1. INTRODUCTION

1.1 Thank you

Thank you for buying LIGHTNING - we hope it is going to give you a lot of enjoyment. We know you are dying to get started with it - but cartridge users please, please control your desires: make a backup copy first. Zoom to 1.4, make a backup copy, and then return. Disk users, set your LIGHTNING disk to write-protect and read on.

1.2 What is LIGHTNING?

LIGHTNING is a magic wand utility which dramatically increases the QL's efficiency by achieving output speeds that we think Sinclair could have engineered into the QL in the first place.

A dictionary definition of LIGHTNING is "a visible discharge of electricity in the atmosphere". This might not seem very relevant to a computer software package, but LIGHTNING really does produce an ELECTRIFYING and VISIBLE speed improvement. The improvement not only affects all aspects of the QL's screen handling, but also CHARGES many mathematical operations with hitherto unknown energy: they can run at 2 - 10 times the normal speed, or more. We shan't push the analogy any further!

LIGHTNING is simplicity itself to use. Once it is loaded ALL programs will AUTOMATICALLY benefit from the various enhancements it provides. The 'magic wand' works not by enhancing individual programs, but by replacing particular parts of QDOS, the QL's operating system. The substitutions are carefully designed to mimic very accurately the way the operating system normally works, except that they are optimised for SPEED, rather than for space (the native QDOS code is optimised for space, and probably more so than was necessary, but the authors of QDOS were presented with a target that Sinclair Research kept moving: we have had the benefit of a far more ordered and reasonable development schedule). Once LIGHTNING is loaded your QL will behave in exactly the same way as it did before, only FASTER!

In addition to its accurate mimicry and enhancement of QDOS, LIGHTNING includes some brand new features: amongst these is a new device, called 'nul', which can greatly simplify the construction of many programs. For example, the option to output to a printer can be taken care of in just one line of code, rather than by including an IF test with each PRINT statement. If you have a program that produces a lot of screen output, or if you want to look at a large file by COPYing it to the screen, another novel feature, the output 'drain' facility, will enable you to skip past sections of the output at up to 100x the normal speed.

Operations not optimised by LIGHTNING (typically because QDOS is already fast when performing them) are handed on to QDOS as if LIGHTNING were not present. This transparency of working is a key to LIGHTNING's magic wand behaviour - install it, forget about it, and benefit from all the enhanced speed it offers.

TURBO users may ignore references to LIGHTNING in section V of the present TURBO Encyclopaedia - the product referred to there will never see the light of day.

What LIGHTNING can do for you is described in the following sections of this manual. Right now you probably want to see LIGHTNING for yourself, so let's think about how to get it started on your QL...

CONTENTS

1. INTRODUCTION
1.1 Thank you
1.2 What is LIGHTNING?
1.3 What you get - a description of the files
1.4 Backing-up LIGHTNING
1.5 Installing LIGHTNING
1.6 Using LIGHTNING
1.7 A word of warning
1.8 Benchmarks
1.9 Commercial use of LIGHTNING

2. USE OF LIGHTNING TO ACCELERATE TEXT
2.1 How fast is it?
2.2 Text Extras - Multiple scrolls, the Drain and Mode changing
2.3 The font loader and the extra fonts
2.4 Possible problems with using strange fonts with LIGHTNING
2.5 The MINI text enhancement routines - for use where RAM is short

3. USE OF LIGHTNING TO ACCELERATE GRAPHICS
3.1 Speed and accuracy
3.2 Circles and ellipses
3.3 Off-screen plotting
3.4 ARC implementation

4. USE OF LIGHTNING TO ACCELERATE MATHS
4.1 Speed and precision
4.2 Changing precision within compiled programs
4.3 Restricted function domains for single-byte precision operation
4.4 Cotangents of angles close to zero
4.5 Accelerating the power function
4.6 The maths library with new names

5. THE NULL HANDLER
5.1 The new device nul

6. WHAT TO DO IF THINGS GO WRONG
6.1 LIGHTNING doesn't load
6.2 Characters disappear
6.3 Text is mis-positioned,' the display gets scarmbled (sic)
6.4 Characters and displays flash or are weird
6.5 Have we forgotten anything?
6.6 Do's and Dont's

7. SUMMARY OF NEW COMMANDS PROVIDED BY LIGHTNING

8. WHERE TO GO NEXT

9. GRATEFUL ACKNOWLEDGEMENTS

10. PERFORMANCE FIGURES

1.3 What you get - a description of the files

LIGHTNING addresses three areas separately for the purposes of speedup. These areas are text (including user-defined graphics, which really means non-standard characters), graphics and maths.

These are the files that are on your master cartridge or disk:

lng_TEXT_ext	The screen text enhancement routines. Normally all the modules you use will be installed automatically (once you have used the automatic configuration program which is part of the master media BOOT file): however, you can also install individual modules at any time by using one of the simple ready-made installation programs.
lng_TEXT_bas	A short SuperBASIC program to install lng_TEXT_ext individually.
lng_GRAF_ext	The screen graphics enhancement routines.
lng_GRAF_bas	Individual installation program for lng_GRAF_ext.
lng_MATH_ext	The mathematical enhancement routines. There are two versions of these in LIGHTNING: this is the version that replaces the relevant routines in QDOS with new routines having the same names (eg; our SIN replaces Sinclair's SIN) so that programs will automatically benefit from the enhancements. It is possible to switch back to the original QDOS routines in many cases - but why should you want to?
lng_MATH_bas	Individual installation program for lng_MATH_ext.
lng_TEXT_mini_ext	A special cut-down version of lng_TEXT_ext, principally for use where RAM is scarce. This package is only 4.5K long and so leaves the maximum amount of memory free for your application programs. It doesn't give you some of the speed and bells and whistles of the full text version.
lng_TEXT_mini_bas	Individual installation program for lng_TEXT_mini_ext.
lng_MATH_newnames_ext	This is the alternative version of the maths package. The difference is that this installs the new functions with different names so that the original QDOS routines remain intact and available in the normal way. This version can only be used when programs are written specifically to use the new routines.
lng_MATH_newnames_bas	Individual installation program for lng_MATH_newnames_bas.
lng_FONT_ext	This contains a machine code routine that will allow new character fonts to be loaded and attached to any SuperBASIC screen window.

d) The program will ask you a few simple questions. If you are not sure how to answer any of these, don't worry: the default (invoked by pressing ENTER) will always be sensible.

Note that you will be instructed to put a cartridge/disk in the second drive. This need not be preformatted but it must NOT be write-protected. Cartridge users note that new cartridges are generally fine, but sometimes you may find that the write-enable tab has been removed. In this case, a small piece of tape may be fixed in place of the tab in order to write-enable the cartridge.

The copy procedure will only take a minute or two. You may want to repeat it so you have several backup media. At the end you should store the original (master) cartridge or disk in a safe place. Use only the backup(s). The only use of the master should be to make backups. The only reason for this is safety - but safety is a very good reason.

The backup procedure allows you to change the device LIGHTNING expects to find its files on. If you have been supplied with LIGHTNING on cartridge, this default will have been set to mdv1_. If you have been supplied with LIGHTNING on disk, this default will have been set to flp1_. BACKUP_bas allows you, for example, to backup LIGHTNING from cartridge to disk with the resultant copy set to run from disk. Or the other way around.

1.5 Installing LIGHTNING

It is necessary to install LIGHTNING (a process that can be made automatic, as you will see) each time your QL is switched on (or reset): once this is done it remains installed until you either deactivate it with a keyword or switch off/reset your QL.

Installing is different from backing up.

LIGHTNING is installed by the BOOT file on a "working copy" of LIGHTNING. We'll soon tell you how to make a "working copy".

In case you are confused: the only difference between your backup copy of LIGHTNING (as produced by BACKUP_bas) and the master we supplied you with is that you may have set up a different default device on the backup. In every other respect, the backup and master are clones. However, the "working copy" of LIGHTNING is a different kettle of fish altogether. The BOOT file on the backup (or master) of LIGHTNING doesn't install LIGHTNING (ie; it doesn't activate any of the text, graphics and maths enhancement routines) - instead, its function is to invoke the CONFIGURE program which then creates a BOOT file on another medium (the "working copy") that DOES install the parts of LIGHTNING you selected. Please understand this.

Let us see how to make a "working copy" of LIGHTNING. There are two methods. Note that both methods involve the automatic setting up of LIGHTNING's extensions to SuperBASIC. You don't need to know how we do this. If you do want to know, it is simply by the normal RESPR - LBYTES - CALL (or several CALLs, one for each of the LIGHTNING modules you've selected). We do this automatically - including working out how much space you need. Remember that RESPR returns the 'Not complete' error if it is used while tasks are running - this is why it is crucial that LIGHTNING is installed right at the beginning, as soon as the QL is switched on.

It isn't a catastrophe if one or more of your application programs is too big to leave enough free space on the medium. Use Method 1 (the clumsy one) described earlier to make a "working copy" of LIGHTNING, and boot from this "working copy" before using LRUN device_BOOT to start the giant.

1.6 Using LIGHTNING

Unlike most software packages, there is little to know about using LIGHTNING, since it will go about its work QUICKLY and AUTOMATICALLY, without further attention. Your only clue that it is present is that things happen faster!

If you would like to have a demonstration of how effective LIGHTNING is, LRUN any of the demonstration programs, DEMO_TEXT_bas, DEMO_GRAF_bas or DEMO_MATH_bas. These will illustrate different aspects of both screen output and mathematical operations enhancement. However, you will very soon observe the effect of LIGHTNING on ALL your normal operations and application programs.

The following sections give details of how to make certain adjustments to LIGHTNING in order to further enhance performance. For now, though, you might like to know that several new commands have been added to SuperBASIC which allow these adjustments to be made. The most dramatic of these are:

_lngOFF	to switch off all the TEXT enhancements. We can't imagine why anyone would wish to do so, other than to act as a reminder of just how sluggish the QL is without them.
_lngON	to switch them back on. That's more like it!

Please distinguish between Text on the one hand, and Graphics and Maths on the other. _lngON and _lngOFF affect text only. We've omitted any 'text' indicator from their names to keep the names short, and because we suspect for many people text acceleration will be the main use to which they put LIGHTNING.

Entering _lngON when you already have it set does no harm - ditto with _lngOFF.

Note that when you type in _lngON, we don't regurgitate any credits onto the screen. If you want to look at the name of the author of LIGHTNING, please refer to the front of the manual - you may look at his name for as long as you want!

When you install the text enhancements, their startup default is _lngON - so you don't need to explicitly enter this.

