# User Guide to ALSCRIPT - Sequence alignment to PostScript Version 2.03

*Geoffrey J. Barton*

Laboratory of Molecular Biophysics
University of Oxford
Rex Richards Building
South Parks Road
Oxford OX1 3QU
U.K.

Tel: (44) 1865-275368
Fax: (44) 1865-510454
e-mail: gjb@bioch.ox.ac.uk

REFERENCE:
Barton, G. J. (1993),
ALSCRIPT a tool to format multiple sequence alignments
Protein Engineering, Volume 6, No. 1, pp.37-40.

# Contents

# 1 Update History

```
VERSION 1.0   19 June     1992
Version 1.1   26 June     1992
Version 1.2   21 October  1992: Add multiple blocks per page option.
Version 1.3   15 November 1992: First Distribution.
Version 1.4    6 December 1992: Add Colour commands.
Version 1.4.1  1 February 1993: Small bug fixes - FULL RELEASE VERSION.
Version 1.4.2 15 February 1993: Make silent_mode toggle.
Version 1.4.3  1 March    1993: Fix bug in colour option.
Version 1.4.4 24 March    1993: Add mask features (should be version 1.5).
Version 1.4.5 25 May      1993: Include alsnum program in distribution.
       7 June     1993: Fix NO_NUMBERS bug in documentation
                              Change defaults for -q option to use MASK.

Version 2.0   23 May    1995: Numerous changes and additions including the
option to colour backgrounds
differently, ommission of idents on
second and subsequent lines, helix,
strand and other special characters,
```

```
relative numbering, error checking of
ranges on input, bounding box,
screening, conservation colouring...
Version 2.03   5 June     1996: Small bug fixes - patches incorporated.
BACKGROUND_REGION and BOUNDING_BOX commands moved to
step 1 section.
```

# 2 Read This First - VERSION 2.0

This manual describes an interim release of ALSCRIPT that includes many additional features over the previously distributed Version 1.4.5. I had hoped to make a lot more changes and improvements before distributing the new version, however I have not had the time to do this. I am distributing Version 2.0 since the new features have been used in a number of published alignment figures. Please see see the Section 9 for details of the new features.

# 3 Related Programs

The AMPS package (Barton, 1990). This performs multiple sequence alignments and databank scanning.
AMAS (Livingstone and Barton, 1993, CABIOS, 9, 745-756). Analysis of Multiply Aligned Sequences. This package uses a sophisticated set-based method to identify patterns of residue conservation in multiple sequence alignments.
All programs are available by anonymous ftp from geoff.biop.ox.ac.uk. Please see the README file for details licencing and registration. You can read manuals for the programs and some related papers at http://geoff.biop.ox.ac.uk/.

# 4 Availability

ALSCRIPT is available free of charge for academic and non-commercial purposes. Distribution is by anonymous ftp from geoff.biop.ox.ac.uk. See the README file on the ftp server for details. You need to register with G. J. Barton before downloading the software.

# 5 Installing ALSCRIPT

See the appropriate section for the computer type you are using:
For PCs see see Section 15.4. For Unix see see Section 15.6 . For VMS see see Section 15.7 .

# 6 Brief Description of ALSCRIPT

ALSCRIPT takes a multiple sequence alignment in AMPS (Barton & Sternberg, 1987, Barton, 1990) block-file format and a set of formatting commands and produces a PostScript file that may be printed on a PostScript laser printer, or viewed using a PostScript previewer (e.g. Sun Microsystem's PageView program). CLUSTAL and GCG format multiple alignment files may also be used (see below). ALSCRIPT is NOT a multiple sequence alignment program, nor is it an alignment editor.
Given a block-file and pointsize (character width/height), ALSCRIPT calculates how many residues can be fitted across the page, and how many sequences will fit down the page, it then prints the alignment at the chosen pointsize on as many pages as are needed. Running ALSCRIPT with a smaller or larger pointsize will automatically re-scale the alignment to fit on fewer or more pages as appropriate. The actual page dimensions may be re-set to any value, so if you have access to an A3 PostScript printer, or typesetting machine, alignments can readily be scaled to maximise the available space.
Each output page has three basic regions. The left hand edge contains identifier codes for each sequence. The main part of the page holds the alignment, and the top part, the position numbers and tick marks. ALSCRIPT commands make use of a character coordinate system for font changes, and other formatting commands. Thus, any residue in the alignment may be referred to by its sequence position number (x-axis) and sequence number (y-axis), similarly, ranges of residue positions, or sequences may also be defined in the character coordinate system.
The basic ALSCRIPT commands allow the following functionality:
Fonts: Any PostScript font at any size may be defined and used on individual residues, regions or identifier codes.
Boxing: Simple rectangular boxes may be drawn around any part of the

alignment. Particular residue types may be selected and automatically "surrounded" by lines. For example, if the characters 'G' and 'P' are selected, then lines will not be drawn between G and P characters, but only where G and P border with other characters.
Shading: Grey shading of any level from black to white may be applied to any region of the alignment, either as a rectangular region, or as residue specific shading. e.g. "shade all Cys residues between positions 6 and 30"
Text: Specific text strings may be added to the alignment at any position and in any font or font size.
Lines: Horizontal or vertical lines may be drawn to the left, right, top or bottom of any residue position or group of positions.
Colour: Characters or character backgrounds may be independently coloured.
The example block file "example1.blc" and command file "example1.als" illustrate most of these commands in action.
Although written with the aim of producing figures for journal submission, ALSCRIPT may be used as a tool for interpreting multiple sequence alignments. For example, the boxing, shading and font changing facilities can be applied to highlight amino acids of a particular type and thus draw attention to clusters of positive or negative charge, hydrophobics, etc.

# 7 New Features in Version 1.4.4

This version introduces the MASK family of commands which allows boxing, shading etc to be applied according to the frequency of occurence of the character types at each position in the alignment. For example, it is possible to box positions where one character is seen in more than N of the sequences. It is also possible to box/shade etc the most frequently occurring character at each position. Commands exist to select which characters will be used in the calculation of frequencies and which will be excluded, thus boxing can be based upon two or more character types at a position. MASK commands also exist to show residues identical to one sequence in the set. See the section on MASK below for details.
NOTE: Although boxing according to the frequency of amino acids seen at a position is a popular method of representation it is not usually the most informative. An analysis that takes into account the physico-chemical properties of the amino acids and also relates the amino acid similarities to the

overall similarity between the sequences is more helpful in identifying functionally important residues. The AMAS program (Livingstone and Barton, 1993) applies a flexible hierarchical set-based approach to this problem.

