:T,M	Assemble to the terminal and to memory.
A:P,281-305	Assemble lines 281 through 305 to the printer.
A:T,O	Assemble to the terminal and write a binary record to the default filename.
A:G,L	Assemble to the default listing file but generate PL/9 source.

CALLING MACE FROM FLEX

+++MACE,[source]	Assemble and check for errors.
+++MACE,[source]+T	Assemble to terminal
+++MACE,[source]+P	Assemble to printer.
+++MACE,[src]+O=[obj],P	Assemble to terminal and write binary file [name] to disk.
+++MACE,[src]+L[=lis]	Assemble to listing file
+++MACE,[src]+O[=obj],$XXXX,L[=list]	Assemble to listing file and write binary file [name] to disk with offset $XXXX.
+++MACE+S=[spool file],[options]	Assemble files in spool mode.

NOTE: Any one, or all of the listing options (T, P or L) may be in force at any given time. i.e it is possible to specify 'A:T,P,L,O' and generate a listing on the terminal, a listing on the printer, a disk file (for spooling later) and the output object file if this is what you require.

ASCII CODE REFERENCE CHART

ASCII	HEX	BINARY	DEC	OCT	ASCII	HEX	BINARY	DEC	OCT
NUL	$00	0000 0000	000	000	SP	$20	0010 0000	032	040
SOH	$01	0000 0001	001	001	!	$21	0010 0001	033	041
STX	$02	0000 0010	002	002	"	$22	0010 0010	034	042
ETX	$03	0000 0011	003	003	#	$23	0010 0011	035	043
EOT	$04	0000 0100	004	004	$	$24	0010 0100	036	044
ENQ	$05	0000 0101	005	005	%	$25	0010 0101	037	045
ACK	$06	0000 0110	006	006	&	$26	0010 0110	038	046
BEL	$07	0000 0111	007	007	'	$27	0010 0111	039	047
BS	$08	0000 1000	008	010	(	$28	0010 1000	040	050
HT	$09	0000 1001	009	011	)	$29	0010 1001	041	051
LF	$0A	0000 1010	010	012	*	$2A	0010 1010	042	052
VT	$0B	0000 1011	011	013	+	$2B	0010 1011	043	053
FF	$0C	0000 1100	012	014	,	$2C	0010 1100	044	054
CR	$0D	0000 1101	013	015	-	$2D	0010 1101	045	055
SO	$0E	0000 1110	014	016	.	$2E	0010 1110	046	056
SI	$0F	0000 1111	015	017	/	$2F	0010 1111	047	057
DLE	$10	0001 0000	016	020	0	$30	0011 0000	048	060
DC1	$11	0001 0001	017	021	1	$31	0011 0001	049	061
DC2	$12	0001 0010	018	022	2	$32	0011 0010	050	062
DC3	$13	0001 0011	019	023	3	$33	0011 0011	051	063
DC4	$14	0001 0100	020	024	4	$34	0011 0100	052	064
NAK	$15	0001 0101	021	025	5	$35	0011 0101	053	065
SYN	$16	0001 0110	022	026	6	$36	0011 0110	054	066
ETB	$17	0001 0111	023	027	7	$37	0011 0111	055	067
CAN	$18	0001 1000	024	030	8	$38	0011 1000	056	070
EM	$19	0001 1001	025	031	9	$39	0011 1001	057	071
SUB	$1A	0001 1010	026	032	:	$3A	0011 1010	058	072
ESC	$1B	0001 1011	027	033	;	$3B	0011 1011	059	073
FS	$1C	0001 1100	028	034	<	$3C	0011 1100	060	074
GS	$1D	0001 1101	029	035	=	$3D	0011 1101	061	075
RS	$1E	0001 1110	030	036	>	$3E	0011 1110	062	076
US	$1F	0001 1111	031	037	?	$3F	0011 1111	063	077

A S C I I   C O D E   R E F E R E N C E   C H A R T

ASCII	HEX	BINARY	DEC	OCT	ASCII	HEX	BINARY	DEC	OCT
@	$40	0100 0000	064	100	`	$60	0110 0000	096	140
A	$41	0100 0001	065	101	a	$61	0110 0001	097	141
B	$42	0100 0010	066	102	b	$62	0110 0010	098	152
C	$43	0100 0011	067	103	c	$63	0110 0011	099	143
D	$44	0100 0100	068	104	d	$64	0110 0100	100	144
E	$45	0100 0101	069	105	e	$65	0110 0101	101	145
F	$46	0100 0110	070	106	f	$66	0110 0110	102	146
G	$47	0100 0111	071	107	g	$67	0110 0111	103	147
H	$48	0100 1000	072	110	h	$68	0110 1000	104	150
I	$49	0100 1001	073	111	i	$69	0110 1001	105	151
J	$4A	0100 1010	074	112	j	$6A	0110 1010	106	152
K	$4B	0100 1011	075	113	k	$6B	0110 1011	107	153
L	$4C	0100 1100	076	114	l	$6C	0110 1100	108	154
M	$4D	0100 1101	077	115	m	$6D	0110 1101	109	155
N	$4E	0100 1110	078	116	n	$6E	0110 1110	110	156
O	$4F	0100 1111	079	117	o	$6F	0110 1111	111	157
P	$50	0101 0000	080	120	p	$70	0111 0000	112	160
Q	$51	0101 0001	081	121	q	$71	0111 0001	113	161
R	$52	0101 0010	082	122	r	$72	0111 0010	114	162
S	$53	0101 0011	083	123	s	$73	0111 0011	115	163
T	$54	0101 0100	084	124	t	$74	0111 0100	116	164
U	$55	0101 0101	085	125	u	$75	0111 0101	117	165
V	$56	0101 0110	086	126	v	$76	0111 0110	118	166
W	$57	0101 0111	087	127	w	$77	0111 0111	119	167
X	$58	0101 1000	088	130	x	$78	0111 1000	120	170
Y	$59	0101 1001	089	131	y	$79	0111 1001	121	171
Z	$5A	0101 1010	090	132	z	$7A	0111 1010	122	172
[	$5B	0101 1011	091	133	{	$7B	0111 1011	123	173
\	$5C	0101 1100	092	134	|	$7C	0111 1100	124	174
]	$5D	0101 1101	093	135	}	$7D	0111 1101	125	175
^	$5E	0101 1110	094	136		$7E	0111 1110	126	176
_	$5F	0101 1111	095	137	DEL	$7F	0111 1111	127	177