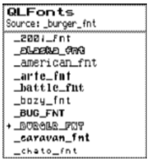
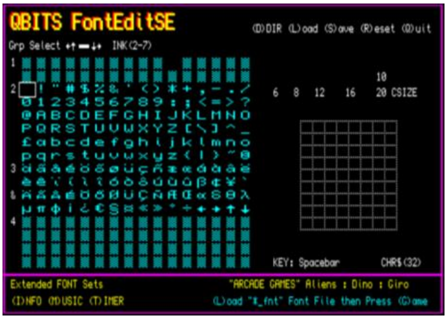

QLFonts
Source: _burger_fnt

_2001_fnt
_alaska_fnt
_american_fnt
_arte_fnt
_battle_fnt
_bozy_fnt
_BUG_FNT
+ _BURGER_FNT
_caravan_fnt
_chato_fnt



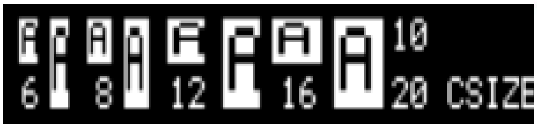
Introduction

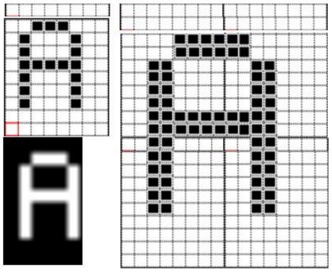
Home Computers in the 1980's used Characters created from Bitmaps. The most common were 8x8 matrixes covering the Western ASCII code set. BITMap Files carry a Header, a number of bytes that might identify type, hold the overall File size, the width and height. For Font _fnt Files this will include A 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, backets 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 :-





CSIZE 0,0 (6x10) CSIZE 3,1 (16x20)

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.

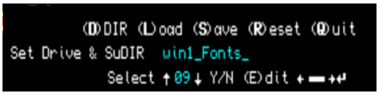
QBITS Font Editor Concept

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 – 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 **SubDIR**ectories if present.

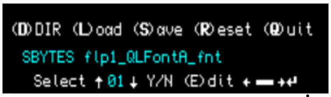
To access a **SubDIR**ectory use (**E**)dit to change for example: 'flp2_' to 'win1_Fonts_', Enter to Return and 'Y' to accept. Then select (**L**)oad.



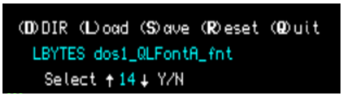
```
      (D)DIR (L)oad (S)ave (R)eset (Q)uit
Set Drive & SuDIR  win1_Fonts_
          Select ↑09↓ Y/N (E)dit ←▬→↵
```

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 Devise has none. Use Up/Down Cursors keys to Select from those '_fnt' Files found, then 'Y' to Load '_fnt' file or 'N' to abort. Load checks for Lowest Character and Number of characters held in First & Second byte of file. It then LBYTES to relative memory address FBase A, B, 1 or 2 then Reads and Displays Fonts to screen.



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

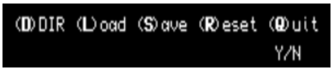


```
(D)DIR (L)oad (S)ave (R)eset (Q)uit
 LBYTES dos1_QLFontA_fnt
  Select ↑14↓ Y/N
```

To **SAVE** use Up/Down Cursors and Select Group (1-2-3-4). The current Font Filename for that Group is displayed. This maybe the Default, LOAD or previous SAVE Filename. Included is a Line Editor to Rename the FONT Filename. When ready to SAVE a check is made: 'DEVICE ERROR' is shown if Device unavailable. 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$ a return to the **QBITSProgs** Menu program.

QBITS FontEditSE - Navigation
Navigate the Font Chart using the Cursor keys and Select the Highlighted Character Outlined by a White rectangle. Press Spacebar and Bitmap Grid is now active and Cursor Keys are used to highlight a Grid Cell. Spacebar toggles between INK and STRIP (BackGnd) colour by setting binary bit to 1 or 0. Select 'N' to return without changing the existing Bitmap, 'Y' to write the changed Bit Pattern into memory.

QBITS FontEditSE - Editor
The heart of the Editor is the Bitmap and what Character Fonts can be created be they Bold, Italics, or Personal and Stylised alphanumerical, Futuristic or to represent an Ancient Language as in Hieroglyphic, Cuneiform or Runes. Maybe for Retro Games Transforming Fonts into Game pieces might also be desirable.

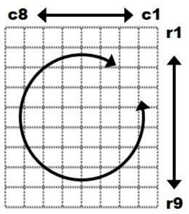
Select a **Font** Outlined with a **White Box**. Press Spacebar to access the **Bitmap** Editor bits are set to 1's & 0's [INK/BkGnd Colours]. The Font is also displayed above in its different **CSIZES.** Below **KEY:** shows **t**he Keys for Printable Fonts and the ASCII Character Code 'n' displayed with **CHR$(n).**

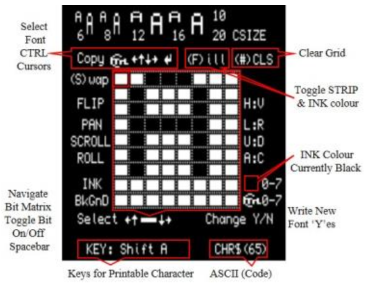
Where slight pattern changes are required copying an existing Chart Font into the Editor might be useful. To **Copy** use CTRL ⬅⬆⬇➡ to highlight a second Chart Font now Outlined within a **Red box**. Press ↵ Enter to action, alternatively you could Press '**S**' and Swap Font Grid Positions. (**F**)ill Sets all to '1' INK colour, (**#**) CLS sets Bitmap cells to '0' BKGnd. Press 1-7 for INK colour Press CTRL 0-7 for BkGnd Colour

.

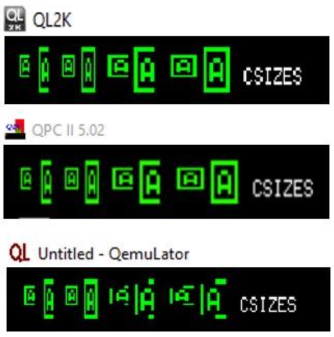
Cell Movement

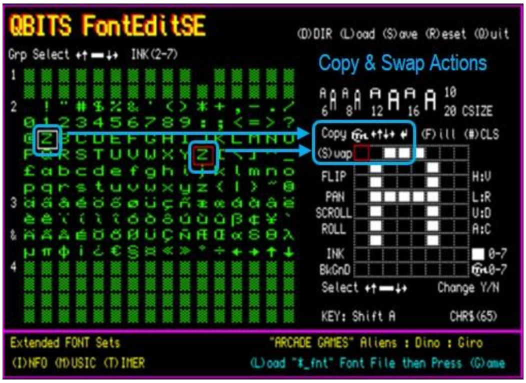
Flip	c1=c8 to c8=c1	**H**
	r1=r9 to r9=r1	**V**
Pan	c1=c8 to c2=c1 c8=c7	**L**
	c1=c2 to c7=c8 c8=c1	**R**
Scroll	r1=r2 to r8=r9 r9=r1	**U**
	r2=r1 to r9=r8 r1=r9	**D**
Roll	r1,c1=r1,c8 r9,c1=r1,c1	**A**
	r1,c1=r9,c1 r9,c8=r1,c8	**C**





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°. Navigate Bitmap with Cursor keys ⬅⬆⬇➡ 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 or 2,0 [6 &12pixels] Print only first six columns of Bitmap.

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 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:

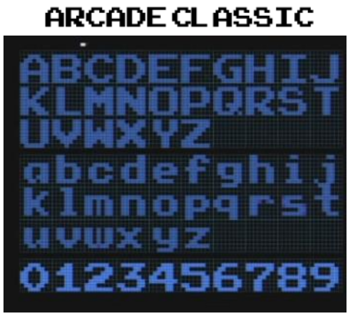
FBase1 = **ALCHP**(876) **LBYTES** FLP1_FONT1,FBase1
FBase2 = **ALCHP**(588) **LBYTES** FLP1_FONT2,FBase2

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

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.

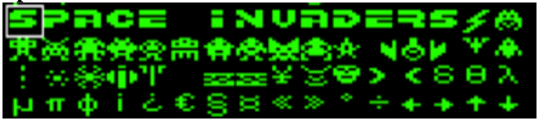
PIXEL Based Font Designs
Pixel-based Characters were popular in the 1980s, when the digital era was just beginning. One typeface stood above the rest the "Atari" font set of 1984 (or known as "Namco" in Japan). Based on the Western ACSII Character Set it lasted until larger resolutions and 3D gaming 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.

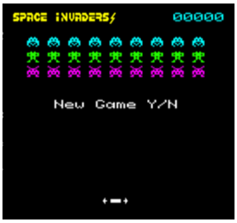


Screens made up of 8x8 bitmap matrix tiles was also used in Arcade Games from the 1970s through to the 1990s. Their chunky and primitive visuals revealed artistry and ingenuity to make full-fledged characters out of just a few pixels. These Fonts were designed by hand on graph paper and coded into the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

QBITS FontEditorSE - ARCADE Games
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.

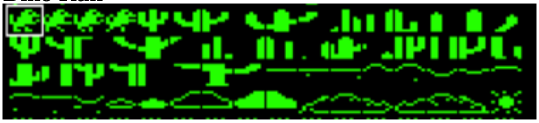
Space Invaders

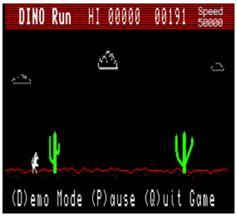




This simplified version has four levels. Move the Defender left or Right using Cursor Keys and Spacebar fires the laser. You have four Lives so watch out for enemy bombs.

Dino Run




DINO Run HI 00000 00191 Speed 50000
(D)emo Mode (P)ause (Q)uit Game

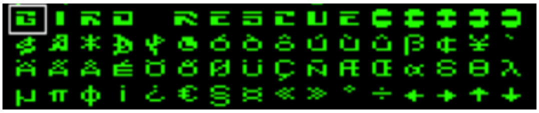
Displayed are High Score and Accumulated Points. The program has a Speed adjustment to overcome CPU Timing issues with different QL Platforms. The PAN control of the screen uses CHAR_INC, press Spacebar to Jump DINO over the approaching Cacti or you lose.

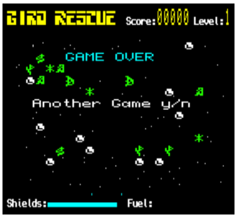
Note: Retro Game Control
The SBASIC keyword **CHAR_INC**#ch, x_inc, y_inc sets Character width and hight beyond those set by CSIZE. Here where the subject Character Font(s) move across the screen, an independent height can be set for **PAN**.