# 8 New Features in Version 1.4.5 - Program alsnum

Version 1.4.5 includes the program "alsnum". This is a temporary solution to the residue numbering problem. Ultimately, these functions will be included as alscript commands.
alsnum creates a set of TEXT commands that can be incorporated into an alscript command file to place sequential numbers at any position in the alignment. The numbers ignore gaps, so the numbering will correspond to the specific sequence position rather than the alignment.
To use the program:
1. Decide where you want the numbers to be placed. For example, you might want the numbers above the third sequence in the alignment. If so, make an extra sequence space above the third sequence using the ADD_SEQ command.
2. Decide what is the number of the first residue of the sequence to be numbered. This will not always be 1 since you may be aligning fragments or domains.
3. Decide the numbering interval (e.g. every 10th amino acid).
4. Run the program.
For example, if you want to add numbers according to sequence 37 of a block file (junk.blc), calling the first residue of the sequence 12, and with an interval of 5, and the numbers are to be placed at the location of sequence 3 in the alignment. Type:
alsnum 37 12 5 3 < junk.blc > junk.text
5. Add the resulting TEXT commands from junk.text to your alscript command file.

# 9 New Features in Version 2.0

Error checking is now done on all ranges input. If you run ALSCRIPT 2.0 on a file that worked with ALSCRIPT 1.4.5, and it complains about out of range numbers, then check your ranges carefully. If you think you are right, then send me a minimal example of the problem and I will investigate. Versions of ALSCRIPT before 2.0 would often work happily with out of range numbers and produce perfectly OK output.
The files **ipns.als** and **ipns.blc** show example command and block file that use most of these new commands. See the *examples* directory.

## 9.1 New Step 1 Commands

SCREENSIZE 120
Usually you should not need to change this value, it alters the screening used by the printer. A value of 120 is used by default. On most 300dpi black and white printers this gives much smoother greys than the default used in earlier versions of ALSCRIPT.
PIR_SAVE filename
Will cause the block file to be saved into the file "filename" in PIR format. This can be useful for moving block file alignments to other programs.
MSF_SAVE filename
Will cause the block file to be saved into the file "filename" in something that approximates GCG .msf format. *Warning! This has not been fully tested.*
NUMBER_COLOUR 4
Sets the colour used for numbering at the top of the alignment (no American spelling at the moment). In this example, colour number 4 has been defined (See the DEFINE_COLOUR command if you are not sure what this means).
SINGLE_PAGE
If this is set, ALSCRIPT assumes everything will be plotted on one page. At the moment, all this does is write the bounding box for the figure, so encapsulating the PostScript. This *may* allow the output of alscript to be imported into word processors etc, but probably not all.
ID_ONLY_ON_FIRST_LINE
If this is present, then sequence identifiers will only be printed on the first line of the alignment. Often this looks better for small alignments than the default.

BACKGROUND_COLOUR 7
Sets the colour used for the background to the alignment. This can be useful for preparing figures for projection. At the moment this only works reliably when the SINGLE_PAGE is also set.
BOUNDING_BOX x y x1 y1
Defines the bounding box for the figure. This is set in points (1/72 inch). NOTE: In version 2.0 this was a STEP 2 Command.
BACKGROUND_REGION x y x1 y1
Defines the region to colour as background - the default is set up for A4 paper so US users may have to fiddle with this. Values are points (1/72 inch). NOTE: In version 2.0 this was a STEP 2 Command.

## 9.2 New Step 2 Commands

COLOUR_TEXT_REGION x y x1 y1 colour
Sets the colour for TEXT command output. Similar syntax to COLOUR_REGION, FONT_REGION etc.
COLOUR_LINE_REGION x y x1 y1 colour
Set the colour for LINEs in a region.
CALCONS x y x1 y1
Calculate conservation values according to Zvelebil *et al.* for the designated region. (See Livingstone & Barton 1993 for details and further refs)
MASK CONSERVATION cutoff
If CONSCAL has been used, then mask residues according to the conservation cutoff.
e.g. MASK CONSERVATION 10 would mask all identities, MASK CONSERVATION 6 would mask reasonably conserved positions. See examples for more on this command.
HELIX x1 y x2
Draw a helix from x1 to x2 of sequence y.
STRAND x1 y x2
Draw a strand from x1 to x2 of sequence y.
COIL x1 y x2
Draw a coil (horizontal line) from x1 to x2 of sequence y.
RELATIVE_TO ¡seqnum¿ ¡startnum¿
Set reference numbers to work relative to sequence number ¡seqnum¿. This means that in all subsequent commands, ALSCRIPT will translate your x

values into absolute position values in the alignment. This is *extremely* useful since you can annotate your alignment using your favourite sequences as a reference point. You no longer have to translate every x position into the alignment position.
¡startnum¿ is optional. If present, it specifies what the first residue in the displayed sequence is. For example, you may be showing residues 200-500 of a sequence, so ¡startnum¿ would be 200 rather than the default of 1. *Warning - this is a very new feature and bounds checking is not fully enabled for it.*
You can use RELATIVE_TO several times in the command file to annotate different sequences. RELATIVE_TO 0 resets to the "normal" alignment numbering.

## 9.3 New special TEXT commands

Some special TEXT commands have been added to allow drawing of alternative shapes etc. In fact this is how the HELIX, STRAND and COIL commands are implemented. The text commands are all prefixed by an @ symbol.
e.g. TEXT 3 6 "@fuparrow"
will draw a filled up arrow at position 3,6.
The alternative text commands are:
@leftarrow - an open left pointing arrow.
@fleftarrow - a filled left pointing arrow.
@uparrow - an open up pointing arrow.
@fuparrow - a filled up pointing arrow.
@downarrow - an open down pointing arrow.
@fdownarrow - a filled down pointing arrow.
@circle - an open circle.
@fcircle - a filled circle.
I plan to make this option more flexible in the near future.

