XMACE

EDITOR/CROSS ASSEMBLER

by

Graham Trott

Editor and Cross-Assembler for the MC6800/1/2/3/8 and HD6301/3

by Graham Trott

Second Release: 1 July 1983 with Version 2.00 Software

COPYRIGHT NOTICE

SERIAL NUMBER NOTICE

This product has been assigned a unique serial number at the time of manufacture. The ASCII code for this serial number, which is encrypted into the body of the product, is also part of the start-up banner. This product is is therefore traceable to the original purchaser in the event of plagiarized copies being discovered.

This product is sold on the basis of being used on a SINGLE microcomputer system by a SINGLE user.

We shall consider it to be an attempt to criminally plagiarize us if duplicate copies of this manual or the accompanying disk are made available for use by other parties, or on other microcomputers. This consideration also applies to, but is not limited to, duplicate copies being produced for use within the original purchasers organisation, establishment, or home for anything other than archival purposes.

WARNING

We at Windrush Micro Systems Limited and the author Graham Trott consider the recognition we receive as a result of the sale of our programs and manuals to be of vital importance in remaining in business.

Unless written arrangements to the contrary have been made between authorized agents of Windrush Micro Systems Limited and the purchaser of this manual and the accompanying computer program we shall consider it to be an attempt to criminally plagiarize us if our company name, the program name, .or the authors name is altered, changed or removed, on or from any of the materials purchased from us regardless of the means by which accomplished. This consideration shall include, but not be limited to, the re-writing of this manual, or its reproduction for distribution under another company, program or trade,name, or any like modification of the accompanying computer program.

WARRANTY NOTICE

Although every effort has been made to insure the accuracy of this material, it is sold AS IS and without warranty. No claim as to the suitability or workability of this material for any particular application or on any particular computer is made. This statement is in lieu of any other statement whether expressed or implied,

Editor and Cross-Assembler for the MC6800/1/2/3/8 and HD6301/3

by Graham Trott

TABLE OF CONTENTS

1.0 INTRODUCTION

This is the second major release of XMACE for the MC6800/1/3. With this release we have added the capability to handle the extra instructions avialable in the Hitachi HD6301 (CMOS 6801). We have also added many new features to the co-resident editor and have improved the disk file handling. In addition the assembler portion of XMACE may now be called from the FLEX command line

ACKNOWLEDGEMENT

Our thanks to Neil Jarman for his efforts in improving the file handling capabilities and editor features of our 6809 assembler MACE. The author has subsequently incorporated these enhancements into this product.

XMACE is a combined editor and assembler designed to enable the user of a FLEX9 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 a rapid edit-assemble-test cycle that is very valuable when in the primary program debugging phase.

Since the object code produced by XMACE is native 6800/1/3 or 6301 code it cannot (for the most part) be executed within the 6809 development system. This will tend to make program debugging somewhat more difficult than normal.

If you are willing to foresake the expanded instruction set available for the 6801/3 and 6301 (i.e. restrict yourself to the 6800 instruction set) it is possible to do ALL of your program development work using MACE, our 6809 assembler. If you take this course of action the bulk of the program development and debugging can be done within your 6809 development system. Once the program is debugged it can then be cross-assembled using this product.

If you require the expanded instructions you will have little recourse but to use XMACE to cross assemble your program and then either program it into EPROM or download it into the target hardware via a serial or parallel link.

Note to beginners:

XMACE is quite forgiving of mistakes so there is no need to understand the whole of this manual (nor all the 6800 instructions); just type in what you think is right and XMACE 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 XMACE gives when it doesn't like what it sees.

TRADEMARK NOTICE

2.0 THE XMACE EDITOR

One of the strongest features of XMACE 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. he 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.

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 XMACE 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 SETXMACE program, described in section six, provides a convenient method of altering the keycodes XMACE recognizes to those available on your terminal.

XMACE 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 XMACE'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 XMACE.

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 XMACE 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.

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 lines specified by the command, and defaults to one if none is given.

<#TARGET> represents the decimal line number specified 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 XMACE.

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.

MODE CONTROL

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 XMACE through the warm start address.

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                              *
*                                                              *
* NEVER RE-ENTER XMACE 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 XMACE, 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 SETXMACE 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.

LINE POSITIONING COMMANDS

<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
would back up 23 lines and print 23 lines.

<CR>
Display the current line.

<ESCAPE>

Hitting the escape key wilt 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 witt get /EOF printed every time.

FILE ORIENTED COMMANDS

N

NEW file. This command erases the file currently in memory to make room for a new file. XMACE 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 XMACE, 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 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.

