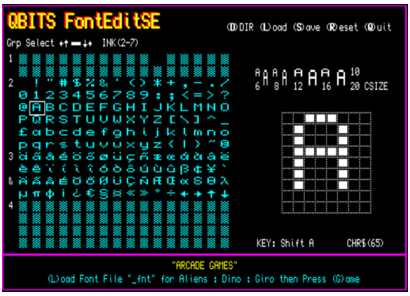


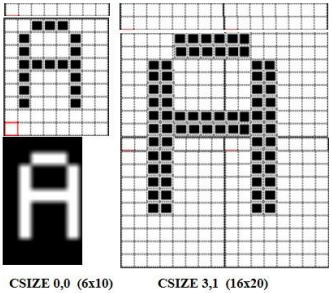
Introduction

Home Computers in the 1980's created Characters from Bitmaps. The most common were 8x8 matrixes covering the Western ASCII code set. A BITMap Files have a header that identifies type, file size, the matrix width and height followed by bytes set with the bit patterns. For Font _fnt files this will include The Start and Number of individual Font Bitmaps. A Computer App processes the Font Bitmap to create a pixel display on screen at an x,y coordinate.

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. A Font header has the first two Bytes hold the Lowest 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 to scale in two widths 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 FontEditor _fnt Files
The QL Character BITMaps can be extended to include CHR$(0 to 31) and (192 to 255). QBITS_FontEdit2 arranges Codes 0 to 255 into four groups: (1) **QLFontA_fnt** (0/128), (2) **QLFont1_fnt** (32/96), (3) **QLFont2_fnt** (128/64), (4) **QLFontB_fnt** (128,128). At start-up **QLFontA** & **QLFontB '_fnt'** files are loaded from the default Storage Device into reserved memory and a Font Chart of CHR$(0 to 255) are displayed to screen.

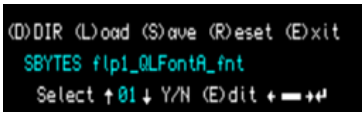
QBITS FontEditor - Navigation
Navigate the Font Chart using the Cursor keys and Select the Highlighted Character with Spacebar. The Bitmap Grid is now active and Navigate with Cursor Keys to highlight a Grid Cell. Use Spacebar to switch between INK and STRIP (background) colour by setting binary bit between 0 & 1. Select 'N' to return without changing the existing Bitmap, 'Y' to write the changed bit pattern into memory.

QBITS (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 selection of 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.

For Load / Save: First Select Storage Device with Up/Down Cursor keys then either 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 or 'N' to abort. The '_fnt' file is checked for Character group to select Memory Address, before File is loaded and Fonts displayed to screen.



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

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

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

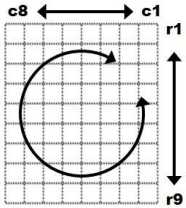
QBITS FontEditor

The heart of the Editor is the Bitmap and what New or Altered Characters Fonts can be created. These may reflect Bold, Italics, or personal stylised alphanumerical Fonts, futuristic designs, or an ancient language such as Cuneiform or Runes. For Retro Games transforming Fonts into game pieces might also be desirable.

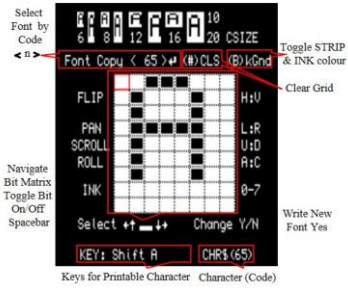
The QBITS **Bitmap** shows bits set to 0's & 1's for chosen Font. The Font under review is also displayed in its different **CSIZES** with **KEY:** showing the key(s) needed for printing and **CHR$(n)** 'n' being the ASCII Character Code.

Where slight pattern changes might be desired, copying an existing Font Character into the Font Grid could be useful. To **COPY** Font use Keys **'<>'** to select and ↵ Enter. For new designs (**#**) CLS clears the Bitmap settings all to '0's. (**B**)kGnd toggles STRIP & INK colours between Black(**0**) and White(**7**). For further actions Flip (**H**)orizontally, or (**V**)ertically, Pan (**L**)eft or (**R**)ight, Scroll (**U**p) & (**D**)own, Rotate (**A**)nti-clockwise or (**C**)lockwise by 90°. To change Font INK colour use (**0-7**). To Navigate the Bitmap use Cursor keys ← ↑ ↓ → and Toggle On/Off Bite INK colour with ▬ Spacebar.

Showing the Fonts in their various CSIZES with different colours can be helpful to identifying possible display problems.

