SPEEDSCREEN

USER GUIDE CONTENTS page

AN INTRODUCTION TO *SPEEDSCREEN*

SPEEDSCREEN optimises QL display handling by replacing slow, general-purpose code in the QL ROM with new, fast routines. You don't need to alter your programs to use it - just load *SPEEDSCREEN* before loading and using your application. The only difference is the extra speed.

SPEEDSCREEN optimises scrolling, printing of small characters, window clearing and cursor operations. The improvement in speed compared with a standard QL depends on the operation: text output is typically between four and twelve times faster than normal. *SPEEDSCREEN* optimises all OVER and UNDER settings, solid colours and 64 stipples.

By default, *SPEEDSCREEN* scrolling is at about twice the normal speed: for example when COPYing to SCR. The new command _SCROLL lets you set higher speeds for free-scrolling displays such as LIST and COPY. _SCROLL lets you make scrolling up to eight times faster than normal. You can adjust the rate to control flicker and match your reading speed.

Why is the QL display so slow?

The QL's original display handling code was written in a great rush and squashed into a very small amount of memory. Routines were chosen for their generality, rather than their speed. Thus one routine prints characters in all CSIZEs and for all values of INK, PAPER, OVER and UNDER. A single tangled block of code is used to scroll and pan, to print blocks and borders, to draw and erase cursors and to clear all or part of any window, regardless of its position. *SPEEDSCREEN* avoids that slow code and uses its own fast, specific routines instead.

How compatible is *SPEEDSCREEN* with existing programs?

SPEEDSCREEN emulates the QL ROM accurately, despite its speed. Displays look just the same, but appear much more quickly. The emulation includes CTRL-F5 and SuperBASIC BREAK. One copy of *SPEEDSCREEN* works with any number of tasks. The action of MODE is changed slightly (see page 10) but apart from that all commands work just as normal.

Features that are not optimised (e.g. CSIZE 2) are handled exactly as usual, at normal speed, and can be mixed with optimised text or user-defined graphics without restriction. *SPEEDSCREEN* does not speed up MODE 8 text output, as this is little used in professional programs. But it can speed up CLS and SCROLL in MODE 8 as well as in MODE 4.

SPEEDSCREEN code is fast but concise. It is supplied in a variety of versions. Large ones speed up display output in many circumstances and add extra commands and facilities. Smaller ones are designed to optimise display output of large programs, such as Psion packages and The Editor on a 128K QL, while leaving lots of memory for data.

What other advantages does *SPEEDSCREEN* have besides speed?

SPEEDSCREEN adds new features to the QL's display handling. There are extra commands to set up character-sets, including more detailed and faster founts. Several founts are supplied, and you can use any other standard QL-format founts you may own. *SPEEDSCREEN* doesn't restrict you to the QL's usual character-sizes. It supports many new sizes, so you can pack lots of information on the screen, yet make sure it is still readable.

SPEEDSCREEN CONFIGURATIONS

SPEEDSCREEN is supplied on disk or tape, in a compacted form. The master medium holds a configuration task, a fount and graphics design utility called QLUDGE_TASK, eight character founts (SERIF, BOLD6, BOLD8, SCIFI, UKSTD1, UKSTD2, FANTASY and SHORT), a demonstration program and components for several versions of *SPEEDSCREEN*.

The configuration program lets you save the founts and utilities onto a 'working' disk or tape, together with a version of *SPEEDSCREEN* tailored for your system. Versions differ in speed, size and compatibility with other programs. We let you select a version that will speed up the programs you use with full compatibility, giving the best possible trade-off between speed and memory usage.

SPEEDSCREEN uses two different techniques to accelerate the QL's text output. These are called *Turbotext* and *Colour Synthesis*. *Turbotext* is the fastest technique, but *Colour Synthesis* is the most versatile.

Turbotext routines speed up output in CSIZE 1,0, when the standard QL fount would normally be used. *Turbotext* does not underline characters, and it is fussy about the position of windows.

SPEEDSCREEN uses *Colour Synthesis* routines if *Turbotext* is not suitable.

