

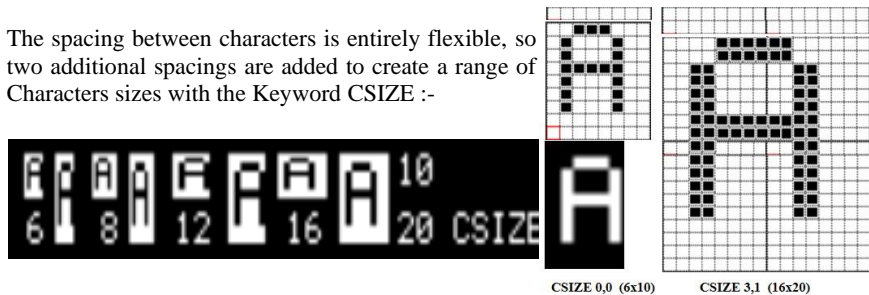
Introduction

Home Computers in the 1980's used Characters created from Bitmaps. The most common were 8x8 matrixes covering the Western ASCII code set. BITMap Files carry a Header, a number of bytes that might identify type, hold the overall File size, the width and height. For Font _fnt Files this will include A Start Font Code and the Number of Fonts. An App processes each Fonts Bitmap to create a pixel display positioned at required screen x,y coordinates.

Sinclair QL Character Fonts

The QL has a default Bitmap group of Patterns for ASCII Character Codes 32 (Space) to 127 ©, the common alphanumeric, punctuation, maths signs, brackets etc. A second group called the extended character set uses Codes 128 to 191. The Font header is the first two Bytes which hold the Lowest ASCII Font Code number followed by a Total number of Fonts. Then 9 Bytes for each Font to give a 9x8 matrix used by the display App. This is scaled in two widths 6 & 12 and two heights of 10 or 20 rows with a single or double blank row added above the (Bitmap) Display.

The spacing between characters is entirely flexible, so two additional spacings are added to create a range of Characters sizes with the Keyword CSIZE :-



When using different CSIZES across the various QL Platforms there is no guarantee of a full Bitmap Pattern being displayed. This can lead to some interesting and at times frustrating outcomes when using Modified Fonts for say Retro Gaming.

QBITS Font Editor Concept

The aspirations for a QBITS Font Editor began in the eighties. The Program was never finished for release, other events taking up ever more of my time. The concept was to Load alternative QL Font sets into memory, display them to screen as a Chart. A selected character would then be shown in a Bitmap display with the option to change the bit patten and save back to memory. After a number of Fonts had been edited in this way the whole Font Set could be saved as a new Font File for use with other Programs.

QBITS FontEditSE ‘_fnt’ Files

The QL Character BITMaps can be extended to include CHR\$(0 to 31) and (192 to 255). QBITS_FontEditSE arranges Codes 0 to 255 in four groups: (1) **QLFontA_fnt** (0/128), (2) **QLFont1_fnt** (32/96), (3) **QLFont2_fnt** (128/64), (4) **QLFontB_fnt** (128,128). At start-up **QLFontA** & **QLFontB** ‘_fnt’ files are loaded from the default Storage Device into reserved memory and the Font Chart displays CHR\$(0 to 255) to screen.

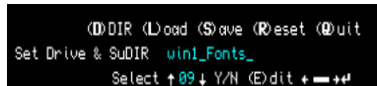
QBITS FontEditSE - Navigation

Navigate the Font Chart using the Cursor keys and Select the Highlighted Character with Spacebar. The Bitmap Grid is now active and Cursor Keys are used to highlight a Grid Cell. Use Spacebar to toggle between INK and STRIP (background) colour by setting binary bit between 1 & 0. Select ‘N’ to return without changing the existing Bitmap, ‘Y’ to write the changed Bit Pattern into memory.

QBITS FontEditSE – MENU (D)DIR (L)oad (S)ave (R)eset (E)xit

Access by pressing the Letter in Brackets and Information is displayed relative to the action requested. (D) allows Changes to Drive and SubDIRectories if present.

To access a SubDIRectory use (E)dit to change for example: ‘flp2_’ to ‘win1_Fonts_’, Enter to Return and ‘Y’ to accept. Then select (L)oad.



```
(D)DIR (L)oad (S)ave (R)eset (Q)uit
Set Drive & SubDIR win1_Fonts_
Select ↑09↓ Y/N (E)dit ←→
```

For Load / Save: First Select Storage Device with Up/Down Cursor keys and then Y/N.

For **LOAD** a search is made of available ‘_fnt’ Front files. A ‘File Not found’ will be displayed if Device has none. Use Up/Down Cursors keys to Select from those ‘_fnt’ Files found, then ‘Y’ to Load or ‘N’ to abort. The ‘_fnt’ file is checked for Character Group and loaded into selected Memory Address then Fonts are displayed to screen.




```
(D)DIR (L)oad (S)ave (R)eset (Q)uit
LBYTES dos1_QLFontA_fnt
Select ↑14↓ Y/N
```



```
(D)DIR (L)oad (S)ave (R)eset (Q)uit
SBYTES flp1_QLFontA_fnt
Select ↑01↓ Y/N (E)dit ←→
```

To **SAVE** use Up/Down Cursors and Select Group (1-2-3-4). The current Font Filename for that Group is displayed. Included is a Line Editor to Rename the Font Filename. When ready to SAVE a check is made, if Device unavailable ‘DEVICE ERROR’ is shown. An ‘Overwrite Y/N’ is given if the file is detected as already existing. If good to go the file is saved with ‘Saving...’ displayed before returning to Font Chart.

Press ‘R’ for **Reset** which prompt with ‘Y/N’, ‘N’ aborts and ‘Y’ Reloads Default Fonts.



```
(D)DIR (L)oad (S)ave (R)eset (Q)uit
Y/N
```



```
(D)DIR (L)oad (S)ave (R)eset (Q)uit
Y/N
```

Pressing ‘Q’ for **Quit** will again prompt with Y/N, ‘N’ returns to program ‘Y’ will action **FontExit** (see Line 1084) which will CLOSE opened channels and Free up Memory then attempt to LRUN dn\$ and return to the **QBITSProgs** Menu program.

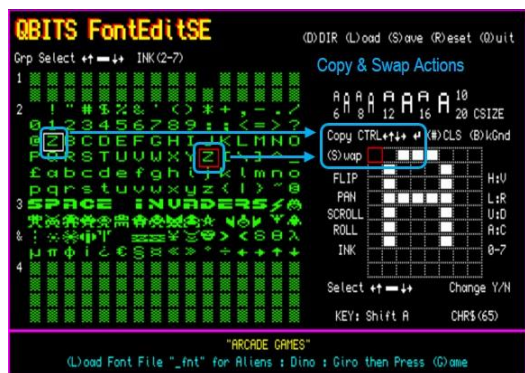
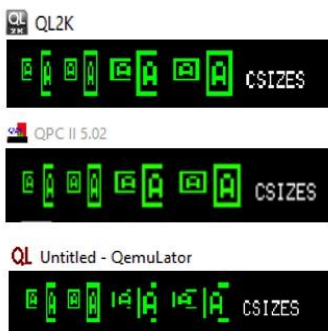
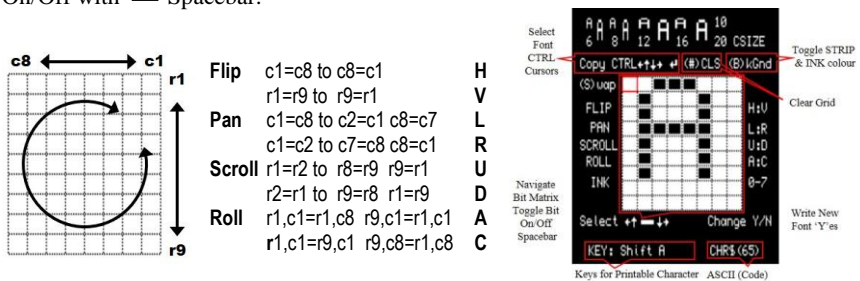
QBITS FontEditSE

The heart of the Editor is the Bitmap and what New or Altered Characters Fonts can be created. These may be Bold, Italics, or Personal and Stylised alphanumerical Fonts, futuristic or represent an ancient language such as Cuneiform or Runes. For Retro Games Transforming Fonts into Game pieces might also be desirable.

Select a **Font** from **Chart** with Cursor keys this is Outlined with a **White Box**. Press Spacebar and QBITS **Bitmap** Editor shows which bits are set to 1's & 0's [INK/BkGnd Colours]. The Font is displayed above in its different **CSIZES**. The Keying for Printable Fonts shown with **KEY:** and **CHR\$(n)** 'n' being the ASCII Character Code.

Where slight pattern changes are required copying an existing Chart Font into the Editor might be useful. To **Copy** use CTRL $\leftarrow \uparrow \downarrow \rightarrow$ to highlight a second Chart Font now Outlined with a **Red box**. Press \leftarrow Enter to action, alternatively you could Press **'S'** and Swap Font Grid Positions. For new designs (**#**) CLS clears Bitmap to '0's. (**B**)kGnd toggles INK & STRIP colours between Black(**0**) and White(**7**).

For further actions Flip (**H**)orizontally, or (**V**)ertically, PAN (**L**)eft or (**R**)ight, SCROLL (**U**p) & (**D**)own, Rotate (**A**)nti-clockwise or (**C**)lockwise by 90°. Use (**0-7**) to change. Font INK colour, Navigate Bitmap with Cursor keys $\leftarrow \uparrow \downarrow \rightarrow$ Toggle Bit INK colour On/Off with **Spacebar**.



QL Fonts & CSIZES

Showing the Fonts in their various CSIZES with different colours can be helpful in identifying possible display problems some QL O/S displays can be very different to that expected. CSIZE 0,2 [6 & 12] Print only first six columns of Bitmap.

QBITS QL Font Installation

CHAR_USE sets or resets one or both character QL Font groups. First memory from the common heap (RAM) area is allocated for loading a Character set. The number of bytes are rounded off to even values for LBYTES to work properly:

FBase1 = **ALCHP**(876) **LBYTES** FLP1_FONT1,FBase1

FBase2 = **ALCHP**(588) **LBYTES** FLP1_FONT2,FBase2

CHAR_USE#ch,FBase1,FBase2 assigns Font Sets to a specified **channel**.

With the Fonts Bitmaps assigned to memory the character pattern can be Read and Overwritten with the PEEK and POKE commands. Each is identified by an offset from the assigned Base address set by **ALCHP**. The first 2 Bytes hold the Lowest Code followed by the Total Number of Font Characters, then a number of 9 Byte blocks each of which represents a Character 8x9 matrix of bits set with 0 & 1's.

When program is finished QBITS Font Editor releases memory with **RECHP**(FBase?) and Resets to the default character sets with **CHAR_USE#ch,0,0**. If **RESPR**(876+588) is used to acquire Memory space the loaded Fonts remain available until a power down or a System Reset.

PIXEL Based Font Designs

Pixel-based Characters were popular in the 1980s, when the digital era was just beginning. One typeface stood above the rest the "Atari" font set of 1984 (or known as "Namco" in Japan). Based on the Western ACSII Character Set it lasted until larger resolutions and 3D gaming became the norm, it stands out as an example of how video game designers used a simple 8-by-8 Bitmap Grid to communicate to players.



Screens made up of 8x8 bitmap matrix tiles was also used in Arcade Games from the 1970s through to the 1990s. Their chunky and primitive visuals revealed artistry and ingenuity to make full-fledged characters out of just a few pixels. These Fonts were designed by hand on graph paper and coded into the game pixel by pixel. The result was a mix of some good ideas and some really bad ones.

Note: Retro Game Control

The SBASIC keyword **CHAR_INC#ch, x_inc, y_inc** sets Character width and height beyond those set by CSIZE. Here where the subject Character Font(s) move across the screen, an independent height can be set for **PAN**.

PAN #ch, +/- distance, 3 (PAN's the Cursor line Left or Right).

For example: **CHAR_INC#ch,16,32** Sets Font width at 16 Pixel with depth 32 rows

QBITS Program Code

To maintain compatibility across the various QL Platforms has its challenges. The original QL came with 128K of RAM, 32K used for the screen and more for running Jobs such as the SuperBASIC Interpreter. QBITS_FontEditSE should load and run on a BBQL but requires TK2 keywords CHAR_USE, CHAR_INC, CURSEN & CURDIS to be present. Extended memory is recommended as the QBITS Editor Special Edition running the ARCADE Games might have problems with original BBQL 128k of RAM.

For high speed QL Platforms scanning through the Font Charts, displays the relative Bitmap and their CSIZE's. For slower Platforms, BBQL etc. this aspect can be disabled see Lines 1041 & 1043 with Bitmap and CSIZE's shown only when a Font is selected.

QBITS FontEditorSE

This Special Edition explores Font designs used in classical ARCADE Games. My choice Space Invaders, Dino Run and Giro Rescue. Each use redefined Fonts of the ASCII extended range 128 to 191. Load relative '_fnt' file into the Editor and then press 'G'. The associated game should now be displayed in WINDOW#3.