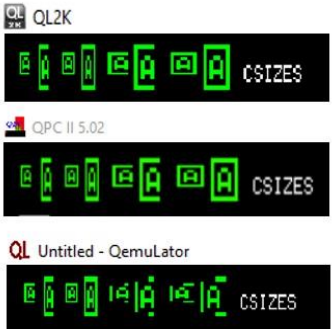


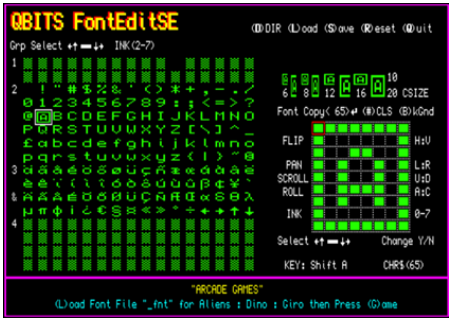
Flip	c1=c8 to c8=c1	**H**
	r1=r9 to r9=r1	**V**
Pan	c1=c8 to c2=c1 c8=c7	**L**
	c1=c2 to c7=c8 c8=c1	**R**
Scroll	r1=r2 to r8=r9 r9=r1	**U**
	r2=r1 to r9=r8 r1=r9	**D**
Roll	r1,c1=r1,c8 r9,c1=r1,c1	**A**
	r1,c1=r9,c1 r9,c8=r1,c8	**C**



QL Fonts & CSIZEs

The bit Pattern 'A' is surrounded with a box. Not all CSIZES show the whole of the box and some QL O/S displays are very different to that expected.





As mentioned before, the Font App across various QL platforms does not retrospectively produce all of the bit pattern held by a character's Bitmap across the range of CSIZEs.

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.

ARCADE CLASSIC

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 a shining example of how video game designers used an 8-by-8 Bitmap Grid to communicate to players.

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

Note: Retro Game Control

A useful SBASIC keyword is **CHAR_INC**#ch, x_inc, y_inc which sets Character width and hight beyond those set by CSIZE. Where the subject Character Font(s) are moved across the screen, an independent height can be set for **PAN.**

Set Font hight **CHAR_INC#ch,40,8** at 40 Pixel rows high and 8 Pixel width then **PAN** #ch, +/- distance, 3 (PAN's the Cursor line Left or Right).

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 Jobs such as the SuperBASIC Interpreter. QBITS_FontEdit2 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 the original BBQL available memory.

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 FontEditor SE

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 game should now be displayed in WINDOW#3.

Space Invaders

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

Dino Run

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

Giro Rescue

The aim rescue all the Escape Pods scattered through space, watch out for space 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 Font Edit Special Edition Code

This Program requires the following files to be available **QLFontA_fnt** & **QLFontB_fnt,** these are the two default QL Font Character Groups. **Aliens_fnt**, **Dino_fnt** and **Giro_fnt** contain the Character Bitmaps used by the Program in the Demo ARCADE Games.

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

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

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

1004 WHEN ERRor :eck=1:CONTINUE:END WHEN

1006 REMark Import QBITSConfig Settings - QPC2
1007 OPEN_IN#9,'ram2_QBITSConfig':INPUT#9,gx\gy\dn$\dev$\dn%\dm%
1008 DIM drv$(15,16):FOR d=0 TO dm%:INPUT#9,Drv$(d):END FOR d:CLOSE#9

1010 REMark Set Font Arrays
1011 DIM Fnt$(9,8),Tmp$(9,8), File$(50,20):fm%=50          Note: fm% (size may require extra  memory).