1.7 A word of warning

Copying, hiring, leasing and duplicating any part of LIGHTNING (program or documentation) for any purpose other than security backup is an offence under copyright law in the UK and abroad. It is punishable by fines, seizure and possibly even imprisonment.

2. USE OF LIGHTNING TO ACCELERATE TEXT

2.1 How fast is it?

LIGHTNING will handle ALL screen text output for character CSIZE 0,0 and CSIZE 1,0. In addition, all the possible character attributes (OVER -1, OVER 0 and OVER 1) and underlining (on or off) are enhanced. These are the sizes that virtually all software uses.

The speed increase achieved depends on several factors: for example, the choice of INK and STRIP (ie; the paper) will affect the final speed, as will OVER and UNDER. Furthermore there can be a big difference between CSIZE 0,0 and CSIZE 1,0.

The effect that your hardware can have is not to be disregarded. While all QLs use substantially the same ROM routines for text, graphics and maths, the speed that LIGHTNING can run at is affected by how "fast" the memory in your computer is - the memory that LIGHTNING has to run from, that is. Unexpanded QLs and certain makes of internal (not plug-in) RAM expansion boards are slowest. Plug-in RAM expansions (including those on disk interface boards) are better - there are many makes, and different board versions from the same manufacturer often differ significantly in speed. Fastest of all is ROM (as there are no wait-states and no time to wait for contents to be refreshed) and CST's RAM-Plus. Note that if expansion RAM is fitted to your QL, the RESPR mechanism ensures LIGHTNING will load into, and run from, "high" memory which is likely to be faster than the standard QL RAM.

The potential for the greatest speed improvement lies with CSIZE 1,0 printing, but there is another factor which is relevant when using this character size: in CSIZE 1,0 the position of a window on the screen can substantially alter the printing speed. This is because there are certain window positions for which each character happens to correspond exactly to each WORD of screen memory. The task of printing the character to the screen can be made very much simpler (and faster) in these circumstances. The default window positions for SuperBASIC channels #0 to #2 are correctly aligned so as to benefit from this effect. The rule is that the pixel position of the left-hand side of the window should be exactly divisible by 8. For example, the following window definitions will be aligned with word boundaries:

```
WINDOW 200,100,64,40
WINDOW #3,400,50,8,100
OPEN #4,scr_222x88a128x150
```

while these will not, and therefore don't allow the very fastest CSIZE 1,0 printing:

```
WINDOW 200,100,62,40
WINDOW #3,400,50,10,100
OPEN #4,scr_222x88a124x150
```

There is another factor to consider: if a border is defined it has the effect of re-positioning the window. It is the resulting position that counts. For example, the following window WILL be correctly aligned for the fastest CSIZE 1,0 printing:

```
OPEN #4,scr_222x88a124x150
BORDER #4,2,2
```

LIGHTNING speeds up the printing of text by a substantial factor, typically 5 times (and up to 15 times) the speed of native QDOS routines. However, when the printing reaches the bottom of a window it is necessary to scroll the entire contents of the window up by one line in order to make room for the next. LIGHTNING does indeed speed up scrolling, but unfortunately there is a physical limit to how fast scrolling can be done. The limit is imposed by QL hardware. Scrolling a window involves moving a lot of data about in memory: the 68008 processor isn't particularly quick in this respect. Once a window is full and each new line of printing involves scrolling, the speedup is governed more by the repeated scroll than by the text printing speed.

In order to reduce the scrolling overhead, LIGHTNING allows a modification to the way window scrolling works. If, when the window is scrolled, everything is moved up by (say) two lines, instead of one, there will be room at the bottom of the window for two lines of printing. A more important consequence is that scrolling will then only happen half as frequently as before. The result of this will be that the overall printing speed will be nearly doubled.

The command which controls this operation is _lngZIP n, where n is the number of lines of text which are moved up on each act of scrolling. You can have any positive integer value of n, up to one less than the height of the window (measured in characters) - higher values are treated as 1. However, for values of n greater than about 5 you may find output appears a bit jumpy - this could be an advantage in some situations. _lngZIP is available to users of the screen text enhancement routines - not the mini version, but the main one.

Note that the value assigned to n will affect output to ALL windows. When you start LIGHTNING it is as if an implicit _lngZIP 1 had been entered - this setting, of course, is the one to be used for exact mimicry of QDOS, which is why we made it our startup default. You can change the _lngZIP setting as often as you want. The multiple scroll only works when the text enhancements are on (ie; an _lngON situation) - single scroll is always performed when in the _lngOFF situation. However, if you toggle the text enhancements' off and on (say with _lngOFF : DO_SOMETHING : _lngON) then whatever you had set _lngZIP to last (whether after an _lngON or an _lngOFF) remains the setting - ie; _lngZIP "remembers".

The _lngZIP setting doesn't affect all scrolling, however. Firstly, it doesn't affect scrolling of character sizes that LIGHTNING does not accelerate. Also, in certain situations it would cause problems if a scroll of other than one line were to be allowed: for example, when a program expects a single-line movement. The TK2 SuperBASIC editor, ED, requires that the window contents are scrolled by only a single line when the cursor is moved beyond the top or bottom of a window. For ED, and for many other programs such as THE EDITOR and QUILL, LIGHTNING will (correctly) realise that scroll should be performed for each line irrespective of the value of n. Note that this override is intelligent: if you have other programs running at the same time as (say) ED, scrolling within these other programs will occur just as you explicitly specified - so if you had entered _lngZIP 3, their screen output will scroll three lines at a time.

It is important to remember that LIGHTNING speeds up the very act of scrolling itself - so even if the _lngZIP setting is 1, scrolling will occur roughly twice as fast.

Please note that the five methods we have suggested for changing the _lngZIP setting also apply to entering any of the other commands to control the operation of LIGHTNING and its component parts. The most common commands that you will want to enter while a program is running are _lngON, _lngOFF and some you haven't yet met - the ones to activate/deactivate the graphics routines, and to vary the precision of the maths routines. Choose between the five methods given above.

There is something else that is common to all the LIGHTNING commands. Their effect is, by and large, global, rather than specific to the program in which they were encountered. If enough LIGHTNING users make reasoned cases for "localisation" (there may be a tiny time penalty for some of the routines) we may localise some of the commands.

LIGHTNING provides another special feature that is useful when large amounts of text are being output to the screen, as well as in other situations. This is the output 'drain' feature. This simple control works as follows: If at any time you press the keys CONTROL and 'Z' (hold CONTROL down, tap Z), then you will find that no further character output to the screen will occur at all! Any program or command that attempts to output text (by text we mean characters - graphic output of circles, lines and the like is unaffected by CTRL/Z) is not stopped, though: it is just that the text output is poured down a drain by LIGHTNING. Since no output is performed (the drain simply gobbles stuff up at an alarming rate) the effective speed of the program is much greater.

One obvious use for this feature arises when COPYing a large text file to the screen. If the part you want to look at isn't at the beginning of the file, you can use the CTRL/Z switch to 'drain' output once copying has commenced. When CTRL/Z is depressed output will continue, but at about 100x the speed of screen printing. Press any key(s) (try CTRL/Z !) to re-enable printing to the screen. You shouldn't wait too long before doing this, in case you should miss the entire output!

Another possible use for this feature arises during SuperBASIC (or other high-level language) program debugging. A common ploy is to insert a number of PRINT statements at various stages of the program which provide information about the progress of the code at run-time. To avoid these statements slowing down execution use CTRL/Z to disable all output: your program will then flash along at full speed until you re-enable output by pressing any key.

If you want to alter the key that is used to disable output, typically because the key already set (CTRL/Z as we supply LIGHTNING) has some other meaning within the software you are running, enter the command:

```
_lngKEYSET
```

You will be asked to press the new key that is to be used as the output switch. This can be any single key, a CONTROL key combination or an ALT key combination. There are a few key combinations that we have not allowed you to choose in order to avoid possible conflicts with other software: if you attempt to set one of these then you will be told to select another key(s). CTRL/Z is a pretty sensible choice by us anyway - you may never need to change it! You program developers out there - please steer clear of CTRL/Z! PS: We first intended to use CTRL/O, but this gave rise to a howl from some of our beta-testers. O stood for Output. But then Z stands for Zzzzz. You can be inventive too!

Each character is made up of nine rows and eight columns of pixels. Not all of these need be printed for some character sizes (CSIZEs). In addition, whenever a character is printed, a blank row of pixels (in the appropriate STRIP colour) is printed at the top: this serves to ensure that there will always be at least a one-pixel gap between successive lines of printing.

For character sizes that are less than 8 pixels wide (like CSIZE 0,0) not all the columns are used: this implies that if a character set is defined that makes use of the full character width, then for CSIZE 0,0 printing it will NOT all be printed - the two rightmost pixel columns (7 and 8) will in fact be lost. The ROM character set avoids hassles by only using columns 2 to 6. Column 1 is omitted to ensure that there is always a gap between consecutive characters on a line.

Provided that you are careful as to which CSIZE you use with non-standard fonts, there is considerable scope for making much more interesting-looking characters. You will find at least 23 character sets, all with names ending in _font, included with LIGHTNING (not all of which are suitable for use with CSIZE 0,0 - the ones ending in _1_font only work in CSIZE 1,0). In addition we provide a font-loader utility that will enable you to install these sets (or fonts) from SuperBASIC or from within compiled programs. You can use other fonts too, "borrowed" from other programs you may own.

Note that LIGHTNING handles so-called fat fonts - ones that use more than the standard number of non-blank columns - completely automatically. You don't need to worry.

The font loader extension (_lng_FONT_ext) can either be installed automatically as part of the general installation of LIGHTNING (you can select this when you make your "working copy" - refer to 1.5), or you can install it at any time by LRUNning the installation program lng_FONT_bas with your backup LIGHTNING media in drive 1.

Once installed, the loader adds the command _lngFONT to SuperBASIC, but before you use this command you may benefit from having a bit of background information about fonts and how the QL stores them.

The character data takes the form of strings of bytes, one byte to define each row of a character. Each character takes up 9 bytes, hence the total number of bytes required to store a character set is

```
        9*( number of characters) + 2
```

The 2 must be added to leave room for some extra information that QDOS (and LIGHTNING) uses to identify characters in the set. If the standard QL ROM font is considered (which defines 96 characters) then the storage required is 96*9+2, or 866 bytes. However, there is more: a second or alternative set of characters may also be used. This is defined in the same way as the first or main set. In order to get QDOS or LIGHTNING to use an alternative character set, the character data for the alternative set must be stored in memory and must be "attached" to each channel in which it is needed. Note that as each channel can have its own character set, there is no need to use the same set on all channels.