# 10 Running ALSCRIPT

## 10.1 Basic Use

I recommend you read through this section, then scan the commands in Section 15.1 to get a feel for what ALSCRIPT can do.
See Section 15.8for alternative methods of invoking ALSCRIPT. In this section, the interactive method is described. The QUICK START method shown in Section 15.8 is useful to format a sequence alignment quickly using standard pointsize and shading.
ALSCRIPT is designed to work with AMPS block file format multiple alignments. If you have a multiple alignment generated by CLUSTAL V or the GCG package, then it must be translated to AMPS block file format.
To translate a GCG .MSF file: Type: msf2blc. To translate a CLUSTAL PIR format file, or any PIR format file: clus2blc.
Both programs prompt for the name of an input file, and an output block file name. A good convention to follow is to name all blockfiles with the extension ".blc".
To run ALSCRIPT simply type:
alscript
you will then be prompted for the name of the ALSCRIPT command file. Having typed the filename, the commands will be executed as you have specified.
A Simple Command File (example.als)
The file example.blc contains a small multiple sequence alignment. The following ALSCRIPT command file will convert this into a PostScript alignment file in 12 point Helvetica.

```
#Comments in ALSCRIPT command files start with a #
#
#Commands are free format - separated by blank, tab or comma characters
#
BLOCK_FILE  example.blc  #define the block file to format
OUTPUT_FILE  example.ps #where to put the result
LANDSCAPE #landscape paper orientation
POINTSIZE 12 #12 point default pointsize
DEFINE_FONT 0 Helvetica DEFAULT #set font 0 to be Helvetica
```

```
SETUP #Tell the program to get on with it.
```

Now try changing the POINTSIZE value to 5 ALSCRIPT will re-format the alignment to make best use of the available paper.
These are all STEP 1 commands - they refer to overall layout, and system settings - for example, the paper size or maximum sequence length. Other commonly used STEP 1 commands are IDENT_WIDTH which reserves more or less width for the sequence identifier codes, NUMBER_SEQS which adds a number to each sequence and LINE_WIDTH_FACTOR which allows the thickness of all boxing lines to be adjusted. See Section 15.1 for details of these and all other STEP 1 commands.
The simple example outlined above can be modified with a variety of STEP 2 commands.
for example file example2.als:

```
# FILE example2.als
#
#Commands are free format - separated by blank, tab or comma characters
#
BLOCK_FILE  example.blc  #define the block file to format
OUTPUT_FILE  example2.ps #where to put the result
LANDSCAPE #landscape paper orientation
POINTSIZE 12 #12 point default pointsize
DEFINE_FONT 0 Helvetica DEFAULT #set font 0 to be Helvetica
DEFINE_FONT 1 Helvetica REL 0.5         #set font 1 to be half sized Helvetica
DEFINE_FONT 3 Helvetica-Bold DEFAULT    #set font 3 to be Bold Helvetica
DEFINE_FONT 4 Times-BoldItalic DEFAULT  #font 4 is Times-BoldItalic
NUMBER_SEQS #Number the sequences at left hand side
SETUP #Tell the program to get on with it.
#
#step 2 commands come after the SETUP command
#
#Here are some examples...
#
SURROUND_CHARS GP ALL #draw lines around all G and P
SHADE_CHARS ILVW ALL 0.6 #shade all I L V and W with value 0.6
BOX_REGION 1 1 2 20 0.8 #rectangular box from positions 1 to 2 of sequences 1 t
```

```
FONT_CHARS C ALL 3              #Use font 3 (BOLD Helvetica) for C characters
ID_FONT ALL 1                   #set identifiers in font 1
```

There are many possible ways of combining these commands and the others shown in Section 15.1 . In general, if you apply multiple commands to the same residue, the effect of the last applied command persists where there would otherwise be a conflict. Thus the intersection of two overlapping SHADed regions would be shaded according to the second SHADE command, not some mixture of the two. Similarly for FONT commands. BOX and SURROUND commands behave in the opposite sense, all BOXing and SURROUNDing persists regardless of how many commands you issue. This makes it possible for example, to SURROUND two different sets of residues as follows:

```
SURROUND_CHARS DE ALL
SURROUND_CHARS DEHKR ALL
```

This would result in D and E characters being partitioned from the rest as well as D E H K R characters (see Example output).

## 10.2 More complex effects - Text Lines, and Masks

Text, lines and masking are meant to be used to annotate the multiple alignment. The TEXT command allows any piece of text to be located anywhere on the alignment. Clearly, however it makes little sense to superimpose the text over the alignment though this can be done! Accordingly, you must first make a space to put the text in. Usually, this will be a few lines below the multiple sequence alignment, but you may want to add text at the top, or somewhere in between two sequences. You can make space in two ways. Either by editing the block-file to introduce "dummy" sequences at the locations you want, or by making use of the ADD_SEQ command.
The ADD_SEQ command has two arguments, the sequence after which you want further sequences to be added, and how many blank sequences you need. Thus, we can reserve space for 5 lines of text underneath a 10 sequence multiple alignment with the following command.

ADD_SEQ 10 5
we can then put text below the alignment at the 20th residue.
TEXT 20 13 "Active Site His"
or any other position.
Similarly, we could draw a vertical line to point out which residue we mean
LINE LEFT 20 13 11
And change the font of the text to number 7 (whatever that has been set to):
FONT_RESIDUE 20 13 7
You can have multiple ADD_SEQ commands, but they must occur in sequence order. Thus:

```
ADD_SEQ 0 5
ADD_SEQ 5 12
```

is legal. But

```
ADD_SEQ 5 12
ADD_SEQ 0 5
```

Is NOT!! NO CHECKING IS performed by the program for this error - so beware!
Note that add_seq commands refer to the actual sequence number as implied by the block file, not the number after applying the add_seq command. Thus, for a four sequence block file, if you want to add space for three sequences before sequence 1 and two sequences after sequence 3, the commands would be:

```
ADD_SEQ 0 3
ADD_SEQ 3 2
```

Text added with the TEXT command will not be split across page breaks, so you may in some circumstances need to fiddle a little with the location/pointsize for the text to get the desired result.
Masking is a technique for drawing irregular shaped outlines, or shaded regions - this should not be confused with the MASK family of commands described below. For example a histogram can be added to the bottom of an alignment by first defining some dummy sequences in the block-file that have letters building up the shape of the histogram, then using the SURROUND_CHARS or SHADE_CHARS commands together with the SUB_CHARS

command to produce the desired effect. An example of this operation being used to show frequencies of secondary structure predictions is shown in example1.als and in the Protein Engineering paper.