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

Giro Rescue





The aim rescue all the Escape Pods scattered through space, watch out for space Debry, 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 FontEditSE - New FONT Designs

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 to take. **QBITS FontEditSE** has the Edit tools to **Copy**, **Swap** and **Modify** the Bitmap layout. Change INK and BackGnd, Fill or Clear the Grid. Flip or Move Grid Cells even Roll 90^0 Clockwise or Anticlockwise. All of which can hasten the creation of New FONT Sets. These can then be Saved in a Format used with the QL O/S for screen PRINT, after loading into memory and set with **CHAR_USE** to a WINDOW Channel.

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 are used in the Code of the following pages. Checkout the use of **DEF PROC QBPrnt** and channel assignments used with the **MUSIC_fnt** and **TIMER_fnt** displays.

QLINFO_fnt	Small Character Fonts for Information Labelling
QLMUSIC_fnt	Music Fonts for Score sheets Notes Rests etc.
QLTIMER_fnt	Clock and Timeout Fonts

QBITS FontEditSE - FONT Displays

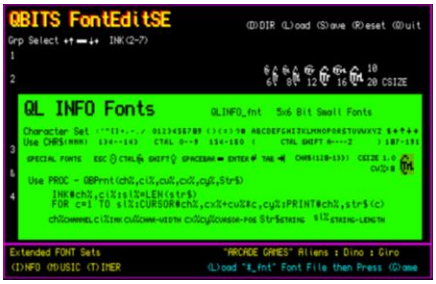
QBITS Font_Info originally written as a standalone Prog this is now included as part of QBITS_FontEditSE.

```
1460 REMark Demo: RETRO ARCADE GAMES - QBITS FontEdit2 SE 2022

1462 DEFine PROCedure Init_Info
1463 base1=ALCHP(588):LBYTES Dev$&'QLINFO_fnt',base1 :CHAR_USE#5,0,base1
1464 base2=ALCHP(588):LBYTES Dev$&'QLMUSIC_fnt',base2:CHAR_USE#6,0,base2
1465 base3=ALCHP(588):LBYTES Dev$&'QLTIMER_fnt',base3:CHAR_USE#7,0,base3
1466 END DEFine

1468 DEFine PROCedure ARCADE
1469 CHAR_INC#0,6,14:INK#0,6
1470 CURSOR#0,260,2:PRINT#0,'"ARCADE GAMES" Aliens : Dino : Giro'
1471 CURSOR#0,6,2:PRINT#0,'Extended FONT Sets'\' (I)NFO (M)USIC (T)IMER'
1472 INK#0,5:CURSOR#0,240,16:PRINT#0,'(L)oad "*_fnt" Font File then Press (G)ame'
1473 END DEFine

1475 DEFine PROCedure FontView
1476 SELect ON K=73,105:CLS#3:Info_Fonts   :PAUSE:K=0
1477 SELect ON K=77,109:CLS#3:Music_Fonts:PAUSE:K=0
1478 SELect ON K=84,116:CLS#3:Timer_Fonts:PAUSE:K=0
1479 IF K=0:CLS#5:FontGrid:FontSets:RETurn :
1480 IF Fg$(3)=='Aliens_fnt' :CLS#3:Space_Invader            Note: Load _fnt File and Press (G)ame
1481 IF Fg$(3)=='Dino_fnt'   :CLS#3:Dino_Run
1482 IF Fg$(3)=='Giro_fnt'    :CLS#3:Giro_Rescue
1483 END DEFine

2050 REMark QBITS FONT Displays

2052 DEFine PROCedure QBPrnt(ch%,ci%,cw%,cx%,cy%,str$)
2053 INK#ch%,ci%:sl%=LEN(str$)-1:IF cw%<5:cw%=5
2054 FOR c=0 TO sl%:CURSOR#ch%,cx%+cw%*c,cy%:PRINT#ch%,str$(c+1)
2055 END DEFine
```

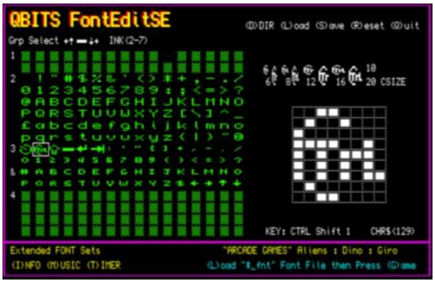
QBITS Extended FONT’s - INFO Fonts

The QL extended range CHR$(128 to 191) was used to provide an Alphanumeric set of 5x6 Bit Matrix FONTS. The Printable key ranges of CHR$(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$(154 to 159) are used for Punctuation, Brackets etc. CHR$(187 to 191) used for ‘$’ and Cursor Arrows.

SPECIAL FONTS ESC CTRL SHIFT SPACEBAR ENTER TAB use CHR$(128 to 133) and require all eight columns, CSIZE 1,0 or 3.0 or 1,1 or 3,1. Creating the Special Fonts was a straight forward adaptation apart from the **CTRL** which took a little time to fit four characters into a readable 8x9 Matrix.



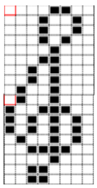
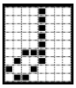
```
QBITS FontEditSE                          (D)DIR (L)oad (S)ave (R)eset (Q)uit
Grp Select  INK(2-7)
                                          6 8 12 16 10 20 CSIZE
QL INFO Fonts          QLINFO_fnt   5x6 Bit Small Fonts
Character Set   0123456789   ABCDEFGHIJKLMNOPQRSTUVWXYZ
Use CHR$(NNN)  134--143   CTRL 0--9  154-160 (   CTRL SHIFT A---Z   ) 187-191
SPECIAL FONTS  ESC CTRL SHIFT SPACEBAR ENTER TAB  CHR$(128-133)  CSIZE 1.0
                                                                  cw%=8
Use PROC - QBPrnt(ch%,ci%,cw%,cx%,cy%,Str$)
   INK#ch%,ci%:sl%=LEN(str$)
   FOR c=1 TO sl%:CURSOR#ch%,cx%+cw%*c,cy%:PRINT#ch%,str$(c)
  ch%CHANNEL ci%INK cw%CHAR-WIDTH cx%cy%CURSOR-POS Str$STRING  sl%STRING-LENGTH
Extended FONT Sets                    "ARCADE GAMES" Aliens : Dino : Giro
(I)NFO (M)USIC (T)IMER                (L)oad "*_fnt" Font File then Press (G)ame
```



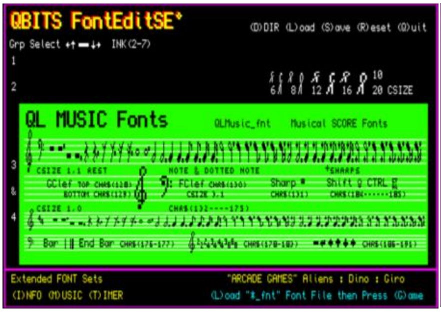
```
QBITS FontEditSE                          (D)DIR (L)oad (S)ave (R)eset (Q)uit
Grp Select  INK(2-7)
                                          6 8 12 16 10 20 CSIZE
KEY: CTRL Shift 1      CHR$(129)
Extended FONT Sets                    "ARCADE GAMES" Aliens : Dino : Giro
(I)NFO (M)USIC (T)IMER                (L)oad "*_fnt" Font File then Press (G)ame
```

```
2060 DEFine PROCedure Info_Fonts
2061 PAPER#5,4:CLS#5:INK#5,0:OVER#5,1:CURSOR 228,60:PRINT Font$
2062 CSIZE#5,2,1:FOR i=0 TO 1:CURSOR#5,6+i,4:PRINT#5,'QL INFO Fonts '
2063 CSIZE#5,0,0:CURSOR#5,230,12:PRINT#5,'QLINFO_fnt   5x6 Bit Small Fonts'
2064 QBPrnt 5,0,6,6,30,'Character Set'
2065 FOR i=0 TO 9:CURSOR#5,92+i*6,30:PRINT#5,CHR$(134+i)
2066 QBPrnt 5,0,6,6,40,'Use CHR$š®®®›  “‘”••‘”‘“'
2067 QBPrnt 5,0,6,160,30,'•‘’“”•——˜™'                    :REMark 0-9
2068 QBPrnt 5,0,6,162,40,'£´²¬ •••™'                     :REMark CTRL 0--9
2069 FOR i=0 TO 6:CURSOR#5,224+i*7,30:PRINT#5,CHR$(154+i)
2070 QBPrnt 5,0,6,228,40,'“•”•‘–•'                       :REMark CHR$(154 - 160)
2071 QBPrnt 5,0,6,272,30,' ¡¢£¤¥¦§¨©ª«¬-®¯°±²³´µ¶·¸¹º'   :REMark CTRL Shift A -- Z
2072 QBPrnt 5,0,5,278,40,'š     £´²¬ ³¨©¦´ ¡••••º     ›' :REMark ‘(   CTRL Shift A-Z   )’
2073 QBPrnt 5,0,8,440,30,'»¼¾¿½'
2074 QBPrnt 5,0,6,438,40,'‘˜—•‘™“'                       :REMark CHR$(187 - 191)
2075 QBPrnt 5,0,5,10,56,'³º¥£©¡¬ ¦¯®´³'                  :REMark ‘SPECIAL FONTS’
2076 QBPrnt 5,0,5,90,56,'¥³£  £´²¬  ³¨©¦´  ³º¡£¥¢¡²  ¥®´¥²  ´¡¢'
2077 CSIZE#5,1,0:QBPrnt 5,0,8,110,55,'€  •   ‚    ƒ   „  …':CSIZE#5,0,0
2078 QBPrnt 5,0,5,330,55,'£¨²»š‘’˜•““‘›  £³©º¥ ‘Œ•'      :REMark CHR$(128 - 133)
2079 QBPrnt 5,0,6,420,65,'£·%•˜'                         :REMark cw%=8
2080 FOR r=0 TO 9:FOR c=0 TO 7:BLOCK#5,1,1,457+c*2,56+r*2,2
2081 CSIZE#5,3,1:CURSOR#5,456,55:PRINT#5,CHR$(129):CSIZE#5,0,0
2082 REMark Proc QBPrnt Channel INK Char Width x,y String' '
2083 QBPrnt 5,0,6,12,76,'Use PROC - QBPrnt(ch%,ci%,cw%,cx%,cy%,Str$)'
2084 QBPrnt 5,0,7,40,89,'INK#ch%,ci%:sl%=LEN(str$)'
2085 QBPrnt 5,0,7,40,98,'FOR c=1 TO sl%:CURSOR#ch%,cx%+cw%*c,cy%:PRINT#ch%,str$(c)'
2086 CURSOR#5,36,112:PRINT#5,'ch%     ci%   cw%      cx%cy%      Str$'
2087 QBPrnt 5,0,5, 54,113,'£¨¡®®¥¬   ©®«   £¨¡²•·©¤´¨      £µ²³¯²•º¯³'
2088 QBPrnt 5,0,5,312,113,'³´²©®§     ³´²©®§•¬¥®§´¨':QBPrnt 5,0,6,352,110,'sl%'
2089 PAPER#5,0  :REMark STRING   STRING_LENGTH
2090 END DEFine
```

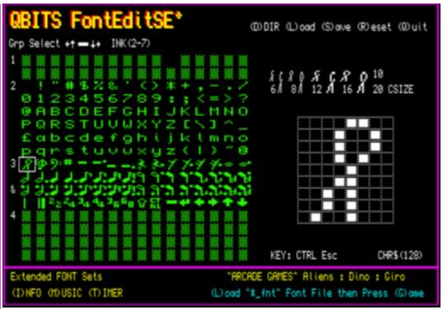
QBITS Extended FONT’s - MUSIC Fonts

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 **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.