Turbotext is used when the pixel co-ordinate of the leftmost column of each character is evenly divisible by eight (X MOD 8 = 0). This means that each line of each character exactly fills a word of display memory, so that *Turbotext* can draw it very quickly. If your program uses CSIZE 1,0 it may worth adjusting window positions slightly to suit *Turbotext*. In many cases windows will already be in the right place. For instance there's no need to move default SCR and CON windows, or the windows set up when the QL is turned on, unless fancy borders are used.

Fast spaces

Many displays - spreadsheets, tabulated data, justified text - contain a large number of spaces. Some versions of *SPEEDSCREEN* recognise space characters, CHR$(32), as special cases and print them specially quickly.

This may cause problems in rare programs that re-define space characters so that they are not blank. You can recognise these programs by the fact that some other symbol normally appears when you type a space. If you're using such a program, or a program that never prints spaces, you should choose a version of *SPEEDSCREEN* that treats a space like any other character. That version will be slower for displays with lots of real blank spaces in them, but at least it will be able to tell the difference between real spaces and re-defined ones.

Colour Synthesis and Fat founts

Sinclair's system documentation says that each character may be composed of up to five columns of coloured dots. Standard QL sets of characters, or 'founts', comply with this rule, so that there is always a blank column of dots between each character in CSIZE 0,0 - six dots wide.

Some desk-top publishing programs define characters containing up to eight columns. Standard versions of *SPEEDSCREEN* cope with characters up to six columns wide, but they don't handle non-standard 'fat founts'.

If your programs use fat founts you need a special version of *SPEEDSCREEN* containing extra code. This copes with characters up to the eight columns wide, but it is 6K larger and between 3 and 7 per cent slower at *Colour Synthesis*. Don't choose it unless you use fat founts.

Turbotext and Fat founts

Turbotext can cope with one fat fount without extra code. However, *Turbotext* can only use one fount, and normally that's a copy of the five column-wide standard ROM fount.

When you configure *SPEEDSCREEN* you can tell *Turbotext* to use a different fount. This can give very fast, attractive displays but *Turbotext* is not used for input or editing, so data typed by the user appears in the normal ROM lettering while output uses the new typeface. *Turbotext* is only used when ROM characters would otherwise appear, so *Colour Synthesis* is used instead of *Turbotext* if you redefine a window's fount.

SPEEDSCREEN VERSIONS

Every version of *SPEEDSCREEN* has its own identifying letter, shown after the code revision number when *SPEEDSCREEN* loads. These are the versions:

Version letter	P	N	F	T	S	C	Z	X
Approx size in K	5.5	11	17	4-6	7	7	11	17
Colour Synthesis	Y	Y	Y	N	Y	Y	Y	Y
Turbotext	N	Y	Y	Y	N	N	Y	Y
Fat founts	N	N	Y	N	N	N	N	Y
Fast spaces	Y	Y	Y	Y	Y	N	N	N

Use version P if running large programs (e.g. The Editor or Psion software) on a QL with only 128K of memory. This is a 'cut down' version which speeds up scrolling and text output for MODE 4, OVER 0, CSIZE 0,0.

Version N is the fastest on an expanded QL. You only need the slower versions F, Z or X if you intend to use fat founts or re-defined spaces.

Versions S and C save some memory if your programs never use CSIZE 1,0.

To choose a version, and copy it to your working disk or tape, you must run CONFIG_TASK. This runs automatically if you reset your QL with the *SPEEDSCREEN* master in drive 1 and press F1 or F2.

Alternatively you can run it by putting the master in drive 1 & typing:

EXEC_W MDV1_CONFIG_TASK (or FLP1, etc, to suit your system)

Once you have configured *SPEEDSCREEN* for your system, you can load it, and the utilities you have chosen, from the working copy it makes for you. You can freely copy any of the configured files to other devices, but you may not supply copies to anyone who has not bought *SPEEDSCREEN*.

Put your master tape or disk in a safe place in case you need to select a different configuration another day. **Do not** try to make a working copy on the master disk or tape. If your master becomes damaged you must return it - or its remains! - to Creative CodeWorks to get a new copy.