# 11 Using Colour

Version 1.4 includes commands to allow the independent colouring of characters, or their backgrounds. Colours are defined in a similar manner to fonts using the DEFINE_COLOUR command (American spelling also allowed). For example:
DEFINE_COLOUR 7 1 0 0
defines colour number 7 to be red - see Section 15.1 for full details of this command. Colours 99 and 100 are pre-defined to white and black. ALSCRIPT assumes the paper colour is white.
The command to colour the text of a character or text string is:
CCOL_CHARS
the command to colour the background of a character is
SCOL_CHARS
both have similar syntax to the FONT_CHARS command.
COLOUR_REGION and COLOUR_RES have similar syntax to SHADE_REGION and SHADE_RES.
An example command file that uses colour is shown in example3.als.

# 12 The MASK command family

The idea behind the MASK command is to build up a set of character positions that will subsequently be boxed, shaded, set in a particular font, etc. For example, lets say we want to box the most frequently occuring character at each position in an alignment.
The command
mask SETUP
tells ALSCRIPT to prepare a mask.
mask FRE ALL
specifies that the most frequently occuring character at each position in the alignment will be masked. This command can be restricted to a region of

the alignment using: mask FRE sx sy ex ey, where sx etc define the region in the same way as for font_region and other commands.
mask BOX ALL
Tells ALSCRIPT to create the boxing lines that will separate the masked characters from non-masked characters - this command may also be restricted to a region of the alignment.
The mask can be reset for re-use using the command:
mask RESET
Two further commands define which characters can be used when calculating the mask. This allows gap-characters, or other amino acids to be excluded from the calculation to avoid unwanted boxing.
mask LEGAL "AVLI"
defines the AVL and I as the only characters that will be used when calculating the mask.
mask ILLEGAL ".-_"
defines .- and _ as characters that will not be used when calculating the mask.
NOTE: the blank character " " cannot be defined in this way. To avoid boxing " " characters substitute blanks for something else (using SUB_CHARS), calculate the mask, then substitute back.

## 12.1 Summary of mask commands

mask SETUP # Prepares for masking
mask LEGAL <qstring> # defines characters to include in ID or FRE calcs - optional.
mask ILLEGAL <qstring> # defines characters to exclude in ID or FRE calcs - optional.
mask ID ALL N # Calculates a mask that flags the character that occurs at least N times at a position. The word ALL can be substituted by four numbers that define a region of the alignment.
mask FRE ALL # calculates a mask that flags the most frequently occuring amino acid at each position. ALL may be replaced by four numbers defining a region of the alignment.
Multiple mask FRE or mask ID commands may be applied, using different LEGAL and ILLEGAL character definitions. In this way more complex effects can be built up.

The mask command also allows characters that are identical to one sequence to be masked.
mask AGREE ALL N
will mask all positions that are identical to the Nth sequence. Thus, for sequences that are very similar to a newly sequenced sequence, all characters identical to the new sequence can be boxed or shaded, or set in a different font or colour etc...
mask NOT ALL
allows the mask to be inverted. Thus, all positions that are NOT in the mask now form the mask. So, having done a mask AGREE, a mask NOT will allow the positions that are not identical to the selected sequence to be highlighted or substituted.
mask SUB ALL <char>
substitutes all characters in the mask with the character <char>.
mask REGION ALL applies a mask to all residues in the defined region.
The following effects can now be applied to the masked characters:
mask BOX ALL # boxes the masked residues - ie surrounds them by lines.
mask SHADE ALL <grey> # shades the masked residues by grey value.
mask FONT ALL <fontnum> # Uses font fontnum to output the masked residues.
mask INVERSE ALL # Inverts the masked characters - ie outputs them in white.
mask CCOL ALL <colnum> # outputs the masked characters in the defined colour.
mask SCOL ALL <colnum> # outputs the backgrounds of the masked characters in the defined colour.
In all commands, the word ALL can be replaced by four numbers defining the region to which the command is applied.
mask RESET # resets the entire mask for re-use

# 13 Printing ALSCRIPT Files

ALSCRIPT produces files in PostScript which may be printed on a PostScript printer (e.g. an Apple LaserWriter). If you don't have a PostScript printer, then you may still be able to use ALSCRIPT if you get hold of the GhostScript software. This is a free package that interprets PostScript commands and can

produce output on a large number of different types of printer. GhostScript runs on most hardware types (including PCs) and can also display output to the screen. The package can be obtained from many different sites on the Internet (In the UK try src.doc.ic.ac.uk).
The actual command you need to type to send a PostScript file to the printer will depend on your system. Consult your system manager for help.
Be warned, ALSCRIPT can create extremely large PostScript files if lots of boxing and shading is done on big alignments. On older printers such output may take a long time to process.

# 14 Conclusion

ALSCRIPT provides a powerful set of formatting and editing commands specifically tailored for multiple sequence alignments. It is best used in conjunction with a PostScript previewer such as Sun's PageView or GhostView since this allows the effect of changing a command to be seen quickly. In the absence of such a tool, simpler effects can be tested out without destroying too many trees in the Laser Printer!
Like most programs, ALSCRIPT is evolving as I find new problems to display, so if you have any suggestions - I shall endeavour to include them in a later version.

# 15 Appendices

## 15.1 ALSCRIPT Command Summary

WARNING: Very little error checking is performed on command input. If you give the wrong number of arguments to a command, then unexpected things may happen, or the program will crash very inelegantly. I hope to fix this in the next version of the program, in the meantime, make sure you give the correct number of arguments to each command.
All commands up to the first space character may be entered in UPPER or lower case or MiXEd case. Qualifiers for commands (e.g. REL) must be written in UPPER case.
Command Reference:

```
<int>  = enter an integer  (e.g. 240)
<float> = enter a floating point number (e.g. 0.45)
<string> = enter a string (e.g.  ARNDql)
<qstring> = enter a quoted string (e.g. "Active Site")
<char>    = enter a single character.
```

### 15.1.1 STEP 1 COMMANDS

These all refer to either system settings - e.g. the maximum allowed sequence length, or to general page layout features. e.g. the longest and shortest side of the paper on which you are plotting.