1013 REMark RAM Allocation
1014 FBaseA=ALCHP(1164):FBaseB=ALCHP(1164):FBase1=ALCHP(876):FBase2=ALCHP(588)

1016 REMark QBITS Special Edition – Showcases Three ARCADE Games

1018 Init_win:ARCADE:FontReset 0:FontMain

1020 DEFine PROCedure Init_win
1021 WINDOW#0,512, 32,gx,gy+224      :PAPER#0,0   :CSIZE#0,0,0:BORDER#0,1,3:CLS#0
1022 WINDOW#1,512,224,gx,gy          :PAPER#1,0   :CSIZE#1,0,0:BORDER#1,1,3:CLS#1
1023 WINDOW#2,512,224,gx,gy          :PAPER#2,0   :CSIZE#2,0,0:BORDER#2,1,3:CLS#2
1024 OPEN#3,scr_:WINDOW#3,276,178,gx+ 18,gy+42 :CSIZE#3,3,0:INK#3,4
1025 OPEN#4,scr_:WINDOW#4,192,146,gx+308,gy+56:CSIZE#4,3,0:INK#4,4
1026 CSIZE#2,2,1:OVER#2,1
1027 INK#2,2:FOR i=0 TO 1:CURSOR#2,6+i,6:PRINT#2,'QBITS QLFont Editor2°'
1028 INK#2,6:FOR i=0 TO 1:CURSOR#2,8+i,4:PRINT#2,'QBITS QLFont Editor2°'
1029 CSIZE#2,0,0:OVER#2,0:INK#1,7
1030 CURSOR#1,280,14:PRINT#1,'(D)DIR (L)oad (S)ave (R)eset (Quit':OVER#1,1
1031 CURSOR#1,281,14:PRINT#1,' D       L       S       R       Q'   :OVER#1,0
1032 CURSOR#1,   4,28:PRINT#1,'Grp Select ⬅ ⬆   ⬇ ➡ INK(2-7)': RESTORE 1034
1033 BLOCK#1,12,3,86,32,7: FOR i=1 TO 5:READ fy,ch$:CURSOR 6,fy:PRINT ch$
1034 DATA 42,'1',64,'2',130,'3',152,'&',174,'4'
1035 END DEFine
```

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


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


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

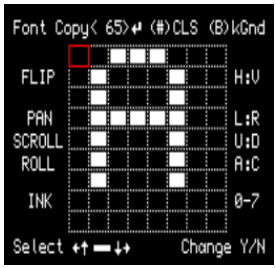
```
1078 DEFine PROCedure FontChar(cx,cy,ci)
1079 INK#3,4:CURSOR#3,2+cx*17,1+cy*11:PRINT#3,CHR$(cn)
1080 BLOCK#3,20,1,cx*17,cy*11,ci:BLOCK#3,20,1,cx*17,12+cy*11,ci
1081 BLOCK#3,1,12,cx*17,cy*11,ci:BLOCK#3,1,12,19+cx*17,cy*11,ci
1082 END DEFine9
```

Note: Highlights a Character within the Font Chart

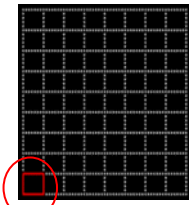
```
1084 DEFine PROCedure FontQuit
1085 CURSOR#1,460,28:PRINT#1,'Y/N':PAUSE:IF KEYROW(5)<>64:RETurn
1086 RECHP FBaseA : RECHP FBase1 : RECHP FBaseB : RECHP FBase2
1087 CLOSE#4:CLOSE#3:CLS#1:CLS#0:: LRUN dn$ :STOP
1088 END DEFine
```

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

```
1100 DEFine PROCedure FontMod
1101 CURSOR#1,312,  82:PRINT#1,'Font Copy <   >← (#)CLS (B)kGnd'
1102 CURSOR#1,312,188:PRINT#1,'Select ←↑  ↓→      Change Y/N'
1103 BLOCK 2,4,402,84,7:BLOCK 12,3,370,192,7
1104 RESTORE 1105:FOR i=1 TO 10:READ a,b,c$:CURSOR a,b:PRINT c$
1105 DATA 318,106,'FLIP',322,126,'PAN',312,137,'SCROLL',318,148,'ROLL'
1106 DATA 470,106,'H:V',470,126,'L:R',470,137,'U:D',470,148,'A:C'
1107 DATA 322,166,'INK',470,166,'0-7'
1108 FontChar cx,cy,7:FontSize:KeyPad:bx=1:by=1:rm=9:cs=8:co=cn
1109 REPeat Chg_Ip
1110   CURSOR 372,82:PRINT FILL$(' ',3-LEN(cn))&cn
1111   FontBit bx,by,7:K=CODE(INKEY$(-1)):FontBit bx,by,248
1112   SELect ON K
1113     =192:bx=bx-1:IF bx<1:bx=1
1114     =200:bx=bx+1:IF bx>8:bx=8
1115     =208:by=by-1:IF by<1:by=1
1116     =216:by=by+1:IF by>9:by=9
1117     =48 TO 55:col=K-48:FOR r=0 TO 8:FontDraw  :REMark Colour change
1118     =      32:BitSwap bx,by                   :REMark Bit Swap 0<>1
1119     =  60,44:cn=cn-1:IF cn<0:cn=0             :REMark <, Lower cn
1120     =  62,46:cn=cn+1:IF cn>255:cn=255         :REMark >. higher cn
1121     =      10:FontPeek:FontDraw               :REMark Change Font Pattern
1122     =      35:FontNew                         :REMark (#) Reset Grid
1123     =  66,98:FontBkGnd                        :REMark (B)kGnd Colour
1124     =104,72:FontFlip 9,-1,0,1                 :REMark (H)orizontal Flip
1125     =118,86:FontFlip 0,1,10,-1                :REMark (V)ertical  Flip
1126     =108,76:FontSlid 0,8,1,0,1                :REMark (L)eft  PAN  Grid
1127     =114,82:FontSlid 0,1,8,1,0                :REMark (R)ight PAN  Grid
1128     =117,85:FontSlid 1,9,1,1                  :REMark (U)p SCROLL  Grid
1129     =100,86:FontSlid 1,1,9,-1                 :REMark (D)n SCROLL  Grid
1130     =  97,65:FontRoll 1                       :REMark (A) Roll Anti-Clockwise
1131     =  99,67:FontRoll 2                       :REMark (C) Roll Clockwise
1132     =110,78:EXIT Chg_Ip                       :REMark (N)o  Change Return
1133     =121,89:cn=co:FontPoke:EXIT Chg_Ip        :REMark (Y)es Change Font
1134   END SELect
1135 END REPeat Chg_Ip
1136 cn=co:CLS#4:FontGrid
1137 END DEFine