LINE EDITING

O<CHAR>

OVERLAY the current line. This command is useful when a change has to be made near the start of a line. XMACE 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 XMACE 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;
    #
```

GLOBAL EDITING

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 XMACE to find the next 23 occurrences of the previously defined <STRING1>, while C8;VAR (note the missing second detimiter) 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. XMACE 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
```

Whenever you specify a filename you can give any combination of drive number, filename and extension, and XMACE 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), and (A:L) 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 (LIS) 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) defaults can be altered is with the SETXMACE command.

The (A:L) 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<CR>                  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 extensions will be updated accordingly.
```

DISK FILE HANDLING

L[=<FILENAME>]

LOAD a disc file. The default drive and file extension specified when configuring XMACE with the SETXMACE 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 SETXMACE.

PLEASE NOTE

XMACE does not make backup copies of files; if you require a backup you must create it explicitly (e.g. S=FILE.BAK). If the fitename 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 XMACE 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 SETXMACE.

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 XMACE 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 XMACE. 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 $2200/1. 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 $2202/3 to the memory address just noted. Now warm start XMACE 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 $2202/3 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 XMACE, i.e. +++GET,XMACE.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 time in the file. Insert this address into to memory location $2202/3.

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

(7) Warm start XMACE 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 XMACE ASSEMBLER

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

The syntax used by XMACE conforms, in general, to the Motorota standard, with several enhancements and a few restrictions. Most existing programs will therefore assemble with a minimum of changes. The syntax of XMACE 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 (*) is treated by the assembler as a comment line.

Comments may be placed on any line, immediately following the operand (if any). If the length of the comment is such that it causes the total assembled line length to exceed the figure specified by the SETXMACE command (see section 6) then it wilt be truncated in the assembly listing.

3.2 LABELS

XMACE 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

XMACE handles either 680012/8 mnemonics only or may be extended to include the 6801/3 and the 6301/3. To select which use the SETXMACE command to configure your copy of XMACE as required.

If you select the 6800 (non extended) instruction set when using SETXMACE you can use the (A:E) command to override the default setting and optionally use the extended instruction set.

When the extended instruction set is selected the following additional mmenmonics for the MC6801/3 are available

ABX Unsigned addition of Accumulator to Index Register

ADDD Add (without carry) Accumulator 'D' to memory and leave the sum in Accumulator 'D'

ASLD Shift Accumulator 'D' left (towards MSB) one bit; the LSB is cleared and the MSB is shifted into carry.

SRN Branch Never

LDD Load the 'D' accumulator from memory.

LSRD Shifts the 'D' Accumulator right (towards LSB) one bit; the MSB is cleared and the LSB is shifted into carry.

MUL Unsigned multiply; Multiplies Accumulator 'A' with Accumulator 'B' and leaves the product in the double Accumulator 'D'.

PSHX Pushes the Index Register onto the Stack.

PULX Puls the Index Register from the Stack.

STD Stores the 'D' Accumulator to memory.

SUBD Subtracts the contents of memory from the 'D' Accumulator and leaves the difference in the 'D' Accumulator.

In addition the following special mnemonics are available:

HCF $4E (6801 only). Halt and Catch Fire.

SKIP1 $81 (CMPA) 6800, $21 (BRN) 6801

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

3.3 MNEMONICS continued

When the extended instruction set is selected the following additional mnemonics are available for the HD6301/3:

AIM	P,M	AND location M with pattern P	DIRECT
AIM	P,M,X		INDEXED
OIM	P,M	OR ditto	
OIM	P,M,X		
EIM	P,M	EOR ditto	
EIM	P,M,X		
TIM	P,M	Bit test loction M using mask pattern P	
TIM	P,M,X		
SSET	B,M	Set bit B of location M	DIRECT
BSET	B,M,X		INDEXED
BCLR	B,M	Clear ditto	
BCLR	B,M,X		
BTGL	B,M	Toggle ditto	
BTGL	B,M,X		
BTST	B,M	Test ditto	
BTST	B,M,X		
XGDX		Exchange D and X	
SLP		Go to sleep (enter power-down mode)	

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 *

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 rather than absolute addresses. FRA LABEL is equivalent to FDB LABEL-*. As an aid to finding ones way around the program the absolute value of LABEL will be printed in the same way as the destination of a branch, e.g. ($XXXX).

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).

RMS 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).

3.6 INVOKING THE ASSEMBLER

The XMACE 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 XMACE 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:E

Select the extended instruction set if not set by the SETXMACE command. This option can enable the extended instruction set mode if you have selected the non extended mode with SETXMACE. If the extended mode is not enabled all non-6800/2/8 instructions will cause errors to be reported during assembly.

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. This option only makes sense if used with the T,P, or L 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.

3.6 INVOKING THE ASSEMBLER continued

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.

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 program counter value (as printed on the object listing).

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.

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.

NOTE: Only one of the listing options (T, P or L) may be in force at any given time. If more than one is specified the first one specified will be used and the remainder ignored

CALLING THE ASSEMBLER FROM FLEX

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

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

Assemble the file specified reporting any errors.

