

Introduction

Home Computers in the 1980's used Bitmap Character sets based on an 8x8 matrix covering Western ASCII codes. Some Files had headers to identify type, overall File size, the number of Bitmap, their width and height. For Font `_fnt` Files this usually includes a Start Font Code and the Number of Fonts. An App processes each Font's Bitmap to create a pixel display positioned at required screen x,y coordinates.

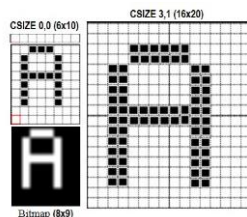
QBITS Font Editor Concept

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 idea was to Load QL Font sets and display them as a Chart. A selected character would then be shown in a Bitmap display with the option to Edit and Save. After a number of Character Fonts had been Edited the Set could be Saved as a New Font File for use with other Programs.

Sinclair QL Character Fonts

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

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



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

QBITS FontEditSE Plus

The Special Edition showcased three ARCADE Game Fonts. SE Plus looks at Designs for Stripped, Narrow, Compact Fonts, with Bold and Proportional Spacing.

QBITS FontEditSE - File Management

The QL Character BITMaps can be extended to include CHR\$(0 to 31) and (192 to 255). QBITS_FontEditSE arranges CHR\$ 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 SubDIRectories if present.

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

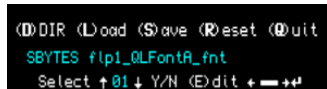


```

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


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

For **LOAD** a search is made of available ‘_fnt’ Front files. A ‘File Not found’ will be displayed if Device has none. Use Up/Down Cursors keys to Select from those ‘_fnt’ Files found, then ‘Y’ to Load ‘_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 (E)xit
SBYTES flp1_QLFontA_fnt
Select ↑01↓ Y/N (E)dit ←→←←
```



```

(D)DIR (L)oad (S)ave (R)eset (E)xit
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 (E)xit
Y/N
```



```

(D)DIR (L)oad (S)ave (R)eset (E)xit
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 by they Bold, Italics, or Personal and Stylised alphanumerical, Futuristic or perhaps 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** within a **White Box**. Press Spacebar to access the **Bitmap** Editor bits are set to 1's & 0's [INK/BkGnd Colours]. Font is displayed above in different **CSIZE's** use **-/+** to view Scalable sizes from 6-70. Below Bitmap **KEY:** shows Keys for Printable Fonts and the ASCII Character Code 'n' is 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←↑↓→** a second Chart Font is now Outlined by a **Red box**. Press **↵** Enter to action, alternatively you could Press **'S'** and Swap Font Grid Positions. **(F)**ill Sets all cells to '1' INK colour, **(#)** CLS sets Bitmap cells to '0' BKGnd. Press 1-7 for INK colour: Press **CTRL** 0-7 to change BkGnd Colour

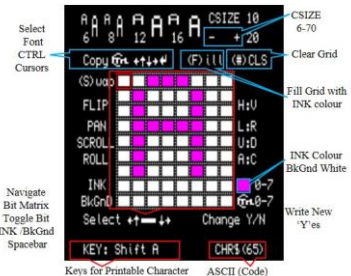
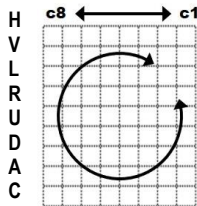
Cell Movement

Flip c1=c8 to c8=c1
r1=r9 to r9=r1

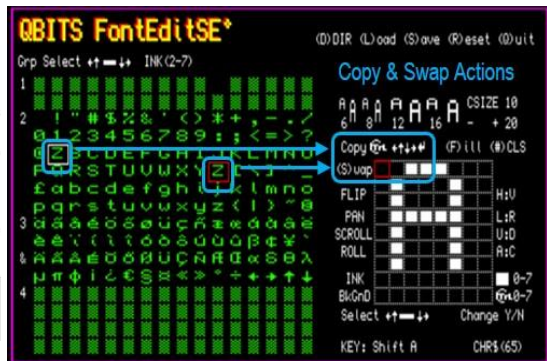
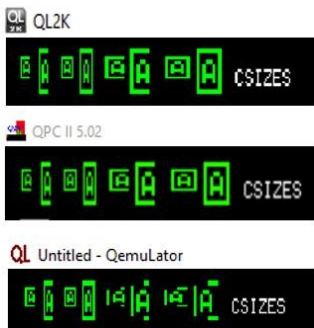
Pan c1=c8 to c2=c1 c8=c7
c1=c2 to c7=c8 c8=c1

Scroll r1=r2 to r8=r9 r9=r1
r2=r1 to r9=r8 r1=r9

Roll r1,c1=r1,c8 r9,c1=r1,c1
r1,c1=r9,c1 r9,c8=r1,c8



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 0,0 or 2,0 [6 & 12pixels] Prints 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, the Font Bitmap and their CSIZE's are displayed, this aspect can be disabled for slower Platforms, BBQL etc. see Lines 1064 & 1066 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 **channel**.

With the Fonts Bitmaps assigned to memory the character pattern can be Read and Overwritten with the PEEK and POKE commands. Each is identified by an offset from the assigned Base address actioned with **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 1's and 0.

Exiting S/SuperBASIC Prog **CHAR_USE** resets to default QL Fonts and memory is released unless **RESPR**(876+588) is used for reserving Memory space, then 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 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 FontEditSE+ - FONT Designs

Tinkering with Bitmap layouts of FONT Designs can take considerable time, but can also produce some amazing results. However, having a specific goal helps in the decisions to take. **QBITS FontEditSE** has the tools to **Copy**, **Swap** and modify Bitmap layouts in a number of ways. Starting with change of **INK** and **BackGnd** colours, **Fill** or **Clear** the Bitmap Grid. **Flip** or move Grid Cells **Lef/Right Up/Down** even Roll **90° Clockwise** or **Anticlockwise**. All of which help in a speedy development of New FONT Sets. These then Saved in a recognised QL Format.

LBYTE Font File into memory and with **CHAR_USE** set to a WINDOW Channel. For multiple Font sets used in the same screen area, assign individual Font sets with **CHAR_USE** to different Channels [ie. Overlapping Windows]. The Font Review Pages selected from the lower **MENU** of **QBITS FontEditSEPlus** show some examples.

QBITS FontEditSE+ - Associated ‘_fnt’ Files

If used **QBITSBoot_bas** sets the default device and configures **QBITSConfig** to a range of available devices. The **QLFontA_fnt** & **QLFontB_fnt** are the default QL Character Groups and are required to be present on the Default Loading Device.

FONT Files with Sprites used by the **SE Showcase ARCADE Games**

Aliens_fnt	Font designs for Alien Sprites Gun Platform etc.
Dino_fnt	Font designs for a walking DINO, Cactus Trees, Clouds etc,
Giro_fnt	Fonts for Rescue ship life pods Debry etc.

FONT Files for Characters sets used by the **SE Plus** Screen displays.

QLNarrow_fnt,	Fonts with a reduced width Character set.
QLXCompact_fnt	Fonts set for Information Labelling etc.
QLMUSIC_fnt	Fonts for Score sheets, Notes Rests Symbols etc
QLTIMER_fnt	Clock and Timeout Fonts

QBITS FontEdit+ - Added PRINT PROCedures

QBBold overwrites a String with a CURSOR offset **x, y+1** to create a **Bold** Font style without having to expand the Character width. This is OK with Uppercase Characters but not so good with lowercase.

QBPrnt Prints a String set with an independent Character width to using CSIZE. The QL default Font's 8x9 Matrix use only 6 columns CSIZE 0,0. For all eight columns use CSIZE 1,0. The CSIZE 2,0 (6>12) and 3,0 (8>16) just double up the number of columns.

Note: A CSIZE change to 20 rows if close to bottom of a Window can cause an unexpected SCROLL UP.

QBPPrt uses a simple method for Proportional Spacing of Alphanumeric Characters. The QL Character set uses six columns, the first is left empty with 2 to 6 used to construct the Character Symbol. For Space all columns are blank, the number 1, Capital I and lowercase f, l, t, leave one and two blank, i, j, the third column is not used. Using a **SElect ON** ASCII code to identity Character, spacing from previous printed symbol is cut by two columns or more. This can reduce a typical line length by as much as 20%.



QBITS QLFonts

This Theme Viewer was written for Character Fonts CHR\$(32 – 127) and has been added to the QBITSProgs Menu.

Select **FontView⁺** from Menu and Press Enter

QLFonts

Source: Type a 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 are listed SCROLL Up/Down to Select a Character Font. Press Enter the program returns to QBITSProgs Menu. However, the display is change to reflect the chosen Font. This has been actioned by use of CHAR_USE referenced earlier.

Source: To Reset Press **Esc**

Source: If you Return without making an 'Entry' CHAR_USE is then Reset to the QL Default Character Fonts CHAR_USE#1,0,0.

QBITS_FontsView_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):sf%=0:ft%=0:fm%=100 :REMark Set FontFiles
1010 FBase1=RESPR(1168) :REMark Heap Allocation
1012 Init_win
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:CUSOR#4,i,1:PRINT#4,'QLFonts'
1019 OVER#4,0:CSIZE#4,0,0:CSIZE#3,1,0
1020 CUSOR#4,2,11:PRINT#4,'Source: ':_WINDOW#4,120,10,gx+70,gy+42:sf%=0
1021 REPEAT Fnt_ip
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 Fnt_ip
1025 END DEFine
```



```

1027 DEFine PROCEDURE FontChar
1028 cy=0:sm=10:if sm>ft%:sm=ft%
1029 FOR sn=1 TO sm:FontFile 3,sn,cy:cy=cy+11
1030 sn=1:sr=1:cy=0
1031 REPEAT Char_lp
1032 INK#3,0:CUSOR#3,2,-11+sr*11:PRINT#3,"½"
1033 PRINT#4,File$(sn)::K=CODE(INKEY$(-1))
1034 INK#3,7:CUSOR#3,2,-11+sr*11:PRINT#3,"½"
1035 SElect ON K
1036 = 27:CLS#3:CLS#4:EXIT Char_lp           Note: Press Esc to Reset Source:
1037 =208:sr=sr-1:sn=sn-1:F_Up
1038 =216:sr=sr+1:sn=sn+1:F_Dn
1039 = 32:RECHP FBase1:CHAR_USE 0,0:CLOSE#3:CLOSE#4:LRUN dn$:STOP
1040 = 10:FontFile 1,sn,cy                   :CLOSE#3:CLOSE#4:LRUN dn$:STOP
1041 END SElect
1042 END REPEAT Char_lp
1043 END DEFine