1139 DEFine PROCedure FontGrid
1140 BLOCK#4,112,90,46,40,bcol
1141 FOR i=0 TO 8:BLOCK#4,1,91,40+i*14,40,248
1142 FOR i=0 TO 9:BLOCK#4,112,1,40,40+i*10,248
1143 END DEFine

1145 DEFine PROCedure FontBit(x,y,hcol)
1146 BLOCK#4,15,1,32+bx*14,30+by*10,hcol:BLOCK#4,15,1,32+bx*14,40+by*10,hcol
1147 BLOCK#4,1,10,32+bx*14,30+by*10,hcol:BLOCK#4,1,10,46+bx*14,30+by*10,hcol1146
1148 END DEFine

1150 DEFine PROCedure FontBkGnd
1151 IF bcol=0:bcol=7:col=0:ELSE bcol=0:col=7
1152 FontGrid:FOR r=0 TO 8:FontDraw
1153 END DEFine
```



Font character number
n=0 to 255

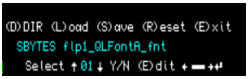


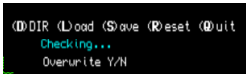
Note: Highlights a Bit Square within the Font BitMap

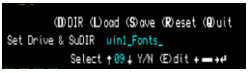
Note: Toggle Black/White STRIP & White/Black INK

```
1155 DEFine PROCedure FontNew
1156 FOR r=0 TO 8:Fnt$(r+1)='00000000':FontDraw                    Note: Set all Matrix Bits to Zeros
1157 END DEFine

1159 DEFine PROCedure FontDraw
1160 FOR c=0 TO 7
1161    IF Fnt$(r+1,c+1)='1':pcol=col:ELSE pcol=bcol
1162    BLOCK#4,11,7,42+c*14,42+r*10,pcol                          Note: Print Pixel colour
1163 END FOR c
1164 END DEFine

1166 DEFine PROCedure BitSwap(bx,by)
1167 IF Fnt$(by,bx)='0':Fnt$(by,bx)='1':ELSE Fnt$(by,bx)='0'
1168 IF Fnt$(by,bx)='0':BLOCK#4,11,7,42+(bx-1)*14,42+(by-1)*10,bcol   Note: STRIP
1169 IF Fnt$(by,bx)='1':BLOCK#4,11,7,42+(bx-1)*14,42+(by-1)*10,col    Note: INK
1170 END DEFine

1172 DEFine PROCedure FontFlip(cf,cz,rf,rz)
1173 FOR r=1 TO 9:Tmp$(r)=Fnt$(r)
1174 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
1175 FOR r=0 TO 8:FontDraw
1176 END DEFine                                                    Note: Flip Horizontal or Vertical

1178 DEFine PROCedure FontSlid(md,a,b,d,e)
1179 FOR r=1 TO 9:Tmp$(r)=Fnt$(r)):rm=9:IF md=1 AND a=9:rm=8       Note: Slide Left  Right
1180 FOR r=1 TO rm
1181    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
1182    IF md=1:Fnt$(a)=Tmp$(b)    :FOR c=1 TO 8:Fnt$(r,c)=Tmp$(r+d,c)  :END FOR c
1183 END FOR r
1184 FOR r=0 TO 8:FontDraw                                        Note: Slide Up Down
1185 END DEFine

1187 DEFine PROCedure FontRoll(rd)
1188 FOR r=1 TO 9:Tmp$(r)=Fnt$(r)                                 Note: Rotate 90* Anticlockwise
1189 FOR r=1 TO 8
1190    IF rd=1:rx=r   :ry=8:FOR c=1 TO 8:Fnt$(ry,rx)=Tmp$(r,c):ry=ry -1:END FOR c
1191    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
1192 END FOR r
1193 FOR r=0 TO 8:FontDraw
1194 END DEFine                                                    Note: Rotate 90* Clockwise

1196 DEFine PROCedure FontSize
1198 CURSOR 318,67:PRINT '6  8  12   16   20 CSIZE'
1197 CURSOR 438,54:PRINT '10':RESTORE 1199:STRIP#4,bcol:INK#4,col
1199 FOR i=1 TO 8:READ a,b,c,d:CSIZE#4,a,b:CURSOR#4,c,d:PRINT#4,CHR$(cn)
1200 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
1201 END DEFine

1203 DEFine PROCedure FontPeek                                    Note: Read Font Bit Pattern from RAM
1204 IF cn<128:addr=FBaseA+2+cn*9
1205 IF cn>127:addr=FBaseB+2+(cn-127)*9
1206 FOR r=0 TO 8:Fnt$(r+1)=BIN$(PEEK(addr+r),8):FontDraw
1207 END DEFine
```

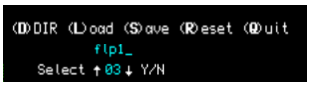
```
1209 DEFine PROCedure FontPoke                    Note: Write New Font Bit Pattern to RAM
1210 IF cn<128:addr=FBaseA+2+cn*9:ELSE addr=FBaseB+2+(cn-127)*9
1211 FOR r=0 TO 8:POKE(addr+r),BIN(Fnt$(r+1))
1212 END DEFine