QBITS FontEditSE*
(D)DIR (L)oad (S)ave (R)eset (Q)uit
Grp Select ←↑→↓ INK(2-7)
6♪ 8♪ 12♪ 16♪ 10 20 CSIZE
QL MUSIC Fonts   QLMusic_fnt   Musical SCORE Fonts
CSIZE 1,1 REST   NOTE & DOTTED NOTE   *SHARPS
GClef TOP CHR$(128) BOTTOM CHR$(129)   FClef CHR$(130) CSIZE 3,1   Sharp * CHR$(131)   Shift & CTRL CHR$(184------185)
CSIZE 1,0   CHR$(132----175)
Bar | End Bar CHR$(176-177)   CHR$(178-183)   CHR$(186-191)
Extended FONT Sets   "ARCADE GAMES" Aliens : Dino : Giro
(I)NFO (M)USIC (T)IMER   (L)oad "*_fnt" Font File then Press (G)ame

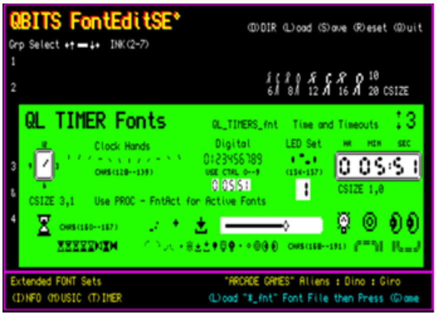


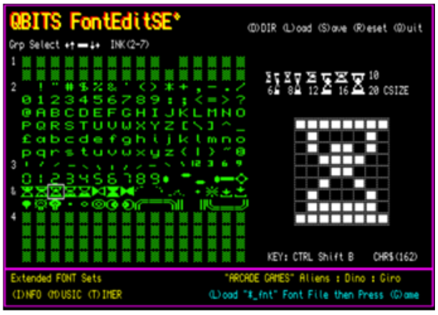
QBITS FontEditSE*
(D)DIR (L)oad (S)ave (R)eset (Q)uit
Grp Select ←↑→↓ INK(2-7)
! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z [\ ] ^ _
£ a b c d e f g h i j k l m n o
p q r s t u v w x y z { | } ~ ©
6♪ 8♪ 12♪ 16♪ 10 20 CSIZE
KEY: CTRL Esc   CHR$(128)
Extended FONT Sets   "ARCADE GAMES" Aliens : Dino : Giro
(I)NFO (M)USIC (T)IMER   (L)oad "*_fnt" Font File then Press (G)ame

```
2100 DEFine PROCedure Music_Fonts
2101 PAPER#6,4:CLS#6:INK#6,0:OVER#6,1:OVER#5,1:CURSOR 228,60:PRINT Font$
2102 CSIZE#6,2,1:FOR i=0 TO 1:CURSOR#6,6+i,4:PRINT#6,'QL MUSIC Fonts'
2103 CSIZE#6,0,0:CURSOR#6,230,12:PRINT#6,'QLMusic_fnt    Musical SCORE Fonts'
2104 FOR i=1 TO 25 STEP 5:BLOCK#6,474,1,6,29+i,7:BLOCK#6,474,1,6,60+i,7
2105 FOR i=1 TO 15 STEP 3:BLOCK#6,474,1,6,91+i,7:BLOCK#6,474,1,6,110+i,7
2106 CSIZE#6,1,1:INK#6,0:STRIP#6,4:OVER#6,1:CURSOR#6,22,28:PRINT#6,CHR$(130)
2107 CURSOR#6,8,20:PRINT#6,CHR$(128):CURSOR#6,8,38:PRINT#6,CHR$(129)
2108 n=0:FOR i=132 TO 175:CURSOR#6,36+n,30:PRINT#6,CHR$(i);:n=n+10:END FOR i
2109 CSIZE#6,0,0 :REMark CSIZE 1,1 RESTS          NOTE & DOTTED NOTE
2110 QBPrnt 5,0,6,20,52,'£³©º¥ ‘Œ‘ ²¥³´          ®¯´¥ & ¤¯´´¥¤ ®¯´¥'
2111 QBPrnt 6,0,6,360,50,'+':QBPrnt 5,0,6,366,52,'³¨¡²º³'                    :REMark Sharp
2112 QBPrnt 6,0,6,32,63,'GClef                  FClef'
2113 QBPrnt 5,0,5,68,64,'´¯° £¨²»š‘ˆ›              £¨²»š‘“•›'
2114 QBPrnt 5,0,5,52,72,'¢¯´´¯- £¨²»š‘™›          £³©º¥ “Œ“
2115 CURSOR#6,0,0:CSIZE#6,3,1        :CURSOR#6,166,59:PRINT#6,CHR$(130)
2116 CURSOR#6,136,51:PRINT#6,CHR$(128):CURSOR#6,136,69:PRINT#6,CHR$(129)
2117 CSIZE#6,1,0
2118 QBPrnt 6,0,6,296,62,'Sharp ƒ   Shift ¸ CTRL ¹'
2119 QBPrnt 5,0,5,296,72,'£¨²»š‘“‘›   £¨²»š‘˜”••••••‘˜•›'
2120 REMark              CHR$(131)    CHR$(184------185)
2121 CURSOR#6,10,88:PRINT#6,CHR$(128):CURSOR#6,10,97:PRINT#6,CHR$(129)
2122 CURSOR#6,22,93:PRINT#6,CHR$(181):CURSOR#6,10,112:PRINT#6,CHR$(130)
2123 n=0:FOR i=132 TO 175:CURSOR#6,40+n,94:PRINT#6,CHR$(i);:n=n+10:END FOR i
2124 QBPrnt 5,0,6,20,84,'£³©º¥ ‘Œ•               £¨²»š‘“’••••‘—•›'
2125 QBPrnt 6,0,6,28,113,'Bar °± End Bar'
2126 QBPrnt 5,0,5,118,114,'£¨²»š‘——•‘——›'                               :REMark CHR$(176-177)
2127 CURSOR#6,200,107:PRINT#6,CHR$(128):CURSOR#6,200,116:PRINT#6,CHR$(129)
2128 QBPrnt 6,0,8,210,113,'²³´µ¶·'                                        :REMark 2/2 2/4 3/44/4 3/8 6/8
2129 QBPrnt 5,0,5,264,114,'£¨²»š‘—˜•‘˜“›'                                 :REMark CHR$(178-183)
2130 QBPrnt 6,0,9,346,113,'º»¼¾¿½'
2131 QBPrnt 5,0,5,404,114,'£¨²»š‘˜–•‘™‘›'                                 :REMark CHR$(186-191)
2132 END DEFine
```

QBITS Extended FONT's - TIMER Fonts

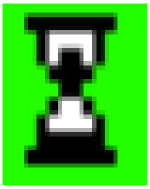
Here the challenge was exploring the effects of Font Combinations and Sequencing for Clocks and Timeout Displays. This involved Backgrounds and Fonts used as Masks to achieve desired results - see Procedure **FntAct**. [Font Action]





```
2150 DEFine PROCedure Timer_Fonts
2151 STRIP#5,4:CLS#7:OVER#7,1:INK#7,0:CURSOR 228,60:PRINT Font$
2152 CSIZE#7,2,1:FOR i=0 TO 1:CURSOR#7,6+i,4:PRINT#7,'QL TIMER Fonts'
2153 CSIZE#7,0,0:CURSOR#7,230,12:PRINT#7,'QL_TIMERS_fnt   Time and Timeouts'
2154 REMark *** Clock Display
2155 CSIZE#7,1,0:FOR a=1 TO 12:CURSOR#7,44+a*12,40:PRINT#7,CHR$(127+a)
2156 QBPrnt 7,0,6,90,30,'Clock Hands'
2157 QBPrnt 5,0,5,90,52,'£¨²»š‘’˜••‘“™›'              :REMark CHR$(128-139)
2158 cx%=18:cy%=36:BLOCK#7,22,25,cx%+1,cy%+2,7
2159 CURSOR#7,cx%,cy%   :PRINT#7,CHR$(184);CHR$(185);CHR$(186)      :REMark  ---
2160 CURSOR#7,cx%,cy%+9 :PRINT#7,CHR$(187);CHR$(32) ;CHR$(188)      :REMark |  |
2161 CURSOR#7,cx%,cy%+18:PRINT#7,CHR$(189);CHR$(190);CHR$(191)    :REMark  ---
2162 CURSOR#7,28,30:PRINT#7,CHR$(140)                               :REMark  12
2163 CURSOR#7,11,48:PRINT#7,CHR$(143);'   ';CHR$(141)               :REMark 9  3
2164 CURSOR#7,28,64:PRINT#7,CHR$(142)                               :REMark   6
2165 REMark *** Digital Display
2166 QBPrnt 7,0,6,234,28,'Digital      LED Set'
2167 QBPrnt 5,0,5,386,28,'¨²    -©®    ³¥£'                    :REMark HR MIN SEC
2168 QBPrnt 7,0,7,220,40,'•‘’“”•——˜™     š›œ•'                 :REMark 0123456789
2169 BLOCK#7,102,23,377,40,7:STRIP#7,7:CURSOR#7,438,46:PRINT#7,':'
2170 CURSOR#7,376,38:PRINT#7,'¸¹¹¹¹¹¹¹¹¹¹¹º'                     :REMark  --------- Top
2171 CURSOR#7,376,47:PRINT#7,'»'                                 :REMark | Left
2172 CURSOR#7,472,47:PRINT#7,'¼'                                 :REMark     Right |
2173 CURSOR#7,376,54:PRINT#7,'½¾¾¾¾¾¾¾¾¾¾¾¿'                      :REMark  --------- Bottom
2174 QBPrnt 5,0,5,222,52,'µ³¥ £´²¬ •••™      š‘•”•‘•—›'        :REMark CTRL 0--9 134-
2175 REMark *** Active Timeouts
2176 QBPrnt 7,0,6,12,76,'CSIZE 3,1':QBPrnt 7,0,6,380,66,'CSIZE 1,0'
2177 FOR a=0 TO 7:CURSOR#7,46+a*9,113:PRINT#7,CHR$(160+a)
2178 QBPrnt 5,0,5,48,98,'£¨²»š‘–•••‘——›'                      :REMark CHR$(160-167)
2179 FOR a=0 TO 15:CURSOR#7,150+a*10,113:PRINT#7,CHR$(168+a)
2180 QBPrnt 5,0,5,320,114,'£¨²»š‘–˜••‘™‘›'                     :REMark ChR$(184-191)
2181 FOR a=0 TO 7:CURSOR#7,400+a*10,113:PRINT#7,CHR$(184+a)
2182 OVER#7,0:STRIP#7,4:QBPrnt 7,0,6,90,76,'Use PROC - FntAct for Active Fonts'
2183 CURSOR#5,450,4:PRINT#5,'¾':CURSOR#5,450,14:PRINT#5,'¿'
2184 c1=0:c2=0:c3=0:c4=0:c5=0:c6=0:c7=1:c8=0:fx%=250:xx%=4:cnt%=0:td%=5
2185 h=0:hr%=0:min%=0:sec%=0:ic%=0:bc%=4:mc%=0:BLOCK#7,120,14,240,94,7
```

```
2186 REPeat Fnt_lp
2187   IF KEYROW(1)=64 :EXIT Fnt_lp:END IF
2188   CURSOR#7,460,4  :PRINT#7,td%
2189   IF td%>0:PAUSE td%*2:ELSE PAUSE                    :REMark Set Timer
2190   IF KEYROW(1)=  4:td%=td%+1:IF td%>9:td%=9:PAUSE 1   :REMark Up/More
2191   IF KEYROW(1)=128:td%=td%-1:IF td%<1:td%=0:PAUSE 1   :REMark Down/Less
2192   CSIZE#7,1,0:sec%=sec%+1 :Digits 7,0,4,230,62        :REMark Digital Clock
2193   cnt%=cnt%+1:fx%=fx%+xx%:BarFnt 250,350,fx%,96       :REMark Sliding Bar
2194   CSIZE#7,3,1:mn%=166:IF c1>4:mn%=167:END IF          :REMark Mask check
2195   ActFnt 7,0, 23,40, 32,128+h :h=h+1   :IF h=12:h=0   :REMark HR Clock
2196   ActFnt 4,7, 22,90,mn%,160+c1:c1=c1+1 :IF c1>5:c1=0  :REMark Egg Timer
2197   ActFnt 4,0,150,88,  32,168+c2  :c2=c2+1 :IF c2=4:c2=0  :REMark Spin
2198   ActFnt 4,0,180,88,  32,172+c3  :c3=c3+1 :IF c3=2:c3=0  :REMark + *
2199   ActFnt 4,0,210,88,  32,174+c4  :c4=c4+1 :IF c4=2:c4=0  :REMark Arrow
2200   ActFnt 4,7,380,88,176,177+c5  :c5=c5+1 :IF c5=2:c5=0  :REMark Lamp
2201   ActFnt 4,0,410,88,  32,179+c6  :c6=c6+1 :IF c6=3:c6=0  :REMark . o O
2202   ActFnt 4,0,440,88,  32,182+c7  :c7=c7+1 :IF c7=2:c7=0  :REMark L Eye
2203   ActFnt 4,0,460,88,  32,182+c8  :c8=c8+c7:IF c8=2:c8=0  :REMark R Eye
2204 END REPeat Fnt_lp
2205 END DEFine
```