```

```

1045 DEFine PROCEDURE F_Up
1046 IF sr<1 AND sn>0:SCROLL#3,11:FontFile 3,sn,0
1047 IF sr<1:sr=1
1048 IF sn<1:sn=1
1049 END DEFine

```

```

1051 DEFine PROCEDURE F_Dn
1052 IF sr>10 AND sn<=ft%:SCROLL#3,-11:FontFile 3,sn,99
1053 IF sr>10:sr=10
1054 IF sn>ft%:sn=ft%
1055 END DEFine

```

```

1057 DEFine PROCEDURE FontFile(ch,sn,cy)
058 LBYTES FDrv$&File$(sn),FBase1:CHAR_USE#ch,FBase1,0
1059 INK#ch,0:CUSOR#ch,12,cy:PRINT#ch,File$(sn)
1060 END DEFine

```

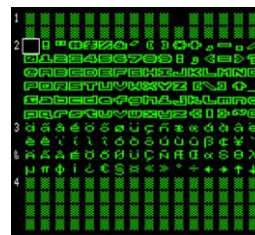
```

1062 DEFine PROCEDURE FntList
1063 IF FDrv$="":RETurn
1064 DELETE FDrv$&'FList':PRINT#4,'Checking...':PAUSE 30
1065 OPEN_NEW#9,FDrv$&'FList':DIR#9,FDrv$:CLOSE#9
1066 OPEN_IN#9,FDrv$&'FList':dl%=LEN(FDrv$):sf%=0
1067 REPEAT dir_lp
1068 IF EOF(#9) OR sf%>fm%:sf%=sf%-1:ft%=sf%:CLOSE#9:EXIT dir_lp
1069 INPUT#9,F$:F$=F$(dl%-4 TO):ft%=LEN(F$)
1070 IF ft%<=20 AND 'fnt' INSTR F$>0 AND F$(ft%)<>'>'
1071 OPEN_IN#99,FDrv$&F$.a%=CODE(INKEY$(#99)):CLOSE#99
1072 IF a%>30% AND a%<33:sf%=sf%+1:File$(sf%)=F$:ft%=sf%
1073 END IF
1074 END REPEAT dir_lp
1075 END DEFine

```

Note: 'fnt' files accessed by Theme Viewer must follow QL Format. The first Byte - start ASCII Code, Byte 2 number of 9Byte Bitmaps 64, 96,127 for File size 588, 876 or 1164 covering ASCII codes 0 to 127. The Extended character range is not covered.

Fonts.ZIP A List of Font Posted on QLFORUM by Andrew (of which some are shown here).



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).**

1016 REMark **QBITS Special Edition Plus – Showcase Three ARCADE Games + SPECIAL FONTS**

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:CUSOR#2,6+i,6:PRINT#2,'QBITS QLFont Editor+'

1031 INK#2,6:FOR i=0 TO 1:CUSOR#2,8+i,4:PRINT#2,'QBITS QLFont Editor+'

1032 CSIZE#2,0,0:OVER#2,0:INK#1,7

1033 CUSOR#1,1,280,14:PRINT#1,'(D)DIR (L)oad (S)ave (R)eset (E)xit':OVER#1,1

1034 CUSOR#1,1,281,14:PRINT#1,' D L S R E ' :OVER#1,0

1035 CUSOR#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\$**:CUSOR 6,fy:PRINT ch\$

1037 DATA 42,'1',64,'2',130,'3',152,'8',174,'4'

1038 **END DEFine**

1040 **DEFine PROCEDURE Init_Fnts**

1041 REMark Editor RAM Allocation

1042 FBaseA=ALCHP(1164):FBaseB=ALCHP(1164):FBase1=ALCHP(876):FBase2=ALCHP(588)

1043 baseC=ALCHP(588):LBYTES Dev\$&'QLXCompact_fnt',baseC :REMark Small Fonts

1044 baseM=ALCHP(588):LBYTES Dev\$&'QLXMUSIC_fnt',baseM :REMark QLX Extn Fonts

1045 baseT =ALCHP(588):LBYTES Dev\$&'QLXTIMER_fnt',baseT :REMark Timeout Symbols

1046 baseN =ALCHP(876):LBYTES Dev\$&'QLNarrow_fnt',baseN :REMark CHAR\$ 32 to 127