The fonts supplied with LIGHTNING are all ready to use. The steps required to install them are outlined as follows in this example that will install the font data ALIEN_font.

```
        100 font_address = RESPR(1024)
```

17

```
ALLCAPS_1_font
▒ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTU
VWXYZ[\]^_£ABCDEFGHIJKLMNOPQRSTUVWXYZ{|}~©

ALLCAPS_font
▒ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTU
VWXYZ[\]^_£ABCDEFGHIJKLMNOPQRSTUVWXYZ{|}~©

AVANT_GARDE_1_font
▒ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTU
VWXYZ[\]↑_£abcdefghijklmnopqrstuvwxyz[|]~©

BOLD_1_font
▒ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTU
VWXYZ[\]^_£abcdefghijklmnopqrstuvwxyz{|}~©

BOLD_font
▒ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTU
VWXYZ[\]^_£abcdefghijklmnopqrstuvwxyz{|}~©

CHUNKY_font
▒ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTU
VWXYZ[\]↑_£abcdefghijklmnopqrstuvwxyz{|}~©

CLASSY_1_font
▒ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTU
VWXYZ[\]^_£abcdefghijklmnopqrstuvwxyz{|}~©

CLASSY_font
▒ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTU
VWXYZ[\]^_£abcdefghijklmnopqrstuvwxyz{|}~©

DESCENDERLESS_font
▒ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTU
VWXYZ[\]^_£abcdefghijklmnopqrstuvwxyz{|}~©

DIGITAL_font
▒ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTU
VWXYZ[\]^_£abcdefghijklmnopqrstuvwxyz{|}~©

ENGINEERING_font
▒ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTU
VWXYZ[\]↑_£abcdefghijklmnopqrstuvwxyz{|}~©

FAMILIAR_1_font
▒ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTU
VWXYZ[\]↑_£abcdefghijklmnopqrstuvwxyz{|}~©
```

2.4 Possible problems with using strange fonts with LIGHTNING

LIGHTNING will deal with all fonts in exactly the same way as QDOS. However, there is one situation where you might get into a small spot of trouble. This is to do with CSIZE 1,0 printing for which LIGHTNING uses a special copy in RAM of the font data that you load: it makes this copy automatically and invisibly. The copy will be updated whenever you output to another channel which uses a different font: consequently, any number of fonts will work fine, each associated with a given channel.

The problem arises if a program installs a font, and then after printing some characters installs another font into the SAME memory area: LIGHTNING will (wrongly) assume that the RAM copy of the font is still OK and doesn't need updating. Hence it will carry on printing using the RAM copy of the first font loaded despite the fact that you have now changed it.

In this situation there are two ways to get LIGHTNING to realise that the font has in fact changed. The first is to output a single character to another channel (in CSIZE 1,0) which isn't using your alternative font, before resuming printing to the required channel. The second way is to re-establish the ROM fonts, print any single character, THEN load and attach your new alternative and carry on from there.

It should be stressed that this problem can ONLY occur for CSIZE 1,0 printing when you attempt to use SUCCESSIVE alternative fonts on a given channel without doing any other CSIZE 1,0 output in between. Quite an unlikely set of conditions really, but at least you won't get confused should it happen..

We use special routines to speed up the printing of spaces. It is possible - just about - that somebody somewhere has written a program that redefines CHR$(32) so that it isn't associated with a space. If you use LIGHTNING's text enhancement mode with such a program, it will screw up. In the extraordinarily unlikely case that this happens, write to us with an SAE and we'll send you a fix.

2.5 The MINI text enhancement routines - for use where RAM is short

The-cut down version of the TEXT routines has been produced for use where RAM is short. Usually, this means on unexpanded QLs - but the MINI extensions will, just like the main text extensions, work on any QL. The aim was to make this version very small so that as much RAM as possible is left free for big application programs. This pruning has been done in such a way as to minimise the loss of speed (compared with the main text extensions) when MINI is used with EDITOR, QUILL and the like.

Specifically, CSIZE 1,0 printing is not enhanced by the mini text version, nor are OVER -1 or +1 (of course all these still work perfectly - but only at their usual QDOS speeds). Features such as multiple scrolls, the null handler and the output drain are not included. Further, the text routines have also been modified so as to optimise for space, wherever this can be done without much of a compromise on speedup. By and large, the mini version is almost as fast as the main version.

There are only four commands available to control lng_TEXT_mini. These are _lngON, _lngOFF, _lngDOMODE and _lng_NOMODE which all work in the same way as for the full version.

3. USE OF LIGHTNING TO ACCELERATE GRAPHICS

3.1 Speed and accuracy

lng_GRAF_ext enhances many of the standard graphics commands, both in MODE 4 and in MODE 8. In particular, the commands for POINT, LINE and CIRCLE and all their variants are improved. In addition, the FILL 1 setting (for which graphics are performed with area filling enabled) is faster too, and certain problems inherent with the QDOS implementation of it are corrected. Note that we are as wary of fixing "bugs" in the QL ROM with LIGHTNING as we were while producing our compilers - the reason for wariness is that many (misguided) programmers rely on ROM quirks for their programs to work - so when we fix the quirk, their programs stop working. But don't worry about the FILL fixes. If anyone out there has written anything actually relying on this ROM bug, get in touch!

The speed improvement achieved over the native QDOS routines varies from better than 4x improvement for point plotting to just 50% improvement for circles. However, as is also the case for text speedups, different hardware setups may give different speedups.

The LIGHTNING graphics routines, as in the case of text output, accurately mimic the way that QDOS draws various shapes. However, while the text enhancement routines never deviate from QDOS even by a single pixel, there are (very rare) occasions when the graphics routines diverge from QDOS by a single pixel.

This isn't a LIGHTNING fault. As a consequence of our accurate algorithms for constructing graphics, we plot things exactly where they should be. Sometimes QDOS gets it wrong by a pixel. Generally speaking these differences are rare, and when they occur they will be much too small to be noticed. Even if they are noticed, they are unlikely to present a problem - after all, we are drawing things where they should have been drawn by QDOS in the first place! However, if you are running an application program which somehow relies on the QDOS positioning (even if slightly inaccurate), you can instruct LIGHTNING to revert to using the native QDOS routines for all graphics functions. The command to do this is:

```
_lngGRAF n
```

where n is a non-negative integer. 0 switches off the LIGHTNING graphics routines and reverts to the QDOS ones, anything else switches LIGHTNING graphics back on. Note that _lngGRAF has no effect on the text or maths enhancement routines - if these were activated when _lngGRAF is used, they will remain activated even after the graphic routines are switched off. And the other way round too. It's quite simple really - all the ON/OFF controls we have provided are entirely distinct.

In the implementation of area fill QDOS sometimes doesn't fill the area requested exactly. This effect is most easily seen in MODE 8: try running the following simple program with and without the LIGHTNING graphics activated:

```
100 REMark - Demo of corrected FILL function
110 FILL 1 : MODE 8
120 FOR i = 0 to 100 STEP 2
130    INK i : CIRCLE 75,50,40
140 END FOR i
```

Without LIGHTNING, you will see that sometimes the newly plotted circle doesn't exactly match the previous one.

23

produce the correct output, it does take several seconds (!) for it to do so - the delay is caused by LIGHTNING 'drawing' those (substantial in this case) parts of the circle that fall outside the window. It is even possible to construct some cases where even LIGHTNING may fail to produce output. The reasons for this failing by QDOS and (to a much lesser extent) by LIGHTNING to cope with these extremes is that both programs use integer arithmetic to compute where the circle (or ellipse) will be drawn on the screen. Integers must be used, otherwise the plotting would be painfully slow. The penalty is that if numbers are involved that are close to the fixed 16 bit limit for integers, 32767, then overflow problems can produce the wrong answers. The situation is made worse since various intermediate results that must be computed while working out where each point of the shape should lie on the screen can be multiples of the required position. It is for this reason that QDOS can fail for relatively small displacements from the window. LIGHTNING, which is more tolerant, will normally cope with offsets of up to 30000: however, this limit will be affected by the scale.

3.4 ARC implementation

This version of lng_GRAF_ext doesn't handle the drawing of arcs. These are currently passed on to, and executed by, QDOS. It is possible that a future release of LIGHTNING may include an implementation of arc drawing. This will be faster than the QDOS code and also cure the problem that is exhibited by QDOS whereby long arcs that subtend a small angle (ie; that have a large radius) aren't plotted.

SUPER SPRITE GENERATOR

This is what Version 2.0 offered:

*Easy drawing and amending of multicoloured sprites. *Upto 16 different sprites on screen simultaneously. *Each sprite can have upto 16 different shapes "frames" for animation. *Upto 256 different planes to control depth of movement. *Automatic real-time collision detection; both Sprite-Sprite (with identity) and Sprite-Object. *Sprite Reversal. *Sprite Inversion. *Comprehensive diagnostics with run-time error trapping and exception-processing systems. *Ultra high speed action - at top speed the motion is faster than the eye can see! *100% Variable speed. *Easy control from within your program, using SuperBASIC Keywords. *Library of preprogrammed sprites. *Easy to use. *Full instructions included. *All processes fully described + flowcharted for clarity. *Ultra large sprites possible. *Full over / under / hit / miss logic. *Freeze Control. *Automatic Screen Border Detection. *Backup copy. *Full Setup / Connect / Extend / Test / Reply facilities. *Solid + "Ghostly" sprites and much, much more. *All you need to know about accessing the system both from SuperBASIC and 68000 machine code supplied with the system.

Version 3.0 has ALL the features of Version 2.0, as well as :

*Amazing, Incredible, guaranteed truly 100% Flicker Free operation. *Choice of two screens. *Upto 256 sprites in total. *Choice of automatic sprite drawing in the background. *Choice of automatic sprite movement. *Ability for each sprite to move at its own independent speed irrespective of the speed of other sprites. *All work done by new Keywords. *Revised, enlarged and improved documentation.

Version 3.5 has ALL the features of Version 3.0, and has greatly improved user-friendliness, ten time improvement on loading speed, etc.

VERSION 4.0 has all the features of v3.5, and in addition a new ma screen 2 windowing, use of borders, new integer keywords for even f

Things don't stop there, though. For most jobs there is no need to use maths functions to the full precision provided by the QL (and LIGHTNING). For example, trigonometric functions are often used when constructing drawings and designs, say for 2D and 3D packages, Astrology programs and so on. In most of these cases the value of one or more trigonometric function is used to work out where a point should be plotted on a single screen. Since the resolution of the QL's screen is only 1 part in 512 (at best), only the first 3 digits or so of the value contribute towards screen position in many of the cases. For this sort of application, you can instruct LIGHTNING to calculate all the above functions to less precision. This has the advantage of increasing the speed quite dramatically.