REPeat Fnt_lp cycles the counters for each FONT sequence.

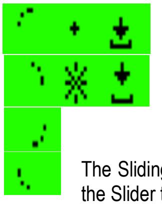
PROC - **FntAct** Prints Font to Screen

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

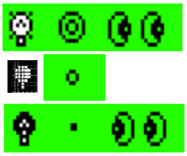
Fonts
Masks

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.

```
2207 DEFine PROCedure ActFnt(bc%,ic%,fx%,fy%,mn%,fn%)
2208 REMark Action BackGnd STRIP bc% INK ic% x/y Mask mn% Font fn%
2209 INK#7,ic%:STRIP#7,bc%:CURSOR#7,fx%,fy%:PRINT#7,CHR$(mn%)          :REMark Mask
2210 INK#7,0:OVER#7,1 :CURSOR#7,fx%,fy%:PRINT#7,CHR$(fn%):OVER#7,0    :REMark Font
2211 END DEFine
```

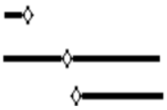


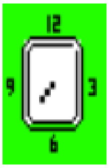
The Font actions can vary as an ON/OFF or swap between two or a number of Fonts displayed in sequence.



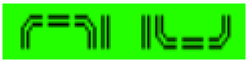
The Sliding Bar has two parts, the Bar and the Slider. In this example the Bar grows left to right with the Slider to max. The Slider then cycles across the Bar until it reaches a set number of loops. The Bar length then diminishes as the Slider moves left to right. With counter reset the process repeats.

```
2213 DEFine PROCedure BarFnt(lx%,rx%,fx%,fy%)
2214 IF fx%<lx% OR  fx%>rx%:xx%=-xx%       :BLOCK#7,8,10,rx%+4,fy%,7
2215 IF fx%>rx% AND cnt%<60:fx%=lx%
2216 IF fx%>rx% AND cnt%>60:mc%=7:fx%=lx%
2217 IF fx%<lx% AND cnt%>99:mc%=0:cnt%=0
2218 STRIP#7,7:INK#7,mc%:CURSOR#7,fx%,fy%    :PRINT#7,CHR$(158)
2219         INK#7,0  :CURSOR#7,fx%+xx%,fy%:PRINT#7,CHR$(159)
2220 END DEFine
```





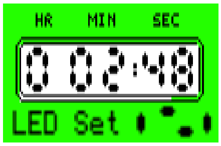
A simple Analogue Clock with CHR$(128 to 139) Fonts one for each of the twelve-hour hand positions. CHR$(140 to 143) 5x5Bit Time Fonts **12 3 6 9**

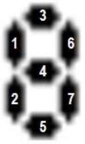


Clock Border Fonts CHR$(184 -191)

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

0123456789
1 4751

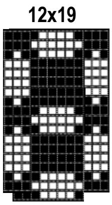


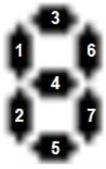


This continued with a more weighted Style of LED with numbers using just Four Fonts CHR$(154-160). To create the numbers 0-9 an Array LED(9,7) is initialised with entries set with relative LED CHR$(n) or to zero if not used. For example: ONE would be 6 & 7 set to CHR$(157). Code then READs the Array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

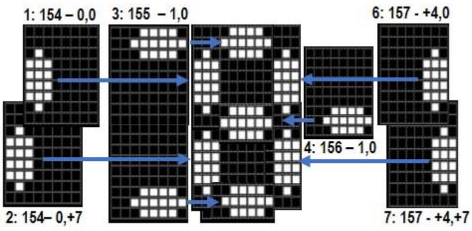
```
2222 DEFine PROCedure Digits(ch%,bc%,ic%,fx%,fy%)
2223 IF sec%=60:sec%=0:min%=min%+1:END IF :st% =sec%  DIV 10:su%=sec%  MOD 10
2224 IF min%=60:min%=0:hr%  =hr%+1   :END IF :mt%=min% DIV 10:mu%=min% MOD 10
2225 IF hr%=10:hr%=0:min%=0:sec%=0:END IF :STRIP#ch%,7
2226 Prt_LED 446,43,st% :Prt_LED 462,43,su% :Prt_LED 332,62,su%
2227 Prt_LED 406,43,mt%:Prt_LED 424,43,mu%:Prt_LED 382,43,hr%
2228 sec$=CHR$(144+st%)&CHR$(144+su%):CURSOR#ch%,fx%+30,fy%:PRINT#ch%,sec$
2229 min$=CHR$(144+mt%)&CHR$(144+mu%):CURSOR#ch%,fx%+12,fy%:PRINT#ch%,min$
2230 hr$=CHR$(144+hr%):CURSOR#ch%,fx%,fy%:PRINT#ch%,hr$:STRIP#ch%,4
2231 END DEFine
```

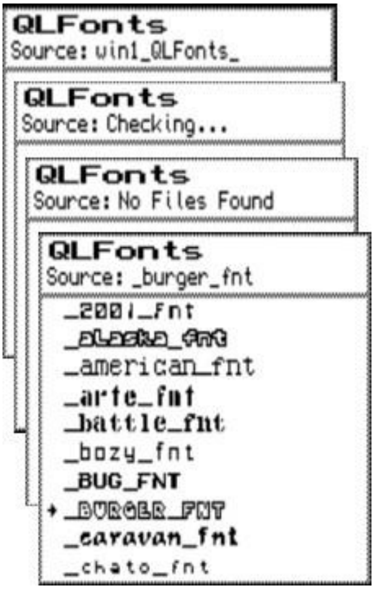
```
2233 DEFine PROCedure Prt_LED(dx%,dy%,dn%)
2234 BLOCK#7,17,18,dx%-2,dy%,7
2235 CURSOR#7,dx%    ,dy%    :PRINT#7,CHR$(LED(dn%,1)):OVER#7,1
2236 CURSOR#7,dx%    ,dy%+7:PRINT#7,CHR$(LED(dn%,2))
2237 CURSOR#7,dx%+1,dy%    :PRINT#7,CHR$(LED(dn%,3))
2238 CURSOR#7,dx%+1,dy%+1:PRINT#7,CHR$(LED(dn%,4))
2239 CURSOR#7,dx%+1,dy%+8:PRINT#7,CHR$(LED(dn%,5))
2240 CURSOR#7,dx%+4,dy%    :PRINT#7,CHR$(LED(dn%,6))
2241 CURSOR#7,dx%+4,dy%+7:PRINT#7,CHR$(LED(dn%,7)):OVER#7,0
2242 END DEFine
```





```
2244  DEFine PROCedure Set_LED
2245 DIM LED(9,7):RESTORE 3192:FOR a=0 TO 9:FOR b=1 TO 7:READ LED(a,b)
2246 DATA 154,154,155,  0,156,157,157
2247 DATA   0,  0,0  ,  0,  0,157,157
2248 DATA   0,154,155,156,156,157,  0
2249 DATA   0,  0,155,156,156,157,157
2250 DATA 154,0  ,0  ,156,0  ,157,157
2251 DATA 154,0  ,155,156,156,0  ,157
2252 DATA 154,154,155,156,156,0  ,157
2253199 DATA   0,  0,155,  0,  0,157,157
2254 DATA 154,154,155,156,156,157,157
2255 DATA 154,  0,155,156,156,157,157
2256 END DEFine
```




QLFonts
Source: win1_QLFonts_

QLFonts
Source: Checking...

QLFonts
Source: No Files Found

QLFonts
Source: _burger_fnt

```
_2001_fnt
_alaska_fnt
_american_fnt
_arte_fnt
_battle_fnt
_bozy_fnt
_BUG_FNT
♦ _BURGER_FNT
_caravan_fnt
_chato_fnt
```

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 earlier.