LOADING *SPEEDSCREEN*

Once you have configured *SPEEDSCREEN* for your system (see page 2) you can load it with the normal commands to add an 'extension' to the QL. The exact instructions vary depending upon the version you are loading, so the *SPEEDSCREEN* configuration task displays the appropriate commands.

The configuration task creates a 'BOOT' program to load *SPEEDSCREEN* automatically. You can run this by re-setting the computer with the version in drive 1, and pressing F1 or F2 as appropriate, or by typing:

LRUN MDV1_BOOT (or FLP1, etc, to suit your system)

SPEEDSCREEN loads into memory reserved with the RESPR command. RESPR does not work (it says 'not complete') if you use it while tasks are running so it makes sense to load *SPEEDSCREEN* at the start of a session.

You should only load *SPEEDSCREEN* once. It adds new SuperBASIC commands and the QL gets confused if you add the same commands repeatedly. Please reset the machine before loading a different version.

Once *SPEEDSCREEN* has been called all currently-open windows, and all SCR and CON windows opened from then on, will have access to it. *SPEEDSCREEN* can handle printing in any number of windows at once. It works just like the normal screen printing routines - but up to 12.5 times faster.

HINTS AND TIPS

FAST SCROLLING

By default, *SPEEDSCREEN* scrolling is at about twice normal speed, for example when COPYing to SCR. The new command _SCROLL lets you select higher speeds for free-scrolling displays such as LIST and COPY.

SPEEDSCREEN uses its own fast scrolling routines whenever a 'New Line' character - CHR$(10) - is printed on the last line of the window, or when characters spill over the end of the window.

SPEEDSCREEN can scroll the screen about twice as fast as the ROM. Any further improvement is prevented by the sheer effort required to move anything up to 32K of slow video memory through the 68008 processor every time a new line is printed.

Normally when text runs off the bottom line of a window the computer moves everything up by one line, and creates a new blank line at the bottom of the window. But *SPEEDSCREEN* prints a single line of characters so fast that the time taken to move the entire window is much greater than the time spent printing the characters.

To make scrolling even faster you must reduce the frequency with which the machine has to roll the page up to get new paper to write upon.

The QL normally rolls up one line whenever the end of the display is crossed. But you can set *SPEEDSCREEN* to roll two lines, or more, and this can dramatically increase the speed at which free-scrolling data appears, for instance when you use LIST, VIEW or COPY to SCR. You do this with a new command, revealed on the next page.

The new command is _SCROLL:

_SCROLL 2 makes *SPEEDSCREEN* roll in two lines instead of 1

This makes free-scrolling displays move much faster, although slightly less smoothly than normal. Larger parameters for _SCROLL make scrolling progressively faster and more jerky.

The command lets you 'tailor' the rate of free scrolling to suit your speed of reading. _SCROLL 2 is about twice as fast as _SCROLL 1 because the computer gets to print two lines between each scrolling operation. _SCROLL 4 is almost twice as fast again, but noticeably less smooth. _SCROLL allows parameters up to 30. _SCROLL 2 is the best setting for BASIC programming. The default, _SCROLL 1, moves displays as normal.

The setting of _SCROLL is only taken into account when *SPEEDSCREEN* character output routines are being used. It doesn't affect the normal SCROLL command, or scrolling in CSIZEs *SPEEDSCREEN* does not recognise.

If the setting of _SCROLL is greater or equal to the number of lines in a window it is ignored; such windows scroll normally, line by line. This means you can use, say, _SCROLL 10 to make the main window go very fast, without losing sight of your commands as soon as you type them into the shallow window at the bottom of the screen!

NEW FOUNTS

SPEEDSCREEN makes it easy to use new characters in your own programs. Whenever the QL prints a character it looks up the shape from a table, called a 'fount'. *SPEEDSCREEN*'s *Colour Synthesis* routines can read just the same founts as the normal QL display code, so you can print new founts as fast as the standard one.