The command to do this is:

```
        _lngPREC n
```

where n is a number that indicates the precision required. Valid precisions include n=4 for the full quadruple-byte precision (the default situation when the LIGHTNING maths routines are first installed), n=2 for double-byte precision (providing approximately 4 significant digits of precision, the same as log tables, and sufficiently accurate to put a man on the moon if we believe NASA's anecdotes) and n=1 for single-byte precision. This last setting provides an extremely fast result, but it is only accurate to about 2.5 significant digits. The use of this last setting must therefore be treated with some care, as problems might result from programs where the result of one calculation provides the input to the next - accumulated errors might become quite large. In any event it is sensible to experiment a bit with the application program if you want to use either _lngPREC 2 or _lngPREC 1 with it (especially if it is the latter) - observe whether the reduced precision makes a difference. Astrologers can turn a Hitler into a Gandhi (or vice versa) by indiscriminate use of _lngPREC 1. Remember that _lngPREC 4 gives a considerable speedup with no loss of precision.

There is one other setting accepted for _lngPREC. If n is given a value of 0 (or less) then LIGHTNING routes you back to QDOS - warts and all, so everything goes slow. This can hence be viewed as an off setting.

Note that _lngPREC, as with other LIGHTNING commands, has a global effect - that is, whether you invoke it from BASIC or from within a compiled task, it will have an effect on all programs currently running on the machine and on all programs that are subsequently loaded into the machine in that session.

4.2 Changing precision within compiled programs

There is one small restriction on the use of the _lngPREC command from within programs compiled by a SuperBASIC compiler. This is that the command will have no effect if the program attempts to switch between one of the LIGHTNING precisions (1, 2 and 4) and the QDOS precision (0), or back. Changes between precisions 1, 2 and 4 themselves will work fine. There is no fix to this that would not put a heavy time penalty on every maths routine access.

4.4 Cotangents of angles close to zero

The full precision (quadruple-byte) LIGHTNING routines work to the limit of precision that can be obtained with the QDOS numbering system (about 9 digits of accuracy). However, as with all numerical operations it is possible that errors in the least significant place can arise as a result of working to a fixed precision. In some situations it is possible that unavoidable rounding errors can become significant.

If COT of very small angles is used then it is possible for errors to become visible in the 8th or possibly +1/-1 in the 7th digit of precision. This is unlikely to cause a problem except in situations where a program uses the difference between COT of two very small angles - in this situation the relative error in the result can become large.

QDOS is pretty accurate for small-angle COTs. But when you actually try COT(0) you will find that the native QDOS routines give the very interesting answer 1. LIGHTNING routines will correctly report an overflow error for COT(0). If you have a program that evaluates COT(0) then it will fail if LIGHTNING maths routines are installed. In this (unlikely) case you should correct your program to avoid the error. It is much better to report an error and return to base than chunter along doing the wrong thing.

4.5 Accelerating the power function

There is no direct way that this function can be substituted on the QL (without changing the QL ROM). If you want to make use of LIGHTNING routines to evaluate X to the power Y then you should use the following substitution:

```
          X raised to Y = EXP (Y* LN(X))
```

This will still provide the advantage of the extra LIGHTNING speed, as well as the choice of precision (2 or 4).

Of course, for small integral values of the exponent it is much faster to perform the multiplications directly:

```
          2.07 raised to -3 =  1/(2.07*2.07*2.07)
```

Note that doing 1/2.07/2.07/2.07 is slower, as divisions take much longer on the QL than multiplications. So don't divide by 100, instead multiply by 0.01.

Powers that are small odd multiples of 1/2 are best dealt with using SQRT:

```
          2.07 raised to -1.5 = 1/(2.07*SQRT(2.07))
```

When writing programs, simplify computations - and get rid of computations exclusively involving constants.

Finally, it is worth mentioning that to receive the full benefit of the enhanced mathematical functions it is best to compile a SuperBASIC program. This is because the interpreter adds a considerable time overhead itself to any mathematically intensive program.

5. THE NULL HANDLER

5.1 The new device nul

This simple device is installed, along with the screen handling code, whenever the main LIGHTNING text routines are loaded. Deceptively simple in what it does, the nul device can be used in a variety of ways. It is used for much the same sort of thing that you might use the CTRL/Z "drain" - but it is more probable that the nul device will be used within programs.

To use the handler, you OPEN a channel to it in the normal way. For example:

```
OPEN #3,nul
```

will set up SuperBASIC channel #3 to the new handler. Once this is done, any output sent to channel #3 will effectively disappear, fast. It doesn't matter how long the output is, all output will be 'swallowed up' equally quickly. For example, compare the times for the following:

```
REMark  Output to a temporary RAM disk dump
FORMAT RAM1_100 : a$=FILL$('*',30000)
BEEP 1000,10
FOR I=1 TO 100
  OPEN_NEW #3,RAM1_dump:PRINT #3,a$:CLOSE #3:DELETE RAM1_dump
END FOR I
BEEP 1000,10

REMark  Output to the device nul
OPEN #3,nul : a$=FILL$('*',30000)
BEEP 1000,10
FOR I=1 to 100
  PRINT #3,a$
END FOR I
CLOSE #3:BEEP 1000,10
```

We chose ramdisk as the destination for the first example as it is normally the fastest device available on the QL, often twenty times faster than even floppy disk. The time for opening, closing and deleting the ramdisk file has no appreciable effect on the execution time of the first program.

If you don't have a ramdisk then you will have to take our word for it that nul is much faster. Try to compare nul with the fastest device you do have - disk drive or microdrive - you will find that there is simply no comparison.

Note that if we output a$ not once but say fifty times to the file, the ramdisk would, of course, have overflowed. The nul device is a black hole - it never overflows and is never satiated - except by a CLOSE command. Also, the time taken for the nul device to gobble up 30000 characters or 300000 characters is not very different.

The main use of nul is in disposing of unwanted output from a program without having to include switches to control each PRINT statement. Consider the following example of what might have to be written to control output to a printer within a program:

6. WHAT TO DO IF THINGS GO WRONG

You shouldn't have any problems with LIGHTNING as the package has been extensively tested on just about everything we could get our hands on. However, it is not possible to absolutely guarantee that problems will never arise as it is impossible to test any complicated piece of software under all possible conditions of input and with all possible operating environments. If you hit what you think is a problem please read this section and then re-read the manual. If this does not yield a result, write to us sending us a copy of the application program you have encountered the problem with. We will fix the problem (if it is possible to fix), supply you with the solution and destroy the copy of the application program.

6.1 LIGHTNING doesn't load

Check very carefully that you have followed the installation procedure described in 1.5 exactly. If you are in any doubt, then we suggest that you make a default installation - ie; go through the installation procedure and select the default (ENTER) response to each question. If the resulting LIGHTNING boot cartridge/disk still doesn't operate correctly then:

Make sure that you have RESET your QL before attempting to use the LIGHTNING "working copy".

If you have any ROM extensions installed, try unplugging/removing them and rebooting. LIGHTNING is compatible with everything we could get our hands on, but you may have something we don't... (no rude comments are invited).

It is worth repeating here that LIGHTNING MUST be installed prior to any other operation (other than simple BASIC operations that do not install extensions). Our installation procedure assumes that certain pointers are set up for QDOS: some packages such as QRAM and SPEEDSCREEN(!) may alter this information. The fix? Install the LIGHTNING extensions first, which is what our automatic installation procedure makes sure happens anyway. Then there is no conflict.

If, when you attempt to use your LIGHTNING boot medium, you get the error message 'Not Complete', it is likely that something else is lurking within your QL and upsetting the LIGHTNING installation code (RESPR won't work if there is a task - other than SuperBASIC - already present).

6.2 Characters disappear

Characters might disappear if the output drain facility is accidentally activated, or accidentally left on. Say you have deactivated screen output and are using a mouse to operate the QL - as the mouse doesn't use keys, output won't be switched back on again. The solution is to be careful not to invoke the output drain facility by mistake. In particular, if you change the drain 'activation' key then don't make it anything that is likely to be pressed accidentally (any of the normal character set 32 - 127 would be an obviously bad choice).

Generally, LIGHTNING should be the FIRST package loaded on your QL. In particular, packages such as QRAM must be loaded after LIGHTNING (refer to 6.1). Also, we do not recommend that both SPEEDSCREEN and LIGHTNING are loaded at the same time. If you must, though, be absolutely sure that LIGHTNING is loaded BEFORE SPEEDSCREEN. If it isn't, some conflicts could occur and some functions, such as MODE changes, might not work correctly.

If you still have problems, then contact Digital Precision by letter (telephone answering staff are unlikely to be able to help you) giving ALL relevant details (QL version and hardware configuration, what you have in your ROM port and so on) of what is going wrong. It would be helpful if you could provide copies of the offending software, which we will destroy after examination. Help us help you.

6.6 Do's and Dont's

Do make sure that LIGHTNING is loaded BEFORE any other utilities are installed on your QL. The automatic system generation program (boot) will organise everything sensibly on your medium - be careful about adding other utilities subsequently to the same medium: some other packages might substitute their boot file in place of the LIGHTNING one.

Don't load both lng_TEXT_ext and _lng_TEXT_mini_ext at the same time, it is likely to give your QL a headache.

Do be very careful when using the single-byte precision with existing programs. You might find that the reduced accuracy will alter the program operation or results. The same warning also applies to the double-byte precision setting except that in this case such problems are very much less likely to occur.

Don't be tempted to use either of the lng_TEXT packages at the same time as any other screen text enhancement program (such as SPEEDSCREEN). At best you will achieve no improvement and you might well degrade the performance of either or both programs.

Don't worry about all the extra features offered by LIGHTNING if all you want is a package to make your existing application software work faster. LIGHTNING has been carefully designed so you should have no problems.