1047 baseL =ALCHP(588):LBYTES Dev\$&'QLXband_fnt',baseL :REMark Lower Half

1048 **END DEFine**

1050 **DEFine PROCEDURE ARCADE**

1051 INK#0,6:CUSOR#0, 32, 4:PRINT#0,'(F)ONTS Review'

1052 INK#0,5:CUSOR#0, 2,18:PRINT#0,'(C)ompact (M)USIC (T)IMER'

1053 INK#0,6:CUSOR#0,280, 4:PRINT#0,'"ARCADE GAMES" Aliens : Dino : Giro'

1054 INK#0,5:CUSOR#0,278,18:PRINT#0,'(L)oad "_fnt" File then Press (G)ame'

1055 **END DEFine**




```

1113 DEFine PROCEDURE FontChar(cx,cy,ci)
1114 INK#3,4:CURSOR#3,2+cx*17,1+cy*11:PRINT#3,CHR$(cn)
1115 BLOCK#3,20,1,cx*17,cy*11,ci:BLOCK#3,20,1,cx*17,12+cy*11,ci
1116 BLOCK#3,1,12,cx*17,cy*11,ci:BLOCK#3,1,12,19+cx*17,cy*11,ci
1117 END DEFine9

```

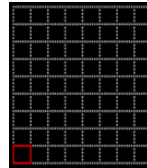
Note: Highlights FONT in Character Chart



```

1119 DEFine PROCEDURE FontGrid
1120 BLOCK#4,112,90,46,40,bcol
1121 FOR i=0 TO 8:BLOCK#4,1,91,40+i*14,40,248
1122 FOR i=0 TO 9:BLOCK#4,112,1,40,40+i*10,248
1123 END DEFine

```



PROCedures for Character PRINT Bold : Character Spacing : Proportional Spacing : Action FONTS

1150 REMark **QBBold** OK for Capitals not so good for lowercase

```

1152 DEFine PROCEDURE QBBold(ch%,bc%,bx%,by%,str$)
1153 INK#ch%,bc%:CURSOR#ch%,bx%,by% :PRINT#ch%,str$
1154 OVER#ch%,1 :CURSOR#ch%,bx%,by%+1:PRINT#ch%,str$:OVER#ch%,0
1155 END DEFine

```

1157 REMark **QBPrnt** Set Character Spacing cw % for String

```

1159 DEFine PROCEDURE QBPrnt(ch%,ci%,cw%,cx%,cy%,str$) :REMark Char Spacing
1160 INK#ch%,ci%:sl%=LEN(str$)-1:IF cw%<5:cw%=5
1161 FOR c=0 TO sl%:CURSOR#ch%,cx%+cw%*c,cy%:PRINT#ch%,str$(c+1)
1162 END DEFine

```

1164 REMark **QBPPrt** Set Proportional Spacing pw% for Char Widths within String

```

1166 DEFine PROCEDURE QBPPrt(ch%,ci%,cw%,cx%,cy%,str$)
1167 OVER#ch%,1:INK#ch%,ci%:sl%=LEN(str$):cx%=cx%-cw%:pw%=cw%
1168 FOR c=1 TO sl%
1169 Cck%=CODE(str$(c)):SElect ON Cck%=32,102,105,106,108,116:pw%=cw%-2
1170 cx%=cx%+pw%:CURSOR#ch%,cx%,cy%:PRINT#ch%,str$(c):pw%=cw%
1171 END FOR c:OVER#ch%,0
1172 END DEFine

```

Note: Char Width pw% is reduced for 'Space', f, i, j, l, t to create Proportional Spacing.

1174 REMark **ActFnt** Action Font - Set Mask and OVER PRINT with Timeout Clock Font

```

1176 DEFine PROCEDURE ActFnt(ch%,cb%,ci%,cx%,cy%,mn%,Fg%)
1177 REMark Action BackGnd STRIP cb% INK ci% x/y Mask mn% Font Fg%
1178 INK#ch%,ci%:STRIP#ch%,cb%:CURSOR#ch%,cx%,cy%:PRINT#ch%,CHR$(mn%)
1179 INK#ch%,0:OVER#ch%,1:CURSOR#ch%,cx%,cy%:PRINT#ch%,CHR$(Fg%):OVER#ch%,0
1180 END DEFine

```

Note: On Exit of SuperBASIC Prog Open Channels #3 and above are Closed and Allocated heap Memory released. STOP can be replaced with an LRUN command to Return to another Program such as LRUN flp1_Boot or win1_Progs_Menu etc

```

1182 DEFine PROCEDURE FontExit
1183 CURSOR#1,460,28:PRINT#1,'Y/N':PAUSE:IF KEYROW(5)<>64:RETurn
1184 CLOSE#3:CLOSE#4:CLOSE#5:CLOSE#6:CLOSE#7:CLS#1:CLS#0: LRUN dn$ :STOP
1185 END DEFine.

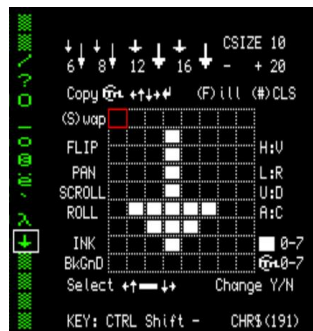
```

Note: Key(s) for Printable Characters CHR\$(32 to 191)

```

1200 DEFine PROCEDURE KeyPad
1201 SElect ON cn=32 TO 93 :Key$=CHR$(cn)
1202 IF cn=32 :Key$='Spacebar'
1203 SElect ON cn=33,34 :Key$='Shift '&(cn-32)
1204 SElect ON cn=36,37 :Key$='Shift '(cn-32)
1205 IF cn=38 :Key$='Shift 7'
1206 IF cn=40 :Key$='Shift 9'
1207 IF cn=41 :Key$='Shift 0'
1208 IF cn=42 :Key$='Shift 8'
1209 IF cn=43 :Key$='Shift ='
1210 IF cn=58 :Key$='Shift ;'
1211 IF cn=60 :Key$='Shift ,'
1212 IF cn=62 :Key$='Shift .'
1213 IF cn=63 :Key$='Shift /'
1214 IF cn=64 :Key$='Shift '"
1215 SElect ON cn=65 TO 90 :Key$='Shift '&CHR$(cn)
1216 IF cn=94 :Key$='Shift 6'
1217 IF cn=95 :Key$='Shift -'
1218 IF cn=96 :Key$='Shift 3'
1219 SElect ON cn=97 TO 122 :Key$=CHR$(cn-32)
1220 IF cn=123 :Key$='Shift ['
1221 IF cn=124 :Key$='Shift \'
1222 IF cn=125 :Key$='Shift ]'
1223 IF cn=126 :Key$='Shift #'
1224 IF cn=127 :Key$='Shift Esc'
1225 IF cn=128 :Key$='CTRL Esc'
1226 IF cn=129 :Key$='CTRL Shift 1'
1227 IF cn=130 :Key$='CTRL Shift '"
1228 SElect ON cn=131 TO 133 :Key$='CTRL Shift '&CHR$(cn-80)
1229 IF cn=134 :Key$='CTRL Shift 7'
1230 IF cn=135 :Key$='CTRL '"
1231 IF cn=136 :Key$='CTRL Shift 9'
1232 IF cn=137 :Key$='CTRL Shift 0'
1233 IF cn=138 :Key$='CTRL Shift 8'
1234 IF cn=139 :Key$='CTRL Shift ='
1235 SElect ON cn=140 TO 153 :Key$='CTRL '&CHR$(cn-96)
1236 IF cn=154 :Key$='CTRL Shift ;'
1237 IF cn=155 :Key$='CTRL ;'
1238 IF cn=156 :Key$='CTRL Shift ;'
1239 IF cn=157 :Key$='CTRL ='
1240 IF cn=158 :Key$='CTRL Shift .'
1241 IF cn=159 :Key$='CTRL Shift /'
1242 IF cn=160 :Key$='CTRL Shift 2'
1243 SElect ON cn=161 TO 186 :Key$='CTRL Shift '&CHR$(cn-96)
1244 IF cn=187 :Key$='CTRL ['
1245 IF cn=188 :Key$='CTRL \'
1246 IF cn=189 :Key$='CTRL ]'
1247 IF cn=190 :Key$='CTRL Shift 6'
1248 IF cn=191 :Key$='CTRL Shift -'
1249 SElect ON cn=0 TO 31,192 TO 255:Key$=""
1250 CURSOR 320,208:PRINT 'KEY: '&Key$;FILL$(' ',14-LEN(Key$));CHR$(';',cn);' '
1251 END DEFINE