The basic QL doesn't include a command to change founts, even though it can work with other founts. *SPEEDSCREEN* adds the new command _FOUNT to SuperBASIC. This lets you set or reset the founts used in any window. The size of a fount is not fixed. Each fount contains a sequence of character patterns, starting at a particular character-code and continuing sequentially for a certain number of characters. QL codes are listed in the Concepts section of the QL User Guide.

Each window has two founts associated with it. The first fount usually contains the shapes for codes 31 to 127, while the second covers codes 127 (again!) to 191. The QL resolves clashes between the two founts by looking at the first fount to see if it includes a particular code. If not, it looks in the second fount. If the second fount does not include the code either, the **first** character in the **second** fount is used.

This explains why there are two different patterns for character code 127. The copyright sign in the first fount is used if CHR$(127) is printed; the chequerboard pattern at the start of the second fount appears for every code that can't be found in either fount.

The character-sets supplied with *SPEEDSCREEN* use the standard groups of codes, but there's no reason why you should do the same. It's easy to make up a fount containing definitions for all 256 possible codes. If you use that as fount 1, the second fount will never be used.

Using the _FOUNT command

Firstly, load the fount into reserved memory. A standard fount 1 uses 875 bytes - two bytes to record the start code and number of patterns, then nine bytes for each pattern. To reserve that much space, type:

```
start=RESPR(875)
```

That sets START to the address of the reserved memory. The next step is to load the chosen fount into that memory:

```
LBYTES "mdv1_fantasy_fount",start
```

Once the fount is loaded you can associate it with any window - say, the listing window, channel 2 - by typing:

```
_FOUNT #2,start
```

Subsequent LIST commands will use the new fount. An address of 0 tells the QL to use the standard fount. To reset the listing fount to the usual character shapes, type:

```
_FOUNT #2,0
```

Once you have loaded a fount you can use it with any number of SuperBASIC windows - just put the required channel number after the hash character. It must be a console or screen channel. If you don't specify a channel the QL assumes channel 1 - the default for PRINT and INPUT.

If you supply a channel number and one address, only the first fount is affected. To change the second fount you must supply a second address:

```
_FOUNT #0,0,start2
```

After this command window 0 will use the standard fount 1 and read a new fount 2 from address START2.

SPEEDSCREEN is supplied with eight founts, and a utility program so you can create more of your own design. These are the founts we supply:

Name	Codes	Length	Details
UKSTD1	31-127	875	The standard fount 1 used in UK QLs
UKSTD2	127-191	587	The second fount built into UK QLs
BOLD6	31-127	875	A heavy face for CSIZE 0,0
SHORT	31-127	875	A vertically condensed face
SERIF	31-127	875	A newspaper-style lettering
FANTASY	31-127	875	A clear 'adventure game' face
SCIFI	31-127	875	An 'Analog Science Fiction' style
BOLD8	31-127	875	A wide heavy CSIZE 1,0 face

SERIF, FANTASY, SCIFI and BOLD8 are 8 pixels wide - wider than standard QDOS founts - so you must print them with the fat fount versions of *SPEEDSCREEN*, or build them into Turbotext versions.

All file names have the extension '_FOUNT'. Lengths are in bytes.

Designing your own founts

These founts were designed with a utility called the 'Quantum Leap User Defined Graphics Editor' (QLUDGE for short). You load it like this:

EXEC_W MDV1_QLUDGE_TASK (or FLP1, etc, to suit your system)

You are asked to type the name of the fount file that you want to alter. Type the name (e.g. MDV1_BOLD6_FOUNT) or just press ENTER to create a new fount, entering the first code and number of characters when asked.

A menu appears. From this menu you can view the entire character set, copy patterns from one character to another, save the fount, or edit individual characters with a joystick or cursor keys. Press F1 for help.

Etymological esoterica, or The A-Z of founts

Print workers called a set of metal character shapes a 'fount'. Computer programmers have borrowed the word and use it to refer to character shapes for diplays, made out of dots rather than alloy. 'Font' is an alternative spelling of the same word.

The shorter spelling makes sense in that it matches the pronuciation and avoids confusion with 'fount', the poetical way to say 'fountain', pronounced, unlike the printing term, the way it is spelt. However, English has never let pronunciation stand in the way of spelling, and it seems unlikely that printers or programmers will confuse people often by digressing into poetry about fountains.