_lngGOOD _lngBAD	1 1	These commands will probably never concern you. A program that performs some direct manipulations on the table of channel definitions might confuse lng_TEXT_ext. If this happens, issue the _lngBAD command to cure the fault, at the expense of (very slightly reduced) text output speed. The default is the opposite, _lngGOOD, for most programs!
_lngKEYSET	1	This should be used to alter the key(s) used to activate the output drain facility (default CTRL/Z). The procedure will prompt you to press the required key combination for the new activation character and then enforces the change.
_lngPREC n _lngFPREC n	4 5	These are used to control the precision and speed of LIGHTNING's maths functions. The default setting, when either lng_MATH library is first loaded, is n=4, or quadruple-byte precision, which is the same as the precision of the native QDOS routines. Other valid values for n are 2 for double-byte precision (about 4 digits accuracy), 1 for single-byte precision (2 to 3 digits accuracy), and, for lng_PREC only, 0 to revert to the native QDOS routines. Note that the fastest setting is n=1, although this should be used with care since it might cause unpredictable results with some programs. Even the full precision provided by the n=4 setting is at least twice as fast as native QDOS.
_lngFONT #c,a	6	This is used to call the font loader to install a new character set located at address a to a SuperBASIC window attached channel c. There are some syntax variants, detailed in s2.3 .

KEY:

1 = lng_TEXT_ext
2 = lng_TEXT_mini_ext
3 = lng_GRAF_ext
4 = lng_MATH_ext
5 = lng_MATH_newnames_ext
6 = lng_FONT_ext

9. GRATEFUL ACKNOWLEDGEMENTS

Digital Precision gives special acknowledgement to John Paine for the ideas behind some of the mathematical algorithms. Thanks also to all the people who have used their valuable time to help test LIGHTNING, to encourage us and to make suggestions about improving it: we couldn't have done it without you! To name a few in alpha-sequence - Peter Jefferies, John Norton, Jonathan Oakley, Jonathan Oakley (intentional repetition - he helped a lot), David Oliver, J S ROM, Hellmuth Stuven, Tony Tebby and the entities CST, Dansoft, QJUMP and Sector Software. As a matter of course, we thank Helmut Aigner (in advance this time) of Vienna for his painstaking proofreading, and blame him for all errors, with as little justification (pun) as ever.

LIGHTNING itself was written almost entirely by Steve Sutton. LIGHTNING's specification was by Freddy Vachha.

Gerry Jackson was responsible for some of the really tricky work on graphics, Chas Dillon for much guidance on text and sensible bells and whistles, and Freddy for some of the maths, mission control and this damn manual. We shall break with tradition and not bore you with any thumbnail/thumbscrew biographies, titles, or appended qualifications of the contributories. LIGHTNING speaks for itself.

You may look forward to other quality programs from the Digital Precision stable in the months and years ahead. It is always our contention that our best program is the one we have not yet written.

10. PERFORMANCE FIGURES

Did you think we would waste our time filling these in?! Do them yourself, or copy from our ads if YOU want to cheat.

8. WHERE TO GO NEXT

If you want even more speed and power from programs, consider buying one of Digital Precision's compilers. We are speed freaks and all our compilers are fast - SUPERFORTH, SPECIAL EDITION SUPERCHARGE, DIGITAL C and TURBO are the fastest compilers available for the QL. And they are all 'true' compilers - not semi-interpretive systems that produce pseudo-code.

LIGHTNING refreshes the parts that even our compilers can't reach. But the compilers refresh the parts that LIGHTNING can't reach. The consequence of this happy symbiosis is that accelerations obtained by combining LIGHTNING with a Digital Precision compiler are MUCH better than multiplicative: if you cannot intuitively understand why this is the case, read on.

Let us say a program spends 40% of its time on actual text printing, and that the average LIGHTNING speedup for these operations is 8x (easily obtained for many common solid ink/paper combinations). Let us also say that the rest of the processing time is spent on operations that TURBO could speed up by 20x (TURBO typically accelerates by between 10x and 100x) and on which LIGHTNING has no effect (so, no graphics or maths functions). Say T seconds was the original time taken to run the program. The only other assumption we shall make is that the areas that LIGHTNING and TURBO speed up are mutually exclusive and exhaustive - a fair assumption in most applications.

The time taken if LIGHTNING was used would thus be 0.6*T + 0.4*T/8 = 0.65*T seconds, yielding an overall speedup of 1.54x . The time taken if TURBO was used would be 0.4*T + 0.6*T/20 = 0.43*T seconds, yielding an overall speedup of 2.33x. If both TURBO and LIGHTNING were used, the time taken would be 0.6*T/20 + 0.4*T/8 = 0.08*T seconds, yielding an overall speedup of 12.5x ! You should note that the speedup of the combined TURBO and LIGHTNING is much greater, than a simple product of overall speedups (2.33 * 1.54 = 3.59x, which is far less than the actual 12.5x that you would obtain).

If this is not enough to tempt you to invest in a Digital Precision compiler for the language of your choice, here is a really super offer. If you quote the medium-name (what you first get when you do a DIR, just above the sector data) for your master LIGHTNING medium when you order (either by letter or phone) a Digital Precision compiler from us, we will give you a 15% discount off the price of the compiler. This offer will lapse one month after you have received your new copy of LIGHTNING. The date of acceptance will be the date you post your order, so foreign customers need not feel discriminated against. But hurry. You are bound to forget if you don't do it NOW.

Of all the Digital Precision compilers the most powerful is the SuperBASIC compiler TURBO, originally designed by S N Goodwin and then implemented by Chas Dillon and Gerry Jackson, with mission control by Freddy Vachha. SPECIAL EDITION SUPERCHARGE is a much cheaper alternative, giving much of the speed of TURBO but fewer extras. DIGITAL C and SUPERFORTH are both written by Gerry Jackson. You'll see most of these names again in the next section.

7. SUMMARY OF NEW COMMANDS PROVIDED BY LIGHTNING

Here is a list of all the extra commands available from SuperBASIC:

COMMAND	AVAILABLE IN (Key at end of table)	EFFECT
_lngOFF	1,2	Disables ALL the TEXT enhancements including multiple scrolling and mode change suppression. The nul driver will still work. Commands associated with the TEXT routines will still be accepted, but the effects of these will not become apparent until TEXT enhancements are switched back on.
_lngON	1,2	Re-enables the screen TEXT enhancements after they have been switched off with the _lngOFF command.
_lngGRAF n	3	Controls the GRAPHICS enhancements. If n is 0, all the lng_GRAF_ext routines are disabled and the native QDOS routines are re-established. If n has any other value then this will re-enable the LIGHTNING graphics enhancements.
_lngZIP n	1	This controls the number of lines of text by which lng_TEXT_ext will move a window's contents whenever printing occurs and the window is full. The default setting, when lng_TEXT_ext is first loaded, is n = 1. Values of 2 or 3 will give dramatic speed improvements when printing large amounts of text to the screen. If n is set to a value that is greater than or equal to the number of lines in a particular window, then an effective value of 1 will be used for that particular window.
_lngDOMODE _lngNOMODE	1,2 1,2	Normally, when the MODE command is issued in BASIC or when a machine code program changes MODE by a direct TRAP call to QDOS, part of the operating system will check the information in all the channels open to the screen and update certain information concerning ink/paper colours, character sizes etc. The operating system will also clear all the screen windows. In some situations this process isn't strictly necessary, and then, since it can be time-consuming, it is beneficial to disable it. The _lngNOMODE command will disable this process. The default, when LIGHTNING's text enhancements are first installed, is that the process will be performed in the normal way.

6.3 Text is mis-positioned, the display gets scarmbled (sic)

It is possible, in rare situations, that a program might perform certain manipulations of the data in the channel definitions without using the standard QDOS trap calls. This might give rise to problems with lng_TEXT_ext since LIGHTNING will normally assume that no such (invisible) alterations have occurred. This is VERY rare.

If you do get a problem, we have already supplied you with a fix. Two extra commands are provided with the main text extensions:

_lngGOOD	The normal situation
_lngBAD	Should problems occur. Naughty would be more accurate than bad, but bad is shorter!

You should never need to use either of these commands unless you can see that things are going wrong. When lng_TEXT_ext is first loaded it is with _lngGOOD activated. If things are going wrong, issue the _lngBAD command as soon as lng_TEXT_ext is loaded (possibly from inside the BOOT program itself on that 'bad' application program medium). This should cure any problems, the only penalty being that lng_TEXT_ext will be slightly less fast.

Note that this problem simply cannot occur with lng_TEXT_MINI_ext, hence _lngGOOD and _lngBAD are not implemented within it.

The only program whose screen handling we have not been able to tame with _lngBAD is Lattice C, which gets up to all sorts of mischief. For this one, switch our text extensions off.

6.4 Characters and displays flash or are weird

If you use the _lngNOMODE command, any subsequent change of mode can cause problems since information in the channel tables doesn't get updated to take account of the new screen resolution (as we explained in 2.2)

The solution is to enter the _lngDOMODE command (which was the default anyway - just don't use _lngNOMODE if there are problems!) and then enter a MODE 4 or MODE 8 command as appropriate. There is a (very remote) possibility that you may have to enter MODE 8:MODE 4 or MODE 4:MODE 8, as appropriate, in order to completely cure the problem. Look at the screen to make sure.

6.5 Have we forgotten anything?

Firstly, check that your problem is indeed caused by LIGHTNING. Issue the appropriate 'off' command (_lngOFF for text, _lngGRAF 0 for graphics, _lngPREC 0 for maths) and confirm that the problem DOESN'T then occur! At least one of the problems reported by one of our beta-testers was absolutely genuine, but occurred whether or not LIGHTNING was present (ie; it was a bug in the application program)!!

If you have any other software loaded as well as LIGHTNING then check to see if the fault still occurs if this other software is NOT present.

```
100 INPUT 'Do you want a hard-copy of the output Y/N ';a$
110 IF a$=='n' : hdcopy=0 : ELSE : hdcopy=1 : OPEN #3,ser1
120 ....
130 ....
140 IF hdcopy : PRINT #3,'Test output to printer, line 1'
150 ....
160 IF hdcopy : PRINT #3,'Test output to printer, line 2'
170 ....
```

and so on, a separate IF construct being required to control each possible PRINT to the printer. Using the nul device, this can be altered to:

```
100 INPUT 'Do you want a hard-copy of the output Y/N ';a$
110 IF a$=='n' : OPEN #3,nul : ELSE : OPEN #3,ser1
120 ....
130 ....
140 PRINT #3,'Test output to printer, line 1'
150 ....
160 PRINT #3,'Test output to printer, line 2'
170 ....
```

This will make your program simpler, shorter and faster.