```
+++XMACE,FILENAME<CR>
```

Assemble the file specified using the default drive number and file extension that XMACE was configured for using the SETXMACE command.

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

Assemble the file specified to the system console.

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

Assemble the file specified to the system printer.

```
+++XMACE,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 SETXMACE defaults.

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

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

```
+++XMACE,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 SETXMACE. Also produce an object file on disk in the same manner.

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

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

NOTE: Only one of the listing options (T, P or L) may be in force at any given time. If more than one is specified the first one specified will be used and the remainder ignored.

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 is created having the following contents:

```
INTRO
MAIN
IOSUBS
```

To assemble the program, use the command A:S=ASM (with any other options that may be required). 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 assemble by itself, these variables are declared in the form

```
TEMP     RMB 2
POINTER  RMB 2
BUFFER   RMS 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
```

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.

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 XMACE 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 XMACE 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 LDY.

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.

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.).

6.0 CONFIGURING XMACE TO YOUR SYSTEM HARDWARE ENVIRONMENT

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

```
*****   XMACE Configuration Program   *****
        ===========================

      For use with XMACE version 2.0X.

This  program  allows you to configure XMACE
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>


                 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


                  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

How many columns are supported?
...................... (currently 132)?       <-------------- number then <CR>

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

6.0 CONFIGURING XMACE TO YOUR SYSTEM HARDWARE ENVIRONMENT continued

INSTRUCTION SET

XMACE can be set to handle the entire 6800 instruction set or may be extended to handle the 6801/6301 instruction sets.

If you do not specify the extended set an error will result when 6801/6301 mnemonics are encountered.

```
Do you want the extended instruction set?     <-------------- 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

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 XMACE is now configured!
=====================================

+++

6.1 XMACE MEMORY MAP

XMACE uses system memory as follows:

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

  ^     +-----------------+  <------- "MEMORY FREE ABOVE"
  |     |     SYMBOL      |
  |     |     TABLES      |
  |     |-----------------|
                  |     |     SOURCE      |
  |     |     PROGRAM     |
  |     |-----------------|  <------- APPROXIMATELY $2880
  |     |      XMACE      |
  |     | VARIABLES/STACK |
$2010   |-----------------|
  |     |     XMACE       |
  |     |   ASSEMBLER     |
$0000   +-----------------+
```

NOTE: XMACE makes use of the FLEX INCHNE vector located at $D3E5. This vector is supposed to point to the system input character, never echo routine.

ALL versions of GIMIX, SSB, and SWTP FLEX we have tested have this vector implemented correctly. Other versions of FLEX, most of which are integrated into 'HYBRID' systems, may not have implemented this vector.

If XMACE refuses to respond to your system console or crashes when it starts up and you do not own one of the above systems the problem is probably caused by the failure of whoever implemented the FLEX for your system to comply with the TSC standards!

The simplest solution to the problem is to find the location of the INCHNE routine in your system monitor (most monitors have one) and put its address into $D3E5. Remember that $D3E5 is part of a VECTOR table NOT a JUMP table so only the address of the routine goes into $D3E5/6.

Once you have put the address of the INCHNE routine into the INCHNE vector $D3E5/6 using your system monitor return to FLEX and do the following:

1. FORMAT a fresh disk and place it in drive #1.
2. +++COPY,0.ERRORS.SYS,1<CR>
3. +++SAVE,0.INCHNE.PAT,D3E5,D3E6,CD00<CR>
4. +++APPEND,0.FLEX.SYS,0.INCHNE.PAT,1.NEWFLEX.SYS<CR>
5. +++LINK,1.NEWFLEX.SYS
6. +++COPY 0,1

You will now have a new FLEX system disk that will have the INCHNE vector implemented.

7.0 COMPATIBILITY

Although XMACE 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 XMACE can be transferred to other systems, the following points should be noted:-

1. Use labels of no more than six characters in length and not containing periods. Do not use local labels at all.

2. Avoid using SKIP1, SKIP2, CON, NOC, FCS or EXT mnemonics, since their meanings may vary from one system to another.

3. The argument of the NAM directive is often restricted to a maximum of six characters.

THIS PAGE INTENTIONALLY LEFT BLANK

XMACE EDITOR COMMAND SUMMARY

SYMBOLS

<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 specified by the command, and defaults to one if none is given.

<#TARGET> represents the decimal line number specified by the command.

MODE CONTROL

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.

LINE POSITIONING COMMANDS

<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.

XMACE EDITOR COMMAND SUMMARY

FILE ORIENTED COMMANDS

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.

LINE EDITING

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

GLOBAL EDITING

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

DISK FILE HANDLING

?	Display the last fitename 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.