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$(#9))
1257 CLOSE#9
1258 IF sg%< 31 AND cg%>   96:addr1=FBaseA:cy= 0:cn1$=File$(sf%):cn2$=cn1$:cn1=0
1259 IF sg%>=31 AND sg%<127:addr1=FBase1:cy= 2:cn2$=File$(sf%):addr2=FBaseA:cn1=96
1260 IF sg%>126 AND cg%<  65:addr1=FBase2:cy= 8:cn3$=File$(sf%):addr2=FBaseB:cn1=64
1261 IF sg%>126 AND cg%>  64:addr1=FBaseB:cy=12:cn4$=File$(sf%):cn3$=cn4$:cn1=0
1262 LBYTES drv$(dn%)&File$(sf%),addr1:cx=0:IF cn1=96:os=sg%:ELSE os=sg%-127
1263 IF cn1=96 OR cn1=64
1264    FOR a=1 TO cn1
1265      FOR b=0 TO 8:POKE addr2+2+(a+os)*9+b,PEEK(addr1+2+a*9+b)
1266    END FOR a
1267 END IF
1268 FontSets:eck=0:INK 7:FontGrid
1269 END DEFine

1271 DEFine PROCedure FontSave
1272 chk=0:eck=0:Lchk=0:sf%=1:ft%=4
1273 File$(1)=cn1$:File$(2)=cn2$:File$(3)=cn3$:File$(4)=cn4$
1274 INK 7:CURSOR 300,42:PRINT 'Select ⬆  ⬇ Y/N  (E)dit ⬅  ➡⬅':
1275 BLOCK 12,3,454,46,7:BLOCK 2,4,480,44,7:INK 5:SelDrv 2
1276 SelFont 2,'SBYTES ':IF Lchk=0:RETurn
1277 FntList 2:BLOCK 200,10,300,42,0       :CURSOR 312,42:INK 7
1278 IF eck=1:PRINT#1,'DEVICE ERROR...':PAUSE 50:RETurn
1279 IF chk=1:PRINT#1,'Overwrite Y/N'  :PAUSE:IF KEYROW(5)<>64:RETurn
1280 INK 5:DELETE drv$(dn%)&File$(sf%) :CURSOR 226,28:CLS 4
1281 CURSOR 310,28:PRINT#1,'Saving...'  :CURSOR 300,42:CLS 4:PAUSE 30
1282 IF sf%=1:addr1=FBaseA:lgth=1154:cn1=127
1283 IF sf%=2:addr1=FBase1:lgth=876 :cn1= 96:addr2=FBaseA:os=31
1284 IF sf%=3:addr1=FBase2:lgth=588 :cn1= 64:addr2=FBaseB:os=  0
1285 IF sf%=4:addr1=FBaseB:lgth=1154:cn1=127
1286 IF cn1=96 OR cn1=64
1287   FOR a=1 TO cn1
1288     FOR b=0 TO 8:POKE addr1+2+a*9+b,PEEK(addr2+2+(a+os)*9+b)
1289   END FOR a
1290 END IF
1291 SBYTES drv$(dn%)&File$(sf%),addr1,lgth
1292 END DEFine

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
```







```
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
```

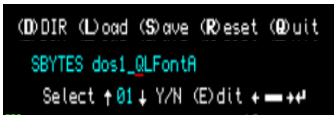



```
1314 DEFine PROCedure SelFont(act%,Act$)
1315 BLOCK 200,10,292,28,0:str$=Act$&drv$(dn%)
1316 IF ft%<1:CURSOR 310,28:PRINT 'No Files Found...':PAUSE 30:RETurn
1317 CURSOR 226,28:PRINT FILL$(' ',23-LEN(str$))&str$
1318 REPeat File_lp
1319   CURSOR 364,28:PRINT File$(sf%):CLS 4
1320   CURSOR 351,42:PRINT FILL$('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
```


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

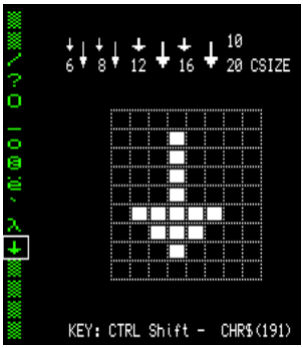
Note This QBITS Editor Restricts ASCII Code Characters used for QL Filenames. Position the underscore with left/Right cursors to a character then **Add** a new or **Delete** the existing character. Adding a Character expands the string to the right, Deleting shrinks the string from right to left. The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



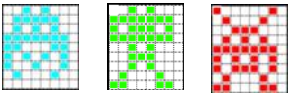