If a program attempts to read data in from the nul device, say using the command

```
INPUT #3,a$
```

in the second example, then the 'End of File' error will be reported. If you are using nul in a program where a read might be performed from it then you should ensure that the EOF function is used (good programming practice anyway). EOF will always return true for any channel that is open to nul, as is illustrated in the next example. This sort of code might be written to enable the QL to communicate with an external device over the ser1 port:

```
100 INPUT 'What port is WIZZ-BANG WIDGET on ';a$
110 IF NOT a$=='ser1' : OPEN #3,nul : ELSE : OPEN #3,a$
120 PRINT #3,'WAKE UP' : REMark string to start WIDGET
130 PAUSE 100          : REMark wait for WIDGET's response
140 IF EOF(#3) : PRINT 'WIDGET absent!' : ELSE : PRINT 'OK'
```

In this case, if the response to the prompt in line 100 isn't ser1 then the code will look for that rather frightening sounding piece of hardware, WIDGET, on the nul driver.

4.6 The maths library with new names

The maths routines with new names and the null handler (the subject of section 5) differ philosophically from the rest of LIGHTNING. While everything else in LIGHTNING is magic-wand stuff that can be used with application programs with internal workings to which you have no access, these two are specifically for programmers only.

If you want to make use of the LIGHTNING fast maths routines and have access to the normal QDOS ones simultaneously (without using the _lngPREC 0 command, which has restrictions as we have already described), then you should make use of the additional maths library, lng_MATH_newnames_ext. This contains all the routines that are in the lng_MATH_ext library, and they all compute values in precisely the same way, taking the same amount of time and giving the same (switchable) accuracy.

The lng_MATH_newnames_ext module can be installed by LRUNning the boot file lng_MATH_newnames_bas. Once this is done the fast routines will be available.

So what is the difference between the two maths libraries? With the newnames extensions each maths function is given a different name from that of the corresponding QDOS function - this new name is the original function name prefixed by an 'F' (for fast), so the fast SIN is FSIN, the fast ATAN is FATAN, the fast EXP is FEXP etc. The QDOS functions are unaffected - so SIN, ATAN, EXP etc themselves will behave exactly as if LIGHTNING was absent.

The new 'F' names can be used from BASIC and from within compiled programs.

A command _lngFPREC is available to alter the precision of these 'F' functions. Its operation is very similar to that of _lngPREC, except that in the case of _lngFPREC a parameter of 0 is not meaningful (there are no QDOS values to fall back on - these functions have distinct names) and will result in a 'bad parameter' error being generated.

The "newnames" version of the fast maths routines will happily co-exist with the normal version, in the unlikely event that you want both to be present at the same time. If you do load both, be careful not to confuse the two precision commands (_lngPREC and _lngFPREC); each works only on its own set of functions.

Unlike with _lngPREC, there are no technical restrictions whatsoever about using _lngFPREC in compiled programs: this is because there is no need to use a parameter of 0 with the _lngFPREC precision command to get access to the standard QDOS functions: they are available all the time, under their usual names.

We realise that the newnames extensions are really only of use to programmers. For this reason, we will allow licence-free commercial use of just these 'newnames' routines (and nothing else - section 1.9 tells you this already) provided full credit is given to Digital Precision Ltd and to LIGHTNING, in documentation and on-screen, for their use. They are not allowed to be supplied with any other speed enhancement program of any kind whatsoever, so don't get ideas.

If the program starts with the LIGHTNING maths active (ie; NOT the _lngPREC 0 situation), then _lngPREC will work as expected inside the task, switching between precisions 1, 2 and 4 freely. In addition, you can use the _lngPREC command from SuperBASIC to switch the precision between values 1, 2 and 4. However, if you try to change the precision to or from the QDOS precision (0), either from within the task or externally, then the compiled task will ignore the attempted change. This could be slightly confusing, especially if there is more than one program running, so generally we advise that if you want to allow a compiled program to use both QDOS and LIGHTNING maths routines then you should consider using the lng_MATH_newnames routines (described in 4.6) and leave the QDOS functions unsubstituted.

Should you use a compiled program that does try to swap between QDOS and LIGHTNING maths then a 'BAD PARAMETER' error will be generated and the program will abort.

4.3 Restricted function domains for single-byte precision operation

We do not allow the _lngPREC 1 setting to operate with the functions LN, LOG10, EXP, COT and ACOT, for reasons associated with their domains, ranges and rapidity of change. Absolute accuracy and ratio accuracy would be very poor for these functions if we allowed single -byte precision with them.

We also impose a limitation on the domain of the trigonometric functions for the _lngPREC 1 setting to operate. This is done in order to avoid the overhead of coping with angles having large absolute values - this would slow these routines down to the extent that they would be no faster than the double-byte precision routines, and would still only have low accuracy.

In the case of TAN and ATAN we impose a limitation on domain to avoid the areas where the rate of change is too great for single -byte precision to be relied on to yield a sensible result.

Arguments that will permit single-byte precision function operation are as follows:

LN	NONE
LOG10	NONE
EXP	NONE
SIN	-PI/2 to +PI/2 (-90 to 90 degrees)
COS	-PI to 0 (-180 to 0 degrees)
TAN	-PI/4 to +PI/4 (-45 to 45 degrees)
COT	NONE
ASIN	-1 to +1
ACOS	-1 to +1
ATAN	-1 to +1
ACOT	NONE
SQRT	ALL

For all the functions in the table, if _lngPREC is set to 1, and if the argument presented is outside this domain then the call is automatically processed at double-byte precision, which is the sensible thing to do. This is LIGHTNING's auto-override: safety is better than speed.

Clearly, a program can be optimised to make the best use of the speed that can be obtained by the single-byte precision setting, by restricting the domains from which arguments will be selected. However, if some arguments fall outside the valid domains for single -byte operation, the only penalty is that each such call will not be quite as fast (but will be much more accurate, as it will have double-byte precision).

4. USE OF LIGHTNING TO ACCELERATE MATHS

4.1 Speed and precision

As with the graphics and text routines, the maths routines can either be loaded automatically by the "working copy" of LIGHTNING that you generate when you configure LIGHTNING, or they can be loaded explicitly by LRUNning the special boot program lng_MATH_bas with your copy of the LIGHTNING backup medium in drive 1.

The mathematical enhancements work on any SuperBASIC program, including programs compiled with any of the SuperBASIC compilers currently available, and any other compiled program that uses the QL's name table. As is usual with LIGHTNING, you need no knowledge whatsoever of the program's workings to benefit from LIGHTNING's effect.

When installed, the following functions are speeded up:

EXP / LN / LOG10 / SIN / ASIN / COS / ACOS / TAN / ATAN / COT / ACOT / SQRT

Note that the speedup is achieved without any appreciable change in precision (for some functions we are actually more accurate, though QDOS is pretty much spot-on most of the time). Your programs will hence operate as before, albeit faster. How much faster? A doubling in speed is par for the course (we get back to hardware differences again). Potterers with a mathematical twist (spot the pun?) will no doubt invest a lot of time in comparing the accuracy of our routines and of QDOS, and in investigating variations in speedup factors over the domains of individual functions. The former is pointless - our aim was to minimise the absolute value of deviations over the whole domain from the exact (double-precision, actually) value, and not to minimise deviations, R.M.S. deviations or relative (ie; ratio) deviations. Note that SuperBASIC only allows the printout of 7 significant figures - the best way to be able to see more is to use Supercharge Special Edition or Turbo. Both QDOS and our routines are accurate to the ninth significant digit - almost always.

A lot of the maths we are using is quite novel - some of it might even be new. Anybody wishing to disassemble our code and understand what we are doing is warned that the task is not inconsiderable - we suggest you start with an easy one like TAN.

Our routines, incidentally, increase the usable domains of EXP, LN and LOG10.

Note that while we have "hit" virtually all the transcendental functions and SQRT, we have not optimised PI, DEG, RAD, ABS, INT, RND and RANDOMISE. The reason? QDOS is very fast with all of these already: we doubt we could appreciably improve on any of their speeds. Potterers are directed at alternatives to RND - the slice from (say) the 6th to 8th decimal digit of SIN(x) where x is in some sequence with a reasonable step (say, > 0.001) and reasonable starting point gives you very pseudo-random looking results. But beware - there are many pitfalls.

Note that the algorithms used in the graphics enhancement routines are stand-alone - they do not use or pay attention to either of our maths enhancement libraries. This is intentional - we did not want you to have to use the maths package(s) if all you wanted was the graphics one. Consequently, changing the precision of the maths routines has no effect whatsoever on the accuracy of drawing of graphics. We apologise to frustrated knob-twiddlers for this, but remind them that they can of course construct their own graphics routines using our fast maths functions, and vary precision to their heart's content.

3.2 Circles and ellipses

lng_GRAF_ext will handle all circles and also all ellipses that are oriented along one of the major axes. The exact rule is that the orientation angle specified must be either 0, PI/2 (90 degrees) or -PI/2 (-90 degrees). If you specify other angles - say 1.3 - the ellipse drawing will not be handled by LIGHTNING but will be passed back to QDOS for execution. Principal reason - we couldn't beat QDOS significantly enough on speed for plotting these skewed ellipses. If any mathematician out there has any ideas, we are listening.

For example, the following calls WILL be handled by LIGHTNING:

```
CIRCLE -20,105,85           : REMark ALL circles are handled
CIRCLE 50,50,30,.5,0        : REMark orientation 0 is OK
CIRCLE 10,50,50,1.5,PI/2    : REMark orientation PI/2 is OK
```

but these will be automatically passed back to QDOS:

```
CIRCLE 50,50,30,.5,.1       : REMark orientation isn't 0,PI/2
CIRCLE 70,30,50,1.99,-1     : REMark or -PI/2 in these ones.
```

Note that the orientation doesn't have to be exactly these required values: for example an angle of PI/2+.00001 will still be handled by LIGHTNING rather than passed back to QDOS.

3.3 Off-screen plotting

The LIGHTNING graphics routines are much more tolerant about drawing shapes and lines off-screen than are the QDOS routines. For example, the following command should produce part of a circle:

```
CIRCLE -500,50,550
```

With QDOS this doesn't produce ANY drawing on the screen. The LIGHTNING circle routine will cope perfectly with this example. Off-screen plotting shouldn't be taken to extremes, though. For example, try:

```
CIRCLE -30000,50,30050
```

You can arrange to use the cut-down version when you configure your LIGHTNING "working copy". However it can also be loaded at any time by LRUNning the lng_TEXT_mini_bas file. Don't be tempted to load both versions of the TEXT routines at the same time: you will not achieve anything useful by doing so, and if you are using an early QL (AH or JM) it will probably result in most of the LIGHTNING commands not working properly.

Note that lng_TEXT_mini is not intended for use on a THOR. All THORs have at least 512K RAM, so this isn't a restriction at all - use the full version.

