IF VER$ =

The title of this article is actually one of the shortest and potentially most useful pieces of code I've found in all my years with the QL.

It's a simple little line of code which lets your programs do different things on different systems. By and large, all versions of QL BASIC have been around long enough for us all to know what each version can do. By checking the version of BASIC we can let a program do different actions, taking advantage of facilities added to Minerva or SBASIC systems for example.

Probably the simplest example is that of a program which needs to be able to open a window to fill the entire screen. On a QDOS machine, it is pretty safe to assume that the display will always be 512x256 pixels (the uQLx emulator can be an exception), but modern SMSQ/E systems can have screens with many more pixels across and down the screen. SBASIC on these systems has extensions which return the width and height of the screen, for example. The SBASIC function SCR_XLIM tells us how many pixels across and SCR_YLIM the number of pixels down.

So, since SBASIC always has a VER$ of "HBA" all we need to do is test the version of BASIC with VER$ and if it turns out to be SBASIC, we can call SCR_XLIM and SCR_YLIM, otherwise assume bog-standard QDOS default values.

Here's a sample little program:

```
1000 DEFine PROCedure Cover_The_Screen
1010   wide = 512 : high = 256 : REMark QDOS defaults
1020   v$ = VER$
1030   IF v$ = 'HBA' THEN
1040     wide = SCR_XLIM
1050     high = SCR_YLIM
1060   END IF
1070   WINDOW #0,wide,high,0,0
1080   CLS #0
1090 END DEFine Cover_The_Screen
```

This little program lets channel #0 cover the entire screen on virtually any system. All it does is assume first it's going to run on an old QDOS system without any high resolution screens, then it checks VER$ to see if it's running on an SBASIC system and if it is, tries the new keywords to make use of the new facilities if they exist on this system.

Before we go any further, a small note about line 1020. Some older versions of QDOS have a minor issue with the VER$ function which can cause problems if used directly in some complex expressions such as IF clauses, so it is usually safest to assume that it is best to copy it to a string variable first, then use the variable in IF phrases or string expressions. If the variable name I've used in these examples is used in your programs, just add something like 1005 LOCal v$ to limit values to local within procedures or functions.

If you have a graphical program which PEEKs and POKEs the display (naughty!) it can be handy to extract the relevant information from the system where keywords are provided to do this. To PEEK and POKE in the display on modern systems, we need to know the following in addition to the above:

(1) the base address of the screen, and
(2) how many bytes across the screen

We can extend the above example program to use the SCR_BASE and SCR_LLEN functions to return the screen base address and screen line length in bytes respectively:

```
1000 DEFine PROCedure Cover_The_Screen
1010   LOCal v$
1020   wide = 512 : high = 256 : REMark QDOS defaults
1030   sbase = 131072 : REMark screen base address on Sinclair QL
```

```
1040   bpl = 128 : REMark line length in bytes on Sinclair QL
1050   v$ = VER$
1060   IF v$ = 'HBA' THEN
1070     wide = SCR_XLIM
1080     high = SCR_YLIM
1090     sbase = SCR_BASE
1100     bpl = SCR_LLEN
1110   END IF
1120   WINDOW #0,wide,high,0,0
1130   CLS #0
1140   PRINT #0,'Screen width = ';wide;' pixels'
1150   PRINT #0,'Screen height = ';high;' pixels'
1160   PRINT #0,'Screen base address = ';sbase
1170   PRINT #0,'Screen line length in bytes = ';bpl
1180 END DEFine Cover_The_Screen
```

On an original QL, you'll get values of 512 pixels wide, 256 pixels high, base address of 131072 and a line length of 128 bytes. On a more modern system with bigger display, you'll gett he results for that system. Where this can come in handy is to reliably save a screen picture. On a QL, this just involves saving 32,768 bytes from address 131072. On a Q40, QPC, QXL etc this might be more complex as both the size and location of the video screen can vary. But it's quite easy to work out, using the SBASIC functions listed above:

```
2000 DEFine PROCedure Save_The_Screen (filename$)
2010   LOCal v$
2020   wide = 512 : high = 256 : REMark QDOS defaults
2030   sbase = 131072 : REMark screen base address on Sinclair QL
2040   bpl = 128 : REMark line length in bytes on Sinclair QL
2050   v$ = VER$
2060   IF v$ = 'HBA' THEN
2070     wide = SCR_XLIM
2080     high = SCR_YLIM
2090     sbase = SCR_BASE
2100     bpl = SCR_LLEN
2110   END IF
2120   SBYTES filename$,sbase,high*bpl
2130 END DEFine Save_The_Screen
```