XMACE ASSEMBLER COMMAND SUMMARY

A	Assemble only showing errors.
A:E	Assemble enabling the extended instruction set.
A:N	Assemble with symbol table only.
A:X	Assemble with cross reference only.
A:T	Assemble with a listing on the terminal.
A:P	Assemble with a printer listing.
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:L,X	Assemble to the default listing file and generate cross reference table.

CALLING XMACE 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.

NOTE: Only one of the listing output directives (T), (P), or (L) may be in force at any one time. If more than one is given the assembler will take the first option and ignore the rest.

ASCII	HEX	BINARY	DEC	OCT
NUL	$00	0000 0000	000	000
SOH	501	0000 0001	001	001
STX	$02	0000 0010	002	002
ETX	$03	0000 0011	003	003
EOT	$04	0000 0100	004	004
ENQ	$05	0000 0101	005	005
ACK	$06	0000 0110	006	006
BEL	$07	0000 0111	007	007
BS	$08	0000 1000	008	010
HT	$09	0000 1001	009	011
LF	S0A	0000 1010	010	012
VT	$08	0000 1011	011	013
FF	$0C	0000 1100	012	014
CR	$0D	0000 1101	013	015
SO	S0E	0000 1110	014	016
S1	$0F	0000 1111	015	017
DLE	$10	0001 0000	016	020
DC1	$11	0001 0001	017	021
DC2	$12	0001 0010	018	022
DC3	$13	0001 0011	019	023
DC4	$14	0001 0100	020	024
NAK	$15	0001 0101	021	025
SYN	$16	0001 0110	022	026
ETB	$17	0001 0111	023	027
CAN	$18	0001 1000	024	030
EM	$19	0001 1001	025	031
SUB	$1A	0001 1010	026	032
ESC	$18	0001 1011	027	033
FS	$1C	0001 1100	028	034
GS	$1D	0001 1101	029	035
RS	$1E	0001 1110	030	036
US	$1F	0001 1111	031	037
SP	$20	0010 0000	032	040
!	$21	0010 0001	033	041
"	$22	0010 0010	034	042
#	$23	0010 0011	035	043
$	$24	0010 0100	036	044
%	$25	0010 0101	037	045
&	$26	0010 0110	038	046
'	$27	0010 0111	039	047
(	$28	0010 1000	040	050
)	$29	0010 1001	041	051
-	$2A	0010 1010	042	052
+	$23	0010 1011	043	053
,	$2C	0010 1100	044	054
-	$2D	0010 1101	045	055
.	$2E	0010 1110	046	056
/	$2F	0010 1111	047	057
0	$30	0011 0000	048	060
1	$31	0011 0001	049	061
2	$32	0011 0010	050	062
3	$33	0011 0011	051	063
4	$34	0011 0100	052	064
5	$35	0011 0101	053	065
6	$36	0011 0110	054	066
7	$37	0011 0111	055	067
8	$38	0011 1000	056	070
9	$39	0011,1001	057	071
:	$3A	0011 1010	058	072
;	$38	0011 1011	059	073
<	$3C	0011 1100	060	074
=	$3D	0011 1101	061	075
>	$3E	0011 1110	062	076
?	$3F	0011 1111	063	077

ASCII	HEX	BINARY	DEC	OCT	ASCII	HEX	BINARY	DEC	OCT
@	$40	0100 0000	064	100	'	$60	0110 0000	096	140
A	$41	0100 0C01	065	101	a	$61	0110 0001	097	141
8	$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	$48	0100 1011	075	113	k	$68	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	100	156
O	$4F	0100 1111	079	117	o	$6F	0110 1111	101	157
P	$50	0101 0000	080	120	p	$70	0111 0000	102	160
Q	$51	0101 0001	081	121	q	$71	0111 0001	103	161
R	$52	0101 0010	082	122	r	$72	0111 0010	104	162
S	$53	0101 0011	083	123	s	$73	0111 0011	105	163
T	$54	0101 0100	084	124	t	$74	0111 0100	106	164
U	$55	0101 0101	085	125	u	$75	0111 0101	107	165
V	$56	0101 0110	086	126	v	$76	0111 0110	108	166
W	$57	0101 0111	087	127	w	$77	0111 0111	109	167
X	$58	0101 1000	088	130	x	$73	0111 1000	110	170
Y	$59	0101 1001	089	131	y	$79	0111 1001	111	171
Z	$5A	0101 1010	090	132	z	$7A	0111 1010	112	172
[	$58	0101 1011	091	133	{	$7B	0111 1011	113	173
\	$5C	0101 1100	092	134	|	$7C	0111 1100	114	174
]	$5D	0101 1101	093	135	}	$7D	0111 1101	115	175
^	$5E	0101 1110	094	136	-	$7E	0111 1110	116	176
_	$5F	0101 1111	095	137	DEL	$7F	0111 1111	117	177