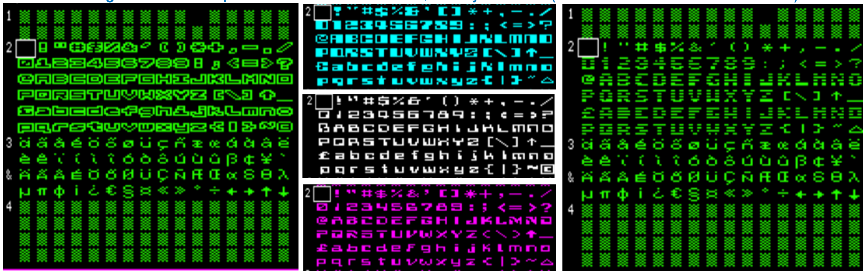
If you Press Enter to Return without making a Source entry, CHAR_USE is then reset to the QL Default Character Fonts CHAR_USE#1,0,0.

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   FntList: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:CURSOR#4,2,16:PRINT#4,'LBYTES '
1030 REPeat lp
1031   INK#3,0:CURSOR#3,2,-11+sr*11:PRINT#3,'½'
1032   PRINT#4,File$(sn)::K=CODE(INKEY$(-1))
1033   INK#3,7:CURSOR#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 LBYTES FDrv$&File$(sn),FBase1:CHAR_USE#ch,FBase1,0
1057 INK#ch,0:CURSOR#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
```

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



QBITS FontEditSE - Associated Files

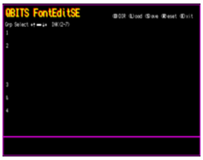
This Program requires the following files to be available **QLFontA_fnt** & **QLFontB_fnt,** these are the two default QL Font Character Groups. Used with **QBITSBoot_bas** this sets the default device and configures **QBITSConfig** used to set a range of available devices.

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

QLINFO_fnt, **QLMUSIC_fnt**, **QLTIMER_fnt** are Font files of alternative characters sets for screen displays.

QBITS FontEditSE - Main Prog Code

```
1000 REMark QBITS_FontEditSE_bas [QBITS Font Editor SE 2024 QL49th - QPC2] vM40

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$(50,20), Fg$(4,20):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 – SPECIAL FONTs + Showcases Three ARCADE Games

1018 Init_win:Init_Info: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 OPEN#5,scr_:WINDOW#5,482,136,gx+  18,gy+84:
1027 OPEN#6,scr_:WINDOW#6,482,136,gx+  18,gy+84:
1028 OPEN#7,scr_:WINDOW#7,482,136,gx+  18,gy+84:
1029 CSIZE#2,2,1:OVER#2,1
1030 INK#2,2:FOR i=0 TO 1:CURSOR#2,6+i,6:PRINT#2,'QBITS QLFont Editor+'
1031 INK#2,6:FOR i=0 TO 1:CURSOR#2,8+i,4:PRINT#2,'QBITS QLFont Editor+'
1032 CSIZE#2,0,0:OVER#2,0:INK#1,7
1033 CURSOR#1,280,14:PRINT#1,'(D)DIR (L)oad (S)ave (R)eset (E)xit':OVER#1,1
1034 CURSOR#1,281,14:PRINT#1,' D        L        S        R        E'    :OVER#1,0
1035 CURSOR#1,    4,28:PRINT#1,'Grp Select  ← ⬆     ⬇ →  INK(2-7)': RESTORE 1034
1036 BLOCK#1,12,3,86,32,7: FOR i=1 TO 5:READ fy,ch$:CURSOR 6,fy:PRINT ch$
1037 DATA 42,'1',64,'2',130,'3',152,'&',174,'4'
1038 END DEFine
```

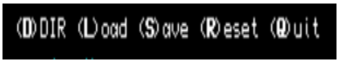


```
QBITS FontEditSE                (D)DIR (L)oad (S)ave (R)eset (E)xit
Grp Select ←↑ ↓→ INK(2-7)
1

2



3

&

4
```

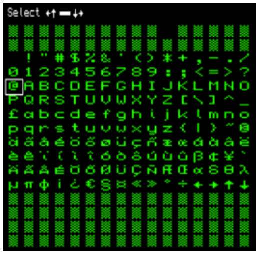
```
1040 DEFine PROCedure FontExit
1041 CURSOR#1,460,28:PRINT#1,'Y/N':PAUSE:IF KEYROW(5)<>64:RETurn
1042 RECHP FBaseA : RECHP FBase1 : RECHP FBaseB : RECHP FBase2
1043 CLOSE#4:CLOSE#3:CLS#1:CLS#0: LRUN dn$ :STOP
1044 END DEFine
```

Note: Closes open channels and releases heap memory. STOP can be replaced with an LRUN command to return to another program such as LRUN flp1_Boot or win1_Progs_Menu etc.

```
1046 DEFine PROCedure FontMain
1047 cx=0:cy=2:x=1:y=1:FontReset 0:chk=0:eck=0
1048 REPeat Main_lp
1049   cn=cx+(cy)*16:BLOCK 280,10,226,28,0:BLOCK 200,12,300,40,0:INK 7
1050   FontChar cx,cy,7:IF eck=0:KeyPad :REMArk FontPeek:FontSize
1051   K=CODE(INKEY$(-1)                Note: REMark out for BBQL and low speed platforms
1052   FontChar cx,cy,0                :REMark IF K<>32:FontGrid
1053   SELect ON K
1054   =192:cx=cx-1:IF cx< 0:cx=15:cy=cy-1:IF cy< 0:cy= 0:cx=0
1055    =200:cx=cx+1:IF cx>15:cx= 0:cy=cy+1:IF cy>15:cy=15:cx=15
1056    =208:cy=cy-1:IF cy< 1:cy= 0
1057    =216:cy=cy+1:IF cy>15:cy=15
1058    =       32:FontPeek:FontMod              :REMark Modify Char
1059    =50 to 56:fcol=K-48:FontSets              :REMark (2-7) Chart INK colour
1060    =65,97,77,109,84,116,71,103103:FontView  :REMark (A)(M)(T)(G)ames
1061    =100,  68:FontDIR                :REMark (D)DIR Set Drive/SubDIR
1062    =108,  76:FontLoad               :REMark (L)oad
1063    =115,  83:FontSave               :REMark (S)ave
1064    =114,  82:FontReset 1            :REMark (R)eset
1065    =113,  81:FontExit               :REMark (Q)uit
1066    =27:STOP                         :REMark Prog Exit
1067   END SELect
1068 END REPeat Main_lp
1069 END DEFine
```

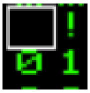


```
1071 DEFine PROCedure FontReset(ck)
1072 IF ck=1:CURSOR#1,412,28:PRINT#1,'Y/N':PAUSE:IF KEYROW(5)<>64:RETurn
1073 LBYTES Dev$&'QLFontA_fnt',FBaseA:Fg4(1)='QLFontA_fnt':Fg$(2)='QLFont1_fnt'
1074 LBYTES Dev$&'QLFontB_fnt',FBaseB:Fg$(4)='QLFontB_fnt':Fg$(3)='QLFont2_fnt'
1075 POKE FBaseA,0,127:POKE FBaseB,127,127
1076 POKE FBase1,32,96:POKE FBase2,127,64
1077 fcol=4:col=7:bcol=0:FontSets:FontGrid:cn=32
1078 END DEFine
```

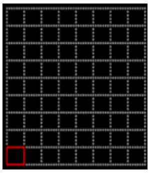


```
1080 DEFine PROCedure FontSets
1081 CHAR_USE#3,FBaseA,FBaseB:CHAR_USE#4,FBaseA,FBaseB
1082 fx=2:fy=1:CSIZE#3,3,0:INK#3,4:CLS#3:CLS#4:INK#1,7
1083 FOR c=0 TO 255
1084   CURSOR#3,fx,fy:PRINT#3,CHR$(c):fx=fx+17:IF fx>260:fx=2:fy=fy+11
1085 END  FOR c
1086 END DEFine
```

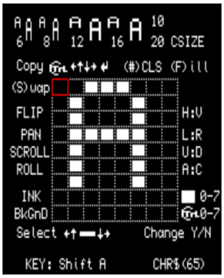
```
1088 DEFine PROCedure FontChar(cx,cy,ci)    Note: Highlights a Character within the Font Chart
1089 INK#3,4:CURSOR#3,2+cx*17,1+cy*11:PRINT#3,CHR$(cn)
1090 BLOCK#3,20,1,cx*17,cy*11,ci:BLOCK#3,20,1,cx*17,12+cy*11,ci
1091 BLOCK#3,1,12,cx*17,cy*11,ci:BLOCK#3,1,12,19+cx*17,cy*11,ci
1092 END DEFine9
```



```
1194 DEFine PROCedure FontGrid
1195 BLOCK#4,112,90,46,40,bcol
1196 FOR i=0 TO 8:BLOCK#4,1,91,40+i*14,40,248
1197 FOR i=0 TO 9:BLOCK#4,112,1,40,40+i*10,248
1198 END DEFine
```



```
1100 DEFine PROCedure FontMod
1101 QBPrnt 1,7,6,312,82,'Copy    ←↑↓→   :QBPrnt 5,7,6,374,0,',,  (F)ill (#)CLS'
1102 QBPrnt 1,7,6,312,188,'Select←↑    ↓→             Change Y/N'
1103 CSIZE#5,3,0:QBPrnt 5,7,16,326,0,'•':QBPrnt 5,7,16,353,105,'ƒ'
1104 CURSOR#5,444,92:PRINT#5,'•':CSIZE#5,0,0:FontINK:FontSize
1105 RESTORE 1106:FOR i=1 TO 13:READ a,b,c$:CURSOR a,b:PRINT c$
1106 DATA 312,112,'FLIP',316,126,'PAN',306,137,'SCROLL',312,148,'ROLL'
1107 DATA 306,96,'(S)wap',460,112,'H:V',460,126,'L:R',460,137,'U:D',460,148,'A:C'
1108 DATA 316,165,'INK',476,165,'0-7',310,176,'BkGnD',476,176,'0-7'
1109 cxo=cx:cyo=cy:FontChar cx,cy,7:KeyPad:bx=1:by=1:rm=9:cs=8:co=cn
1110 REPeat Chg_lp
1111   cn=cx+(cy)*16:FontChar cx,cy,2:FontChar cxo,cyo,7
1112   FontBit bx,by,7:K=CODE(INKEY$(-1)):FontBit bx,by,248:Fontchar cx,cy,0
1113   SELect ON K
1114    =192:bx=bx-1:IF bx<1:bx=1
1115    =200:bx=bx+1:IF bx>8:bx=8
1116    =208:by=by-1:IF by<1:by=1
1117    =216:by=by+1:IF by>9:by=9
1118    =196:cx=cx-1:IF cx< 0:cx=15:cy=cy-1:IF cy< 0:cy= 0:cx=0
1119    =204:cx=cx+1:IF cx>15:cx= 0:cy=cy+1:IF cy>15:cy=15:cx=15
1120    =212:cy=cy-1:IF cy< 1:cy= 0
1121    =220:cy=cy+1:IF cy>15:cy=15
1122    =48 TO 55: col=K-48:FontINK          :REMark 0-7 Change INK Colour
1123    =      32:BitSwap bx,by              :REMark Bit Swap 0<>1
1124    =      10:FontPeek:FontDraw          :REMark Change Font Pattern
1125    =      35:FontCLS                    :REMark (#) Reset Grid
1126    =144 TO 153:bcol=K-144:FontINK       :REMark CTRL 0-7 Change BkGnD Colour
1127    =83,115:FontSwap:EXIT Chg_lp         :REMark (S)wap Chart Fonts
1128    =104,72:FontFlip 9,-1,0,1            :REMark (H)orizontal Flip
1129    =118,86:FontFlip 0,1,10,-1           :REMark (V)ertical   Flip
1130    =108,76:FontSlid 0,8,1,0,1           :REMark (L)eft  PAN  Grid
1131    =114,82:FontSlid 0,1,8,1,0           :REMark (R)ight PAN  Grid
1132    =117,85:FontSlid 1,9,1,1             :REMark (U)p SCROLL  Grid
1133    =100,86:FontSlid 1,1,9,-1            :REMark (D)n SCROLL  Grid
1134    =  97,65:FontRoll 1                  :REMark (A) Roll Anti-Clockwise
1135    =  99,67:FontRoll 2                  :REMark (C) Roll Clockwise
1136    =102,70:FontFill                     :REMark (F)ill with Ink Colour
1137    =110,78:EXIT Chg_lp                  :REMark (N)o  Change Return
1138    =121,89:cn=co:FontPoke:EXIT Chg_lp   :REMark (Y)es Change Font
1139   END SELect
1140 END REPeat Chg_lp
1141 cn=co:cx=cxo:cy=cyo:CLS#4:FontGrid
1142 END DEFine
```