Space Invaders



This simplified version has four levels. Move the Defender left or Right using Cursor Keys and Spacebar fires the laser. You have four Lives so watch out for enemy bombs.



Dino Run



Displayed are High Score and Accumulated Points. The program has Speed adjustment to overcome CPU Timing issues with different QL Platforms. Screen PAN's uses CHAR_INC, press Spacebar to Jump DINO over the approaching Cacti or you lose.



Giro Rescue



The aim rescue all the Escape Pods scattered through space, watch out for space Debris, they can weaken your shields and you only have limited fuel to complete each level.



Note: In utilising Code from their published free programs, I wish to give thanks to Dilwyn Jones DINO version for QL - 2022 and Henry Wrighton GIRO - Aug 1989.



QBITS QLFonts

To view the various Character Fonts a viewer has been added to the QBITS Progs Menu. Select and Press Enter.

Type in the QLFonts Source Device Name such as 'win1_QLFonts_' and Press Enter. If not available a 'No Files Found' is returned.

All being well a list of Character Fonts appear in the window below. Scroll Up/down and Select a Character Font and Press Enter the program returns to QBITS Progs Menu. However, the display is change to reflect the chosen Font. This has been actioned by use of CHAR_USE referenced to on previous page.

If you Press Enter to Return without making a Source entry, CHAR_USE is then reset to the QL Default Character Fonts.

QBITS QLFonts _bas Code

1000 REMark **QBITS_QLFonts_bas** [QBITS Character Font Viewer 2024 - QPC2] vM20

1002 Dev\$='win1_':MODE 4:gx=0:gy=0 :REMark Basic Settings

1004 **WHEN ERROR :eck=1:CONTINUE:END WHEN**

1006 REMark **Import QBITSConfig Settings - QPC2**

1007 OPEN _IN#9,Dev\$&'QBITSConfig':INPUT#9,gx\gy\dn\$:CLOSE#9

1009 DIM File\$(100,20):fm%=100 :REMark Set FontFiles

1010 FBase1=RESPR(1168) :REMark Heap Allocation

1012 Init_win:FontChar

1014 **DEFine PROCedure Init_win**

1015 OPEN#4,con_:WINDOW#4,180, 24,gx+20,gy+30:BORDER#4,1,255:PAPER#4,7:CLS#4

1016 OPEN#3,scr_:WINDOW#3,180,112,gx+20,gy+54:BORDER#3,1,255:PAPER#3,7:CLS#3

1017 OVER#4,1:CSIZE#4,2,0

1018 INK#4,0:FOR i=2 TO 3:CURSOR#4,i,1:PRINT#4,'QLFonts'

1019 OVER#4,0:CSIZE#4,0,0:CSIZE#3,1,0:

1020 CURSOR#4,2,11:PRINT#4,'Source: ':WINDOW#4,120,10,gx+70,gy+42:sf%=0

1021 **REPEAT lp**

1022 INPUT#4,FDrv\$:sf%=0:IF FDrv\$="":CLOSE#3:CLOSE#4:CHAR_USE 0,0:LRUN dn\$:STOP

1023 **FontList**:IF sf%<1:PRINT#4,'No Files Found':PAUSE 30:ELSE IF sf%>0:**FontChar**

1024 **END REPEAT lp**

1025 **END DEFine**

```

1027 DEFine PROCEDURE FontChar
1028 cy=0:FOR sn=1 TO 10:FontFile 3,sn,cy:cy=cy+11
1029 sn=1:sr=1:cy=0:CUSOR#4,2,16:PRINT#4,'LBYES '
1030 REPeat lp
1031 INK#3,0:CUSOR#3,2,-11+sr*11:PRINT#3,'½'
1032 PRINT#4,File$(sn)::K=CODE(INKEY$(-1))
1033 INK#3,7:CUSOR#3,2,-11+sr*11:PRINT#3,'½'
1034 SElect ON K
1035 =208:sr=s r-1:sn=sn -1:F_Up
1036 =216:sr=sr+1:sn=sn+1:F_Dn
1037 = 32:RECHP FBase1:CHAR_USE 0,0:CLOSE#3:CLOSE#4:LRUN dn$:STOP
1038 = 10:FontFile 1,sn,cy :CLOSE#3:CLOSE#4:LRUN dn$:STOP
1039 END SElect
1040 END REPeat lp
1041 END DEFine

1043 DEFine PROCEDURE F_Up
1044 IF sr<1 AND sn>0:SCROLL#3,11:FontFile 3,sn,0
1045 IF sr<1:sr=1
1046 IF sn<1:sn=1
1047 END DEFine

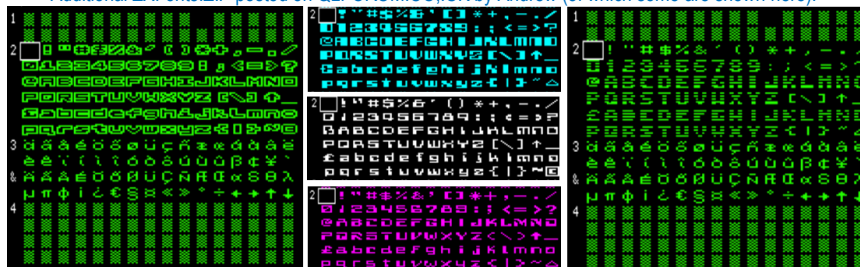
1049 DEFine PROCEDURE F_Dn
1050 IF sr>10 AND sn<ft%:SCROLL#3,-11:FontFile 3,sn,187
1051 IF sr>10:sr=10
1052 IF sn>ft%:sn=ft%
1053 END DEFine

1055 DEFine PROCEDURE FontFile(ch,sn,cy)
1056 LBYES FDrv$&File$(sn),FBase1:CHAR_USE#ch,FBase1,0
1057 INK#ch,0:CUSOR#ch,12,cy:PRINT#ch,File$(sn)
1058 END DEFine

1060 DEFine PROCEDURE FntList
1061 IF FDrv$=" :RETurn
1062 DELETE FDrv$&'FList':PRINT#4,'Checking...':PAUSE 30
1063 OPEN_NEW#9,FDrv$&'FList':DIR#9,FDrv$:CLOSE#9
1064 OPEN_IN#9, FDrv$&'FList':dl%=LEN(FDrv$):sf%=0
1065 REPeat dir_lp
1066 IF EOF(#9) OR sf%>fm%:sf%=sf%-1:ft%=sf%:CLOSE#9:EXIT dir_lp
1067 INPUT#9,F$:F$=F$(dl%-4 TO):fl%=LEN(F$)
1068 IF fl%<=20 AND '_fnt' INSTR F$>0:sf%=sf%+1:File$(sf%)=F$
1069 END REPeat dir_lp
1070 END DEFine

```

Additional ZXFonts.ZIP posted on QLFORUM.CO..UK by Andrew (of which some are shown here).



QBITS FontEdit - New FONT Designs

Just Tinkering with Bitmap layouts in Creating FONT Designs can take considerable time, but can produce some amazing results. However, having a specific goal helps in the decisions taken. QBITS FontEditSE has the tools to Copy, Swap and Modify Font Bitmaps to speed the creation of a number of Fonts. These can then be Saved in a Format used with the QL O/S for screen PRINT, having been loaded into memory and set with CHAR_USE to a WINDOW Channel.

QLLABEL_fnt	is a set of small Character Fonts for Labelling.
QLMUSIC_fnt	is a set of Music Fonts for Score sheets Notes Rests etc.
QLTIMER_fnt	is a set of Clock and Timeout Fonts

To use multiple Font sets in the same screen area, assign each of them with CHAR_USE to different Channels [ie. Overlapping Windows]. Some examples of this practice are used in the following code. Checkout the use of **DEF PROC QBPrnt** and channel assignments in **Music_Fonts & TIMER_Fonts**.

QBITS FONT Info - Code

3000 REMark **QBITS_FONTS_Info_bas** [QBITS Font Info Display 2024 QL40th - QPC2] v40

3002 dev\$='dos1_':gx=40:gy=40

```
3004 OPEN#5,con_:_WINDOW#5,496,136,gx+8,gy+79:PAPER#5,4
3005 OPEN#6,con_:_WINDOW#6,496,136,gx+8,gy+79:PAPER#6,4
3006 OPEN#7,con_:_WINDOW#7,496,136,gx+8,gy+79:PAPER#7,4:CLS#5:CLS#0
3007 base1=ALCHP(588):LBYTES dev$&QLLABEL_fnt',base1:CHAR_USE#5,0,base1
3008 base2=ALCHP(588):LBYTES dev$&QLMUSIC_fnt',base2:CHAR_USE#6,0,base2
3009 base3=ALCHP(588):LBYTES dev$&QLTIMER_fnt',base3:CHAR_USE#7,0,base3
3010 WINDOW 512,222,gx,gy:PAPER 0:BORDER 1,5:CLS:CSIZE 2,1:OVER 1
3011 INK 2:FOR i=0 TO 1:CURLSOR 12+i,4:PRINT "QBITS FONTS Info"
3012 INK 6:FOR i=0 TO 1:CURLSOR 14+i,5:PRINT "QBITS FONTS Info"
3013 INK 7:FOR i=0 TO 1:QBPrnt 1,7,11,12,30,'(L)abel (M)usic (T)imers'
3014 CSIZE 0,0:CURLSOR 260,12:PRINT 'Fonts Created with QBITS_FontEditSE_bas'
3015 OVER 0:Set_LED:Font$='QLLABEL':Label_Fonts:CURLSOR 60,60
3016 PRINT 'base=ALCHP(588):LBYTES '&dev$&Font$&'_fnt,base:CHAR_USE#ch,0,base'
```

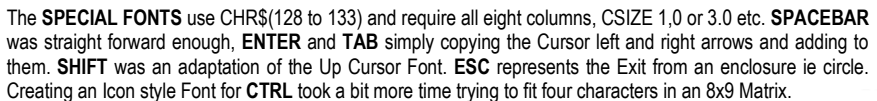
3018 **REPEAT lp**

```
3019 IF KEYROW(4)= 1:Font$='QLLABEL':Label_Fonts
3020 IF KEYROW(2)=64:Font$='QLMUSIC':Music_Fonts
3021 IF KEYROW(6)=64:Font$='QLTIMER':Timer_Fonts
3022 :
3023 IF KEYROW(1)= 8:LRUN dev$&'QBITS_FontEditSE_bas': CLOSE#6:STOP
3024 END REPEAT lp
```

Procedure **QBPrnt** Prints a string character-by-character with a set pixel spacing of Fonts. This enables the writing of Characters to screen with varying Character widths.

```
3026 DEFine PROCEDURE QBPrnt(ch%,ci%,cw%,cx%,cy%,str$)
3027 INK#ch%,ci%:sl%=LEN(str$)-1:IF cw%<5:cw%=5
3028 FOR c=0 TO sl%:CURLSOR#ch%,cx%+cw%*c,cy%:PRINT#ch%,str$(c+1)
3029 END Define
```

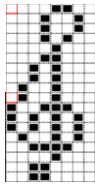

The QL extended range CHR\$(128 to 191) was used to provide an Alphanumeric set of 5x6Bit Matrix FONTS. The Printable key ranges of CHR\$(144-153) **CTRL 0 - 9** being used for numerals 0-9 and CHR\$(161-186) **CTRL Shift A-Z** for the A-to-Z characters. Groups CHR\$(134 to 143) and CHR\$(154 to 159) are used for Punctuation and Brackets etc. CHR\$(187 to 191) used for '\$' and Cursor Arrows. These can all be used with CSIZE 0.0 to 3.1.