```



```

1300 DEFINE PROCEDURE FontMod
1301 QBPrnt 1,7,6,312,82,'Copy ↵↵↵↵ :QBPrnt 5,7,6,374,0,',, (F)ill (#)CLS'
1302 QBPrnt 1,7,6,312,188,'Select↵↵ ↵↵ Change Y/N'
1303 CSIZE#5,3,0:QBPrnt 5,7,16,326,0,'':QBPrnt 5,7,16,353,105,'f'
1304 CURSOR#5,444,92:PRINT#5,'':CSIZE#5,0,0:FontINK:FontSize
1305 RESTORE 1106:FOR i=1 TO 13:READ a,b,c$:CURSOR a,b:PRINT c$
1306 DATA 312,112,'FLIP',316,126,'PAN',306,137,'SCROLL',312,148,'ROLL'
1307 DATA 306,96,'(S)wap',460,112,'H:V',460,126,'L:R',460,137,'U:D',460,148,'A:C'
1308 DATA 316,165,'INK',476,165,'0-7',310,176,'BkGnD',476,176,'0-7'
1309 cxo=cx:cyo=cy:FontChar cx,cy,7:KeyPad:bx=1:by=1:rm=9:cs=8:co=cn
1310 REPEAT Chg_lp
1311   cn=cx+(cy)*16:FontChar cx,cy,2:FontChar cxo,cyo,7
1312   FontBit bx,by,7:K=CODE(INKEY$(-1)):FontBit bx,by,248:Fontchar cx,cy,0
1313   SELECT ON K
1314     =192:bx=bx-1:IF bx<1:bx=1
1315     =200:bx=bx+1:IF bx>8:bx=8
1316     =208:by=by-1:IF by<1:by=1
1317     =216:by=by+1:IF by>9:by=9
1318     =196:cx=cx-1:IF cx<0:cx=15:cy=cy-1:IF cy<0:cy=0:cx=0
1319     =204:cx=cx+1:IF cx>15:cx=0:cy=cy+1:IF cy>15:cy=15:cx=15
1320     =212:cy=cy-1:IF cy<1:cy=0
1321     =220:cy=cy+1:IF cy>15:cy=15
1322     =48 TO 55: col=K-48:FontINK
1323     = 32:BitSwap bx,by
1324     = 10:FontPeek
1325     = 35:FontCLS
1326     =144 TO 153:bcol=K-144:FontINK
1327     =83,115:FontSwap:EXIT Chg_lp
1328     =104,72:FontFlip 9,-1,0,1
1329     =118,86:FontFlip 0,1,10,-1
1330     =108,76:FontSlid 0,8,1,0,1
1331     =114,82:FontSlid 0,1,8,1,0
1332     =117,85:FontSlid 1,9,1,1
1333     =100,86:FontSlid 1,1,9,-1
1334     = 97,65:FontRoll 1
1335     = 99,67:FontRoll 2
1336     =102,70:FontFill
1337     =110,78:EXIT Chg_lp
1338     =121,89:cn=co:FontPoke:EXIT Chg_lp
1339 END SELECT
1340 END REPEAT Chg_lp
1341 cn=co:cx=cxo:cy=cyo:CLS#4:FontGrid
1342 END DEFINE

```

:REMark 0-7 Change INK Colour
 :REMark Bit Swap 0<>1
 :REMark Change Font Pattern
 :REMark (#) Reset Grid
 :REMark CTRL 0-7 Change BkGnD Colour
 :REMark (S)wap Chart Fonts
 :REMark (H)orizontal Flip
 :REMark (V)ertical Flip
 :REMark (L)eft PAN Grid
 :REMark (R)ight PAN Grid
 :REMark (U)p SCROLL Grid
 :REMark (D)n SCROLL Grid
 :REMark (A) Roll Anti-Clockwise
 :REMark (C) Roll Clockwise
 :REMark (F)ill with Ink Colour
 :REMark (N)o Change Return
 :REMark (Y)es Change Font



```

1344 DEFINE PROCEDURE FontINK
1345 BLOCK 12,8,460,166,col:FOR r=0 TO 8:FontDraw
1346 END DEFINE

```

Note: Display INK and BkGnd Colour

```

1348 DEFINE PROCEDURE FontFill
1349 FOR r=0 TO 8:Fnt$(r+1)="11111111":FontDraw
1350 END DEFINE

```

Note: Set all Matrix Bits to One's

```

1352 DEFINE PROCEDURE FontCLS
1353 FOR r=0 TO 8:Fnt$(r+1)="00000000":FontDraw
1354 END DEFINE

```

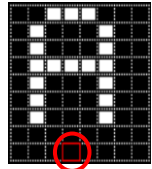
Note: Set all Matrix Bits to Zeros

```

1356 DEFine PROCEDURE FontDraw
1357 FOR c=0 TO 7
1358   IF Fnt$(r+1,c+1)='1':pcol=col:ELSE pcol=bcol
1359   BLOCK#4,11,7,42+c*14,42+r*10,pcol
1360 END FOR c
1361 END DEFine

```

Note: Print Pixel Colour



```

1363 DEFine PROCEDURE FontBit(x,y,hcol)  Note: Highlights a Bit Square within the Font BitMap
1364 BLOCK#4,15,1,32+bx*14,30+by*10,hcol:BLOCK#4,15,1,32+bx*14,40+by*10,hcol
1365 BLOCK#4,1,10,32+bx*14,30+by*10,hcol:BLOCK#4,1,10,46+bx*14,30+by*10,hcol1146
1366 END DEFine

```

```

1368 DEFine PROCEDURE BitSwap(bx,by)
1369 IF Fnt$(by,bx)='0':Fnt$(by,bx)='1':ELSE Fnt$(by,bx)='0'
1370 IF Fnt$(by,bx)='0':BLOCK#4,11,7,42+(bx-1)*14,42+(by-1)*10,bcol
1371 IF Fnt$(by,bx)='1':BLOCK#4,11,7,42+(bx-1)*14,42+(by-1)*10,col
1372 END DEFine

```

Note: STRIP

Note: INK

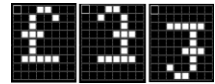


```

1374 DEFine PROCEDURE FontFlip(cf,cz,rf,rz)
1375 FOR r=1 TO 9:Tmp$(r)=Fnt$(r)
1376 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
1377 FOR r=0 TO 8:FontDraw
1378 END DEFine