### 15.1.2 REQUIRED STEP 1 COMMANDS

BLOCK_FILE <string>
Gives the name of the file that contains the multiple sequence alignment to be formatted. File names should be fully qualified i.e. not relative to the current directory. If no block file command is given, ALSCRIPT will expect to read the block file from standard input.
OUTPUT_FILE <string>
Defines the output file name. This command should be near the beginning of the command list. e.g. OUTPUT_FILE Figure1.ps
You MUST define an output file unless the -p option (See Section 15.8 ) is used.
DEFINE_FONT <int> <string> (<int>/DEFAULT)/(REL <float>)
Defines a font to use later: e.g.

```
DEFINE_FONT 0 Helvetica 10
DEFINE_FONT 2 Times-Roman 2
```

defines font number 0 to be 10 point Helvetica, and font number 2 to be 2 point Times-Roman. Font 0 is always used as the default font. You MUST define at least the font 0 font.
DEFINE_FONT 3 Times-Roman DEFAULT
sets font 3 to be Times-Roman at whatever the default pointsize is as set by the POINTSIZE command.
DEFINE_FONT 4 Helvetica REL 0.5
sets font 4 to be helvetica at half the default pointsize.

NOTE: Font names must be written exactly as shown in Section 15.3 .
SETUP
Signals the end of the STEP 1 commands.

### 15.1.3 OPTIONAL STEP 1 COMMANDS

ADD_SEQ <int> <int>
Allows extra sequence positions to be created in an existing alignment. This permits additional annotations to be interspaced either above, below, or anywhere in the middle of an alignment. For example:
ADD_SEQ 0 10
would create an additional 10 sequences - all set to the blank character before the first sequence in the block file that has been read in.
ADD_SEQ 3 1
would add an extra sequence after sequence 3.
IMPORTANT: If you use the ADD_SEQ facility to add sequences anywhere except after the last sequence, then remember that the sequence numbers will alter. All formatting commands that follow this command must use the new sequence numbering. Thus in the first example:
ADD_SEQ 0 10
what was sequence number 1 becomes sequence 11. Sequences 1-10 are the new blank sequences to be used for annotation. Note that the sequence numbers only change for commands AFTER the SETUP command, thus, multiple add_seq commands refer to the sequence number as implied by the block file.
POINTSIZE <int>
Defines the pointsize to be used to scale the plot and space the characters. Default is 10 point.
NUMBER_SEQS
If present, then the sequence number is output with the identifier code. This is useful for finding the coordinates of residues to box or otherwise highlight.
LANDSCAPE
Specifies that alignments will be plotted with the longest paper axis horizontal. (Can get longer alignments on a page this way).
PORTRAIT
Specifies that alignments will be plotted with the longest paper axis vertical (can get more sequences on a page this way).

IDENT_WIDTH <int>
Units are characters.
Reserves <int> characters at left of every page for plotting identifiers. Note that not all this space need be used, if a smaller pointsize is used to plot out the identifier codes, than is used for the main alignment.
LINE_WIDTH_FACTOR <float>
Value greater than 0 that scales the default line width. The linewidth is obtained by multiplying the pointsize by this factor.
X_SPACE_FACTOR <float>
Y_SPACE_FACTOR <float>
This determines the spacing between adjacent residues in the X and Y directions. The spacing is calculated as: POINTSIZE + POINTSIZE * X_SPACE_FACTOR or POINTSIZE + POINTSIZE * Y_SPACE_FACTOR as appropriate. Defaults are 0.2 and 0.0 respectively.
X_SHIFT_FACTOR <float>
Y_SHIFT_FACTOR <float>
These determine the shift relative to the residue drawing position that is given to the boxing lines. The shift is calculated as follows
(POINTSIZE + POINTSIZE * X_SPACE_FACTOR) * X_SHIFT_FACTOR
similarly for Y_SHIFT_FACTOR.
The defaults are 0.3 and 0.0 respectively.
Fiddling with the X_SPACE/SHIFT values is useful to fine tune the appearance of the alignment.
MAX_INPUT_LEN <int>
Units are characters. Defines the maximum number of characters possible in the input line length. This must be greater than the maximum number of sequences (MAX_NSEQ).
e.g. MAX_INPUT_LEN 600
Increases the default value of 500 characters to 600 characters.
MAX_NSEQ <int>
Units are characters. Defines the maximum number of sequences that may be read by the program. This parameter has a large default (500). You may need to reduce it on computers with small memories.
MAX_ILEN <int>
Units are characters. The maximum length allowed for a sequence identifier code.
MAX_SEQ_LEN <int>

Defines the maximum length allowed for a sequence alignment - this may need to be reduced from the 8000 default value on smaller computers.
X_OFFSET <int> Units of points (1/72 inch).
Defines the offset along the X-axis that the alignments will be shifted prior to printing. Fiddle with this value to get a nice offset from the bottom left hand corner of the page if your page size is not A4.
Y_OFFSET <int> Units of points (1/72 inch).
As for X_OFFSET, only Y axis.
MAX_SIDE <int> Units of inches.
Defines the length of the longest side of the printer page.
MIN_SIDE <int> Units of inches.
Defines the length of the shortest side of the printer page.
VERTICAL_SPACING <int>
Defines the vertical spacing in character units between blocks of sequences when more than one block will fit on a page - default is 0.
DEFINE_COLOUR <int> <float> <float> <float>
DEFINE_COLOR
Defines a colour - the first number is a number by which the colour will be referred. The following three numbers are the intensities of red, green and blue respectively. Thus:
DEFINE_COLOUR 1 0 0.2 0.8
sets colour number 1 to be a colour with no red, 0.2 green and 0.8 blue. The exact appearance of this colour will depend on the output device. If you find suitable combinations of colours for your printer, then please let me know and I shall distribute your suggestions with the program.
DO_TICKS
If present, then tick marks are drawn below the numbers at the top of the page. Otherwise no ticks are shown.
NUMBER_INT <int>
Specifies the interval for writing residue position numbers. Default is 10
NO_NUMBERS
Switches all residue numbering off.