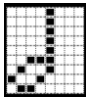
3031 DEFine PROCedure Label_Fonts

Page 40

QBITS FONT Info - MUSIC Fonts



The eight columns of QL 8x9 Bitmap are used so only use CSIZE 1,0 or 3,1.
The most difficult was the **GCleft** which required an upper and lower Font.
For Notes **Semibreve** a horizontal Ellipse and an angled Ellipse style Head for **Minim**, filled for **Crotchet** and then Flags 1 & 2 added to Stem for **Quaver** and **Semiquaver**. That left **Dots** and **Sharps** to be added.
The **RESTS** required a little more experimentation leaving the Bar, End Bar and Beat Signatures to finish off.



```

3062 DEFINE PROCEDURE Music_Fonts
3063 PAPER#6,4:CLS#6:INK#6,0:OVER#6,1:OVER#5,1:CUSOR#228,60:PRINT Font$
3064 CSIZE#6,2,1:FOR i=0 TO 1:CUSOR#6,6+i,4:PRINT#6,'QL MUSIC Fonts'
3065 CSIZE#6,0,0:CUSOR#6,210,12:PRINT#6,'QLMusic_fnt created for Musical SCORE'
3066 FOR i=1 TO 25 STEP 5:BLOCK#6,480,1,6,29+i,7:BLOCK#6,480,1,6,60+i,7
3067 FOR i=1 TO 15 STEP 3:BLOCK#6,480,1,6,91+i,7:BLOCK#6,484,1,6,110+i,7
3068 CSIZE#6,1,1:INK#6,0:STRIP#6,4:OVER#6,1:CUSOR#6,24,28:PRINT#6,CHR$(130)
3069 CUSOR#6,10,20:PRINT#6,CHR$(128):CUSOR#6,10,38:PRINT#6,CHR$(129)
3070 n=0:FOR i=132 TO 175:CUSOR#6,40+n,30:PRINT#6,CHR$(i):n=n+10:END FOR i
3071 CSIZE#6,0,0:REMARK CSIZE 1,1 RESTS      NOTE & DOTTED NOTE
3072 QBPrnt 5,0,6,20,52,'£'»$ 'CE'»$      @~'¥ & ¢~'¥ ¢ @~'¥
3073 QBPrnt 6,0,6,360,50,'+':QBPrnt 5,0,6,366,52,'³'»$ 'CE'»$ :REMARK Sharp
3074 QBPrnt 6,0,6,32,63,'GCleft      FCleft'
3075 QBPrnt 5,0,5,68,64,'£'»$ 'CE'»$      £'²'»$ 'CE'»$
3076 QBPrnt 5,0,5,52,72,'£'»$ 'CE'»$      £'²'»$ 'CE'»$
3077 CUSOR#6,0,0:CSIZE#6,3,1      :CUSOR#6,166,59:PRINT#6,CHR$(130)
3078 CUSOR#6,136,51:PRINT#6,CHR$(128):CUSOR#6,136,69:PRINT#6,CHR$(129)
3079 CSIZE#6,1,0
3080 QBPrnt 6,0,6,296,62,'Sharp f Shift_CTRL "
3081 QBPrnt 5,0,5,296,72,'£'»$ 'CE'»$      £'²'»$ 'CE'»$
3082 REMARK      CHR$(131)      CHR$(184-----185)
3083 CUSOR#6,10,88:PRINT#6,CHR$(128):CUSOR#6,10,97:PRINT#6,CHR$(129)
3084 CUSOR#6,22,93:PRINT#6,CHR$(181):CUSOR#6,10,112:PRINT#6,CHR$(130)
3085 n=0:FOR i=132 TO 175:CUSOR#6,40+n,94:PRINT#6,CHR$(i):n=n+10:END FOR i
3086 QBPrnt 5,0,6,20,84,'£'»$ 'CE'»$      £'²'»$ 'CE'»$
3087 QBPrnt 6,0,6,28,113,'Bar °± End Bar'
3088 QBPrnt 5,0,5,118,114,'£'»$ 'CE'»$      :REMARK CHR$(176-177)
3089 CUSOR#6,200,107:PRINT#6,CHR$(128):CUSOR#6,200,116:PRINT#6,CHR$(129)
3090 QBPrnt 6,0,8,210,113,'²'»$ 'CE'»$      :REMARK 2/2 2/4 3/4 4/4 3/8 6/8
3091 QBPrnt 5,0,5,264,114,'£'»$ 'CE'»$      :REMARK CHR$(178-183)
3092 QBPrnt 6,0,9,346,113,'°»'»$ 'CE'»$      :REMARK CHR$(186-191)
3093 QBPrnt 5,0,5,404,114,'£'»$ 'CE'»$      :REMARK CHR$(186-191)
3094 END DEFINE

```

Here the challenge was exploring the effects of Font combinations and sequencing for Clocks and Timeout Displays. This involved Backgrounds and Fonts used as Masks to achieve desired results - see Procedure **FontAct**. [Font Action]



Page 42

```

3132 REPEAT Fnt_Ip
3133 IF KEYROW(1)=64 :EXIT Fnt_Ip:END IF
3134 CURSOR#7,460,4 :PRINT#7,td%
3135 IF td%>0:PAUSE td%*2:ELSE PAUSE :REMark Set Timer
3136 IF KEYROW(1)= 4:td%=td%+1:IF td%>9:td%=9:PAUSE 1 :REMark Up/More
3137 IF KEYROW(1)=128:td%=td% -1:IF td%<1:td%=0:PAUSE 1 :REMark Down/Less
3138 CSIZE#7,1,0:sec%=sec%+1:Digits 7,0,4,230,62 :REMark Digital Clock
3139 cnt%=cnt%+1:fx%=fx%+xx%:FntBar 250,350,fx%,96 :REMark Sliding Bar
3140 CSIZE#7,3,1:mn%=166:IF c1>4:mn%=167:END IF :REMark Mask check
3141 FntAct 7,0, 23,40, 32,128+h :h=h+1 :IF h=12 :h=0 :REMark HR Clock
3142 FntAct 4,7, 22,90,mn%,160+c1:c1=c1+1 :IF c1>5:c1=0 :REMark Egg Timer
3143 FntAct 4,0,150,88, 32,168+c2:c2=c2+1 :IF c2=4:c2=0 :REMark Spin
3144 FntAct 4,0,180,88, 32,172+c3:c3=c3+1 :IF c3=2:c3=0 :REMark + *
3145 FntAct 4,0,210,88, 32,174+c4:c4=c4+1 :IF c4=2:c4=0 :REMark Arrow
3146 FntAct 4,7,380,88,176,177+c5:c5=c5+1 :IF c5=2:c5=0 :REMark Lamp
3147 FntAct 4,0,410,88, 32,179+c6:c6=c6+1 :IF c6=3:c6=0 :REMark . o O
3148 FntAct 4,0,440,88, 32,182+c7:c7=c7+1 :IF c7=2:c7=0 :REMark L Eye
3149 FntAct 4,0,460,88, 32,182+c8:c8=c8+f7 :IF c8=2:c8=0 :REMark R Eye
3150 END REPEAT Fnt_Ip
3151 END DEFINE

```



REPEAT Fnt_Ip cycles the counters for each FONT sequence.

PROC - FntAct Prints Font to Screen

Three things are involved the Background [STRIP] Mask [INK] Font [INK]

Fonts

Masks

Mask is printed first then FONT is overlayed on top.
Counter incremented and Next Font sequence is Printed.
Once the sequence is complete the counter is reset.



```

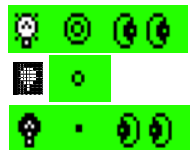
3153 DEFINE PROCEDURE FntAct(bc%,ic%,fx%,fy%,mn%,fn%)
3154 REMark Action BackGnd STRIP bc% INK ic% Cursor Position fx%,fy% Mask mn% Font fn%
3155 INK#7,ic%:STRIP#7,bc%:CURSOR#7,fx%,fy%:PRINT#7,CHR$(mn%) :REMark Mask
3156 INK#7,0:OVER#7,1 :CURSOR#7,fx%,fy%:PRINT#7,CHR$(fn%):OVER#7,0 :REMark Font
3157 END DEFINE

```



The Fonts sequence can vary as an ON/OFF or swap between two or a number of Fonts acting in sequence.

Lamp Mask

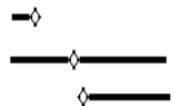


The Sliding Bar has two parts, the Bar and the Slider. In this example the Bar grows left to right with the Slider to max. The Slider then cycles across the Bar until it reaches a set number of loops. The Bar length then diminishes as the Slider moves left to right. With counter reset the process repeats.

```

159 DEFINE PROCEDURE FntBar(lx%,rx%,fx%,fy%)
3160 IF fx%<lx% OR fx%>rx% :xx%=-xx%:BLOCK#7,8,10,rx%+4,fy%,7
3161 IF fx%>rx% AND cnt%<60 :fx%=lx%
3162 IF fx%>rx% AND cnt%>60 :mc%=7:fx%=lx%
3163 IF fx%<lx% AND cnt%>99 :mc%=0:cnt%=0
3164 STRIP#7,7:INK#7,mc%:CURSOR#7,fx%,fy%:PRINT#7,CHR$(158)
3165 INK#7,0:CURSOR#7,fx%+xx%,fy%:PRINT#7,CHR$(159)
3166 END DEFINE

```



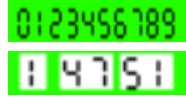


A simple Analogue Clock with CHR\$(128 to 139) Fonts one for each of the twelve-hour hand positions. CHR\$(140 to 143) 5x5Bit Time Fonts **12 3 6 9**



Clock Border Fonts CHR\$(184 -191)

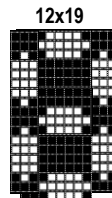
Digital Clock use CHR\$(144-153) CTRL 0 - 9 for Fonts with an LED Style of Numbers.



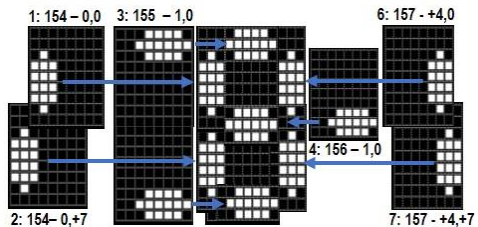
This continued with a more weighted Style of LED with numbers using just Four Fonts CHR\$(154-160). To create the numbers 0-9 an Array LED(9,7) is initialised with entries set with relative LED CHR\$(n) or to zero if not used. For example: ONE would be 6 & 7 set to CHR\$(157). Code then READs the Array entries and positions Font with CURSOR for PRINT.

```
3168 DEFINE PROCEDURE Digits(ch%,bc%,ic%,fx%,fy%)
3169 IF sec%=60:sec%=0:min%=min%+1:END IF :st% =sec% DIV 10:su%=sec% MOD 10
3170 IF min%=60:min%=0:hr% =hr%+1 :END IF :mt%=min% DIV 10:mu%=min% MOD 10
3171 IF hr%=10:hr%=0:min%=0:sec%=0:END IF :STRIP#ch%,7
3172 Prt_LED 446,43,st% :Prt_LED 462,43,su% :Prt_LED 332,62,su%
3173 Prt_LED 406,43,mt%:Prt_LED 424,43,mu%:Prt_LED 382,43,hr%
3174 sec$=CHR$(144+st%)&CHR$(144+su%):CURSOR#ch%,fx%+30,fy%:PRINT#ch%,sec$
3175 min$=CHR$(144+mt%)&CHR$(144+mu%):CURSOR#ch%,fx%+12,fy%:PRINT#ch%,min$
3176 hr$=CHR$(144+hr%):CURSOR#ch%,fx%,fy%:PRINT#ch%,hr$:STRIP#ch%,4
3177 END DEFINE
```

```
3179 DEFine PROCEDURE Prt_LED(dx%,dy%,dn%)
3180 BLOCK#7,17,18,dx%-2,dy%,7
3181 CURSOR#7,dx% ,dy% :PRINT#7,CHR$(LED(dn%,1)):OVER#7,1
3182 CURSOR#7,dx% ,dy%+7:PRINT#7,CHR$(LED(dn%,2))
3183 CURSOR#7,dx%+1,dy% :PRINT#7,CHR$(LED(dn%,3))
3184 CURSOR#7,dx%+1,dy%+1:PRINT#7,CHR$(LED(dn%,4))
3185 CURSOR#7,dx%+1,dy%+8:PRINT#7,CHR$(LED(dn%,5))
3186 CURSOR#7,dx%+4,dy% :PRINT#7,CHR$(LED(dn%,6))
3187 CURSOR#7,dx%+4,dy%+7:PRINT#7,CHR$(LED(dn%,7)):OVER#7,0
3188 END DEFine
```



```
3190 DEFINE PROCEDURE Set_LED
3191 DIM LED(9,7):RESTORE 3192:FOR a=0 TO 9:FOR b=1 TO 7:READ LED(a,b)
3192 DATA 154,154,155, 0,156,157,157
3193 DATA 0, 0,0 , 0, 0,157,157
3194 DATA 0,154,155,156,156,157, 0
3195 DATA 0, 0,155,156,156,157,157
3196 DATA 154,0 ,0 ,156,0 ,157,157
3197 DATA 154,0 ,155,156,156,0 ,157
3198 DATA 154,154,155,156,156,0 ,157
3199 DATA 0, 0,155, 0, 0,157,157
3200 DATA 154,154,155,156,156,157,157
3201 DATA 154, 0,155,156,156,157,157
3202 END DEFINE
```



QBITS FontEditSE Code

This Program requires the following files to be available **QLFontA_fnt** & **QLFontB_fnt**, these are the two default QL Font Character Groups. **QBITSBoot_bas** sets the default device and configures **QBITSConfig** used to set range of available devices.

Aliens_fnt, **Dino_fnt**, **Giro_fnt** contain Character Bitmaps used by the Program in Demo ARCADE Games.

QQLABEL_fnt, **QLMUSIC_fnt**, **QLTIMER_fnt** are Font files used by **QBITS_FONT_Info_bass** Program.