Printers derived their spelling from 'foundry' - the place where the alloy letters were cast. Of course, they might just have chosen it to use up excess stocks of the letter 'u'....

NEW CHARACTER-SIZES

SPEEDSCREEN doesn't limit you to the QL's usual range of CSIZEs. It lets you adjust the distance between characters on a pixel-by-pixel basis. This gives you lots of extra character-sizes, and makes it easy to get the best possible compromise between clarity and effective use of the screen area. For instance you can create a new CSIZE midway between CSIZE 0,0 and 1,0, add extra space between lines, or compress text vertically to get 31 readable lines on a screen.

The distance, in pixels, between the start of each character and the start of the next one on the same line is called the XSTEP. The distance in pixels from the top of one line of characters to the top of the next is the YSTEP. Standard CSIZEs use the following XSTEPs and YSTEPs:

CSIZE	0,0	0,1	1,0	1,1	2,0	2,1	3,0	3,1
XSTEP	6	6	8	8	12	12	16	16
YSTEP	10	20	10	20	10	20	10	20

Whenever you change CSIZE the XSTEP and YSTEP are automatically set to the default values shown in the table above.

SPEEDSCREEN lets you specify any sensible STEP you like, either vertically or horizontally. For instance, this command:

_XSTEP #2,7

sets the horizontal gap between characters printed in the listing window to 7. That changes CSIZE 0,0 into a size midway between CSIZE 0,0 and 1,0. Quill uses just such a size in its 60 column mode; it's a neat compromise that lets you print lots of text clearly on a TV display.

If you want commands and error-messages to be spaced out more than usual you could enter:

_YSTEP #0,12

to put an extra two pixels between each lines of characters printed in the command window.

It is useful to be able to increase the number of characters that can be shown on the QL screen. You can do this by setting a YSTEP less than the usual ten pixels. The snag is that character-sets use the lower nine rows of every character, and UNDER uses the ninth row.

The file SHORT_FOUNT contains a re-defined character-set designed for use with a YSTEP of 8. It is almost as easy to read as the normal QL fount, but lets you print 6 extra lines of text on the screen. All the characters fit into the top seven rows of the pattern, leaving the last two rows blank.

Non-standard XSTEP values stop *SPEEDSCREEN* using its very fastest routines for printing on black paper, but screen output is still much faster than normal. *SPEEDSCREEN* can cope with any YSTEP at top speed.

Valid STEPs

The XSTEP and YSTEP commands use channel 1, like PRINT and _FOUNT, unless you specify an explicit channel number. The STEP itself can be any whole number between 6 and 40. Fractions are rounded up or down to the nearest whole number. XSTEPs below six are not desirable, because they make it impossible to print letters like 'm' and 'w' properly.

XSTEP and YSTEP only work with SCR and CON windows - other devices give a 'bad parameter' report if you try to change their STEP.

Character overlaps

XSTEP and YSTEP do not change the actual size of characters, but only the distance between each one and the next. It follows that you can get odd results if screens are drawn from the bottom upwards (as in some editors) when a small YSTEP is being used. The bottom of one line may overwrite the top of the line below. The answer is to use CLS or SCROLL to clear areas before printing, and then to print in OVER 1 instead of OVER 0, so that blank paper at the bottom of a character doesn't mess up lower lines.

Everything works fine if you print screens from the top down, as most programs do. Remember that the last line is taller than the others when you set up your window size, or the last line may spill over the bottom of the window slightly.

Don't use a reduced YSTEP on the very bottom line of the screen without allowing space below the window for lines at the bottom of each letter. It is important to ensure that displays which use reduced YSTEPs don't print off the bottom of the screen; the next memory area contains vital information (system variables). The QL will stop if this is overwritten. That's why you can display no more than 31 lines with a YSTEP of 8: the bottom part of text printed on the 32nd line would overflow the screen.