This works out where the screen starts in memory, then saves a chunk from memory depending on the height of the screen (the variable high) and the number of bytes per line.

To reload a screen, just LBYTES the screen to the relevant base address. We need to be careful that the screen is the same size as the previously saved file, the length of which we can check with the FLEN function of Toolkit 2 (built into SBASIC).

```
2150 DEFine PROCedure Load_Screen (filename$)
2160   LOCal fl,v$
2170   fl = FLEN(\filename$)
2180   bpl = 128
2190   sbase = 131072
2200   high = 256
2210   v$ = VER$
2220   IF v$ = 'HBA' THEN
2230     bpl = SCR_LLEN
2240     sbase = SCR_BASE
2250     high = SCR_YLIM
2260   END IF
2270   screenlength = bpl * high
2280   IF fl = screenlength THEN
2290     LBYTES filename$,sbase
2300   ELSE
```

```
2310     PRINT #0,'Unsuitable screen'
2320   END IF
2330 END DEFine Load_Screen
```

It is possible to harness Minerva extensions like this too. For example, QDOS has no function to tell you the base address of the system variables, because they are always at a fixed address of 163840. On a Minerva system, a second 32K screen may be in use at that address, so the system variables have to move to another address. They may also be at a different address on SMSQ/E systems, because the screen can vary in size for example. So we need a method of finding out where they are without having to resort to machine code. The answer is that Minerva and SBASIC have a special form of the VER$ function which can return the base address of the system variables - VER$(-2)

```
3000 DEFine FuNction SystemVariables
3010   LOCal v$
3020   v$ = VER$
3030   sv = 163840 : REMark default for old QDOS systems
3040   IF v$ = 'JSL1' OR v$ = 'HBA' THEN sv = VER$(-2)
3050   RETurn sv
3060 END DEFine SystemVariables
```

So you can see that the underlying principle is to apply sensible default values which apply to an original QL with a Sinclair ROM, then test the version of BASIC and if Minerva ("JSL1") or SBASIC ("HBA") are detected, use extended facilities.

Qliberator will allow you to compile programs like this, where extensions are typed into a program but not always used, depending on system it runs on. Turbo won't - it tests for the extension being present as the compiled task starts and stops with an error message if a program includes, say, the keyword SCR_LLEN, but the system doesn't have that installed. So the technique is less useful in Turbo compiled programs.

Exists

This is another very useful extension which can be used for the same sort of programming concept. It checks through the name table for a given keyword and returns 1 if found, or 0 if not.

This allows a program to check for the existence of a particular keyword and take different actions depending on whether or not it was found on the system. Again, less useful in compiled programs, although very useful in interpreted programs.

EXISTS is a basic extension function by Phil Borman, available from the Toolkits page on my website at http://www.dilwyn.me.uk/tk/index.html

A simple line like `IF EXISTS('TK2_EXT')` lets you check if Toolkit 2 extensions exist on your system.

A good example might be checking if the Jochen Merz menu extension exists on this system. Just check for an extension you know exists in menu_rext:

```
LET mr = EXISTS('FILE_SELECT$')
```

You can even check of pointer environment is installed:

```
LET pe = EXISTS('CKEYON')
```

This checks if the CKEYON (Cursor Keys On) extension is present, though this is less reliable as the authors of the pointer environment might one day choose to leave this out of pointer environment or SMSQ/E!

However, simple programming techniques like the above can help your program make use of facilities on more recent systems, while still being able to run on older systems. I've found techniques like this to

be a useful and very basic way of updating old programs to run on modern systems without sacrificing the ability to run on older systems.