1000 REMark **QBITS_FontEditSE_bas** [QBITS Font Editor SE Review 2023 - QPC2]

1002 dev\$='win1_' :MODE 4:gx=0:gy=0:CHAR_USE 0,0 :REMark Basic Settings

1004 **WHEN ERROR** :eck=1:**CONTINUE:END WHEN**

1006 REMark **Import QBITSConfig Settings - QPC2**

1007 OPEN _IN#9,dev\$&'QBITSConfig':INPUT#9,gx\gy\dn\$\dev\$\dn%\dm%

1008 DIM drv\$(15,16):FOR d=0 TO dm%:INPUT#9,Drv\$(d):END FOR d:CLOSE#9

1010 REMark **Set Font Arrays**

1011 DIM Fnt\$(9,8),Tmp\$(9,8), File\$(50,20):fm%=50 **Note: fm% (size may require extra memory).**

1013 REMark **RAM Allocation**

1014 FBaseA=ALCHP(1164):FBaseB=ALCHP(1164):FBase1=ALCHP(876):FBase2=ALCHP(588)

1016 REMark **QBITS Special Edition – Showcases Three ARCADE Games**

1018 Init_win:**ARCADE:FontReset 0:FontMain**

1020 **DEFine PROCedure Init_win**

1021 WINDOW#0,512, 32,gx,gy+224 :PAPER#0,0 :CSIZE#0,0,0:BORDER#0,1,3:CLS#0

1022 WINDOW#1,512,224,gx,gy :PAPER#1,0 :CSIZE#1,0,0:BORDER#1,1,3:CLS#1

1023 WINDOW#2,512,224,gx,gy :PAPER#2,0 :CSIZE#2,0,0:BORDER#2,1,3:CLS#2

1024 OPEN#3,scr_:WINDOW#3,276,178,gx+ 18,gy+42 :CSIZE#3,3,0:INK#3,4

1025 OPEN#4,scr_:WINDOW#4,192,146,gx+308,gy+56:CSIZE#4,3,0:INK#4,4

1026 CSIZE#2,2,1:OVER#2,1

1027 INK#2,2:FOR i=0 TO 1:CUSOR#2,6+i,6:PRINT#2,'QBITS QLFont Editor2'

1028 INK#2,6:FOR i=0 TO 1:CUSOR#2,8+i,4:PRINT#2,'QBITS QLFont Editor2'

1029 CSIZE#2,0,0:OVER#2,0:INK#1,7

1030 CUSOR#1,280,14:PRINT#1,'(D)DIR (L)oad (S)ave (R)eset (E)xit':OVER#1,1

1031 CUSOR#1,281,14:PRINT#1,' D L S R E ':OVER#1,0

1032 CUSOR#1, 4,28:PRINT#1,'Grp Select ← ↑ ↓ → INK(2-7)': **RESTORE 1034**

1033 BLOCK#1,12,3,86,32,7: FOR i=1 TO 5:**READ fy,ch\$**:CUSOR 6,fy:PRINT ch\$

1034 DATA 42,'1',64,'2',130,'3',152,'&',174,'4'

1035 **END DEFine**




```

1037 DEFine PROCEDURE FontMain
1038 cx=0:cy=2:x=1:y=1:FontReset 0:chk=0:eck=0
1039 REPEAT Main_lp
1040   cn=cx+(cy)*16:BLOCK 280,10,226,28,0:BLOCK 200,12,300,40,0:INK 7
1041   FontChar cx,cy,7:IF eck=0:KeyPad : REMark FontPeek:FontSize
1042   K=CODE(INKEY$(-1))           Note: REMark out for BBQL and low speed platforms
1043   FontChar cx,cy,0              : REMark IF K<>32:FontGrid
1044   SELECT ON K
1045     =192:cx=cx-1:IF cx< 0:cx=15:cy=cy-1:IF cy< 0:cy= 0:cx=0
1046     =200:cx=cx+1:IF cx>15:cx= 0:cy=cy+1:IF cy>15:cy=15:cx=15
1047     =208:cy=cy-1:IF cy< 1:cy= 0
1048     =216:cy=cy+1:IF cy>15:cy=15
1049     = 32:FontPeek:FontMod      :REMark Modify Char
1050     =50 to 56:fc=K-48:FontSets  :REMark (2-7) Chart INK colour
1051     =100, 68:FontDIR            :REMark (D)DIR Set Drive/SubDIR
1052     =103, 71:FontGame          :REMark (G)ames
1053     =108, 76:FontLoad          :REMark (L)oad
1054     =115, 83:FontSave          :REMark (S)ave
1055     =114, 82:FontReset 1       :REMark (R)eset
1056     =113, 81:FontExit          :REMark (Q)uit
1057   END SELECT
1058 END REPEAT Main_lp
1059 END DEFine

```

ⓓ DIR ⓓ Load ⓓ Save ⓓ Reset ⓓ Quit

```

1061 DEFine PROCEDURE FontReset(chk)
1062 IF ch=1:CURSOR#1,412,28:PRINT#1,"Y/N":PAUSE:IF KEYROW(5)<>64:RETURN
1063 LBYTES Dev$&QLFontA_fnt',FBaseA:cn1$=QLFontA_fnt':cn2$=QLFont1_fnt'
1064 LBYTES Dev$&QLFontB_fnt',FBaseB:cn4$=QLFontB_fnt':cn3$=QLFont2_fnt'
1065 POKE FBaseA,0,127:POKE FBaseB,127,127
1066 POKE FBase1,32,96:POKE FBase2,127,64
1067 fcol=4:col=7:bcol=0:FontSets:FontGrid:cn=32
1068 END DEFine

```

```

1070 DEFine PROCEDURE FontSets
1071 CHAR_USE#3,FBaseA,FBaseB:CHAR_USE#4,FBaseA,FBaseB
1072 fx=2:fy=1:CSIZE#3,3,0:INK#3,4:CLS#3:CLS#4:INK#1,7
1073 FOR c=0 TO 255
1074   CURSOR#3,fx,fy:PRINT#3,CHR$(c):fx=fx+17:IF fx>260:fx=2:fy=fy+11
1075 END FOR c
1076 END DEFine

```



```

1078 DEFine PROCEDURE FontChar(cx,cy,ci)
1079 INK#3,4:CURSOR#3,2+cx*17,1+cy*11:PRINT#3,CHR$(cn)
1080 BLOCK#3,20,1,cx*17,cy*11,ci:BLOCK#3,20,1,cx*17,12+cy*11,ci
1081 BLOCK#3,1,12,cx*17,cy*11,ci:BLOCK#3,1,12,19+cx*17,cy*11,ci
1082 END DEFine

```

Note: Highlights a Character within the Font Chart



```

1084 DEFine PROCEDURE FontExit
1085 CURSOR#1,460,28:PRINT#1,"Y/N":PAUSE:IF KEYROW(5)<>64:RETURN
1086 RECHP FBaseA : RECHP FBase1 : RECHP FBaseB : RECHP FBase2
1087 CLOSE#4:CLOSE#3:CLS#1:CLS#0:: LRUN dn$ :STOP
1088 END DEFine

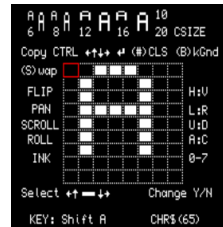
```

Note: Closes open channels and releases heap memory. STOP can be replaced with an LRUN command to return to another program such as LRUN flip1_Boot or win1_Progs_Menu etc.


```

1100 DEFINE PROCEDURE FontMod
1101 CURSOR#1,312, 82:PRINT#1,'Font Copy < > ← (#)CLS (B)kGnd'
1102 CURSOR#1,312,188:PRINT#1,'Select ← ↑ ↓ → Change Y/N'
1103 BLOCK 2,4,402,84,7:BLOCK 12,3,370,192,7
1104 RESTORE 1105:FOR i=1 TO 10:READ a,b,c$:CURSOR a,b:PRINT c$
1105 DATA 318,106,'FLIP',322,126,'PAN',312,137,'SCROLL',318,148,'ROLL'
1106 DATA 470,106,'H:V',470,126,'L:R',470,137,'U:D',470,148,'A:C'
1107 DATA 322,166,'INK',470,166,'0-7'
1108 FontChar cx,cy,7:cxo=cx:cyo=cy:FontSize:KeyPad:bx=1:by=1:rm=9:cs=8:co=cn
1109 REPEAT Chg_lp
1110 cn=cn+(cy)*16:FontChar cx,cy,2:FontChar cxo,cyo,7
1111 FontBit bx,by,7:K=CODE(INKEY$(-1)):FontBit bx,by,248:Fontchar cx,cy,0
1112 SELECT ON K
1113 =192:bx=bx-1:IF bx<1:bx=1
1114 =200:bx=bx+1:IF bx>8:bx=8
1115 =208:by=by-1:IF by<1:by=1
1116 =216:by=by+1:IF by>9:by=9
1117 =196:cx=cx-1:IF cx< 0:cx=15:cy=cy-1:IF cy< 0:cy= 0:cx=0
1118 =204:cx=cx+1:IF cx>15:cx= 0:cy=cy+1:IF cy>15:cy=15:cx=15
1119 =212:cy=cy-1:IF cy< 1:cy= 0
1120 =220:cy=cy+1:IF cy>15:cy=15
1121 =48 TO 55:col=K-48:FOR r=0 TO 8:FontDraw :REMark Colour change
1122 = 32:BitSwap bx,by :REMark Bit Swap 0<>1
1123 = 10:FontPeek:FontDraw :REMark Change Font Pattern
1124 = 35:FontNew :REMark (#) Reset Grid
1125 = 66,98:FontBkGnd :REMark (B)kGnd Colour
1126 =104,72:FontFlip 9,-1,0,1 :REMark (H)orizontal Flip
1127 =118,86:FontFlip 0,1,10,-1 :REMark (V)ertical Flip
1128 =108,76:FontSlid 0,8,1,0,1 :REMark (L)eft PAN Grid
1129 =114,82:FontSlid 0,1,8,1,0 :REMark (R)ight PAN Grid
1130 =117,85:FontSlid 1,9,1,1 :REMark (U)p SCROLL Grid
1131 =100,86:FontSlid 1,1,9,-1 :REMark (D)n SCROLL Grid
1132 = 97,65:FontRoll 1 :REMark (A) Roll Anti-Clockwise
1133 = 99,67:FontRoll 2 :REMark (C) Roll Clockwise
1134 =110,78:EXIT Chg_lp :REMark (N)o Change Return
1135 =121,89:cn=co:FontPoke:EXIT Chg_lp :REMark (Y)es Change Font
1136 END SELECT
1137 END REPEAT Chg_lp
1138 cn=co:CLS#4:FontGrid
1139 END DEFINE

```



```

1141 DEFINE PROCEDURE FontGrid
1142 BLOCK#4,112,90,46,40,bcol
1143 FOR i=0 TO 8:BLOCK#4,1,91,40+i*14,40,248
1144 FOR i=0 TO 9:BLOCK#4,112,1,40,40+i*10,248
1145 END DEFINE

```

```

1147 DEFINE PROCEDURE FontBit(x,y,hcol)
1148 BLOCK#4,15,1,32+bx*14,30+by*10,hcol:BLOCK#4,15,1,32+bx*14,40+by*10,hcol
1149 BLOCK#4,1,10,32+bx*14,30+by*10,hcol:BLOCK#4,1,10,46+bx*14,30+by*10,hcol
1150 END DEFINE

```

Note: Highlights a Bit Square within the Font BitMap

```

1152 DEFINE PROCEDURE FontBkGnd
1153 IF bcol=0:bcol=7:col=0:ELSE bcol=0:col=7
1154 FontGrid:FOR r=0 TO 8:FontDraw
1155 END DEFINE

```

Note: Toggle Black/White STRIP & White/Black INK

```

1157 DEFINE PROCEDURE FontNew
1158 FOR r=0 TO 8:Fnt$(r+1)='00000000':FontDraw
1159 END DEFINE

```

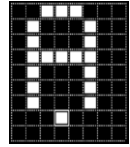
Note: Set all Matrix Bits to Zeros

```

1161 DEFINE PROCEDURE FontDraw
1162 FOR c=0 TO 7
1163 IF Fnt$(r+1,c+1)='1':pcol=col:ELSE pcol=bccl
1164 BLOCK#4,11,7,42+c*14,42+r*10,pcol
1165 END FOR c
1166 END DEFINE

```

Note: Print Pixel colour



```

1168 DEFINE PROCEDURE BitSwap(bx,by)
1169 IF Fnt$(by,bx)='0':Fnt$(by,bx)='1':ELSE Fnt$(by,bx)='0'
1170 IF Fnt$(by,bx)='0':BLOCK#4,11,7,42+(bx-1)*14,42+(by-1)*10,bcol
1171 IF Fnt$(by,bx)='1':BLOCK#4,11,7,42+(bx-1)*14,42+(by-1)*10,col
1172 END DEFINE

```

Note: STRIP

Note: INK

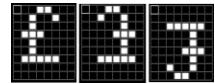


```

1174 DEFINE PROCEDURE FontFlip(cf,cz,rf,rz)
1175 FOR r=1 TO 9:Tmp$(r)=Fnt$(r)
1176 FOR r=1 TO 9:FOR c=1 TO 8:Fnt$(r,c)=Tmp$(rf+r*rz,cf+c*cz):END FOR c:END FOR r
1177 FOR r=0 TO 8:FontDraw
1178 END DEFINE

```

Note: Flip Horizontal or Vertical



```

1180 DEFINE PROCEDURE FontSlid(md,a,b,d,e)
1181 FOR r=1 TO 9:Tmp$(r)=Fnt$(r):rm=9:IF md=1 AND a=9:rm=8
1182 FOR r=1 TO rm
1183 IF md=0:Fnt$(r,a)=Tmp$(r,b):FOR c=1 TO 7:Fnt$(r,c+d)=Tmp$(r,c+e):END FOR c
1184 IF md=1:Fnt$(a)=Tmp$(b) :FOR c=1 TO 8:Fnt$(r,c)=Tmp$(r+d,c) :END FOR c
1185 END FOR r
1186 FOR r=0 TO 8:FontDraw
1187 END DEFINE