### 15.1.4 STEP 2 COMMANDS

All these are optional formatting commands.
IMPORTANT PLEASE READ THIS NOTE:

For those commands that accept region definitions (e.g. SURROUND_CHARS) it is easiest to think of the region being defined in terms of X and Y coordinates, where X is the sequence residue coordinate and Y is the sequence number coordinate. Thus 3 7 means the 3rd residue in sequence 7. 3 7 12 42 means the rectangular box bounded by residue 3 of sequence 7 and residue 12 of sequence 42.
SURROUND_CHARS <string> ALL
Draw lines round, but not between the characters that are in the string. e.g.
SURROUND_CHARS GP ALL
will draw lines round all G and P characters in the alignment, but not between adjacent G and P characters.
SURROUND_CHARS <string> <int> <int> <int> <int>
Similar command, but the surrounding is restricted to the region defined by the four integers.
e.g.
SURROUND_CHARS ILVW 3 12 7 32
would surround ILVW characters that occur in the region defined from residue positions 3-7 of sequences 12 to 32.
SHADE_CHARS <string> ALL <float>
Shade all characters in the <string> by the grey value given by <float>. e.g.
SHADE_CHARS GP ALL 0.5
would shade all G and P characters in the alignment by the grey value 0.5.
SHADE_CHARS <string> <int> <int> <int> <int> <float>
restricts the shading to the region defined by the four integers. Thus
SHADE_CHARS ILVW 3 12 7 32 0.7
would shade I L V and W characters from residues 3-7 of sequences 12-32 inclusive with a grey value of 0.7.
FONT_CHARS <string> ALL <int>
e.g.
FONT_CHARS GP ALL 7
would use font 7 to write out all G and P characters. Font 7 MUST have been defined using the DEFINE_FONT commands above.
FONT_CHARS <string> <int> <int> <int> <int> <int>
Similar to previous command, but restricts the effect to the region defined by the first four integers. The font must have been defined by the DEFINE_FONT command.
e.g.

FONT_CHARS ILVW 3 45 9 70 7
Would set the font to 7 for I L V and W characters for residues 3-9 of sequences 45-70 inclusive. The font must have been defined by the DEFINE_FONT command.
FONT_REGION <int> <int> <int> <int> <int>
Define the font to use throughout the region specified by the first four integers.
e.g.
FONT_REGION 3 12 20 40 10
Use font 10 for residues from residues 3-20 of sequences 12-40. The font must have been defined using the DEFINE_FONT command.
FONT_RESIDUE <int> <int> <int>
Set the font for use with a single residue position - most useful when used with the TEXT command.
e.g.
FONT_RESIDUE 3 7 2
Use font 2 for residue 3 of sequence 7. Font 2 must have been defined using the DEFINE_FONT command.
LINE <string> <int> <int> <int>
There are four commands of this type for drawing horizontal or vertical lines on the alignment.
LINE LEFT <int> <int> <int>
Draw a line to the left of the character positions indicated.
e.g.
LINE LEFT 3 12 24
Draw a vertical line starting at residue 3 of sequence 12 and ending at residue 3 of sequence 24.
LINE TOP 3 12 24
Draw a horizontal line above the character positions from residue 3 of sequence 12 to residue 24 of sequence 12.
Similar commands are:
LINE BOTTOM <int> <int> <int> Draw a line at bottom of character position.
LINE RIGHT <int> <int> <int> Draw a line at right of character position.
BOX_REGION <int> <int> <int> <int>
Draw a box around the region indicated by the four integers.
e.g.

BOX_REGION 2 5 30 7
Would box from residue 2 of sequence 5 to residue 30 of sequence 7.
SHADE_REGION <int> <int> <int> <int> <float>
Shade the region indicated by the integers with the grey value shown by the float. e.g.
SHADE_REGION 30 40 35 46 0.2
Would shade from residue 30-35 of sequences 40-46 with a grey value of 0.2.
SHADE_RES <int> <int> <float>
Shade just one amino acid with the grey value.
e.g.
SHADE_RES 3 4 0.7
Shades residue 3 of sequence 7. (Note: this can also be achieved with the SHADE_REGION command, but requires 2 extra numbers)
TEXT <int> <int> <qstring>
Place the text string at the location indicated.
e.g.
TEXT 30 70 "Active Site His"
would put the text Active Site His starting at position 30 of sequence 70. (Use FONT_RESIDUE or FONT_REGION commands to set the font of the text). Text added with the TEXT command will not be split across page breaks, so you may in some circumstances need to fiddle a little with the location/pointsize for the text to get the desired result.
ID_FONT ALL <int>
Set the font for all identifier codes to the font number shown by <int>. e.g.
ID_FONT ALL 3
Would set all the identifier codes to font 3.
ID_FONT <int> <int>
Set the font for a specific identifier to font number. e.g.
ID_FONT 12 4
Use font 4 for the identifer of sequence 12, default font for all other identifiers.
SUB_CHARS ALL <char> <char>
Substitute the characters indicated.
e.g.
SUB_CHARS ALL + *
would change all occurences of + to * in the alignment.
SUB_CHARS <int> <int> <int> <int> <char> <char>
restrict the substitution to the region shown.

e.g.
SUB_CHARS 1 1 7 8 % *
would substitute * for % from residue 1-7 of sequences 1-8. NOTE: To substitute for or with the space character use the word SPACE. e.g. to change all space characters to -.
SUB_CHARS ALL SPACE -
SUB_ID <int> <qstring>
Replace the numbered identifier by the string. e.g.
SUB_ID 34 "Predicted Secondary Structure"
would replace whatever the identifier of sequence 34 was, by the text shown. This is useful when used in conjunction with the ADD_SEQ command shown under the STEP 1 commands.
INVERSE_CHARS <string> ALL/Range (similar syntax to FONT_CHARS but no font number)
Print the selected characters in white. This clearly will only work if you first use the SHADE_CHARS command to shade the characters with something other than white.
CCOL_CHARS <string> ALL <int>
Colour all characters in the <string> by the colour defined by <int>.
e.g.
CCOL_CHARS GP ALL 12
would colour all G and P characters in the alignment by the colour 12. This colour MUST have been defined by the DEFINE_COLOUR command.
CCOL_CHARS <string> <int> <int> <int> <int> <int>
restricts the colouring to the region defined by the four integers. Thus
CCOL_CHARS ILVW 3 12 7 32 7
would colour I L V and W characters from residues 3-7 of sequences 12-32 inclusive with the colour 7.
SCOL_CHARS: This has identical syntax to SCOL_CHARS, but colours the background of the character, rather than the letter itself.
COLOUR_REGION <int> <int> <int> <int> <int>
COLOR_REGION
Colour the region indicated by the integers with the colour number given as the last number.
e.g.
COLOUR_REGION 30 40 35 46 2
Would colour from residue 30-35 of sequences 40-46 with the colour 2.