```

Note: Flip Horizontal or Vertical



```

1380 DEFine PROCEDURE FontSlid(md,a,b,d,e)
1381 FOR r=1 TO 9:Tmp$(r)=Fnt$(r):rm=9:IF md=1 AND a=9:rm=8
1382 FOR r=1 TO rm
1383   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
1384   IF md=1:Fnt$(a)=Tmp$(b):FOR c=1 TO 8:Fnt$(r,c)=Tmp$(r+d,c):END FOR c
1385 END FOR r
1386 FOR r=0 TO 8:FontDraw
1387 END DEFine

```

Note: Slide Left or Right



Note: Slide Up or Down



```

1389 DEFine PROCEDURE FontRoll(rd)
1390 FOR r=1 TO 9:Tmp$(r)=Fnt$(r)
1391 FOR r=1 TO 8
1392   IF rd=1:rx=r:ry=8:FOR c=1 TO 8:Fnt$(ry,rx)=Tmp$(r,c):ry=ry-1:END FOR c
1393   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
1394 END FOR r
1395 FOR r=0 TO 8:FontDraw
1396 END DEFine

```

Note: Rotate 90° Anticlockwise



Note: Rotate 90° Clockwise



```

1398 DEFine PROCEDURE FontSize
1399 CURSOR 318,67:PRINT '6 8 12 16 20 CSIZE'
1400 CURSOR 438,54:PRINT '10':RESTORE 1199:STRIP#4,bcol:INK#4,col
1401 FOR i=1 TO 8:READ a,b,c,d:CSIZE#4,a,b:c:CURSOR#4,c,d:PRINT#4,CHR$(cn)
1402 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
1403 END DEFine

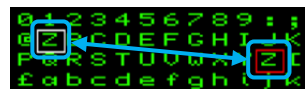
```



```

1405 DEFine PROCEDURE FontSwap
1406 FOR r=1 TO 9:Tmp$(r)=Fnt$(r):END FOR r:FontPoke:FontDraw:sn=cn:cn=co:FontPoke
1407 FOR r=1 TO 9:Fnt$(r)=Tmp$(r):END FOR r:cn=sn:FontPoke
1408 CURSOR#3,2+cx*17,1+cy*11:PRINT#3,CHR$(cn)
1409 END DEFine

```



```

1411 DEFine PROCEDURE FontPeek
1412 IF cn<128:addr=FBaseA+2+cn*9
1413 IF cn>127:addr=FBaseB+2+(cn-127)*9
1414 FOR r=0 TO 8:Fnt$(r+1)=BIN$(PEEK(addr+r),8):FontDraw
1415 END DEFine

```

Note: Read Font Bit Pattern from RAM

```

1417 DEFine PROCEDURE FontPoke
1418 IF cn<128:addr=FBaseA+2+cn*9:ELSE addr=FBaseB+2+(cn-127)*9
1419 FOR r=0 TO 8:POKE(addr+r),BIN(Fnt$(r+1))
1420 END DEFine

```

Note: Write New Font Bit Pattern to RAM

```

1450 DEFine PROCEDURE FontLoad

```

```

1451 chk=0:eck=0:Lchk=0:sf%=1:ft%=0:FOR i=1 TO fm%:File$(i)="
1452 INK 7:CURSOR 300,42:PRINT 'Select ↑ ↓ Y/N':INK 5:SelDrv 1:FntList 1
1453 SelFont 1,'LBYTES':IF Lchk=0:RETurn:ELSE CURSOR 226,28:CLS 4
1454 CURSOR 310,28:PRINT#1,'Loading...' :CURSO 300,42:CLS 4:PAUSE 30
1455 OPEN _IN#9,drv$(dn%)&File$(sf%)
1456 sg%=CODE(INKEY$(#9)):cg%=CODE(INKEY$(#9))
1457 CLOSE#9
1458 IF sg%<31 AND cg%> 96:addr1=FBaseA:cy= 0:Fg$(1)=File$(sf%):Fg$(2)=Fg$(1):cn1=0
1459 IF sg%>=31 AND sg%<127:addr1=FBase1:cy= 2:Fg$(2)=File$(sf%):addr2=FBaseA:cn1=96
1460 IF sg%>126 AND cg%< 65:addr1=FBase2:cy= 8:Fg$(3)=File$(sf%):addr2=FBaseB:cn1=64
1461 IF sg%>126 AND cg%> 64:addr1=FBaseB:cy=12:Fg$(4)=File$(sf%):Fg$(3)=Fg$(4):cn1=0
1462 LBYTES drv$(dn%)&File$(sf%),addr1:cx=0:IF cn1=96:os=sg%:ELSE os=sg%-127
1463 IF cn1=96 OR cn1=64
1464 FOR a=1 TO cn1
1465 FOR b=0 TO 8:POKE addr2+2+(a+os)*9+b,PEEK(addr1+2+a*9+b)
1466 END FOR a
1467 END IF
1468 FontSets:eck=0:INK 7:FontGrid
1469 END DEFine

```

```

@DIR Load Save Reset Quit
LBYTES dos1_0LFontA_fnt
Select ↑ 14 ↓ Y/N

```

```

@DIR Load Save Reset Quit
Loading...

```

```

1471 DEFine PROCEDURE FontSave

```

```

1472 chk=0:eck=0:Lchk=0:sf%=1:ft%=4
1473 FOR i=1 to 4:File$(i)=Fg$(i)
1474 INK 7:CURSOR 300,42:PRINT 'Select ↑ ↓ Y/N (E)dit ← →':
1475 BLOCK 12,3,454,46,7:BLOCK 2,4,480,44,7:INK 5:SelDrv 2
1476 SelFont 2,'SBYTES':IF Lchk=0:RETurn
1477 FntList 2:BLOCK 200,10,300,42,0 :CURSOR 312,42:INK 7
1478 IF eck=1:PRINT#1,'DEVICE ERROR...':PAUSE 50:RETurn
1479 IF chk=1:PRINT#1,'Overwrite Y/N' :PAUSE:IF KEYROW(5)<>64:RETurn
1480 INK 5:DELETE drv$(dn%)&File$(sf%) :CURSOR 226,28:CLS 4
1481 CURSOR 310,28:PRINT#1,'Saving...' :CURSOR 300,42:CLS 4:PAUSE 30
1482 IF sf%=1:addr1=FBaseA:lgth=1154:cn1=127
1483 IF sf%=2:addr1=FBase1:lgth=876 :cn1= 96:addr2=FBaseA:os=31
1484 IF sf%=3:addr1=FBase2:lgth=588 :cn1= 64:addr2=FBaseB:os= 0
1485 IF sf%=4:addr1=FBaseB:lgth=1154:cn1=127
1486 IF cn1=96 OR cn1=64
1487 FOR a=1 TO cn1
1488 FOR b=0 TO 8:POKE addr1+2+a*9+b,PEEK(addr2+2+(a+os)*9+b):Fg$(sf%)=File$(sf%)
1489 END FOR a
1490 END IF
1491 SBYTES drv$(dn%)&File$(sf%),addr1,lgth
1492 END DEFine

```

```

@DIR Load Save Reset Quit
SBYTES f\p1_0LFontA_fnt
Select ↑ 01 ↓ Y/N (E)dit ← →

```

```

@DIR Load Save Reset Quit
Checking...
DEVICE ERROR...

```

```

@DIR Load Save Reset Quit
Checking...
Overwrite Y/N

```

```

@DIR Load Save Reset Quit
Saving...