```

Note: Slide Left Right



Note: Slide Up Down



```

1189 DEFINE PROCEDURE FontRoll(rd)
1190 FOR r=1 TO 9:Tmp$(r)=Fnt$(r)
1191 FOR r=1 TO 8
1192 IF rd=1:rx=r :ry=8:FOR c=1 TO 8:Fnt$(ry,rx)=Tmp$(r,c):ry=ry-1:END FOR c
1193 IF rd=2:rx=9-r:ry=1:FOR c=1 TO 8:Fnt$(ry,rx)=Tmp$(r,c):ry=ry+1:END FOR c
1194 END FOR r
1195 FOR r=0 TO 8:FontDraw
1196 END DEFINE

```

Note: Rotate 90° Anticlockwise



Note: Rotate 90° Clockwise



```

1198 DEFINE PROCEDURE FontSize
1199 CURSOR 318,67:PRINT '6 8 12 16 20 CSIZE'
1200 CURSOR 438,54:PRINT '10':RESTORE 1199:STRIP#4,bcol:INK#4,col
1201 FOR i=1 TO 8:READ a,b,c,d:CSIZE#4,a,b:CURSOR#4,c,d:PRINT#4,CHR$(cn)
1202 DATA 0,0,12,0,0,1,22,0,1,0,34,0,1,1,46,0,2,0,60,0,2,1,78,0,3,0,94,0,3,1,1114,0
1203 END DEFINE

```



```

1205 DEFINE PROCEDURE FontPeek
1206 IF cn<128:addr=FBASEA+2+cn*9
1207 IF cn>127:addr=FBASEB+2+(cn-127)*9
1208 FOR r=0 TO 8:Fnt$(r+1)=BIN$(PEEK(addr+r),8):FontDraw
1209 END DEFINE

```

Note: Read Font Bit Pattern from RAM

1211 **DEFine PROCEDURE FontPoke**

Note: Write New Font Bit Pattern to RAM

1212 IF cn<128:addr=FBASEA+2*cn*9:ELSE addr=FBASEB+2*(cn-127)*9

1213 FOR r=0 TO 8:POKE(addr+r),BIN(Fnt\$(r+1))

1214 **END DEFine**

```
ⓂDIR Ⓛoad Ⓢave Ⓡeset Ⓚuit
LBYTES dos1_0LFontA_fnt
Select ↑14 ↓ Y/N
```

1250 **DEFine PROCEDURE FontLoad**

1251 chk=0:eck=0:Lchk=0:sf%=1:ft%=0:FOR i=1 TO fm%:File\$(i)="-

1252 INK 7:CURSOR 300,42:PRINT 'Select ↑ ↓ Y/N':INK 5:SelDrv 1:FntList 1

1253 SelFont 1,'LBYTES ':IF Lchk=0:RETurn :ELSE CURSOR 226,28:CLS 4

1254 CURSOR 310,28:PRINT#1,'Loading...':CURSO 300,42:CLS 4:PAUSE 30

1255 OPEN _IN#9,drv\$(dn%)&File\$(sf%)

1256 sg%=CODE(INKEY\$(#9)):cg%=CODE(INKEY\$(#9))

1257 CLOSE#9

```
ⓂDIR Ⓛoad Ⓢave Ⓡeset Ⓚuit
Loading...
```

1258 IF sg%< 31 AND cg%> 96:addr1=FBASEA:cy= 0:cn1\$=File\$(sf%):cn2\$=cn1\$:cn1=0

1259 IF sg%>=31 AND sg%<127:addr1=FBASE1:cy= 2:cn2\$=File\$(sf%):addr2=FBASEA:cn1=96

1260 IF sg%>126 AND cg%< 65:addr1=FBASE2:cy= 8:cn3\$=File\$(sf%):addr2=FBASEB:cn1=64

1261 IF sg%>126 AND cg%> 64:addr1=FBASEB:cy=12:cn4\$=File\$(sf%):cn3\$=cn4\$:cn1=0

1262 LBYTES drv\$(dn%)&File\$(sf%),addr1:cx=0:IF cn1=96:os=sg%:ELSE os=sg%-127

1263 IF cn1=96 OR cn1=64

1264 FOR a=1 TO cn1

1265 FOR b=0 TO 8:POKE addr2+2+(a+os)*9+b,PEEK(addr1+2+a*9+b)

1266 END FOR a

1267 END IF

1268 FontSets:eck=0:INK 7:FontGrid

1269 **END DEFine**

1271 **DEFine PROCEDURE FontSave**

1272 chk=0:eck=0:Lchk=0:sf%=1:ft%=4

1273 File\$(1)=cn1\$:File\$(2)=cn2\$:File\$(3)=cn3\$:File\$(4)=cn4\$

1274 INK 7:CURSOR 300,42:PRINT 'Select ↑ ↓ Y/N (E)dit ← → ←':

1275 BLOCK 12,3,454,46,7:BLOCK 2,4,480,44,7:INK 5:SelDrv 2

1276 SelFont 2,'SBYTES ':IF Lchk=0:RETurn

1277 FntList 2:BLOCK 200,10,300,42,0 :CURSOR 312,42:INK 7

1278 IF eck=1:PRINT#1,'DEVICE ERROR...':PAUSE 50:RETurn

1279 IF chk=1:PRINT#1,'Overwrite Y/N':PAUSE:IF KEYROW(5)<>64:RETurn

1280 INK 5:DELETE drv\$(dn%)&File\$(sf%):CURSOR 226,28:CLS 4

1281 CURSOR 310,28:PRINT#1,'Saving...':CURSOR 300,42:CLS 4:PAUSE 30

1282 IF sf%=1:addr1=FBASEA:lgth=1154:cn1=127

1283 IF sf%=2:addr1=FBASE1:lgth=876 :cn1= 96:addr2=FBASEA:os=31

1284 IF sf%=3:addr1=FBASE2:lgth=588 :cn1= 64:addr2=FBASEB:os= 0

1285 IF sf%=4:addr1=FBASEB:lgth=1154:cn1=127

1286 IF cn1=96 OR cn1=64

1287 FOR a=1 TO cn1

1288 FOR b=0 TO 8:POKE addr1+2+a*9+b,PEEK(addr2+2+(a+os)*9+b)

1289 END FOR a

1290 END IF

1291 SBYTES drv\$(dn%)&File\$(sf%),addr1,lgth

1292 **END DEFine**

```
ⓂDIR Ⓛoad Ⓢave Ⓡeset Ⓚuit
SBYTES f\p1_0LFontA_fnt
Select ↑01 ↓ Y/N (E)dit ← → ←
```

```
ⓂDIR Ⓛoad Ⓢave Ⓡeset Ⓚuit
Checking...
DEVICE ERROR...
```

```
ⓂDIR Ⓛoad Ⓢave Ⓡeset Ⓚuit
Checking...
Overwrite Y/N
```

```
ⓂDIR Ⓛoad Ⓢave Ⓡeset Ⓚuit
Saving...
```

1294 **DEFine PROCEDURE FontDIR**

1295 CURSOR 226,28:PRINT 'Set Drive & SuDIR'

1296 INK 7:CURSOR 300,42:PRINT 'Select↑ ↑ Y/N (E)dit ← → ←'

1297 BLOCK 12,3,454,46,7:BLOCK 2,4,480,44,7:INK 5:SelDrv 3

1298 **END DEFine**

```
ⓂDIR Ⓛoad Ⓢave Ⓡeset Ⓚuit
Set Drive & SuDIR uini_Fonts_
Select ↑09 ↓ Y/N (E)dit ← → ←
```

```

1300 DEFINE PROCEDURE SelDrv(act%)
1301 REPEAT drv_ip
1302 CURSOR 342,28:PRINT drv$(dn%):CLS 4
1303 CURSOR 351,42:PRINT FILL$('0',2-LEN(dn%))&dn%
1304 K=CODE(INKEY$(-1))
1305 SElect ON K
1306   =208:dn%=dn%-1:IF dn%<1:dn%=dm%
1307   =216:dn%=dn%+1:IF dn%>dm%:dn%=1
1308   =101,69:IF act%=3:EditName 3,342,16,drv$(dn%)
1309   =121,89,110,78:EXIT drv_ip
1310 END SElect
1311 END REPEAT drv_ip
1312 END DEFine

```

```

DIR (Load (Save (Reset (Quit
      flp1_
Select ↑ 03 ↓ Y/N

```

Yes or No Selects Device

```

1314 DEFINE PROCEDURE SelFont(act%,Act%)
1315 BLOCK 200,10,292,28,0:str$=Act$&drv$(dn%)
1316 IF ft%<1:CURSOR 310,28:PRINT 'No Files Found...':PAUSE 30:RETURN
1317 CURSOR 226,28:PRINT FILL$(' ',23-LEN(str%))&str$
1318 REPEAT File_ip
1319 CURSOR 364,28:PRINT File$(sf%):CLS 4
1320 CURSOR 351,42:PRINT FILL$('0',2-LEN(sf%))&sf%
1321 K=CODE(INKEY$(-1))
1322 SElect ON K
1323   =208:sf%=sf%-1:IF sf%<1:sf%=ft%
1324   =216:sf%=sf%+1:IF sf%>ft%:sf%=1
1325   =101,69:IF act%=2:EditName 2,364,16,File$(sf%)
1326   =110,78:Lchk=0:EXIT File_ip
1327   =121,89:Lchk=1:EXIT File_ip
1328 END SElect
1329 END REPEAT File_ip
1330 END DEFine

```

```

DIR (Load (Save (Reset (Quit
      No Files Found...
Select ↑ 03 ↓ Y/N

```

No abort action

Yes Load / Save Font File

```

1332 DEFINE PROCEDURE FntList(act%)
1333 BLOCK 280,10,226,28,0:CURSOR 310,28:PRINT 'Checking...'
1334 PAUSE 20:DELETE drv$(dn%)&'FList'
1335 OPEN_NEW#9,drv$(dn%)&'FList':DIR#9,drv$(dn%):CLOSE#9
1336 OPEN_IN#9,drv$(dn%)&'FList':dl%=LEN(drv$(dn%))
1337 REPEAT dir_ip
1338 IF act%=1
1339   IF EOF(#9) OR sf%>fm%:sf%=sf%-1:ft%=sf%:CLOSE#9:EXIT dir_ip
1340   INPUT#9,F$:F$=F$(dl%-4 TO):fl%=LEN(F$)
1341   IF fl%<=20 AND '_fnt' INSTR F$>0:File$(sf%)=F$:sf%=sf%+1
1342 END IF
1343 IF act%=2
1344   IF EOF(#9):CLOSE#9:chk=0:EXIT dir_ip
1345   INPUT#9,F$:F$=F$(dl%-4 TO)
1346   IF F$==File$(sf%):CLOSE#9:chk=1:EXIT dir_ip
1347 END IF
1348 END REPEAT dir_ip
1349 END DEFine

```

```

DIR (Load (Save (Reset (Quit
      Checking...
Select ↑ 03 ↓ Y/N

```

Note: Checking '_nft' Files to LOAD

Note: Checking If SAVE File Exists

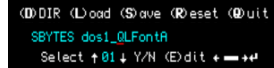
Note: Before Loading GrpChk checks Lowest Character and Number of Characters held in first & second byte of file. It then LBYTES to the relative FBase A, B, 1 or 2 memory address allocated.