```
1351 DEFine PROCedure EditName(act%,x,sm%,str$)
1352 IF act%=2:sl%=('_fnt' INSTR str$)-1:str$=str$(1 TO sl%)
1353 temp$=str$:sl%=LEN(str$):cp%=1
1354 REPeat Ed_lp
1355   Ln_Prn:Ln_Cur:k$=INKEY$(#0,-1):K=CODE(k$)
1356   SELect ON K
1357     = 10:EXIT Ed_lp
1358     = 48 TO 57, 65 TO 90,95, 97 TO 122:Add_chr     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
```

```
1400 DEFine PROCedure KeyPad
1401 SELect ON cn=32  TO 93    :Key$=CHR$(cn)
1402 SELect ON cn=32           :Key$='Spacebar'
1403 SELect ON cn=33,34        :Key$='Shift '&(cn-32)
1404 SELect ON cn=36,37        :Key$='Shift '&(cn-32)
1405 SELect ON cn=38           :Key$='Shift 7'
1406 SELect ON cn=40           :Key$='Shift 9'
1407 SELect ON cn=41           :Key$='Shift 0'
1408 SELect ON cn=42           :Key$='Shift 8'
1409 SELect ON cn=43           :Key$='Shift ='
1410 SELect ON cn=58           :Key$='Shift ;'
1411 SELect ON cn=60           :Key$='Shift ,'
1412 SELect ON cn=62           :Key$='Shift .'
1413 SELect ON cn=63           :Key$='Shift /'
1414 SELect ON cn=64           :Key$="Shift '"
1415 SELect ON cn=65  TO  90   :Key$='Shift '&CHR$(cn)
1416 SELect ON cn=94           :Key$="Shift 6"
1417 SELect ON cn=95           :Key$="Shift -"
1418 SELect ON cn=96           :Key$="Shift 3"
1419 SELect ON cn=97  TO 122   :Key$=CHR$(cn-32)
1420 SELect ON cn=123          :Key$='Shift ['
1421 SELect ON cn=124          :Key$='Shift \'
1422 SELect ON cn=125          :Key$='Shift ]'
1423 SELect ON cn=126          :Key$='Shift #'
1424 SELect ON cn=127          :Key$='Shift Esc'
1425 SELect ON cn=128          :Key$='CTRL Esc'
1426 SELect ON cn=129          :Key$='CTRL Shift 1'
1427 SELect ON cn=130          :Key$="CTRL Shift '"
1428 SELect ON cn=131 TO 133   :Key$='CTRL Shift '&CHR$(cn-80)
1429 SELect ON cn=134          :Key$='CTRL Shift 7'
1430 SELect ON cn=135          :Key$="CTRL '"
1431 SELect ON cn=136          :Key$='CTRL Shift 9'
1432 SELect ON cn=137          :Key$='CTRL Shift 0'
1433 SELect ON cn=138          :Key$='CTRL Shift 8'
1434 SELect ON cn=139          :Key$='CTRL Shift ='
1435 SELect ON cn=140 TO 153   :Key$='CTRL '&CHR$(cn-96)
1436 SELect ON cn=154          :Key$='CTRL Shift ;'
1437 SELect ON cn=155          :Key$='CTRL ;'
1438SELect ON cn=156           :Key$='CTRL Shift ,'
1439 SELect ON cn=157          :Key$='CTRL ='
1440 SELect ON cn=158          :Key$='CTRL Shift .'
1441 SELect ON cn=159          :Key$='CTRL Shift /'
1442 SELect ON cn=160          :Key$='CTRL Shift 2'
1443 SELect ON cn=161 TO 186   :Key$='CTRL Shift '&CHR$(cn-96)
1444 SELect ON cn=187          :Key$='CTRL ['
1445 SELect ON cn=188          :Key$='CTRL \'
1446 SELect ON cn=189          :Key$='CTRL ]'
1447SELect ON cn=190           :Key$='CTRL Shift 6'
1448 SELect ON cn=191          :Key$='CTRL Shift -'
1449 SELect ON cn=0 TO 31,192 TO 255 :Key$=''
1450 CURSOR 320,208:PRINT 'KEY: '&Key$;FILL$(' ',14-LEN(Key$));'CHR$(';cn;')  '
1451 END DEFine
```

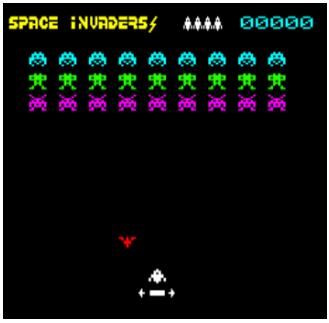
Note: Identify Key(s) for Printable Characters



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

1455 DEFine PROCedure ARCADE
1456 INK#0,6:CURSOR#0,212,4::PRINT#0,'"ARCADE GAMES"':INK#0,5:CURSOR#0,54,16
1457 PRINT#0,'(L)oad Font File "_fnt" for Aliens : Dino : Giro then Press (G)ame'
1458 END DEFine

1460 DEFine PROCedure FontGame                    Note: Load _fnt File and Press (G)ame
1461 IF cn3$=='Aliens_fnt':CLS#3:Space_Invader
1462 IF cn3$=='Dino_fnt'   :CLS#3:Dino_Run
1463 IF cn3$=='Giro_fnt'   :CLS#3:Giro_Rescue
1464 END DEFine

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

1502 DEFine PROCedure Space_Invader
1503 pd=2                          :REMark Game Speed pd=pause delay set (1 to 5)
1504 CSIZE#3,1,0:INK#3,6:b=0:pd=2
1505 FOR a=128 TO 142:b=b+1:CURSOR#3,b*8,8:PRINT#3,CHR$(a)    Note: Font Grp 3 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                   Note: 4 Levels of Play
1524 DIM GL%(4,3),GA%(3,9):RESTORE 1526
1525 FOR a=1 TO 4:READ GL%(a,1),GL%(a,2),GL%(a,3)
1526 DATA 143,144,145,146,147,148,149,150,151,152,153,154
1527 END DEFine

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                        Note: Ship Lives/Level
1541 END DEFine
```



```
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)   Note: Defender Ship
1574 END DEFine
```

```
1576 DEFine PROCedure FShip
1577 BEEP 2000,1,255,200,4,2,0,0,0
1578 FOR i=1 TO 5                                    Note: Fire at Enemy
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                   Note: Check Array Entry
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   Note: Explode Alien and Delete
1586 END DEFine
```

```
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
```