```

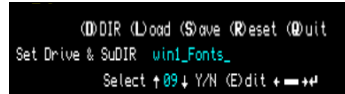

1494 DEFINE PROCEDURE FontDIR

1495 CURSOR 226,28:PRINT 'Set Drive & SuDIR'

1496 INK 7:CURSOR 300,42:PRINT 'Select↑ ↑ Y/N (E)dit ← →←'

1497 BLOCK 12,3,454,46,7:BLOCK 2,4,480,44,7:INK 5:SelDrv 3

1498 END DEFINE



1500 DEFINE PROCEDURE SelDrv(act%)

1501 REPEAT drv_ip

1502 CURSOR 342,28:PRINT drv\$(dn%):CLS 4

1503 CURSOR 351,42:PRINT FILL\$(0',2-LEN(dn%))&dn%

1504 K=CODE(INKEY\$(-1))

1505 SELECT ON K

1506 =208:dn%=dn%-1:IF dn%<1:dn%=dm%

1507 =216:dn%=dn%+1:IF dn%>dm%:dn%=1

1508 =101,69:IF act%=3:EditName 3,342,16,drv\$(dn%)

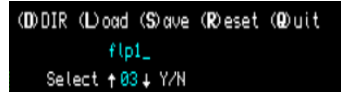
1509 =121,89,110,78:EXIT drv_ip

Yes or No Selects Device

1510 END SELECT

1511 END REPEAT drv_ip

1512 END DEFINE



1514 DEFINE PROCEDURE SelfFont(act%,Act\$)

1515 BLOCK 200,10,292,28,0:str\$=Act\$&drv\$(dn%)

1516 IF ft%<1:CURSOR 310,28:PRINT 'No Files Found...':PAUSE 30:RETURN

1517 CURSOR 226,28:PRINT FILL\$(' ',23-LEN(str%))&str\$

1518 REPEAT File_ip

1519 CURSOR 364,28:PRINT File\$(sf%):CLS 4

1520 CURSOR 351,42:PRINT FILL\$(0',2-LEN(sf%))&sf%

1521 K=CODE(INKEY\$(-1))

1522 SELECT ON K

1523 =208:sf%=sf%-1:IF sf%<1:sf%=ft%

1524 =216:sf%=sf%+1:IF sf%>ft%:sf%=1

1525 =101,69:IF act%=2:EditName 2,364,16,File\$(sf%)

1526 =110,78:Lchk=0:EXIT File_ip

No abort action

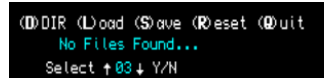
1527 =121,89:Lchk=1:EXIT File_ip

Yes Load / Save Font File

1528 END SELECT

1529 END REPEAT File_ip

1530 END DEFINE



1532 DEFINE PROCEDURE FntList(act%)

1533 BLOCK 280,10,226,28,0:CURSOR 310,28:PRINT 'Checking...'

1534 PAUSE 20:DELETE drv\$(dn%)&'FList'

1535 OPEN_NEW#9,drv\$(dn%)&'FList':DIR#9,drv\$(dn%):CLOSE#9

1536 OPEN_IN#9,drv\$(dn%)&'FList':dl%=LEN(drv\$(dn%))

1537 REPEAT dir_ip

1538 IF act%=1

1539 IF EOF(#9) OR sf%>fm%:sf%=sf%-1:ft%=sf%:CLOSE#9:EXIT dir_ip

1540 INPUT#9,F\$:F\$=F\$(dl%-4 TO):fl%=LEN(F\$)

Note: Checking _nft' Files to LOAD

1541 IF fl%<=20 AND '_fnt' INSTR F\$>0:File\$(sf%)=F\$:sf%=sf%+1

1542 END IF

1543 IF act%=2

1544 IF EOF(#9):CLOSE#9:chk=0:EXIT dir_ip

1545 INPUT#9,F\$:F\$=F\$(dl%-4 TO)

1546 IF F\$==File\$(sf%):CLOSE#9:chk=1:EXIT dir_ip

Note: Checking If SAVE File Exists

1547 END IF

1548 END REPEAT dir_ip

1549 END DEFINE

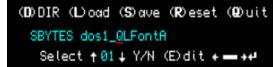


Note This QBITS Editor Restricts ASCII Code Characters used for QL Filenames. Position the underscore with left/Right cursors to a character then **Add** a new or **Delete** the existing character. Adding a Character expands the string to the right, Deleting shrinks the string from right to left. The 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.

```

1551 DEFINE PROCEDURE EditName(act%,x,sm%,str$)
1552 IF act%=2:sl%=(' _fnt' INSTR str$)-1:str$=str$(1 TO sl%)
1553 temp$=str$:sl%=LEN(str$):cp%=1
1554 REPEAT Ed_lp
1555   Ln_Prn:Ln_Cur:k$=INKEY$(#0,-1):K=CODE(k$)
1556   SELECT ON K
1557     = 10:EXIT Ed_lp
1558     = 48 TO 57, 65 TO 90,95, 97 TO 122:Add_chr
1559     =194   :IF cp%>1:cp%=cp%-1:Del_chr
1560     =202,32:Del_chr
1561     =192   :IF cp%>1:cp%=cp%-1
1562     =200   :IF cp%<sl%+1:cp%=cp%+1
1563   END SELECT
1564 END REPEAT Ed_lp
1565 IF sl%=0:str$=temp$
1566 IF act%=2:str$=str$&'_fnt'
1567 IF act%=3 AND str$(sl%)<>'_':IF sl%<sm%:str$=str$&'_' ELSE str$(sl%)='_'
1568 IF act%=3 AND str$(5)<>'_' :str$(5)='_'
1569 END DEFINE

```

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

Note: Return with Original str\$
Note: Check for ' _fnt' Suffix

```

1571 DEFINE PROCEDURE Ln_Prn
1572 IF LEN(str$)>sm%:str$=str$(1 TO sm%):cp%=sm%
1573 INK 5:CURSOR x,28:PRINT str$:CLS 4
1574 END DEFINE

```

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

```

1576 DEFINE PROCEDURE Ln_Cur
1577 BLOCK 6,1,x+cp%*6-6,37,2
1578 END DEFINE

```

Note: Character Highlight Cursor Bar

```

1580 DEFINE PROCEDURE Add_chr
1581 IF cp% = 1 AND sl% = 0 :str$=str$&k$
1582 IF cp%>=1 AND cp%<sl%:str$=str$(1 TO cp%-1)&k$&str$(cp% TO sl%)
1583 IF cp%>=1 AND cp%=sl%:str$=str$(1 TO cp%-1)&k$&str$(cp%)
1584 IF cp%> 1 AND cp%>sl%:str$=str$&k$
1585 IF cp%=sm%:str$(cp%)=k$
1586 IF sl%<sm% :sl% =sl% +1:ELSE sl%=sm%
1587 IF cp%<sm%:cp%=cp%+1:ELSE cp%=sm%
1588 END DEFINE

```

Note: Checks for Adding Character to String

```

1590 DEFINE PROCEDURE Del_chr
1591 IF cp%=sl%:str$=str$(1 TO sl%-1):sl%=sl%-1
1592 IF cp%>=1 AND cp%<sl%:str$=str$(1 TO cp%-1)&str$(cp%+1 TO sl%):sl%=sl%-1
1593 IF cp%=sm%:str$=str$(1 TO sm%-1):cp%=cp%-1:sl%=sm%-1
1594 IF cp%=1 AND sl%=1:str$="" :sl%=0
1595 END DEFINE

```

Note: Checks when Deleting Character from String

QBITS FontEditorSE - ARCADE Games

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

Space Invaders



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



Dino Run



Displayed are High Score and Accumulated Points. The program has 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 height beyond those set by CSIZE. Here where the subject Character Font(s) move across the screen, an independent height can be set for **PAN**.