Note This QBITS Editor Restricts ASCII Code Characters used for QL Filenames. Position the underscore with left/Right cursors to a character then **Add** a new or **Delete** the existing character. Adding a Character expands the string to the right, Deleting shrinks the string from right to left. The QL Delete uses CTRL Right Cursor for character above underline, the **Spacebar** also acts as a **Delete** key. To Delete character to left of underline CTRL Left Backspace still applies.

```

1351 DEFINE PROCEDURE EditName(act%,x,sm%,str$)
1352 IF act%=2:sl%=(' _fnt' INSTR str$)-1:str$=str$(1 TO sl%)
1353 temp$=str$:sl%=LEN(str$):cp%=1
1354 REPEAT Ed_lp
1355   Ln_Prn:Ln_Cur:k$=INKEY$(#0,-1):K=CODE(k$)
1356   SELECT ON K
1357     = 10:EXIT Ed_lp
1358     = 48 TO 57, 65 TO 90,95, 97 TO 122:Add_chr
1359     =194   :IF cp%>1:cp%=cp%-1:Del_chr
1360     =202,32:Del_chr
1361     =192   :IF cp%>1:cp%=cp%-1
1362     =200   :IF cp%<sl%+1:cp%=cp%+1
1363   END SELECT
1364 END REPEAT Ed_lp
1365 IF sl%=0:str$=temp$
1366 IF act%=2:str$=str$&'_fnt'
1367 IF act%=3 AND str$(sl%)<>'_' :IF sl%<sm%:str$=str$&'_' :ELSE str$(sl%)='_'
1368 IF act%=3 AND str$(5)<>'_' :str$(5)='_'
1369 END DEFINE

```

Note: Restricted ASCII Codes
Delete Character to left of cursor
Delete Character above cursor

```

1371 DEFINE PROCEDURE Ln_Prn
1372 IF LEN(str$)>sm%:str$=str$(1 TO sm%):cp%=sm%
1373 INK 5:CURSOR x,28:PRINT str$:CLS 4
1374 END DEFINE

```

Note: Prints Str\$ CLS to end of Cursor Line

```

1376 DEFINE PROCEDURE Ln_Cur
1377 BLOCK 6,1,x+cp%*6-6,37,2
1378 END DEFINE

```

Note: Character Highlight Cursor Bar

```

1380 DEFINE PROCEDURE Add_chr
1381 IF cp% = 1 AND sl% = 0 :str$=str$&k$
1382 IF cp%>=1 AND cp%<sl%:str$=str$(1 TO cp%-1)&k$&str$(cp% TO sl%)
1383 IF cp%>=1 AND cp%=sl%:str$=str$(1 TO cp%-1)&k$&str$(cp%)
1384 IF cp%> 1 AND cp%>sl%:str$=str$&k$
1385 IF cp%=sm%:str$(cp%)=k$
1386 IF sl%<sm%:sl%=sl%+1:ELSE sl%=sm%
1387 IF cp%<sm%:cp%=cp%+1:ELSE cp%=sm%
1388 END DEFINE

```

Note: Checks for Adding Character to String

```

1390 DEFINE PROCEDURE Del_chr
1391 IF cp%=sl%:str$=str$(1 TO sl%-1):sl%=sl%-1
1392 IF cp%>=1 AND cp%<sl%:str$=str$(1 TO cp%-1)&str$(cp%+1 TO sl%):sl%=sl%-1
1393 IF cp%=sm%:str$=str$(1 TO sm%-1):cp%=cp%-1:sl%=sm%-1
1394 IF cp%=1 AND sl%=1:str$="" :sl%=0
1395 END DEFINE

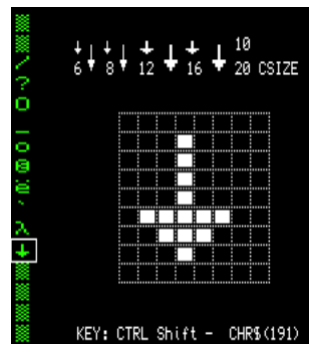
```

Note: Checks when Deleting Character from String

1400 DEFINE PROCEDURE KeyPad

```
1401 SELECT ON cn=32 TO 93 :Key$=CHR$(cn)
1402 IF cn=32 :Key$='Spacebar'
1403 SELECT ON cn=33,34 :Key$='Shift '&(cn-32)
1404 SELECT ON cn=36,37 :Key$='Shift '&(cn-32)
1405 IF cn=38 :Key$='Shift 7'
1406 IF cn=40 :Key$='Shift 9'
1407 IF cn=41 :Key$='Shift 0'
1408 IF cn=42 :Key$='Shift 8'
1409 IF cn=43 :Key$='Shift ='
1410 IF cn=58 :Key$='Shift .'
1411 IF cn=60 :Key$='Shift ,'
1412 IF cn=62 :Key$='Shift /'
1413 IF cn=63 :Key$='Shift /'
1414 IF cn=64 :Key$='Shift '"
1415 SELECT ON cn=65 TO 90 :Key$='Shift '&CHR$(cn)
1416 IF cn=94 :Key$='Shift 6'
1417 IF cn=95 :Key$='Shift -'
1418 IF cn=96 :Key$='Shift 3'
1419 SELECT ON cn=97 TO 122 :Key$=CHR$(cn-32)
1420 IF cn=123 :Key$='Shift ['
1421 IF cn=124 :Key$='Shift \'
1422 IF cn=125 :Key$='Shift ]'
1423 IF cn=126 :Key$='Shift #'
1424 IF cn=127 :Key$='Shift Esc'
1425 IF cn=128 :Key$='CTRL Esc'
1426 IF cn=129 :Key$='CTRL Shift 1'
1427 IF cn=130 :Key$='CTRL Shift '"
1428 SELECT ON cn=131 TO 133 :Key$='CTRL Shift '&CHR$(cn-80)
1429 IF cn=134 :Key$='CTRL Shift 7'
1430 IF cn=135 :Key$='CTRL '"
1431 IF cn=136 :Key$='CTRL Shift 9'
1432 IF cn=137 :Key$='CTRL Shift 0'
1433 IF cn=138 :Key$='CTRL Shift 8'
1434 IF cn=139 :Key$='CTRL Shift ='
1435 SELECT ON cn=140 TO 153 :Key$='CTRL '&CHR$(cn-96)
1436 IF cn=154 :Key$='CTRL Shift .'
1437 IF cn=155 :Key$='CTRL ,'
1438 IF cn=156 :Key$='CTRL Shift ,'
1439 IF cn=157 :Key$='CTRL ='
1440 IF cn=158 :Key$='CTRL Shift .'
1441 IF cn=159 :Key$='CTRL Shift /'
1442 IF cn=160 :Key$='CTRL Shift 2'
1443 SELECT ON cn=161 TO 186 :Key$='CTRL Shift '&CHR$(cn-96)
1444 IF cn=187 :Key$='CTRL ['
1445 IF cn=188 :Key$='CTRL \'
1446 IF cn=189 :Key$='CTRL ]'
1447 IF cn=190 :Key$='CTRL Shift 6'
1448 IF cn=191 :Key$='CTRL Shift .'
1449 SELECT ON cn=0 TO 31,192 TO 255:Key$=""
1450 CURSOR 320,208:PRINT 'KEY: '&Key$;FILL$(' ',14-LEN(Key$));CHR$(':',cn);' '
1451 END DEFINE
```

Note: Identify Key(s) for Printable Characters



1453 REMark **Demo: RETRO ARCADE GAMES - QBITS FontEdit2 SE 2022**

1455 **DEFine PROCedure ARCADE**

1456 INK#0,6:CURSOR#0,212,4::PRINT#0,"ARCADE GAMES":INK#0,5:CURSOR#0,54,16

1457 PRINT#0, '(L)oad Font File "_fnt" for Aliens : Dino : Giro then Press (G)ame'

1458 **END DEFine**

1460 **DEFine PROCedure FontGame**

Note: Load _fnt File and Press (G)ame

1461 IF cn3\$=="Aliens_fnt":CLS#3:**Space_Invader**

1462 IF cn3\$=="Dino_fnt":CLS#3:**Dino_Run**

1463 IF cn3\$=="Giro_fnt":CLS#3:**Giro_Rescue**

1464 **END DEFine**

1500 REMark **ARCADE - Space Invaders** a simplified version for QL by QBITS 2022

1502 **DEFine PROCedure Space_Invader**

1503 pd=2:REMark Game Speed pd=pause delay set (1 to 5)

1504 CSIZE#3,1,0:INK#3,6:b=0:pd=2

1505 FOR a=128 TO 142:b=b+1:CURSOR#3,b*8,8:PRINT#3,CHR\$(a) **Note: Font Grp 3 Codes 128 to 191**

1506 CURSOR 127,200:PRINT#3, ← →:BLOCK 12,3,137,204,7

1507 **InitLevel:DrawAliens** 1:!=5:INK#3,7

1508 **REPEAT Invader_lp**

1509 IF !=4 OR at%>0

1510 CSIZE#3,2,0:CURSOR#3,54,80:PRINT#3,'New Game Y/N'

1511 **REPEAT ans**

1512 IF KEYROW(5)=64:z%=0:!=1:at%=27:tnum=0:num=0:CLS#3,3:**EXIT ans**

1513 IF KEYROW(7)=64:**EXIT Invader_lp**

1514 **END REPEAT ans**

1515 **END IF**

1516 **DrawAliens** !=:INK#3,7:CURSOR#3,74,80:PRINT#3,'LEVEL '!=

Note != level

1517 FOR z=6 TO 0 STEP -1:CURSOR#3,124,96:PRINT#3,z:PAUSE 25

1518 BLOCK#3,120,30,72,80,0:i=8:n=5:sl%=0:ay=3:**Invaders**

1519 **END REPEAT Invader_lp**

1520 CLS#3:**FontSets**

1521 **END DEFine**

1523 **DEFine PROCedure InitLevel**

1524 DIM GL%(4,3),GA%(3,9):**RESTORE 1526**

1525 FOR a=1 TO 4:**READ GL%(a,1),GL%(a,2),GL%(a,3)**

1526 DATA 143,144,145,146,147,148,149,150,151,152,153,154

1527 **END DEFine**

Note: 4 Levels of Play



1529 **DEFine PROCedure DrawAliens**(!=)

Note: Display Aliens by Level

1530 BLOCK#3,260,10,6,150,0:at%=27:**GScore 0**

1531 FOR a=1 TO 3:FOR b=1 TO 9:GA%(a,b)=GL%(!=,a):END FOR b:END FOR a

1532 INK#3,5:FOR i=1 TO 9:CURSOR#3,i*24,28:PRINT#3,CHR\$(GA%(1,i))

1533 INK#3,4:FOR i=1 TO 9:CURSOR#3,i*24,40:PRINT#3,CHR\$(GA%(2,i))

1534 INK#3,3:FOR i=1 TO 9:CURSOR#3,i*24,52:PRINT#3,CHR\$(GA%(3,i))

1535 **END DEFine**



1537 **DEFine PROCedure SLives**

1538 CSIZE#3,1,0:INK#3,7

1539 CURSOR#3,148,8:PRINT#3,FILL\$(CHR\$(159),4-sl%)&'

1540 CSIZE#3,3,0:sl%=sl%+1

1541 **END DEFine**

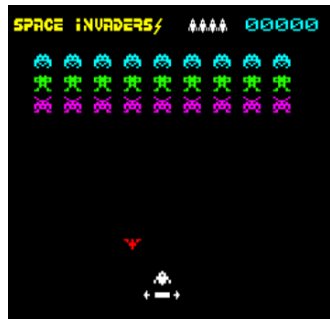


Note: Ship Lives/Level


```

1543 DEFine PROCEDURE Invaders
1544 sl%=0:n=5:at%=27:SLives
1545 REPEAT lp
1546 i=i+1:IF i>7:i=1:an=n+RND(-1 TO 1)
1547 DShip 7,n:x=an*24:y=60+i*12
1548 IF at%=0:l%=l%+1:RETurn
1549 IF at%>0 AND sl%>4:l%=5:RETurn
1550 IF KEYROW(1)=64:FShip:ay=3:GScore l%*50
1551 IF KEYROW(1)= 2:DShip 0,n:n=n-1:IF n<1:n=1
1552 IF KEYROW(1)=16:DShip 0,n:n=n+1:IF n>9:n=9
1553 INK#3,2:CUSOR#3,x,y:PRINT#3,CHR$(158):PAUSE pd
1554 INK#3,2:CUSOR#3,x,y:PRINT#3,'':PAUSE pd
1555 IF an=n AND KEYROW(1)=64
1556 BEEP 2000,1,255,200,4,2,0,0,0
1557 INK#3,7:CUSOR#3,x,140:PRINT#3,CHR$(160):PAUSE pd
1558 CLS#3,3:FExplode x,y:GScore 200:i=8
1559 END IF
1560 IF an=n AND i=7
1561 BEEP 3000,20,80,80,-8,15,15,i=8:SLives
1562 CURSOR#3,x,150:PRINT#3,CHR$(162):PAUSE pd*5
1563 END IF
1564 END REPEAT lp
1565 END DEFine

```



Note: Falling Bombs Chr\$(158)

```

1567 DEFine PROCEDURE GScore(num)
1568 CURSOR#3,192,8:CSIZE#3,2,0:INK#3,5:tnum=tnum+num
1569 PRINT#3,FILL$(0',5-LEN(tnum))&tnum:CSIZE#3,3,0
1570 END DEFine

```



```

1572 DEFine PROCEDURE DShip(col,n)
1573 INK#3,col:CUSOR#3,n*24,150:PRINT#3,CHR$(159)
1574 END DEFine

```



Note: Defender Ship

```

1576 DEFine PROCEDURE FShip
1577 BEEP 2000,1,255,200,4,2,0,0,0
1578 FOR i=1 TO 5
1579 OVER#3,1:CUSOR#3,n*24,150-i*18:PRINT#3,CHR$(160):PAUSE pd
1580 OVER#3,0:CUSOR#3,n*24,150-i*18:PRINT#3,' '
1581 END FOR i
1582 IF ay=3 AND GA%(ay,n)=32:ay=2
1583 IF ay=2 AND GA%(ay,n)=32:ay=1
1584 IF ay=1 AND GA%(ay,n)=32:ay=3:RETurn
1585 FExplode n*24,16+ay*12:GA%(ay,n)=32:at%=at%-1
1586 END DEFine

```

Note: Fire at Enemy



Note: Check Array Entry

Note: Explode Alien and Delete



```

1588 DEFine PROCEDURE FExplode(x,y)
1589 FOR i=1 TO 6
1590 BEEP 3000,20,60,80,-8,15,15,15
1591 INK#3,2:CUSOR#3,x,y:PRINT#3,CHR$(160):PAUSE pd
1592 INK#3,6:CUSOR#3,x,y:PRINT#3,CHR$(161):PAUSE pd
1593 END FOR i
1594 CURSOR#3,x,y:PRINT#3,' '
1595 END DEFine

```



Note: Load `Dino_fnt` and Press (G)ame

DINO Run is no doubt-based on the 2014 Google Chrome Dino Game a recent addition to horizontal moving genre of ARCADE style games. It possibly links back to the Retro 1980's BC Quest for Tires, where Caveman Thor had to avoid various hazards by jumping over pits or ducking tree branches etc.