COLOUR_RES <int> <int> <int>
Colour just one amino acid with the defined colour.
e.g.
COLOUR_RES 3 4 7
Colours residue 3 of sequence 7. (Note: this can also be achieved with the COLOUR_REGION command, but requires 2 extra numbers)

## 15.2 AMPS Block file format

The first part of a block-file contains the identifier codes of the sequences that are to follow. Each code is prefixed by the > symbol, codes must not contain spaces.
e.g.

```
>HAHU
>Trypsin
>A0046
>Seq1
```

etc.
ALSCRIPT counts the number of > symbols in the beginning of the file until a * symbol is found. The * signals the beginning of the multiple alignment which is stored VERTICALLY, thus columns are individual sequences, whilst rows are aligned positions. The * symbol must lie over the first sequence. A further star in the same column signals the end of the alignment. ALSCRIPT uses the number of > symbols at the beginning of the file to work out how many columns to read from the * position. It is therefore important that the only > symbols in the file are those that define the identifiers, and the only * symbols are those defining the start and end of the multiple alignment. The block file can contain additional text, providing that there are no more > or * symbols in the file than those used to define the identifiers or alignment start and end.
A simple, small block-file is shown here.

```
>Seq_1
>A0231
>HAHU
```

```
>Four_Alpha
>Globin
>GLobin_C
*
ARNDLQ
AAAAAA
PPPPPP
PP PPP
WW WWW
LLLLLL
IIVVLL
*
```

## 15.3 PostScript Fonts

```
Times-Roman,
Times-Italic,
Times-Bold,
Times-BoldItalic,
Helvetica,
Helvetica-Oblique,
Helvetica-Bold,
Helvetica-BoldOblique,
Courier,
Courier-Oblique,
Courier-Bold
Courier-BoldOblique,
AvantGarde-Book,
AvantGarde-BookOblique,
AvantGarde-Demi,
AvantGarde-DemiOblique,
Bookman-Demi,
Bookman-DemiItalic,
Bookman-Light,
Bookman-LightItalic,
Helvetica-Narrow,
Helvetica-Narrow-Bold,
```

```
Helvetica-Narrow-BoldOblique,
Helvetica-Narrow-Oblique,
NewCenturySchblk-Roman,
NewCenturySchlbk-Bold,
NewCenturySchblk-Italic,
NewCenturySchblk-BoldItalic,
Palatino-Roman,
Palatino-Bold,
Palatino-Italic,
Palatino-BoldItalic
ZapfChancery-MediumItalic.
Symbol
```

## 15.4 386 DOS installation

IMPORTANT - The programs on this disk will ONLY WORK on a PC with a 386 or better processor. See the Technical Notes section for details of why. Directions:

1. Create a directory on your hard disk. e.g. mkdir ALSCRIPT.

2. Copy the Contents of the floppy disk into this directory.

   ```
   e.g.   copy a:*.* c:\alscript.
   ```

3. Edit your AUTOEXEC.BAT file and add

   ```
   C:\ALSCRIPT to your path.
   ```

4. Edit your AUTOEXEC.BAT file and add the following two lines. set DOS4GVM=@ALSCRIPT.VMC set DOS4G=quiet

The first line is an instruction to read instructions from the file ALSCRIPT.VMC. This sets up a permanent swap file on your hard disk. By default, the swap file is about 12MBytes in size. If you do not have this much free space on your disk, then edit the ALSCRIPT.VMC file to reduce the swap file size, or alternatively, do not put this line in your autoexec.bat.

The programs will run without this swap file, but you will be limited in the size of alignment you can process by the amount of RAM you have installed. I have only tested this program on a 486/33 with 8MBytes RAM and a 386/33 with 4MBytes so I do not know the practical limitations of machines with smaller memories. Any feedback would be appreciated.
5. Type AUTOEXEC.BAT to initialise the changes, or better still, reset the computer.
6. You should now be able to run all three programs in the package from anywhere on your disk. msf2blc, clus2blc and alscript. If you get memory allocation errors when you try to run alscript, then use the MAX_NSEQ and MAX_SEQ_LEN commands to reduce the default limits. If the program still won't run, then think about buying some more memory!!
The programs msf2blc and clus2blc should run OK, but if you try to process alignments that are too large for your computer, you may get a "malloc error" which will stop the program. If this happens and you are not using the virtual memory option discussed above, then try adding the line set DOS4GVM=@filename to your autoexec.bat file. If you don't have enough disk space to do this, then buy a bigger disk, or more memory.

## 15.5 TECHNICAL NOTES

The executables included in this package were compiled with the WATCOM C compiler. This is a full 32 bit compiler that makes good use of the 386 processor and does not work on the 16 bit 286. It also has the advantage of allowing the flat memory model to be used. In practice this means that porting programs like alscript from Unix computers like the Sun, is straightforward. In order to access the memory of the computer in this way, an extra program called a dos extender is required - this is called DOS4GW.EXE. DOS4GW is automatically invoked every time you run one of the programs and is responsible for managing the memory and creating the swap file discussed above.