```
1144 DEFine PROCedure FontINK
1145 BLOCK 12,8,460,166,col:FOR r=0 TO 8:FontDraw
1146 END DEFine
```

Note: Display **INK** and **BkGnd** Colour

```
1148 DEFine PROCedure FontFill
1149 FOR r=0 TO 8:Fnt$(r+1)='11111111':FontDraw
1150 END DEFine
```

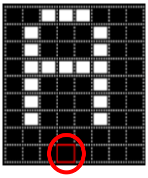
Note: Set all Matrix Bits to One's

```
1152 DEFine PROCedure FontCLS
1153 FOR r=0 TO 8:Fnt$(r+1)='00000000':FontDraw
1154 END DEFine
```

Note: Set all Matrix Bits to Zeros

```
1156 DEFine PROCedure FontDraw
1157 FOR c=0 TO 7
1158    IF Fnt$(r+1,c+1)='1':pcol=col:ELSE pcol=bcol
1159    BLOCK#4,11,7,42+c*14,42+r*10,pcol
1160 END FOR c
1161 END DEFine
```

Note: Print Pixel Colour

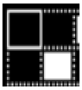


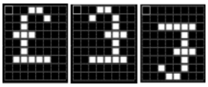
```
1163 DEFine PROCedure FontBit(x,y,hcol)
1164 BLOCK#4,15,1,32+bx*14,30+by*10,hcol:BLOCK#4,15,1,32+bx*14,40+by*10,hcol
1165 BLOCK#4,1,10,32+bx*14,30+by*10,hcol:BLOCK#4,1,10,46+bx*14,30+by*10,hcol1146
1166 END DEFine
```

Note: Highlights a Bit Square within the Font BitMap

```
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



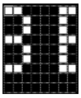


```
1174 DEFine PROCedure FontFlip(cf,cz,rf,rz)
1175 FOR r=1 TO 9:Tmp$(r)=Fnt$(r)
1176 FOR r=1 TO 9:FOR c=1 TO 8:Fnt$(r,c)=Tmp$(rf+r*rz,cf+c*cz):END FOR c:END FOR r
1177 FOR r=0 TO 8:FontDraw
1178 END DEFine
```

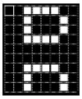
Note: Flip **H**orizontal or **V**ertical

```
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 **L**eft **R**ight

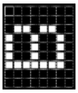


Note: Slide **U**p **D**own

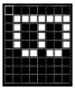


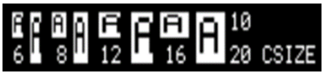
```
1189 DEFine PROCedure FontRoll(rd)
1190 FOR r=1 TO 9:Tmp$(r)=Fnt$(r)
1191 FOR r=1 TO 8
1192    IF rd=1:rx=r   :ry=8:FOR c=1 TO 8:Fnt$(ry,rx)=Tmp$(r,c):ry=ry -1:END FOR c
1193    IF rd=2:rx=9-r:ry=1:FOR c=1 TO 8:Fnt$(ry,rx)=Tmp$(r,c):ry=ry+1:END FOR c
1194 END FOR r
1195 FOR r=0 TO 8:FontDraw
1196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

Note: Rotate 90* **A**nticlockwise



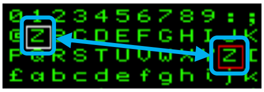
Note: Rotate 90* **C**lockwise





```
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 FontSwap
1206 FOR r=1 TO 9:Tmp$(r)=Fnt$(r):END FOR r:FontPeek:FontDraw:sn=cn:cn=co:FontPoke
1207 FOR r=1 TO 9:Fnt$(r)=Tmp$(r):END FOR r:cn=sn:FontPoke
1208 CURSOR#3,2+cx*17,1+cy*11:PRINT#3,CHR$(cn)
1209 END DEFine
```

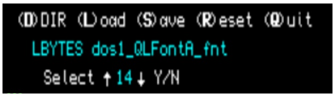


Note: Read Font Bit Pattern from RAM

```
1211 DEFine PROCedure FontPeek
1212 IF cn<128:addr=FBaseA+2+cn*9
1213 IF cn>127:addr=FBaseB+2+(cn-127)*9
1214 FOR r=0 TO 8:Fnt$(r+1)=BIN$(PEEK(addr+r),8):FontDraw
1215 END DEFine
```

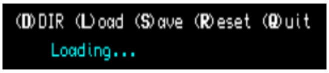
Note: Write New Font Bit Pattern to RAM

```
1217 DEFine PROCedure FontPoke
1218 IF cn<128:addr=FBaseA+2+cn*9:ELSE addr=FBaseB+2+(cn-127)*9
1219 FOR r=0 TO 8:POKE(addr+r),BIN(Fnt$(r+1))
1220 END DEFine
```

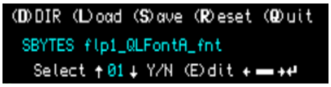


```
(D)DIR (L)oad (S)ave (R)eset (Q)uit
 LBYTES dos1_QLFontA_fnt
  Select ↑14↓ Y/N
```

```
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$(#9))
1257 CLOSE#9
1258 IF sg%< 31 AND cg%>  96:addr1=FBaseA:cy=  0:Fg$(1)=File$(sf%):Fg$(2)=Fg$(1):cn1=0
1259 IF sg%>=31 AND sg%<127:addr1=FBase1:cy=  2:Fg$(2)=File$(sf%):addr2=FBaseA:cn1=96
1260 IF sg%>126 AND cg%<  65:addr1=FBase2:cy=  8:Fg$(3)=File$(sf%):addr2=FBaseB:cn1=64
1261 IF sg%>126 AND cg%>  64:addr1=FBaseB:cy=12:Fg$(4)=File$(sf%):Fg$(3)=Fg$(4):cn1=0
1262 LBYTES drv$(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
```

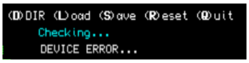


```
(D)DIR (L)oad (S)ave (R)eset (Q)uit
   Loading...
```

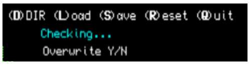


```
(D)DIR (L)oad (S)ave (R)eset (Q)uit
 SBYTES flp1_QLFontA_fnt
  Select ↑01↓ Y/N (E)dit ←—→↵
```

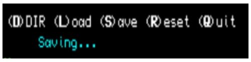
```
1271 DEFine PROCedure FontSave
1272 chk=0:eck=0:Lchk=0:sf%=1:ft%=4
1273 FOR i=1 to 4:File$(i)=Fg$(i)
1274 INK 7:CURSOR 300,42:PRINT 'Select ⬆   ⬇ Y/N  (E)dit ⬅   ➡⬅':
1275 BLOCK 12,3,454,46,7:BLOCK 2,4,480,44,7:INK 5:SelDrv 2
1276 SelFont 2,'SBYTES ':IF Lchk=0:RETurn
1277 FntList 2:BLOCK 200,10,300,42,0       :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):Fg$(sf%)=File$(sf%)
1289   END FOR a
1290 END IF
1291 SBYTES drv$(dn%)&File$(sf%),addr1,lgth
1292 END DEFine
```



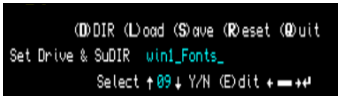
```
(D)DIR (L)oad (S)ave (R)eset (Q)uit
    Checking...
    DEVICE ERROR...
```



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

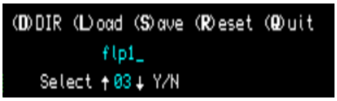


```
(D)DIR (L)oad (S)ave (R)eset (Q)uit
   Saving...
```



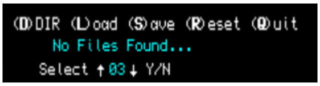
```
(D)DIR (L)oad (S)ave (R)eset (Q)uit
Set Drive & SuDIR  win1_Fonts_
        Select ↑09↓ Y/N (E)dit ←▬→←
```

```
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
```



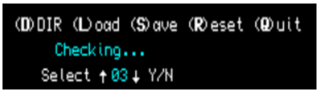
```
(D)DIR (L)oad (S)ave (R)eset (Q)uit
     flp1_
  Select ↑03↓ Y/N
```

```
1300 DEFine PROCedure SelDrv(act%)
1301 REPeat drv_lp
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_lp                Yes or No Selects Device
1310  END SELect
1311 END REPeat drv_lp
1312 END DEFine
```



```
(D)DIR (L)oad (S)ave (R)eset (Q)uit
   No Files Found...
  Select ↑03↓ Y/N
```

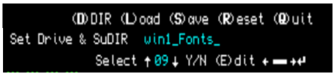
```
1314 DEFine PROCedure SelFont(act%,Act$)
1315 BLOCK 200,10,292,28,0:str$=Act$&drv$(dn%)
1316 IF ft%<1:CURSOR 310,28:PRINT 'No Files Found...':PAUSE 30:RETurn
1317 CURSOR 226,28:PRINT FILL$(' ',23-LEN(str$))&str$
1318 REPeat File_lp
1319   CURSOR 364,28:PRINT File$(sf%):CLS 4
1320   CURSOR 351,42:PRINT FILL$('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_lp              No  abort action
1327     =121,89:Lchk=1:EXIT File_lp              Yes Load / Save Font File
1328   END SELect
1329 END REPeat File_lp
1330 END DEFine
```



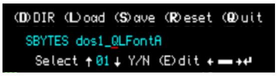
```
(D)DIR (L)oad (S)ave (R)eset (Q)uit
   Checking...
  Select ↑03↓ Y/N
```

```
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$(dn%))
1337 REPeat dir_lp
1338  IF act%=1
1339    IF EOF(#9) OR sf%>fm%:sf%=sf%-1:ft%=sf%:CLOSE#9:EXIT dir_lp
1340    INPUT#9,F$:F$=F$(dl%-4 TO):fl%=LEN(F$)         Note: Checking _nft' Files to LOAD
1341    IF fl%<=20 AND '_fnt' INSTR F$>0:File$(sf%)=F$:sf%=sf%+1
1342  END IF
1343  IF act%=2
1344    IF EOF(#9):CLOSE#9:chk=0:EXIT dir_lp
1345    INPUT#9,F$:F$=F$(dl%-4 TO)
1346    IF F$==File$(sf%):CLOSE#9:chk=1:EXIT dir_lp    Note: Checking If SAVE File Exists
1347  END IF
1348 END REPeat dir_lp
1349 END DEFine
```

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.