In this version DINO has to simply jump over Cacti to survive. With suspected CPU timing issue across the various QL Platforms a Speed Adjuster has been added. The timings can be set from '0' to '50000'. Use +/- and/or simply Enter at start of each Game.



1602 DEFine PROCEDURE Dino_Run

```

1603 dino_colour%      = 7
1604 cactus_colour%    = 4
1605 ground_colour%    = 2
1606 cloud_colour%     = 255
1607 background_col    = 0
1608 slowdown_steps    = 25000 :REMark 1/n Game Delay [?Dependant on Processor Speed]
1609 pany%              = 90 :REMark Top Left of Main PAN Block
1610 byPerCent          = 1 :REMark Percentage change of Slowdown delays
1611 every%             = 1000 :REMark every this much of score
1612 sound_on%         = 1 :REMark 0=sound off 1=sound on
1613 demo_mode%        = 0 :REMark auto-jumping
1614 hiscore=score     = 0 :dy%=18:sl=5000
1615 :
1616 DIM backg$(2,34) :REMark Cactus Height 0-2 Locations 0-34 of ground steps
1617 :
1618 REPEAT Dino_program
1619 REMark *** Title & Score bar ***
1620 PAPER#3,background_col:CLS#3:BLOCK#3,276,22,0,0,80
1621 OVER#3,1:CSIZE#3,1,1:STRIP#3,80:INK#3,7
1622 FOR i=0 TO 1:CUSOR#3,8+i,2:PRINT#3,' DINO Run'
1623 CUSOR#3,100,2:PRINT#3,'HI '&FILL$('0',5-LEN(hiscore))&hiscore;
1624 OVER#3,0:CSIZE#3,0,0:DSpeed:STRIP#3,0
1625 :
1626 REMark *** Set Ground level at pany%+50 [ie. (3*18)-4] ***
1627 RANDOMISE:CSIZE#3,1,0:INK#3,ground_colour%
1628 CUSOR#3,0,pany%+50:FOR a=1 TO 34:PRINT#3,CHR$(RND(168 TO 178));
1629 :
1630 REMark *** Insert a couple of Random Cactii ***
1631 REMark 0=cactus 3 (top) 1=cactus 2 (middle) 2=cactus 1 (bottom)
1632 CHAR_INC#3,8,18:FOR y=0 TO 2:backg$(y)=FILL$(' ',34)
1633 backg$(2,RND(18 TO 22))=CHR$(132):backg$(2,RND(32 TO 34))=CHR$(132)
1634 OVER#3,1:CSIZE#3,1,1:INK#3,cactus_colour%
1635 FOR a=0 TO 2:CUSOR#3,0,pany%+(18*a):PRINT#3,backg$(a);
1636 OVER#3,0:CSIZE#3,1,0

```

```

1638 REMark *** Place Random Cloud ***
1639 STRIP#3,background_col :INK#3,ground_colour%:cloudy%=48
1640 rn=RND(1 TO 5) :REMark avoids Turbo error
1641 SElect ON rn
1642 =1 : cloud_colour% = 7
1643 =2 : cloud_colour% = 56
1644 =3 : cloud_colour% = 63
1645 =4 : cloud_colour% = 248
1646 =5 : cloud_colour% = 255
1647 END SElect
1648 CSIZE#3,1,0:INK#3,cloud_colour%:CURSOR#3,8*RND(10 TO 30),cloudy%
1649 IF RND(1 TO 2)=1
1650 PRINT#3,CHR$(181)&CHR$(182);
1651 ELSE
1652 PRINT#3,CHR$(183)&CHR$(184);
1653 END IF
1654 CSIZE#3,1,1:INK#3,7
1655 cloud_gap%=RND(10 TO 20):cloud_wide=0:cloud_run%=0
1656 slowdown=slowdown_steps :REMark disable [Set to 0 for BBQL]
1657 :
1658 cactus_gap%=RND(10 TO 20) :REMark columns blank before a cactus
1659 cactus_wide =0 :REMark 1/2/3
1660 cactus_high =0 :REMark 1/2/3
1661 cactus_run% =0 :REMark Cactus width (1/2/3) drawn
1662 dinobasey% =pany%+36 :REMark Dinosaur 'ground level'
1663 dinoy% =dinobasey% :REMark Dino moves up from here
1664 score =0 :REMark Score is one point per step
1665 cloudy% =RND(36 TO 48) :REMark Location above cactus
1666 counter% =0 :REMark Dino Animation Frame Counter
1667 ticker =0 :REMark Speed Control Delay
1668 jumping% =0 :REMark <>0 Dino jumping (+ve=up -ve=down)
1669 demo_mode =0 :REMark Demo mode OFF=0 ON=1
1670 :
1671 REPeat Dino_GAME
1672 OVER#3,0:CURSOR#3,6,158:CSIZE#3,1,1
1673 IF demo_mode%=1:CLS#3,3:PRINT#3,'Demo Mode On'
1674 IF demo_mode%=0:PRINT#3,'(D)emo Mode (P)ause (Q)uit Game'
1675 INK#3,dino_colour%:OVER#3,-1 :REMark Show Dino
1676 CURSOR#3,32,dinoy%:PRINT#3,CHR$(128+(counter% MOD 4));
1677 :
1678 REMark *** Speed control - ignore (slowdown-1) passes of the loop ***
1679 IF slowdown>0
1680 REPeat wait
1681 ticker=ticker+1:IF ticker<slowdown:NEXT wait
1682 ticker=0:EXIT wait
1683 END REPeat wait
1684 END IF

```

Demo Mode On

(D)emo Mode (P)ause (E)xit Game

<pre> 1685 key=CODE(INKEY\$) 1686 SElect ON key 1687 =81,113:CURSOR#3,40,158:CLS#3,3:EXIT Dino_GAME :REMark (Q)uit 1688 =10,32 :REMark Manual Jump 1689 IF jumping%=0 AND demo_mode%=0 1690 jumping%= -1:dinoy%=dinobasey%:dy%=18 1691 IF sound_on%:BEEP 100,100,0,0,0,0,0 1692 END IF 1693 =80,112 :REMark (P)ause 1694 OVER#3,0:CURSOR#3,6,158:CLS#3,3:PRINT#3,'PAUSED':CLS#3,4 1695 k\$=INKEY\$(-1):CURSOR#3,40,158:CLS#3,3:IF k\$=='Q':EXIT Dino_GAME 1696 =68,100 :REMark (D)emo_mode (auto-jumping) 1697 demo_mode%=NOT demo_mode% :dy%=18 1698 OVER#3,0:CURSOR#3,6,158:CSIZE#3,1,1:INK#3,7 1699 END SElect 1700 OVER#3,-1:INK#3,dino_colour% :REMark Erase Dino before PAN'ing 1701 CURSOR#3,32,dinoy% :PRINT#3,CHR\$(128+(counter% MOD 4)) 1702 OVER#3,0:y%=(dinobasey%-dinoy%) DIV 18 :REMark Detect Obstacle 1703 IF y%<3 1704 IF backg\$(2-y%,5)<>' ' 1705 IF sound_on% : BEEP 5000,0,50,50,1,0,0 1706 OVER#3,-1:INK#3,dino_colour% 1707 CURSOR#3,32,dinoy%:PRINT#3,CHR\$(128+(counter% MOD 4)); 1708 FOR a=1 TO 10:BLOCK#3,10,18,32,dinoy%,7:PAUSE 5 1709 OVER#3,0:CURSOR#3,12,158:CLS#3,3 1710 m = RND(1 TO 3) 1711 SElect ON m 1712 =1 : PRINT#3,'Ouch! ' ; 1713 =2 : PRINT#3,'Oops! ' ; 1714 =3 : PRINT#3,'Oh dear! ' ; 1715 END SElect 1716 EXIT Dino_GAME 1717 END IF 1718 END IF 1719 IF jumping%<>0 1720 IF jumping%=-1 1721 dinoy%=dinobasey%-72*(SIN(RAD(dy%))) 1722 dy%=dy%+18:IF dy%=90:jumping%=1 :REMark change direction 1723 ELSE 1724 dinoy%=dinobasey%-72*(SIN(RAD(dy%))) :REMark Upward 1725 dy%=dy%-18:IF dy%=0:jumping%=0:dinoy%=dinobasey% 1726 END IF 1727 ELSE 1728 IF demo_mode%=1 1729 IF backg\$(2,10)<>' ' 1730 jumping%=-1:dinoy%=dinobasey%:dy%=18 1731 IF sound_on% : BEEP 100,10,0,0,0,0,0 1732 END IF 1733 END IF 1734 END IF </pre>	<div style="display: flex; align-items: center;"> <div style="font-size: 4em; margin-right: 10px;">}</div> <div> M E N U C H O I S E </div> </div> <div style="display: flex; align-items: center; margin-top: 10px;"> <div style="font-size: 4em; margin-right: 10px;">}</div> <div> D E T E C T O B S T A C L E </div> </div> <div style="display: flex; align-items: center; margin-top: 10px;"> <div style="font-size: 4em; margin-right: 10px;">}</div> <div> D I N O J U M P </div> </div>
--	---



```

1735 REMark *** PAN Cactus Display to Left - Height 4x18 Pixels -4(overlap)
1736 INK#3,cactus_colour%:CURSOR#3,0,pany%
1737 CHAR_INC#3,8,68:PAN#3,-8,3:CHAR_INC#3,8,18
1738 FOR y=2 TO 0 STEP -1:backg$(y)=backg$(y,2 TO 34)&' '
1739 REMark *** Add any extra Objects to right of background array
1740 IF cactus_gap%>0
1741     cactus_gap%=cactus_gap%-1 :REMark Decrement count to next cacti
1742     IF cactus_gap%=0
1743         cactus_run%=1 :REMark how far are we across a cactus?
1744         cactus_wide=RND(1 TO 3) :REMark cactus width 1-3
1745         cactus_high=RND(1 TO 3) :REMark cactus height 1-3
1746     END IF
1747 END IF
1748 IF cactus_run%>0
1749     SElect ON cactus_high
1750     =1:REMark cactus 1 high
1751         base_char =131+(cactus_wide=2)+(3*(cactus_wide=3))
1752         backg$(2,34)=CHR$(base_char+cactus_run%)
1753     =2:REMark cactus 2 medium
1754         base_char =137+(cactus_wide=2)+(3*(cactus_wide=3))
1755         backg$(1,34)= CHR$(base_char+cactus_run%)
1756         backg$(2,34)= CHR$(base_char+6+cactus_run%)
1757     =3:REMark cactus 3 low
1758         base_char =149+(cactus_wide=2)+(3*(cactus_wide=3))
1759         backg$(0,34)=CHR$(base_char+cactus_run%)
1760         backg$(1,34)=CHR$(base_char+6+cactus_run%)
1761         backg$(2,34)=CHR$(base_char+12+cactus_run%)
1762     END SElect
1763     cactus_run%=cactus_run% + 1
1764     IF cactus_run%>cactus_wide
1765         cactus_gap%=RND(10 TO 20):REMark start a new gap between cacti
1766         cactus_run%=0:cactus_high=0:cactus_wide=0 :REMark RESet
1767     END IF
1768 END IF
1769 :
1770 REMark *** Move Ground : Update Cacti
1771 OVER#3,1:CSIZE#3,1,0:INK#3,ground_colour%
1772 CURSOR#3,260,pany%+50:PRINT#3,CHR$(RND(168 TO 178));
1773 CSIZE#3,1,1:INK#3,cactus_colour%
1774 FOR y=2 TO 0 STEP -1:CURSOR#3,260,pany%+(18*y):PRINT#3,backg$(y,34);
1775 OVER#3,0
1776 :
1777 REMark *** Handle the score IF >99999 wrap around
1778 STRIP#3,80:INK#3,7:score=score+1:IF score>99999:score=0
1779 CURSOR#3,180,2:PRINT#3,FILL$(0',5-LEN(score))&score:STRIP#3,0
1780 :
1781 REMark *** Speed up Slightly as Score gets Higher (if set to do so)
1782 IF every%>0
1783     IF (score MOD every%)=0
1784         slowdown=slowdown-(slowdown*byPerCent/100)
1785     END IF
1786 END IF
1787 :