## 15.6 Unix Installation

ALSCRIPT is distributed with executables for Sun (SunOS 4.1.3), Silicon Graphics (IRIX 5.3), DEC ALPHA OSF/1 and Sun Solaris (2.4). The executables are stored in the subdirectories bin/sun, bin/sgi, bin/osf and bin/sol. If these are OK for your system, then just add the apporpriate directory to

your path, or put links to /usr/local/bin or somewhere that is on all users paths.
The source code for ALSCRIPT is contained in a directory hierarchy. The top directory contains a README file and the BUILD script. Subdirectories are: **examples** which contains example command and alignment files, **doc** which contains LaTeXand PostScript copies of the manual - a subdirectory of this contains an HTML version of the manual, and **src** which contains the source code and Makefiles for the package. There may also be a directory called **bin**. If present this will contain subdirectories with executables for the programs in the package. Makefiles to build alscript, msf2blc, clus2blc and alsnum are included in the **src** directory. Versions for Sun (acc compiler .sun), Silicon Graphics (.sgi), DEC OSF/1 (.osf) are included.
There is a utility csh script called BUILD. Simply type ./BUILD sun to compile alscript on the Sun, ./BUILD sgi for Silicon Graphics or BUILD gcc for use with gcc compiler. See instructions in the file BUILD. The BUILD script will create a /bin directory and subdirecotry if not already present. You can create makefiles for different computers and the BUILD script should still function.

## 15.7 VAX/VMS Installation

The standard VAX C compiler is not ANSI. Accordingly, ALSCRIPT will require changes to the source code to compile on a VAX.
The DEC C++ compiler works OK for alscript. Alscript will also compile on Dec ALPHA under OpenVMS. A descrip.mms file is included for this purpose.
*WARNING: I've not tested Version 2.0 of ALSCRIPT on VMS*

## 15.8 Alternative ways of invoking ALSCRIPT

The documentation above describes the interactive mode of running ALSCRIPT. However, it may be more convenient to run the program as a pipe under Unix or MS-DOS. Examples are shown here.
ALSCRIPT is a program for producing pretty versions of multiple sequence aligments. ALSCRIPT will also format single sequences. A full description of the program is given in the file "alscript.doc".
Ways of running alscript:

1. Interactive mode: just type alscript. You will be prompted for a command file name. The command file will define the AMPS blocfile, and name of the file to store the PostScript output - see alscript.doc for details.

2. alscript <command_file> has same effect as 1, But does not prompt for the command file e.g. alscript example1.als

3. alscript -q < <blocfile> > <PostScript> Quick mode - uses default commands, reads alignment from stdin, writes PostScript to stdout. This mode creates a command file called ALPSQ.COM.

   e.g. alscript -q < example1.blc > example1.ps

4. alscript -f <command_file> Similar effect to 2.

5. alscript -f <command_file> -s Silent operation: No messages are written to stderr, unless fatal. Silent operation may be toggled by the silent_mode command in the command file.

6. alscript -f <command_file> -p < <blocfile> > <PostScript> Make alscript work like a pipe - blocfile is read from stdin, postscript is written to stdout. Messages are written to stderr. To supress messages include the -s flag too

   e.g. alscript -f example1.als -p -s < example1.blc > example1.ps

Using alscript as a pipe has the advantage of allowing the blocfile to be created on the fly by the programs msf2blc or clus2blc. For example if we have a GCG .msf file called "pileup.msf" we can run alscript with default shading/fonts and send the results straight to the PostScript printer "lpr" as follows:
msf2blc -q <pileup.msf — alscript -q -s — lpr

## 15.9 Program Crashes and Known Bugs

We've used ALSCRIPT on Sun Workstations and Silicon Graphics for some time, with very large alignments and command files with thousands of commands. All seems to work OK, the program has not crashed on us at all!!

However, the command interpreter in ALSCRIPT is very simple and the program will crash if you give any command the wrong number of arguments (e.g. leaving out the shade value in a shade_chars command).
If you do make the program crash, have checked all the documentation and your numbers, and the program still crashes. Then send me the command file and block file that causes the crash and I will try to investigate.
Suggestions for improvements to the program are always welcome.

## 15.10 Wish List for next version!!

A command interpreter that does more error checking will be included. Currently, no checking is done to make sure that the correct number of arguments are given to a command.
Sequences will be able to be given unique labels and region commands refer to these labels or ranges of labels. This will permit a sequence to be deleted or added to the alignment without having to update the .als file.
The relative numbering option will be extended to allow numbering relative to a position. e.g. 456+7 would be 7 residues after position 456. This will allow annotation of positions that may be in insertions relative to the reference sequence.
Special TEXT commands will be extended to allow alternative shapes to be drawn and scaled in various ways.
Tree drawing and generalised graphics. An option to draw arbitrary lines on an alignment will be added. This will permit line graphics to be added to an alignment. The initial reason for this will be to show dendrograms (trees) alongside the alignment, but simple line graphs could also be plotted under the alignment.
Fiddle factors will be introduced to allow fine positioning of individual characters. For example, if you like your "I" characters to be centred rather than left justified, this will be possible.
In single_page mode, it will be possible to add arbitrary text to an alignment for final annotation, e.g. titles etc.
Variable height/width sequence lines will be permitted (maybe).

## 15.11 Acknowledgements

I thank all those who have emailed me with suggestions for improvements to alscript. I've tried to include some of these in the current distribution (e.g. screening).

## 15.12 References

1. Barton, G. J. (1993),
"ALSCRIPT A tool to format multiple sequence alignments",
Protein Engineering, Volume 6, No. 1, pp.37-40.

2. Barton, G. J. (1990),
"Protein Multiple Sequence Alignment and Flexible Pattern Matching",
Methods in Enzymology,
183,403-428.

3. Barton, G. J. and Sternberg, M. J. E. (1987),
"A Strategy for the Rapid Multiple Alignment of Protein Sequences:
Confidence Levels From Tertiary Structure Comparisons",
Journal of Molecular Biology,
198,327-337

4. Higgins, D. G. and Sharp, P. M. (1989),
"Fast and sensitive multiple sequence alignments on a microcomputer",
CABIOS,
5,151--153

5. Devereux, J. Haeberli, P. Smithies, O. (1984),
"A comprehensive set of sequence analysis programs for the VAX",
Nucleic Acids Res.
12, 387-395

6. Livingstone, C. D. and Barton, G. J. (1993),
"Protein Sequence Alignments: A Strategy for the Hierarchical analysis
of residue conservation"

Computer Applications in the Biosciences,
9, 745-756.