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

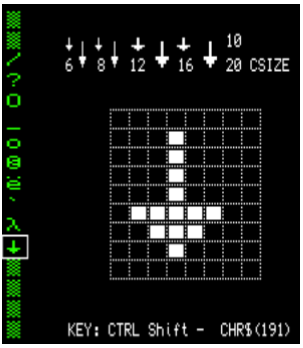


```
(D)DIR (L)oad (S)ave (R)eset (Q)uit
 SBYTES dos1_QLFontA
  Select ↑01↓ Y/N (E)dit ←▬→↵
```

```
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   Note: Restricted ASCII Codes
1359     =194    :IF cp%>1:cp%=cp%-1:Del_chr          Delete Character to left of cursor
1360     =202,32:Del_chr                              Delete Character above cursor
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$=temp$                              Note: Return with Original str$
1366 IF act%=2:str$=str$&'_fnt'                       Note: Check for '_fnt' Suffix
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

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               Note: Prints Str$ CLS to end of Cursor Line
1374 END DEFine

1376 DEFine PROCedure Ln_Cur
1377 BLOCK 6,1,x+cp%*6-6,37,2                         Note: Character Highlight Cursor Bar
1378 END DEFine

1380 DEFine PROCedure Add_chr                         Note: Checks for Adding Character to String
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

1390 DEFine PROCedure Del_chr                         Note: Checks when Deleting Character from String
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: Key(s) for Printable Characters CHR$(32 to 191)

```
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
```

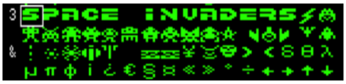


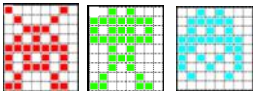
```
KEY: CTRL Shift -  CHR$(191)
```

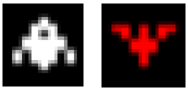
QBITS ARCADE - SPACE INVADERS

One form of Alien Attack or Space Invaders of the 1980's Retro Games Genre was based on shooting and blowing up a swarm of Aliens before they could bomb you out of existence. You PAN a Spacecraft or Gun back and forth across the bottom of the screen firing missiles to knockout the Aliens.

In deciding to code my own version of the Game so I created as a FONT Set to include Characters for a banner, various Aliens and Bitmaps designs for other parts of the Game...

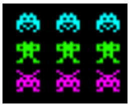






Note: Load Aliens_fnt and Press (**G**)ame

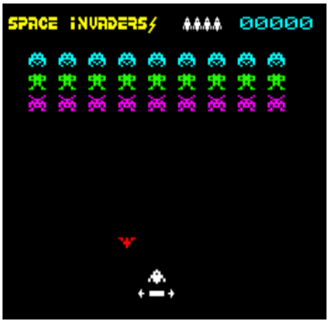
```
1502 DEFine PROCedure Space_Invader
1503 pd=2                              :REMark Game Speed pd=pause delay set (1 to 5)
1504 CSIZE#3,1,0:INK#3,6:b=0:pd=2
1505 FOR a=128 TO 142:b=b+1:CURSOR#3,b*8,8:PRINT#3,CHR$(a)    Note: Font Grp 3 Codes128 to 191
1506 CURSOR 127,200:PRINT'←     →':BLOCK 12,3,137,204,7
1507 InitLevel:DrawAliens 1:l%=5:INK#3,7
1508 REPeat Invader_lp
1509   IF l%>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:l%=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 l%:INK#3,7:CURSOR#3,74,80:PRINT#3,'LEVEL ';l%             Note l% 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)                 Note: 4 Levels of Play
1526 DATA 143,144,145,146,147,148,149,150,151,152,153,154
1527 END DEFine

1529 DEFine PROCedure DrawAliens(l%)                              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%(l%,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
```

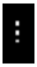


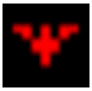
```
1537 DEFine PROCedure SLives
1538 CSIZE#3,1,0:INK#3,7
1539 CURSOR#3,148,8:PRINT#3,FILL$(CHR$(159),4-sl%)&' '
1540 CSIZE#3,3,0:sl%=sl%+1
1541 END DEFine
```

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:CURSOR#3,x,y:PRINT#3,CHR$(158):PAUSE pd
1554   INK#3,2:CURSOR#3,x,y:PRINT#3,' ':PAUSE pd
1555   IF an=n AND KEYROW(1)=64
1556     BEEP 2000,1,255,200,4,2,0,0,0
1557     INK#3,7:CURSOR#3,x,140:PRINT#3,CHR$(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,-8,15,15,15: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:CURSOR#3,n*24,150:PRINT#3,CHR$(159)
1574 END DEFine
```

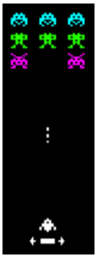
Note: Defender Ship

```
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:CURSOR#3,n*24,150-i*18:PRINT#3,CHR$(160):PAUSE pd
1580   OVER#3,0:CURSOR#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
```

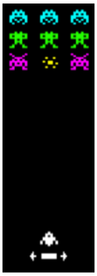
Note: Fire at Enemy

Note: Check Array Entry

Note: Explode Alien and Delete

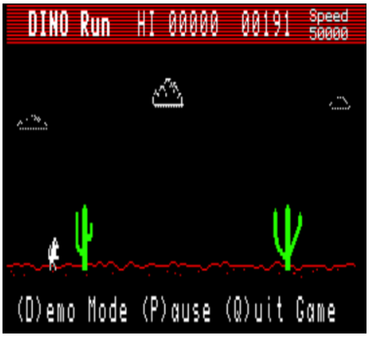


```
1588 DEFine PROCedure FExplode(x,y)
1589 FOR i=1 TO 6
1590   BEEP 3000,20,60,80,-8,15,15,15
1591   INK#3,2:CURSOR#3,x,y:PRINT#3,CHR$(160):PAUSE pd
1592   INK#3,6:CURSOR#3,x,y:PRINT#3,CHR$(161):PAUSE pd
1593 END FOR i
1594 CURSOR#3,x,y:PRINT#3,' '
1595 END DEFine
```



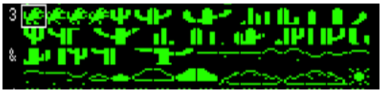
QBITS ARCADE - DINO Run

This 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, in which Caveman Thor had to avoid various hazards by jumping over pits or ducking tree branches etc.


DINO Run HI 00000 00191 Speed 50000
(D)emo Mode (P)ause (Q)uit Game

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.

1600 REMark **ARCADE – Dino Run** based on version for QL by Dilwyn Jones 2022

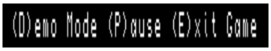


Note: Load Dino_fnt and Press (**G**)ame

```
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$(2,34)               :REMark Cactus Height 0-2 Locations 0-34 of ground steps
1617 :
1618 REPeat Dino_program
1619    REMark *** Title & Score bar ***
1620    PAPER#3,background_col:CLS#3:BLOCK#3,276,22,0,0,80
1621    OVER#3,1:CSIZE#3,1,1:STRIP#3,80:INK#3,7
1622    FOR i=0 TO 1:CURSOR#3,8+i,2:PRINT#3,' DINO Run'
1623    CURSOR#3,100,2:PRINT#3,'HI '&FILL$('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    CURSOR#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$(132)
1634    OVER#3,1:CSIZE#3,1,1:INK#3,cactus_colour%
1635    FOR a=0 TO 2:CURSOR#3,0,pany%+(18*a):PRINT#3,backg$(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  :
1675  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
```

```
1685    key=CODE(INKEY$)
1686    SELect ON key
1687       =81,113:CURSOR#3,40,158:CLS#3,3:EXIT Dino_GAME  :REMark (Q)uit
1688       =10,32                                   :REMark Manual Jump
1689          IF jumping%=0 AND demo_mode%=0
1690             jumping%= -1:dinoy%=dinobasey%:dy%=18
1691             IF sound_on%:BEEP 100,100,0,0,0,0,0,0
1692          END IF
1693      =80,112                                   :REMark (P)ause
1694          OVER#3,0:CURSOR#3,6,158:CLS#3,3:PRINT#3,'PAUSED':CLS#3,4
1695          k$=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$(128+(counter% MOD 4));
1708         FOR a=1 TO 10:BLOCK#3,10,18,32,dinoy%,7:PAUSE 5
1709         OVER#3,0:CURSOR#3,12,158:CLS#3,3
1710         rn = RND(1 TO 3)
1711         SELect ON rn
1712             =1 : PRINT#3,'Ouch!    ';
1713             =2 : PRINT#3,'Oops!    ';
1714             =3 : PRINT#3,'Oh dear! ';
1715          END SELect
1716          EXIT Dino_GAME
1717    END IF
1718   END IF
1719   IF jumping%<>0
1720      IF jumping%=-1
1721          dinoy%=dinobasey%-72*(SIN(RAD(dy%)))
1722          dy%=dy%+18:IF dy%=90:jumping%=1           :REMark change direction
1723      ELSE
1724          dinoy%=dinobasey%-72*(SIN(RAD(dy%)))       :REMark Upward
1725          dy%=dy%-18:IF dy%=0:jumping%=0:dinoy%=dinobasey%
1726      END IF
1727   ELSE
1728      IF demo_mode%=1
1729         IF backg$(2,10)<>' '
1730            jumping%=-1:dinoy%=dinobasey%:dy%=18
1731            IF sound_on% : BEEP 100,10,0,0,0,0,0,0
1732         END IF
1733      END IF
1734   END IF
```

PAUSED

MENU CHOISE

DETECT OBSTACLE

DINO JUMP

```
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 cactii
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 :
```