I D I S   I N T E L L I G E N T   D I S A S S E M B L E R

IDIS is an indispensable tool for all serious assembler programmers: it has many new features not found in other disassemblers. It is also ideal for beginners at assembler programming. Perhaps its largest benefits are to those who only want to potter about with other people's 68000 code - without first mastering all the intricacies of assembly language.

COMPATIBILITY with the full MOTOROLA 68000 instruction set is of course one feature of IDIS. But while most other disassemblers for the QL do not have full compatibility, a few of them do. So what makes IDIS so special?

AUTOMATIC and intelligent disassembly is IDIS's hallmark. Subroutines called by jump instructions (like JSR and Bcc) are automatically disassembled. References to addresses are automatically replaced by labels, which are far more human-friendly than long numbers. Further, program and data are automatically discriminated between, of immense value when attempting to understand someone else's code (and a huge time-saver too). With IDIS, a lot is going on behind the scenes - you command, we interpret, search, analyse and then obey -, all at lightning speed.

REASSEMBLING of disassembled code may take place instantly, because IDIS produces source code containing labels instead of absolute addresses (which is all that other disassemblers / monitors are capable of). With IDIS you can even merge disassembly files together.

OUTPUT to screen and other devices (mdv, ser, ram...) is supported.

MULTITASKING allows IDIS to run simultaneously with other programs. IDIS is compatible with TASKMASTER and QRAM!

REPORTS of errors and warnings are user-friendly.

FAST loading and running characterises IDIS.

EASE of use is very important - no long rules to remember.

EXTRA MEMORY is not required. IDIS runs even on 128K QLs.

MONITOR program feeling lonely? IDIS is the ideal complement to all Monitors and Debuggers.

18

```
HELLENIC_font
▒ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTU
VWXYZ[\]^_£abcdefghijklmnopqrstuvwxyz{|}~©

INVADER_font
▒ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTU
VWXYZ[\]^_£abcdefghijklmnopqrstuvwxyz{|}~©

ITALIC_font
▒ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTU
VWXYZ[\]^_£abcdefghijklmnopqrstuvwxyz{|}~©

METRO_font
▒ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTU
VWXYZ[\]^_£abcdefghijklmnopqrstuvwxyz{|}~©

MICRO_font
▒ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTU
VWXYZ[\]^_£abcdefghijklmnopqrstuvwxyz{|}~©

OLDE_1_font
▒ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTU
VWXYZ[\]^_£abcdefghijklmnopqrstuvwxyz{|}~©

SIMPLE_font
▒ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTU
VWXYZ[\]^_£abcdefghijklmnopqrstuvwxyz{|}~©

STANDARD_font
▒ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTU
VWXYZ[\]^_£abcdefghijklmnopqrstuvwxyz{|}~©

STENCIL_font
▒ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTU
VWXYZ[\]^_£abcdefghijklmnopqrstuvwxyz{|}~©

STUTTER_font
▒ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTU
VWXYZ[\]^_£abcdefghijklmnopqrstuvwxyz{|}~©

UNIVERSAL_font
▒ !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTU
VWXYZ[\]^_£abcdefghijklmnopqrstuvwxyz{|}~©
```

We haven't supplied you with a font designer program with LIGHTNING - there are lots of them around, and most of them aren't bad - the ones in TURBO Toolkit, EYE-Q and Speedscreen are all fine, and most of the mags have got type-in ones that work, sort of. If you design a nice font or fonts, and they are reasonably original, please send them to Digital Precision for inclusion with LIGHTNING (we are much better at producing software programs than at producing fonts) - you'll be generously rewarded with free software of your choice.

This reserves some memory for storing the font data. Note that the RESPR command will accept any number for the required amount of space to acquire. However, as space is only ever allocated in blocks of 512 bytes, we choose 1024 (the smallest multiple of 512 as big as than the length of the font). This will allow a character set of up to 113 characters to be loaded, which is adequate for all the fonts supplied with LIGHTNING.

```
        110 LBYTES MDV1_ALIEN_font,font_address
```

This loads the font data from the cartridge/disk file into the space reserved. Obviously you should alter MDV1_ according to the location of the file.

```
        120 _lngFONT #N,font_address
```

This tells the operating system (QDOS or LIGHTNING) to use the new font for the SuperBASIC channel #N, where N should be a screen or console channel. If you omit #N, that is if line 120 was "120 _lngFONT font_address", then the loader will assume that you mean channel #1, the normal SuperBASIC default. You can omit the second parameter as well: if this is done (or if you use a value of 0 for it) then this will re-establish the standard ROM font on the specified channel.

You can load any of the supplied fonts in this way. You might have realised that our method is only ever substituting one of the fonts, and is never altering the alternative or second character set. This is because all of the character sets supplied with LIGHTNING are designed to be used as the first font out of the two available. If, however, you have any other character sets that are suitable as alternative second character sets then you can use the _lngFONT utility to load them by adding the address of the second font after the first:

```
     120 _lngFONT #N,font_address_1,font_address_2
```

Of course you should make sure that you reserve memory - say 1024 bytes each for the two fonts - and load the relevant font data BEFORE using the _lngFONT command. Note that if you want to add a new second font but leave the first font as the ROM one, then substitute 0 for the "font_address_1" parameter.

You can improve the appearance of many of your existing programs by a judicious change of font. Experiment with the fonts supplied - try them out using the program below:

```
     100 _addr=RESPR(1024):device$="FLP1_":REMark Adjust devicename!
     110 LBYTES device$ & "lng_FONT_ext", _addr: CALL _addr
     120 _addr=RESPR(1024):MODE 4:CLS:CSIZE 1,0
     130 REPeat view_font
     140   INPUT"Enter Font name (excluding _font): ";fontname$
     150   LBYTES device$ & fontname$ & "_font", _addr
     160   _lngFONT 0:PRINT fontname$ & "_font"
     170   _lngFONT _addr
     180   FOR Z=31 TO 127:PRINT CHR$(Z);
     190   PRINT:PRINT
     200 END REPeat view_font
```

If you used the above program to display each of the fonts supplied with LIGHTNING (fonts as at 10th June 1988), this is what you would see on the screen:

A channel-specific drain is also provided by LIGHTNING - you'll come to it in section 5.

Another feature provided by LIGHTNING is the facility to disable part of normal QDOS operation. This is less surgical than it sounds!

Whenever commands such as MODE, NEW and RUN are encountered, QDOS will clear all the windows currently open to the screen. During MODE, it will also reset certain information in the header table for each channel open to the screen so that the correct character attributes and colours are set up. All this often takes a long time.

LIGHTNING will happily duplicate this process. However, you can also arrange for LIGHTNING to disable this (sometimes excessive) housekeeping - you will then benefit from further speed improvement.

Some programs use calls to MODE prior to updating their displays. Examples include the Psion suite of programs - in QUILL, whenever the F2 key is used to recall or remove the help windows a MODE call is made. This MODE call doesn't really alter Quill's mode, it just clears the screen - so 90% of its work is updating data that does not need to be updated (as it has not changed). Hence if you use QUILL, or other programs that get up to similar shenanigans, disabling the redrawing of all the windows and the associated checking of all channel information will noticeably accelerate operation, without causing any problems.

There are two commands to allow this MODE checking feature to be switched on or off. These are _lngNOMODE to disable the feature (and hence make things go faster) and _lngDOMODE to switch it back on. The default, when LIGHTNING is loaded, is that _lngDOMODE is in force. Generally, we suggest that you leave the mode checking active (_lngDOMODE) unless you are intending to use programs which will benefit in speed from it being disabled. To find out whether a particular program will benefit or not, experiment - see how fast it is at screen clearing first when _lngDOMODE is in force, and then when _lngNOMODE is in force. If it is faster the second time, the program is using MODE to perform screen clearing - if there is no difference, it isn't. If it was faster, and if screen clearing is used a lot (as in Quill) then it is sensible to switch to _lngNOMODE.

_lngNOMODE may cause problems (you can't miss the mess on the screen straight away if it does) if the program is actually using MODE for the purpose for which MODE was designed - a genuine change of the mode setting (from 4 to 8 or vice versa). But we can't think of any programs that actually change MODE - almost all programs use MODE 4, and a small number use MODE 8. This paragraph exists just in case somebody out there is switching MODE within a program (ie; using both) - if there is, the fix, of course, is to use _lngDOMODE (the default).

Unlike _lngZIP, _lngNOMODE and _lngDOMODE have effect even when in the _lngOFF situation. You can set either of the mode treatments at any point in time - subsequent or earlier _lngON and _lngOFF commands will not affect the treatment of mode. Of course you can toggle between _lngNOMODE and _lngDOMODE at will.

2.3 The font loader and the extra fonts

Like the QL, LIGHTNING supports alternative character sets in addition to the normal 'standard' one that is present in ROM. A set of characters is a font. 'Font' has an archaic but quaint variant - unknown outside this country - 'fount'. We'll stick with font.

Setting the _lngZIP parameter to high values is particularly useful when COPYing a large file (with or without so-called "non-printable" characters) to the screen and when LISTing a long SuperBASIC program. In these situations the improvement in output speed arising from a scroll setting of just 2 or 3 is startling.

There is a small problem with the use of scroll settings greater than 1. Some add-on commands (Supertoolkit extensions EXTRAS and WDIR are examples) generate an automatic 'freeze screen' function, similar to CTRL/F5, when a screenful of data has been generated: they are trying to be helpful by pausing output to give you time to read it. It is not possible for the program that generates the freeze screen to know that LIGHTNING has scrolled the window by more than a line. The result? Lines can get 'lost' - the first few lines printed after a freeze screen is generated can be scrolled off the top of the window before the next pause, resulting in your never seeing them, unless you can read at maybe 5000 characters per second! For this reason, it is recommended that you set the scroll rate to 1 while using these commands.