```

U
P
D
A
T
E

C
A
C
T
I

Note: UPDATE SCORE

Note: Adjust Speed

```

1788 REMark *** Increment Dino Frame Counter
1789 counter%=counter%+1:IF counter% >4:counter%=0
1790 :
1791 REMark *** Handle Clouds Once Every 4 Frames
1792 IF (counter% MOD 4)=0
1793     CURSOR#3,0,cloudy%:CHAR_INC#3,8,36:PAN#3,-8,3:CHAR_INC#3,8,18
1794     cloud_gap%=cloud_gap%-1
1795     IF cloud_gap%=0
1796         cloud_run% =1 :REMark how far across
1797         cloud_wide =RND(2 TO 3) :REMark cloud width 1-3
1798         cloud_high =RND(0 TO 1) :REMark cloud height 0-1
1799         cloudyy% =cloudy% + RND(0 TO 26-(9*cloud_high))
1800         rn=RND(1 TO 5) :REMark Cloud - Random Colour/Stipple
1801         Select ON rn
1802             =1:cloud_col%=7
1803             =2:cloud_col%=56
1804             =3:cloud_col%=63
1805             =4:cloud_col%=248
1806             =5:cloud_col%=255
1807         END SElect
1808         SElect ON cloud_wide
1809             =1:IF RND(1 TO 10)=10
1810                 cloud_base_char=191:cloud_high=0:cloud_col%=6
1811             ELSE
1812                 cloud_base_char=179+(RND>.5) :REMark small cloud
1813             END IF
1814             =2:cloud_base_char=180+(2*(RND>.5))
1815             =3:cloud_base_char=184+(3*(RND>.5))
1816         END SElect
1817     END IF
1818     IF cloud_run%>0
1819         CSIZE#3,1,cloud_high:INK#3,cloud_col%
1820         CURSOR#3,33*8,cloudy%:PRINT#3,CHR$(cloud_base_char+cloud_run%);
1821         INK#3,7:cloud_run%=cloud_run%+1
1822         IF cloud_run%>cloud_wide
1823             cloud_gap%=RND(10 TO 20):cloud_run%=0:cloud_wide=0
1824         END IF
1825     END IF
1826 END IF
1827 END REPEAT Dino_GAME
1828 CSIZE#3,1,1:PRINT#3,'< Another Game Y/N >'
1829 REPEAT Ans_Ip
1830     IF KEYROW(7)=64:EXIT Dino_program
1831     IF KEYROW(5)=64:EXIT Ans_Ip
1832 END REPEAT Ans_Ip
1833 IF score>hiscore:hiscore=score
1834 END REPEAT Dino_program
1835 CSIZE#3,0,0:CLS#3:FontSets
1836 END DEFINE

```

U
P
D
A
T
E

C
L
O
U
D

< Another Game Y/N >

1838 DEFine PROCEDURE DSpeed

1839 CSIZE#3,0,0 :CURSOR#3,232, 2:PRINT#3,'Speed'

1840 sl=slowdown_steps:CURSOR#3,224,12:PRINT #3,'- +'

1841 REPEAT Sp_lp

1842 CURSOR#3,232,12:PRINT#3,FILL\$('0',5-LEN(sl));sl:K=CODE(INKEY\$(-1))

1843 IF K=43 OR K=61:IF sl<50000:sl=sl+500

1844 IF K=45 OR K=95:IF sl>= 500:sl=sl -500

1845 IF K=10:slowdown_steps=sl:EXIT Sp_lp

1846 END REPEAT Sp_lp

1847 CURSOR#3,220,12:PRINT#3,' ';FILL\$('0',5-LEN(sl));sl; ' '

1848 END DEFine DSpeed

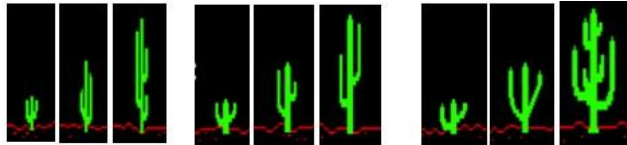
DINO Run HI 00000 Speed
-25000 +

Note: Adjust with +/- then Enter

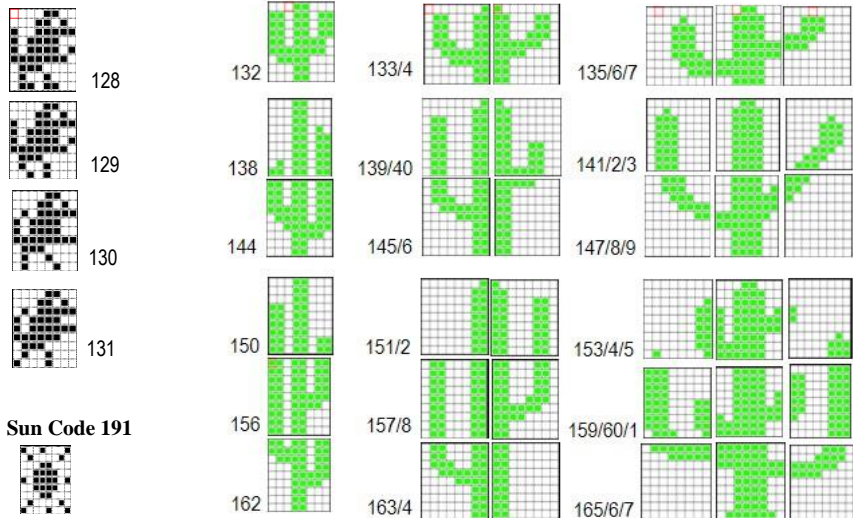
DINO_fnt uses the extended character code set 128 to 191

Cacti Codes 132 to 167

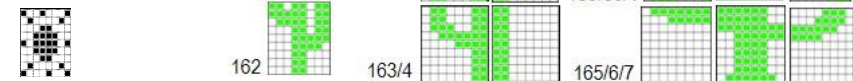
[3 widths 3 heights]



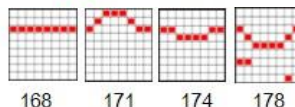
Dino Codes 128 to 131



Sun Code 191

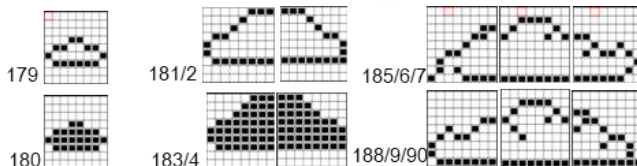


Ground Level Codes 168 to 178



Clouds Codes 179 to 190

[Three widths White/Black]




```

1904 DEFine PROCEDURE main
1905 yy=0:xx=0:x=120:y=80:ix=2:iy=1:fu=90:fin=0:pm=1
1906 OVER#3,0:BLOCK#3,90,4,52,170,5:BLOCK#3,90,4,180,170,5
1907 INK#3,7:CUSOR#3,x,y:PRINT#3,pod$(pm):OVER#3,-1
1908 REPEAT loop
1909   get_keys:PAUSE 5
1910   IF x+xx< 0 OR x+xx>260:xx=-xx :beeps 2
1911   IF y+yy<20 OR y+yy>150:yy=-yy :beeps 2
1912   IF al((x)/12,(y)/10)=1:pod_clear :beeps 3
1913   IF al((x)/12,(y)/10)=2:shield 1 :beeps 4
1914   CURSOR#3,x,y:PRINT#3,pod$(pm):IF fin :PAUSE 30:EXIT loop
1915   x=x+xx:y=y+yy:pm=(pm MOD 5)+1:CUSOR#3,x,y:PRINT#3,pod$(pm)
1916 END REPEAT loop
1917 END DEFine

```

```

1919 DEFine PROCEDURE get_keys
1920 K=KEYROW(1):tx=xx:ty=yy
1921 SElect ON K
1922   =128:yy=yy+iy
1923   =144:yy=yy+iy:xx=xx+ix
1924   =16 :xx=xx+ix
1925   =20 :yy=yy-iy:xx=xx+ix
1926   =4 :yy=yy-iy
1927   =6 :yy=yy-iy:xx=xx-ix
1928   =2 :xx=xx-ix
1929   =130:yy=yy+iy:xx=xx-ix
1930 END SElect
1931 SElect ON K=128,144,16,20,4,6,2,130:fuel 1
1932 IF xx>11 OR xx<-11 THEN xx=tx
1933 IF yy> 8 OR yy< -8 THEN yy=ty
1934 END DEFine

```



```

1936 DEFine PROCEDURE pod_clear
1937 al(x/12,y/10)=0:AT#3,'':sc=sc+10
1938 Giroscore sc,l:cap=cap-1:IF cap=0:fin=2
1939 END DEFine

```

Note: CHR\$(149)



```

1941 DEFine PROCEDURE fuel(f)
1942 fu=f-f:BLOCK#2,f,4,196+fu,211,0:IF fu<=0:fin=1
1943 END DEFine

```



```

1945 DEFine PROCEDURE shield(s)
1946 sh=sh-s:BLOCK#2,s,4, 70+sh,211,0:IF sh<=0:fin=1
1947 END DEFine

```



```

1949 DEFine PROCEDURE beeps(b)
1950 SElect ON b
1951   =1:BEEP 100,0
1952   =2:BEEP 3000,0,200,2,3
1953   =3:BEEP 500,0
1954   =4:BEEP 1000,10,20,1,1
1955   =5:BEEP 1000,0,150,100,1
1956   =6:BEEP 1000,100
1957 END SElect
1958 END DEFine

```

```

1960 DEFINE PROCEDURE Init_Giro
1961 CLS#3:FOR i=1 TO 100:INK#3,(RND(1 TO 5)):POINT#3,RND(5 TO 110),RND(10 TO 85)
1962 Star_Ship 40,50:PAUSE 20:CSIZE#3,2,1:OVER#3,1
1963 INK#3,6:FOR i=0 TO 1:CURL#3,1+i,1:PRINT#3,'GIRO RESCUE' Note: CHR$(128 to 138)
1964 INK#3,2:FOR i=0 TO 2:CURL#3,90+i,22:PRINT#3,'RED ALERT'
1965 CSIZE#3,2,0:OVER#3,0:INK#3,5
1966 RED_Alert 40,50:PAUSE 20:BEEP:CSIZE#3,0,0:INK#3,5
1967 CURSOR#3,58,132:PRINT#3,'Press Any Key to begin RESCUE'
1968 CURSOR#3,36,144:PRINT#3,'Use ←↑→ keys to Guide your Giro Pod'
1969 CURSOR#3,86,154:PRINT#3,'to pick up Survivors'
1970 PAUSE:FOR i=1 TO 9:PAUSE 3:BLOCK#3,276,i*16,0,90-i*8,0
1971 OVER#3,1:INK#3,7:RESTORE 1975
1972 FOR a=1 TO 4
1973 READ x,y,str$:FOR b=0 TO 1:CURL#3,x+b,y:PRINT#3,str$:END FOR b
1974 END FOR a
1975 DATA 146,9,'Score:',228,9,'Level:',2,166,'Shields:',148,166,'Fuel:'
1976 OVER#3,0:CURL#3,0,0:CSIZE#3,2,0
1977 END DEFINE

```

```

1979 DEFINE PROCEDURE Star_Ship(sx,sy)
1980 FILL#3,1:INK#3,7:CIRCLE#3,sx+4,sy-3,8:FILL#3,0
1981 FILL#3,1:OVER#3,1:LINE#3,sx+40,sy+8
1982 RESTORE 1988:FOR i=1 TO 12:READ x,y:LINE#3 TO sx+x,sy+y
1983 FILL#3,0:OVER#3,0:INK#3,0
1984 LINE#3,sx+12,sy TO sx+26,sy+4 TO sx+28,sy+6 TO sx+40,sy+8
1985 LINE#3,sx+44,sy+4 TO sx+32,sy+1 TO sx+30,sy+1.2
1986 CIRCLE#3,sx+4,sy-3,8:ARC#3,sx-4,sy-2 TO sx+6,sy-4,PI/2
1987 FOR i=1 TO 6:CIRCLE#3,sx+12.5+i*2.2,sy-4.2+i*.6,1
1988 DATA 30,10,20,8,18,6,0,2,10,0,26,4,28,6,40,8,44,4,40,0,12,-8,10,0
1989 END DEFINE

```

Note: The QBITS Intro Screen to Giro Rescue



```

1991 DEFINE PROCEDURE RED_Alert(sx,sy)
1992 RESTORE 2000:BEEP 0,10,100,5,-5,10,10,15
1993 FOR i=1 TO 8
1994 INK#3,RND(2 TO 5):CURSOR#3,70,42:PRINT#3,'Abandon Ship'
1995 READ x,y:INK#3,248:LINE#3,sx+12.5+i*2.2,sy-5+i*.6 TO sx+x,sy+y
1996 FILL#3,1:INK#3,7:CIRCLE#3,sx+x,sy+y,1:FILL#3,0:PAUSE 5
1997 READ x,y:INK#3,248:LINE#3,sx+6+i*2.2,sy+5+i*.6 TO sx+x,sy+y
1998 FILL#3,1:INK#3,7:CIRCLE#3,sx+x,sy+y,1:FILL#3,0:PAUSE 9
1999 END FOR i
2000 DATA 4,-18,-24,10,10,-22,-18,18,18,-23,-2,15,28,-23,18
2001 DATA 34,-18,15,15,46,-21,20,18,55,-18,24,19,55,-5,28,15
2002 END DEFINE

```