UPDATE CACTI

```
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
```

Note: UPDATE SCORE

```
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)
```

Note: Adjust Speed

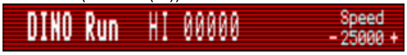
```
1785   END IF
1786 END IF
1787 :
```

```
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,cloudyy%: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_lp
1830     IF KEYROW(7)=64:EXIT Dino_program
1831     IF KEYROW(5)=64:EXIT Ans_lp
1832   END REPeat Ans_lp
1833   IF score>hiscore:hiscore=score
1834 END REPeat Dino_program
1835 CSIZE#3,0,0:CLS#3:FontSets
1836 END DEFine
```

UPDATE CLOUD

< Another Game Y/N >

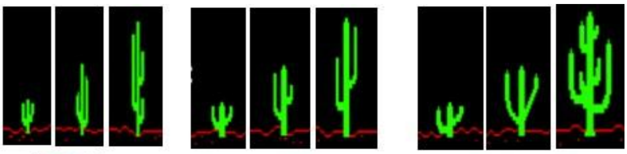
```
1838 DEFine PROCedure DSpeed
1839 CSIZE#3,0,0      :CURSOR#3,232, 2:PRINT#3,'Speed'
1840 sl=slowdown_steps:CURSOR#3,224,12:PRINT #3,'-      +'
1841 REPeat Sp_lp
1842   CURSOR#3,232,12:PRINT#3,FILL$('0',5-LEN(sl));sl:K=CODE(INKEY$(-1))
1843   IF K=43 OR K=61:IF sl<50000:sl=sl+500
1844   IF K=45 OR 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 +

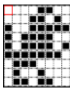
Note: Adjust with +/- then Enter

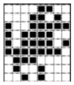
DINO_fnt uses the extended character code set 128 to 191

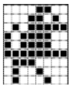
Cacti Codes 132 to 167
[3 widths 3 heights]

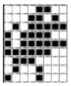


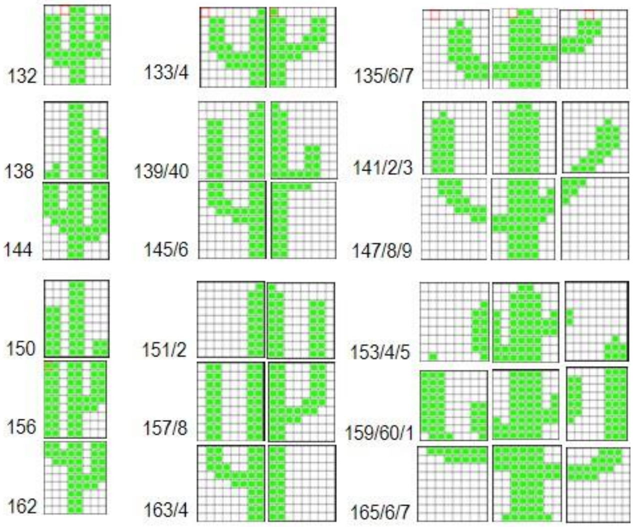
Dino Codes 128 to 131


128

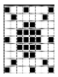

129


130

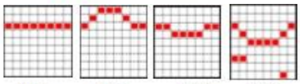

131



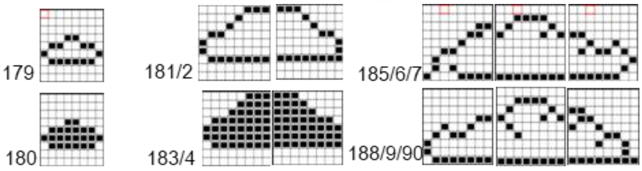
Sun Code 191



Ground Level Codes 168 to 178



Clouds Codes 179 to 190
[Three widths White/Black]

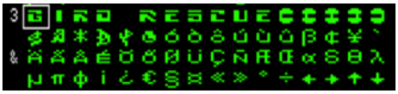


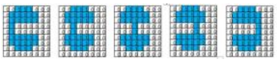
QBITS ARCADE - Giro Rescue

A Game where you manoeuvre a Rescue Ship to pick up Survivors Life Pods.

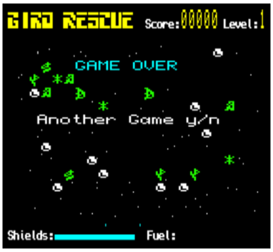
```
1850 REMark ARCADE - Giro_Rescue based on version by Henry Wrighton Aug 1989
```

Note: Load Giro_fnt and Press (**G**)ame





```
1852 DEFine PROCedure Giro_Rescue
1853 Init_Giro
1854 REPeat Giro
1855   set_level          :REMark New Game Setup
1856   REPeat Rescue
1857     tcap=tcap+1:cap=tcap:fu=90:sh=90:Giroscore sc
1858     INK#3,4:set_junk:INK#3,7:set_pod:main
1859     IF fin=1 OR tcap>15:EXIT Rescue :ELSE fin=0
1860     REPeat Bonus
1861       fuel 2:sc=sc+(2*(tcap-9)):Giroscore sc:beeps 1
1862       PAUSE 5:IF fin:EXIT Bonus
1863     END REPeat Bonus
1864   END REPeat Rescue
1865   OVER#3,0:fin=0:i=0:CURSOR#3,32,80:PRINT#3,'Another Game y/n'
1866   REPeat Key_lp
1867    i=(i+1) MOD 7:INK#3,i:AT#3,4,6:PAUSE 5:PRINT#3,'GAME OVER'
1868    IF KEYROW(5)=64 THEN EXIT Key_lp
1869    IF KEYROW(7)=64 THEN EXIT Giro
1870   END REPeat Key_lp
1871 END REPeat Giro
1872 CLS#3:CSIZE#2,0,0:FontSets
1873 END DEFine
```

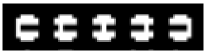


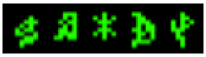
```
1875 DEFine PROCedure Giroscore(sc)
1876 CSIZE#2,1,1:CURSOR#2,200,42:PRINT#2,FILL$('0',5-LEN(sc))&sc
1877 CURSOR#2,282,42:PRINT#2,tcap-9:CSIZE#2,0,0
1878 END DEFine
```

Note: sc score
Note: tcap levels

```
1880 DEFine PROCedure set_level
1881  DIM al(36,19):BLOCK#3,276,140,0,22,0:INK#3,7
1882  sh=90:sc=0:tcap=9:pod$='œáàâë':pm=1:junk$='èêïíì'
1883  FOR i=1 TO 80:INK RND(1 TO 5):POINT#3,RND(5 TO 110),RND(12 TO 84)
1884  INK#3,4:FOR K=1 TO 12:set_junk
1885 END DEFine
```

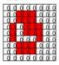
Note: pod$ CHR$(139 to 143)



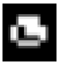


Note: junk$ CHR$(144 to 148)

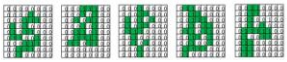
```
1887 DEFine PROCedure set_pod
1888 FOR i=1 TO tcap
1889  REPeat pp
1890    ay=RND(3 TO 14):ax=RND(2 TO 20)
1891    IF ax<>11 AND ay<>8 AND al(ax,ay)=0:EXIT pp
1892  END REPeat pp
1893  al(ax,ay)=1:AT#3,ay,ax:PRINT#3,"•":beeps 5
1894 END FOR i
1895 END DEFine
```



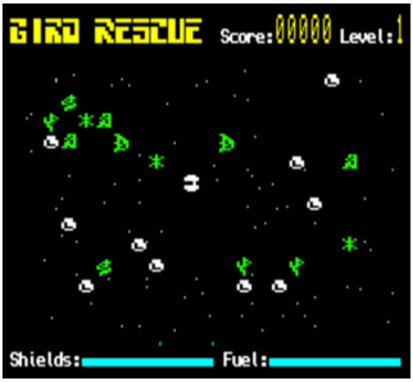
Note: Escape Pod CHR$(149)



```
1897 DEFine PROCedure set_junk
1898  REPeat junk
1899    x=RND(2 TO 20):y=RND(3 TO 14):IF x<>10 AND y<>8:EXIT junk
1900  END REPeat junk
1901  al(x,y)=2:AT#3,y,x:PRINT#3,junk$(RND(1 TO 5)):beeps 6
1902 END DEFine
```

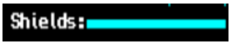


```
1904 DEFine PROCedure main
1905 yy=0:xx=0:x=120:y=80:ix=2:iy=1:fu=90:fin=0:pm=1
1906 OVER#3,0:BLOCK#3,90,4,52,170,5:BLOCK#3,90,4,180,170,5
1907 INK#3,7:CURSOR#3,x,y:PRINT#3,pod$(pm):OVER#3,-1
1908 REPeat loop
1909   get_keys:PAUSE 5
1910   IF x+xx< 0 OR x+xx>260:xx=-xx    :beeps 2
1911   IF y+yy<20 OR y+yy>150:yy=-yy   :beeps 2
1912   IF al((x)/12,(y)/10)=1:pod_clear   :beeps 3
1913   IF al((x)/12,(y)/10)=2:shield 1       :beeps 4
1914   CURSOR#3,x,y:PRINT#3,pod$(pm):IF fin :PAUSE 30:EXIT loop
1915   x=x+xx:y=y+yy:pm=(pm MOD 5)+1:CURSOR#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,y/10,x/12:PRINT#3,'•':sc=sc+10
1938  Giroscore sc,l:cap=cap-1:IF cap=0:fin=2
1939 END DEFine
```

Note: CHR$(149)



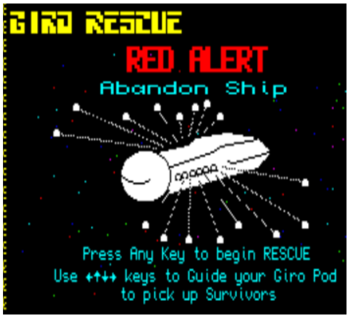
```
1941 DEFine PROCedure fuel(f)
1942 fu=fu-f:BLOCK#2,f,4,196+fu,211,0:IF fu<=0:fin=1
1943 END DEFine

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:CURSOR#3, 1+i, 1:PRINT#3,'GIRO RESCUE'
1964 INK#3,2:FOR i=0 TO 2:CURSOR#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 Ship'
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$:FOR b=0 TO 1:CURSOR#3,x+b,y:PRINT#3,str$: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:CURSOR#3,0,0:CSIZE#3,2,0
1977 END DEFine
```

Note: CHR$(128 to138)
äãåéöõøüçñæ

```
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


GIRO RESCUE
RED ALERT
Abandon Ship
Press Any Key to begin RESCUE
Use ←↑↓→ keys to Guide your Giro Pod
to pick up Survivors

```
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,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,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,-23,8,18
2001 DATA 34,-18,15,15, 46,-21, 20,18, 55,-18,24,19, 55,-5,28,15
2002 END DEFine
```