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

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

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

```
1602 DEFine PROCedure Dino_Run
1603 dino_colour%      = 7
1604 cactus_colour%    = 4
1605 ground_colour%    = 2
1606 cloud_colour%     = 255
1607 background_col    = 0
1608 slowdown_steps    = 25000     :REMark 1/n Game Delay [?Dependant on Processor Speed]
1609 pany%             = 90        :REMark Top Left of Main PAN Block
1610 byPerCent         = 1         :REMark Percentage change of Slowdown delays
1611 every%           = 1000      :REMark every this much of score
1612 sound_on%        = 1         :REMark 0=sound off 1=sound on
1613 demo_mode%       = 0         :REMark auto-jumping
1614 hiscore=score    = 0 :dy%=18:sl=5000
1615 :
1616 DIM backg$(2,34)              :REMark Cactus Height 0-2 Locations 0-34 of ground steps
1617 :
1618 REPeat Dino_program
1619    REMark *** Title & Score bar ***
1620    PAPER#3,background_col:CLS#3:BLOCK#3,276,22,0,0,80
1621    OVER#3,1:CSIZE#3,1,1:STRIP#3,80:INK#3,7
1622    FOR i=0 TO 1:CURSOR#3,8+i,2:PRINT#3,' DINO Run'
1623    CURSOR#3,100,2:PRINT#3,'HI '&FILL$('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
```

```
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
```

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  :
1777  REMark *** Handle the score IF >99999 wrap around
1778  STRIP#3,80:INK#3,7:score=score+1:IF score>99999:score=0
1779  CURSOR#3,180,2:PRINT#3,FILL$('0',5-LEN(score))&score:STRIP#3,0
1780  :
1781  REMark *** Speed up Slightly as Score gets Higher (if set to do so)
1782  IF every%>0
1783    IF (score MOD every%)=0
1784      slowdown=slowdown-(slowdown*byPerCent/100)
1785    END IF
1786  END IF
1787  :
```

UPDATE CACTI

Note: UPDATE SCORE

Note: Adjust Speed

```
1788   REMark *** Increment Dino Frame Counter
1789   counter%=counter%+1:IF counter% >4:counter%=0
1790   :
1791   REMark *** Handle Clouds Once Every 4 Frames
1792   IF (counter% MOD 4)=0
1793     CURSOR#3,0,cloudy%:CHAR_INC#3,8,36:PAN#3,-8,3:CHAR_INC#3,8,18
1794     cloud_gap%=cloud_gap%-1
1795     IF cloud_gap%=0
1796       cloud_run% =1          :REMark how far across
1797       cloud_wide =RND(2 TO 3) :REMark cloud width 1-3
1798       cloud_high =RND(0 TO 1) :REMark cloud height 0-1
1799       cloudyy%   =cloudy% + RND(0 TO 26-(9*cloud_high))
1800       rn=RND(1 TO 5)         :REMark Cloud - Random Colour/Stipple
1801       SELect ON rn
1802         =1:cloud_col%=7
1803         =2:cloud_col%=56
1804         =3:cloud_col%=63
1805         =4:cloud_col%=248
1806         =5:cloud_col%=255
1807       END SELect
1808       SELect ON cloud_wide
1809         =1:IF RND(1 TO 10)=10
1810             cloud_base_char=191:cloud_high=0:cloud_col%=6
1811           ELSE
1812             cloud_base_char=179+(RND>.5)              :REMark small cloud
1813           END IF
1814         =2:cloud_base_char=180+(2*(RND>.5))
1815         =3:cloud_base_char=184+(3*(RND>.5))
1816       END SELect
1817     END IF
1818     IF cloud_run%>0
1819       CSIZE#3,1,cloud_high:INK#3,cloud_col%
1820       CURSOR#3,33*8,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