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

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

Giro Rescue



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



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

QBITS 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. I used QBITS FontEdutSE to created Characters for a banner, various Aliens and designs for other parts of the Game...



Note: Load Aliens_fnt and Press (G)ame

1600 REMARK ARCADE – SPACE INVADERS a simplified version for QL By QBITS 2022

1602 **DEFine PROCEDURE Space_Invader**

1603 pd=2 :REMARK Game Speed pd=pause delay set (1 to 5)

1604 CSIZE#3,1,0:INK#3,6,b=0:tnum=0:num=0:pd=2

1605 FOR a=128 TO 142:b=b+1:CUSOR#3,b*8,8:PRINT#3,CHR\$(a) **Note:** Font Grp 3 Codes128 to 191

1606 CUSOR 127,200:PRINT'← →':BLOCK 12,3,137,204,7

1607 InitLevel:DrawAliens 1:!=5:INK#3,7

1608 **REPEAT Invader_lp**

1609 IF !=>4 OR at%>0

1610 CSIZE#3,2,0:CUSOR#3,54,80:PRINT#3,'New Game Y/N'

1611 **REPEAT ans**

1612 IF KEYROW(5)=64:z%=0:!=1:at%=27:tnum=0:num=0:CLS#3,3:**EXIT ans**

1613 IF KEYROW(7)=64:**EXIT Invader_lp**

1614 **END REPEAT ans**

1615 **END IF**

1616 **DrawAliens !=:**INK#3,7:CUSOR#3,74,80:PRINT#3,'LEVEL '!= **Note != level**

1617 FOR z=6 TO 0 STEP -1:CUSOR#3,124,96:PRINT#3,z:PAUSE 25

1618 BLOCK#3,120,30,72,80,0:i=8:n=5:s!=0:ay=3:**Invaders**

1619 **END REPEAT Invader_lp**

1620 CLS#3:FontSets

1621 **END DEFine**

1623 **DEFine PROCEDURE InitLevel**

1624 DIM GL%(4,3),GA%(3,9):**RESTORE 1526**

1625 FOR a=1 TO 4:**READ GL%(a,1),GL%(a,2),GL%(a,3)**

Note: 4 Levels of Play

1626 DATA 143,144,145,146,147,148,149,150,151,152,153,154

1627 **END DEFine**

1629 **DEFine PROCEDURE DrawAliens(!%)**

Note: Display Aliens by Level

1630 BLOCK#3,260,10,6,150,0:at%=27:**GScore 0**

1631 FOR a=1 TO 3:FOR b=1 TO 9:GA%(a,b)=GL%(!=,a):END FOR b:END FOR a

1632 INK#3,5:FOR i=1 TO 9:CUSOR#3,i*24,28:PRINT#3,CHR\$(GA%(1,i))

1633 INK#3,4:FOR i=1 TO 9:CUSOR#3,i*24,40:PRINT#3,CHR\$(GA%(2,i))

1634 INK#3,3:FOR i=1 TO 9:CUSOR#3,i*24,52:PRINT#3,CHR\$(GA%(3,i))

1635 **END DEFine**



1637 **DEFine PROCEDURE SLives**

1638 CSIZE#3,1,0:INK#3,7

1639 CUSOR#3,148,8:PRINT#3,FILL\$(CHR\$(159),4-s!%)&'

1640 CSIZE#3,3,0:s!=s!%+1

1641 **END DEFine**



Note: Ship Lives/Level

1643 **DEFine PROCedure** Invaders

1644 sl%=0:n=5:at%=27:**SLives**

1645 **REPeat lp**

1646 i=i+1:IF i>7:i=1:an=n+RND(-1 TO 1)

1647 **DShip** 7,n:x=an*24:y=60+i*12

1648 IF at%=0:l%=l%+1:RETurn

1649 IF at%>0 AND sl%>4:l%=5:RETurn

1650 IF KEYROW(1)=64:**FShip**:ay=3:**GScore** l%*50

1651 IF KEYROW(1)= 2:**DShip** 0,n:n=n-1:IF n<1:n=1

1652 IF KEYROW(1)=16:**DShip** 0,n:n=n+1:IF n>9:n=9

1653 INK#3,2:CUSOR#3,x,y:PRINT#3,CHR\$(158):PAUSE pd

1654 INK#3,2:CUSOR#3,x,y:PRINT#3,'':PAUSE pd

1655 IF an=n AND KEYROW(1)=64

1656 BEEP 2000,1,255,200,4,2,0,0

1657 INK#3,7:CUSOR#3,x,140:PRINT#3,CHR\$(160):PAUSE pd

1658 CLS#3,3:**FExplode** x,y:**GScore** 200:i=8

1659 **END IF**

1660 IF an=n AND i=7

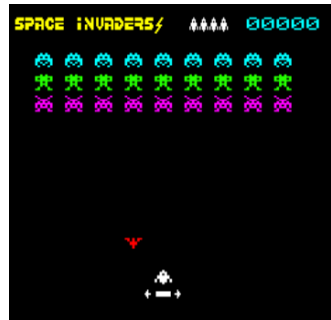
1661 BEEP 3000,20,80,80,-8,15,15,i=8:**SLives**

1662 CUSOR#3,x,150:PRINT#3,CHR\$(162):PAUSE pd*5

1663 **END IF**

1664 **END REPeat lp**

1665 **END DEFine**



Note: Falling Bombs Chr\$(158)

1667 **DEFine PROCedure** GScore(num)

1668 CUSOR#3,192,8:CSIZE#3,2,0:INK#3,5:tnum=tnum+num

1669 PRINT#3,FILL\$(0',5-LEN(tnum))&tnum:CSIZE#3,3,0

1670 **END DEFine**



1672 **DEFine PROCedure** DShip(col,n)

1673 INK#3,col:CUSOR#3,n*24,150:PRINT#3,CHR\$(159) **Note:** Defender Ship

1674 **END DEFine**



1676 **DEFine PROCedure** FShip

1677 BEEP 2000,1,255,200,4,2,0,0

1678 **FOR** i=1 **TO** 5

Note: Fire at Enemy

1679 OVER#3,1:CUSOR#3,n*24,150-i*18:PRINT#3,CHR\$(160):PAUSE pd

1680 OVER#3,0:CUSOR#3,n*24,150-i*18:PRINT#3,''

1681 **END FOR** i

1682 IF ay=3 AND GA%(ay,n)=32:ay=2

Note: Check Array Entry

1683 IF ay=2 AND GA%(ay,n)=32:ay=1

1684 IF ay=1 AND GA%(ay,n)=32:ay=3:RETurn

1685 **FExplode** n*24,16+ay*12:GA%(ay,n)=32:at%=at%-1 **Note:** Explode Alien and Delete

1686 **END DEFine**



1688 **DEFine PROCedure** FExplode(x,y)

1689 **FOR** i=1 **TO** 6

1690 BEEP 3000,20,60,80,-8,15,15,15

1691 INK#3,2:CUSOR#3,x,y:PRINT#3,CHR\$(160):PAUSE pd

1692 INK#3,6:CUSOR#3,x,y:PRINT#3,CHR\$(161):PAUSE pd

1693 **END FOR** i

1694 CUSOR#3,x,y:PRINT#3,''

1695 **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.

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.