CHAR_INC faults

Some Toolkits include a command CHAR_INC, which adjusts the XSTEP and YSTEP with one single command. This works fine as long as you don't type anything silly. There is **no range checking** in versions of the CHAR_INC command that we have seen: even negative STEPs are passed. The QL's ROM and *SPEEDSCREEN* both get very upset if you accidentally set STEP to a daft value. We recommend that you use _XSTEP and _YSTEP in programs, rather than CHAR_INC, so that dangerous settings are rejected.

TURNING *SPEEDSCREEN* OFF AND ON

You can enable or disable *SPEEDSCREEN* at any time. This is useful when performing comparative timings, or when you want to use a program that is not compatible with the version of *SPEEDSCREEN* you have loaded. To put the QL in the state it was in before you loaded *SPEEDSCREEN*, type:

```
_SPEED 0
```

You can do this as a command or in a program. The commands _FOUNT, _XSTEP and _YSTEP still work, but new settings of _SCROLL are ignored until you re-enable *SPEEDSCREEN*. Note: utilities such as ICE, QRAM and Taskmaster **automatically** turn SPEEDSCREEN back on at once!

There are two ways to enable *SPEEDSCREEN*. _SPEED 1 turns it on and prints the version number in window #0. _SPEED 2 turns it on 'silently'.

MODE CHANGES

Machine-code programs that change display MODE can confuse *SPEEDSCREEN* unless they update character-sizes and colours to suit the new MODE. The QL ROM MODE '**trap**' is not clever enough to do this for tasks unless the standard, slow ROM display routines are in use. Colours and CSIZEs are NOT automatically altered if MODE 8 is selected from machine-code while *SPEEDSCREEN* is turned on.

SPEEDSCREEN automatically selects MODE 4 when it is loaded. MODE in compiled or interpreted BASIC is fine; even the SV.TVMOD bug is fixed.

However, for example, you may see strange (but harmless) effects if you load *SPEEDSCREEN* then load Psion Quill, switch to the 40 column (MODE 8) display and return to BASIC. The 40 column Quill mode will work fine, but SuperBASIC channels will still be connected to *SPEEDSCREEN*s MODE 4 routines when you return, giving very fast but strange-looking flashing characters in MODE 8. Everything will be fine if you type MODE 4 to get the display and SuperBASIC back in synch.

MODE 4 clears the screen to black, but doesn't redraw the PAPER of every active window. This speeds up programs that use MODE to tidy the screen.

QRAM

SPEEDSCREEN works well with current versions of QRAM. However, beware: several versions of QRAM have been shipped, and some were bug-ridden.

QRAM users should watch out for three things:

(1) *SPEEDSCREEN* must be loaded before QRAM. QRAM sets up a background process to monitor the MODE, removing the quirks noted at the foot of page 10 but imposing a small time penalty.

(2) *SPEEDSCREEN* should not be turned off with _SPEED 0, as that would disconnect old versions of QRAM, prompting them to crash the system! More recent versions of QRAM just ignore _SPEED 0.

(3) You should not use early versions of QRAM with *SPEEDSCREEN* if you intend to use MODE 8 and MODE 4 windows at the same time. Later versions (such as German QRAM) work fine with mixed modes. At the time of writing we understood that QRAM was being re-coded for the third time. If you have an early version we advise you to obtain an upgrade from QJUMP Ltd, the publishers.

PRINTING ON STIPPLED PAPER

SPEEDSCREEN accelerates CLS and SCROLL for all paper colours, although it only speeds up printing in solid colours and the stipples 128-191. If you want to print on another stipple colour - say, 8-127 or 192-255 - you may still be able to take advantage of *SPEEDSCREEN*.

The trick is to clear or scroll the required part of the screen before printing, and then to print with OVER 1 set. This prints ink on top of the existing paper, whatever colour it happens to be. The technique can be useful even if you're using a paper colour that *SPEEDSCREEN* can optimise, because printing in OVER 1 is significantly faster that output in coloured paper with OVER 0. If your program uses a scrolling display, or never prints on top of existing text, use OVER 1 instead of OVER 0.