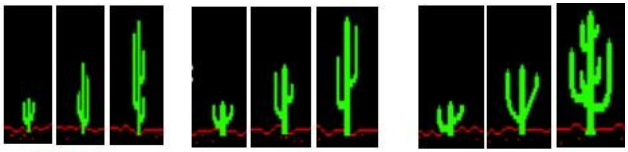
< Another Game Y/N >

```
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
```

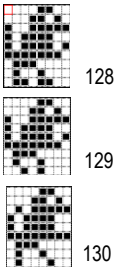
Note: Adjust with +/- then Enter

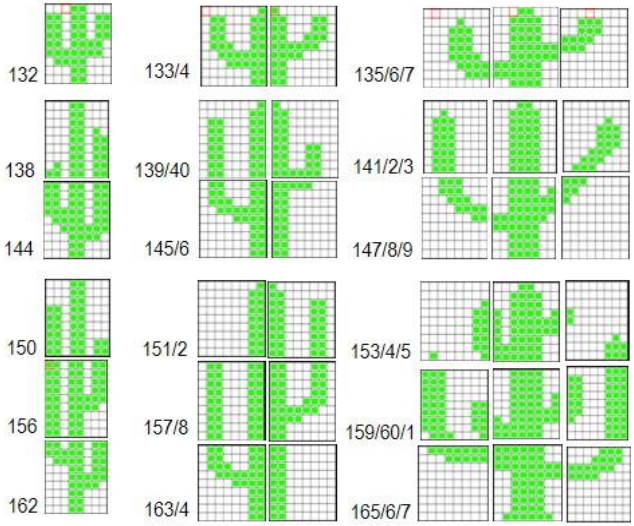
DINO_fnt uses the extended character code set 128 to 191

Cacti Codes 132 to 167
[3 widths 3 heights]



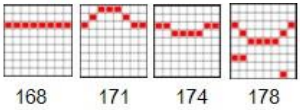
Dino Codes 128 to 131



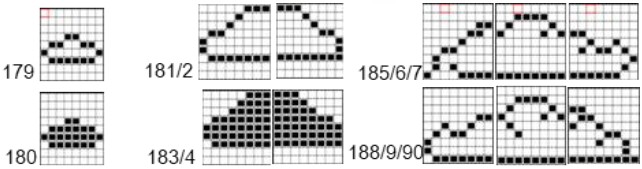


Sun Code 191

Ground Level Codes 168 to 178



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



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

```
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
```

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


```
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$='ÌÍÎÏÐ':pm=1:junk$='ÑÒÓÔÕ'
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)

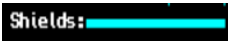
```
1887 DEFine PROCedure set_pod
1888 FOR i=1 TO tcap
1889   REPeat pp
1890     ay=RND(3 TO 14):ax=RND(2 TO 20)
1891     IF ax<>11 AND ay<>8 AND al(ax,ay)=0:EXIT pp
1892   END REPeat pp
1893   al(ax,ay)=1:AT#3,ay,ax:PRINT#3,"•":beeps 5
1894 END FOR i
1895 END DEFine
```

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

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
```

Note: CHR$(149)



```
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'   Note: CHR$(128 to138)
1964 INK#3,2:FOR i=0 TO 2:CURSOR#3,90+i,22:PRINT#3,'RED ALERT'      äãåéöõøúçñ
1965 CSIZE#3,2,0:OVER#3,0:INK#3,5
1966 RED_Alert 40,50:PAUSE 20:BEEP:CSIZE#3,0,0:INK#3,5
1967 CURSOR#3,58,132:PRINT#3,'Press Any Key to begin RESCUE'
1968 CURSOR#3,36,144:PRINT#3,'Use ←↑↓→ keys to Guide your Giro Pod'
1969 CURSOR#3,86,154:PRINT#3,'to pick up Survivors'
1970 PAUSE:FOR i=1 TO 9:PAUSE 3:BLOCK#3,276,i*16,0,90-i*8,0
1971 OVER#3,1:INK#3,7:RESTORE 1975
1972 FOR a=1 TO 4
1973   READ x,y,str$:FOR b=0 TO 1: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

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

Note: The QBITS Intro Screen to Giro Rescue

```
1991 DEFine PROCedure RED_Alert(sx,sy)
1992 RESTORE 2000:BEEP 0,10,100,5,-5,10,10,15
1993 FOR i=1 TO 8
1994   INK#3,RND(2 TO 5):CURSOR#3,70,42:PRINT#3,'Abandon Ship'
1995   READ x,y:INK#3,248:LINE#3,sx+12.5+i*2,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
```