If you want to know how to change the _lngZIP setting when running an application program to whose internal workings you have no access (typically, this means every program except the ones you wrote - in programs you've written, simply insert _lngZIP n at the appropriate place(s) inside the program), here are five ways of doing so:

(a) Edit the BOOT file and put in _lngZIP n (where n is the number you want it set to) at any place AFTER the main LIGHTNING text extensions are enabled - ie; after the CALL statement(s).
This method must not be used on AH or JM QLs, as you will get the error message 'bad name'. The reason for this failure is that on early QLs the fact that _lngZIP (or indeed any add-on command - this problem has nothing to do with LIGHTNING at all, but is a quirk that impinges upon the use of all add-on commands on early QLs) didn't have a meaning until the extensions were enabled, and that the extensions weren't enabled when the boot program was loaded and started, gets the computer into a pickle. Later QLs are more forgiving - as long as the extensions are enabled before the command is actually encountered in the program, all is well. So if you have an AH or JM QL, stick to one of the following four methods - or put the _lngZIP command in YOUR_ORIGINAL_BOOT, which is loaded only after the extensions have been enabled.

or (b) Toggle into SuperBASIC by using CTRL/C (hold the CTRL key down, tap C) while 'in' the application program, enter _lngZIP n, and toggle back with CTRL/C (having made sure you have a cursor to toggle back to before having toggled out of the program in the first place!). This only works if the application program multitasks.

or (c) Use a multitasking utility such as Taskmaster or QRAM to "force" your machine code application program to multitask. Get into BASIC, enter _lngZIP n and then get back.

or (d) If your application program is in BASIC, BREAK into it once it is going, and enter _lngZIP n. Then enter CONTINUE to resume processing.

or (e) If you chose Method 1 for making a "working copy", simply enter _lngZIP n after booting up the "working copy" but before starting up your application program.

The effect of different character attributes and INK/STRIP colours is quite involved. However, as a general rule the fastest printing will result from WHITE ink on BLACK strip. Either GREEN or RED ink on black strip will be almost as fast. All other solid INK/STRIP combinations will be slightly less fast than these: stipples will be slightly slower still.

All combinations will be affected by the use of OVER -1 and OVER 1: in most situations either will speed up printing a tiny bit.

Of course we realise that in many cases when you are running someone else's program you will not have any control over window positions and ink/paper colours. However, on those programs that do allow some or all of these to be adjusted - The Editor is such a program - LIGHTNING users who wish to squeeze the last ounce of speed from their toy will benefit from fine tuning.

It pays to reduce the number of times that a string is presented to QDOS for printing: QDOS spends an awfully long time on parameter processing, so it's sensible to reduce the number of parameters! If you have access to program source, please ensure that strings (whether explicit ones - ie; within quotes - or variables) are concatenated before being presented for printing. Hence:

```
        PRINT #n, ONE_STRING$ ; ANOTHER_STRING$;
```

should be replaced by

```
        PRINT #n, ONE_STRING$ & ANOTHER_STRING$;
```

If you have comma separators between strings, and you know what the number of spaces to be inserted is going to be, you may be better off replacing the lot by a concatenated string which includes the spaces. Hence:

```
        PRINT "Hello","Goodbye";
```

should be replaced by

```
        PRINT "Hello   Goodbye";
```

Finally, QDOS takes a long time dealing with PRINT instructions that end without a semi-colon or comma - in fact, in these cases, it assumes a semi-colon, outputs a null string and then performs the line feed. To stop QDOS doing this, replace

```
        PRINT "We are coming to the end."
```

with the faster

```
        PRINT "We are coming to the end." & CHR$(10);
```

We have come to the end. Whew!

2.2 Text Extras - Multiple scrolls, the Drain and Mode changing

When LIGHTNING is first installed, the printing of text will exactly mimic normal QL output. There is, however, an optional extra facility which allows you to alter the way LIGHTNING handles scrolling of the screen in certain situations. This can result in a dramatic improvement in performance.

Piracy is the enemy not only of software publishers but also of software users. By stealing the intellectual property of others, pirates make it unprofitable for authors to produce good programs: this harms you. Pirates are usually unconcerned about the harm they do - their activities are characterised by greed and stupidity. We feel very strongly about piracy, and will give a cash reward of two hundred pounds to any person giving us information leading to the successful prosecution of these persons. We will keep your name out of the prosecution - anonymity is guaranteed. If you are offered or if you see any Digital Precision program lacking our high-quality documentation, you are probably looking at a pirate copy.

Digital Precision has successfully tracked down a number of software pirates in the UK and Europe. We actively seek your help in eradicating software piracy within the QL market. Thank you!

Users who want to know the bare minimum about using LIGHTNING may actually stop reading the manual at this point.

1.8 Benchmarks

Benchmarks are like members of the opposite sex (whichever) - you can't live with them, but you can't live without them.

If you want to write benchmarks for LIGHTNING routines, we suggest the following:

(a) A lot of iteration if you are using the QL's clock, which has a least count of 1 second and can hence cause considerable errors. A second inaccuracy in a ten second test makes nonsense of the results. Give it half an hour.

(b) Separate results for interpreted and compiled results - the interpretive overhead may be very considerable!

(c) Computation of an overall test overhead, to be subtracted from all timings, both with LIGHTNING and without. This is especially relevant if you do not have a compiler and hence cannot judge the interpreted overhead. If you are benchmarking text, you might get this overhead figure by PRINTing a null string instead of the test string at the appropriate place in the program. If you are benchmarking maths or graphics, put in as many floating point variable assignments as there are arguments - so where you have A=SIN(X) put X=X, and where you have CIRCLE(X,Y,Z) put X=X:Y=Y:Z=Z. This ensures that you are measuring the right thing, and not being swamped by the QDOS parameter processing overhead.

(d) Supply a wide and random (but legal!) range of input argument - so don't confine yourself to one CSIZE, INK, PAPER, UNDER, OVER, stipple, window position or string length, or to lines of only one length, or to silly arguments for maths functions (note how blindingly fast we are on TAN(0)!). Make it hard for us!

(e) Use the different LIGHTNING options. Vary precision, scroll interval, mode change protocol, CTRL/Z drain and so on. Experiment.

(f) Take advantage of the hints and tips we have provided in section 2.1 and elsewhere: speed both with and without LIGHTNING will benefit.

Method 1 is to create a "working copy" to reside on a medium all by itself. Boot up with this "working copy", and LIGHTNING will be installed. Take out this "working copy" and put in your application program medium (say Quill). Enter LRUN device_BOOT (if you reset to boot up the application program, you'll wipe out LIGHTNING!) and that's it. This method is simple but clumsy - you've got to keep switching media (unless you get your application program to boot up from a different device).

Method 2 provides an alternative: to have a "working copy" of LIGHTNING on all of your application media (ie; on the disks or cartridges from which you run your other programs). This alternative method will ensure that each time you boot up the medium, LIGHTNING will be automatically installed just before the 'normal' boot is executed - you don't have to change media.

The special BOOT program on your LIGHTNING backup (and master) medium will allow you to choose between methods 1 and 2. To make a "working copy", use the following procedure:

a) Place your LIGHTNING backup copy in drive 1, and reset your QL.

b) Press F1 or F2 as appropriate.

c) The BOOT program on the LIGHTNING cartridge/disk will now take charge: it will present you with a series of questions. If at this stage you are uncertain as to how you want to set up LIGHTNING, select the default (ENTER) response for each question. This ensures Method 1 is selected - so you must have a spare cartridge/disk to hand on which we will build a "working copy" of LIGHTNING.

d) If you want to install LIGHTNING on an existing application cartridge/disk, you will need to override the default response to certain questions. If you are uncertain as to how to respond to the rest, the default (ENTER) response will always be the 'safe' choice. Please note that Method 2 involves installation procedure can rename an existing BOOT file on your cartridge/disk. For this reason, in case anything should go wrong, such as a tape sticking, we suggest that you make a backup copy of your application medium (you're already using a backup of the LIGHTNING medium) BEFORE you commence installation, and use this application backup as a target. Also, ensure that the target copy of the application program you submit for installing a "working copy" of LIGHTNING on is not write-protected - if it is read only, we can't rename BOOT.

It is possible (though improbable) that an error may occur during installation. The most likely error is that there will be insufficient room on your existing cartridge/disk to accommodate the extra LIGHTNING files. In this case you will get a message 'DRIVE FULL' and the installation procedure will stop. If this should happen, the simplest way to recover your application cartridge/disk is to re-copy it from the master. If you prefer instead to recover the medium, the only file that we will have amended is the original BOOT file, which we will have renamed YOUR_ORIGINAL_BOOT. Delete the new BOOT and rename (by copying, if you don't have RENAME on your QL setup) YOUR_ORIGINAL_BOOT as BOOT. You are likely to also find partially complete LIGHTNING files on the medium - delete them.

To avoid the possibility of running out of space on the medium, you should check that there is enough free space (using the DIR command: refer to the QL User Guide) before starting the installation. For the default (and biggest) LIGHTNING system you will need roughly 66 free sectors. If there is this much space available you should have no problems. If the available space is less than this, there might still be enough room, since some of the LIGHTNING configurations are considerably shorter than the default configuration. In this situation the easiest way to find out is to try it and see!

lng_FONT_bas	Individual installation program for lng_FONT_ext.
BOOT	The 'BOOT' file will call the main configuring program...
CONFIGURE_bas	This special program deals with setting up a "working copy" of LIGHTNING. It can produce a "working copy" on an existing device or format a new cartridge/disk to which you may later add other software/data . There is no need for any complicated decisions in setting up LIGHTNING: to obtain the best results for your QL configuration and applications you need only answer a few simple questions and CONFIGURE will do the rest.
BACKUP_bas	Clones LIGHTNING. Also lets you change LIGHTNING's default devices.
DEMO_TEXT_bas DEMO_GRAF_bas DEMO_MATHS_bas	Demonstration programs that illustrate the power of the various LIGHTNING enhancements. Each of these SuperBASIC programs should be LRUN after the appropriate enhancement package is installed. Note that they may be compiled for optimum results.
xxxxxxxx_font	You will find many files with names ending in _font. These are alternative character sets which can be installed for use in your programs by the lng_FONT_ext utility.
UPDATES_doc	This Quill file will include anything we forgot to put into the manual.

1.4 Backing-up LIGHTNING

```
***************************************************************
******* DON'T DO ANYTHING WITH LIGHTNING UNTIL YOU HAVE *******
*********** MADE A BACKUP OF YOUR DISK OR CARTRIDGE. **********
***************************************************************
```

Disk owners should back up the LIGHTNING disk (which is not physically protected against copying in any way whatsoever) either by using the WCOPY add-on command, by copying files individually or by using BACKUP_bas. If either of the first two methods is used, default devices referred to within the program suite will, of course, not be changed.

We have included more detailed instructions for cartridge users, as they may not have available useful extensions like WCOPY. To make a backup of the master LIGHTNING cartridge, the following procedure should be adopted:

a) Reset your QL with no cartridges or disks in place. Select F1 or F2, in the normal way.

b) Put LIGHTNING in drive 1.

c) Enter the command:

```
        LRUN device_BACKUP_bas       where device is mdv1_ or flp1_
```

LIGHTNING

by
Steve Sutton

ASSISTED BY
G Jackson
F Vachha
C Dillon

DIGITAL PRECISION

The program that puts go-faster stripes on your QL — automatically