If you use OVER -1 on stippled paper you can get the same effect as you would by using stippled inks that *SPEEDSCREEN* does not directly support.

GETTING BEST RESULTS FROM *SPEEDSCREEN*

SPEEDSCREEN works best when printing long sequences of characters on the screen. If every character is printed with a separate call to the operating system (as, for no very sensible reason, on the first edition of the CP/Mulator) the system spends much more time getting ready to print than it actually spends printing characters. To ensure top speed, always concatenate strings before printing them. For example, write: **PRINT "Hello " & name$ & CHR$(10);** rather than **PRINT "Hello ";name$** .

FURTHER INFORMATION

Any last-minute updates to the *SPEEDSCREEN* documentation can be found in the Quill document WHATSUP_DOC, automatically written out when you configure a version of *SPEEDSCREEN*. Please read that document to find out about extra features in your copy.

SPEEDSCREEN PERFORMANCE

SPEEDSCREEN is fastest when printing text in the default fount, in CSIZE 1,0 with windows 'on grid' (see below). Black paper is printed fastest in OVER 0, particularly in conjunction with white ink. Underlining, and coloured PAPER in OVER 0, stop *SPEEDSCREEN* using its fastest routines.

These are the speed-improvements found when re-drawing screens with a short program (PRINTSPEED, supplied). The program repeatedly re-draws a screen containing 1000 mixed text characters.

MODE	CSIZE	INK	PAPER	OVER	UNDER	Speed factor
4	0,0	white	black	0	0	5.3
4	0,0	all	black	0	0	4.7
4	0,0	all	all	0	all	3.8
4	0,0	all	all	-1	all	4.8
4	0,0	all	all	1	all	4.3
4	1,0	white	black	0	0	9.9 / 12.6
4	1,0	all	black	0	0	8.3 / 10.5
4	1,0	all	all	0	0	6.6 / 8.4
4	1,0	all	all	-1	0	7.7 / 9.7
4	1,0	all	all	1	0	7.7 / 9.7

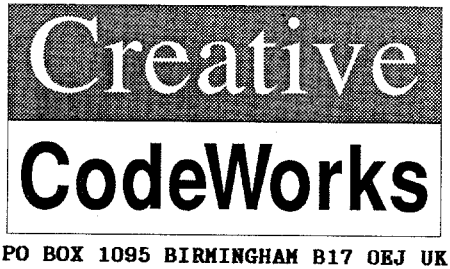
Two speeds are shown for CSIZE 1,0, because *SPEEDSCREEN*, like QDOS, goes extra fast in that size if a window is on-grid (i.e. the memory used for the window starts on a word boundary). The standard QL windows - SCR, CON and F1/F2 defaults - use this format.

The first figure shows speed-ups for windows that are already on-grid. The second shows the speed-up factor that *SPEEDSCREEN* can provide if output is changed from CSIZE 0,0 to CSIZE 1,0, or if a CSIZE 1,0 window is moved on-grid. *SPEEDSCREEN* timings for off-grid and underlined CSIZE 1,0 match those for CSIZE 0,0. 'ALL', in terms of colour, means solid colours 0-7 and stipples 128-191. Figures were correct on 1st November 1987, but may change as *SPEEDSCREEN* is developed. Both *SPEEDSCREEN* and QDOS benefit slightly if channel definition blocks are in fast memory.

Variation between machines

All QLs use the same ROM display code. The maximum improvement *SPEEDSCREEN* can give depends upon just one thing: the speed of the memory in your computer. Programs in ROM run quickly because the processor has exclusive control over them, so it can read instructions or data from them as fast as it can go. Access to most types of RAM must be shared between the processor and video or 'refresh' circuitry, which gets in the way. *SPEEDSCREEN* is coded to minimise this effect, but it can be degraded by 31 per cent, typically, if it runs in shared video RAM. Expanded systems run at an in-between speed, depending upon their type and date of manufacture. Newer boards are generally slower. Only CST RAM-Plus is as fast as ROM

Creative CodeWorks is developing a 16K SPEEDSCREEN ROM cartridge.



PO BOX 1095 BIRMINGHAM B17 0EJ UK