1700 REMARK **ARCADE – Dino Run** based on version for QL by Dilwyn Jones 2022

1702 DEFine PROCEDURE Dino_Run

```
1703 dino_colour%      = 7
1704 cactus_colour%    = 4
1705 ground_colour%    = 2
1706 cloud_colour%     = 255
1707 background_col    = 0
1708 slowdown_steps    = 25000      :REMark 1/n Game Delay [?Dependant on Processor Speed]
1709 pany%              = 90         :REMark Top Left of Main PAN Block
1710 byPerCent          = 1          :REMark Percentage change of Slowdown delays
1711 every%             = 1000       :REMark every this much of score
1712 sound_on%          = 1          :REMark 0=sound off 1=sound on
1713 demo_mode%         = 0          :REMark auto-jumping
1714 hiscore=score      = 0 :dy%=18:sl=5000
```



Note: Load Dino_fnt and Press (G)ame

```
1715 :
1716 DIM backg$(2,34)      :REMark Cactus Height 0-2 Locations 0-34 of ground steps
1717 :
```

1718 REPEAT Dino_program

```
1719 REMark *** Title & Score bar ***
1720 PAPER#3;background_col:CLS#3:BLOCK#3,276,22,0,0,80
1721 OVER#3,1:CSIZE#3,1,1:STRIP#3,80:INK#3,7
1722 FOR i=0 TO 1:CORSOR#3,8+i,2:PRINT#3,' DINO Run'
1723 CURSOR#3,100,2:PRINT#3,'HI '&FILL$(0',5-LEN(hiscore))&hiscore;
1724 OVER#3,0:CSIZE#3,0,0:DSpeed:STRIP#3,0
1725 :
1726 REMark *** Set Ground level at pany%+50 [ie. (3*18)-4] ***
1727 RANDOMISE:CSIZE#3,1,0:INK#3,ground_colour%
1728 CURSOR#3,0,pany%+50:FOR a=1 TO 34:PRINT#3,CHR$(RAND(168 TO 178));
1729 :
1730 REMark *** Insert a couple of Random Cactii ***
1731 REMark 0=cactus 3 (top) 1=cactus 2 (middle) 2=cactus 1 (bottom)
1732 CHAR_INC#3,8,18:FOR y=0 TO 2:backg$(y)=FILL$('',34)
1733 backg$(2,RND(18 TO 22))=CHR$(132):backg$(2,RND(32 TO 34))=CHR$(132)
1734 OVER#3,1:CSIZE#3,1,1:INK#3,cactus_colour%
1735 FOR a=0 TO 2:CORSOR#3,0,pany%+(18*a):PRINT#3,backg$(a);
1736 OVER#3,0:CSIZE#3,1,0
```



```

1738 REMark *** Place Random Cloud ***
1739 STRIP#3,background_col :INK#3,ground_colour%:cloudy%=48
1740 rn=RND(1 TO 5) :REMark avoids Turbo error
1741 SElect ON rn
1742 =1 : cloud_colour% = 7
1743 =2 : cloud_colour% = 56
1744 =3 : cloud_colour% = 63
1745 =4 : cloud_colour% = 248
1746 =5 : cloud_colour% = 255
1747 END SElect
1748 CSIZE#3,1,0:INK#3,cloud_colour%:CURSOR#3,8*RND(10 TO 30),cloudy%
1749 IF RND(1 TO 2)=1
1750 PRINT#3,CHR$(181)&CHR$(182);
1751 ELSE
1752 PRINT#3,CHR$(183)&CHR$(184);
1753 END IF
1754 CSIZE#3,1,1:INK#3,7
1755 cloud_gap%=RND(10 TO 20):cloud_wide=0:cloud_run%=0
1756 slowdown=slowdown_steps :REMark disable [Set to 0 for BBQL}
1757 :
1758 cactus_gap%=RND(10 TO 20) :REMark columns blank before a cactus
1759 cactus_wide =0 :REMark 1/2/3
1760 cactus_high =0 :REMark 1/2/3
1761 cactus_run% =0 :REMark Cactus width (1/2/3) drawn
1762 dinobasey% =pany%+36 :REMark Dinosaur 'ground level'
1763 dinoy% =dinobasey% :REMark Dino moves up from here
1764 score =0 :REMark Score is one point per step
1765 cloudy% =RND(36 TO 48) :REMark Location above cactus
1766 counter% =0 :REMark Dino Animation Frame Counter
1767 ticker =0 :REMark Speed Control Delay
1768 jumping% =0 :REMark <>0 Dino jumping (+ve=up -ve=down)
1769 demo_mode =0 :REMark Demo mode OFF=0 ON=1
1770 :
1771 REPEAT Dino_GAME
1772 OVER#3,0:CURSOR#3,6,158:CSIZE#3,1,1
1773 IF demo_mode%=1:CLS#3,3:PRINT#3,'Demo Mode On'
1774 IF demo_mode%=0:PRINT#3,'(D)emo Mode (P)ause (Q)uit Game'
1775 INK#3,dino_colour%:OVER#3,-1 :REMark Show Dino
1776 CURSOR#3,32,dinoy%:PRINT#3,CHR$(128+(counter% MOD 4));
1777 :
1778 REMark *** Speed control - ignore (slowdown-1) passes of the loop ***
1779 IF slowdown>0
1780 REPEAT wait
1781 ticker=ticker+1:IF ticker<slowdown:NEXT wait
1782 ticker=0:EXIT wait
1783 END REPEAT wait
1784 END IF

```

Demo Mode On

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

```

1785 key=CODE(INKEY$)
1786 SElect ON key
1787 =81,113:CURSOR#3,40,158:CLS#3,3:EXIT Dino_GAME :REMark (Q)uit
1788 =10,32 :REMark Manual Jump
1789 IF jumping%=0 AND demo_mode%=0
1790     jumping%=-1:dinoy%=dinobasey%:dy%=18
1791     IF sound_on%:BEEP 100,100,0,0,0,0,0
1792     END IF
1793 =80,112 :REMark (P)ause
1794 OVER#3,0:CURSOR#3,6,158:CLS#3,3:PRINT#3,'PAUSED':CLS#3,4
1795 k$=INKEY$(-1):CURSOR#3,40,158:CLS#3,3:IF k$=='Q':EXIT Dino_GAME
1796 =68,100 :REMark (D)emo_mode (auto-jumping)
1797 demo_mode%=NOT demo_mode% :dy%=18
1798 OVER#3,0:CURSOR#3,6,158:CSIZE#3,1,1:INK#3,7
1799 END SElect
1800 OVER#3,-1:INK#3,dino_colour% :REMark Erase Dino before PAN'ing
1801 CURSOR#3,32,dinoy% :PRINT#3,CHR$(128+(counter% MOD 4))
1802 OVER#3,0:y%=(dinobasey%-dinoy%) DIV 18 :REMark Detect Obstacle
1803 IF y%<3
1804     IF backg$(2-y%,5)<>' '
1805     IF sound_on% : BEEP 5000,0,50,50,1,0,0
1806     OVER#3,-1:INK#3,dino_colour%
1807     CURSOR#3,32,dinoy%:PRINT#3,CHR$(128+(counter% MOD 4));
1808     FOR a=1 TO 10:BLOCK#3,10,18,32,dinoy%,7:PAUSE 5
1809     OVER#3,0:CURSOR#3,12,158:CLS#3,3
1810     m = RND(1 TO 3)
1811     SElect ON m
1812     =1 : PRINT#3,'Ouch!  ';
1813     =2 : PRINT#3,'Oops!  ';
1814     =3 : PRINT#3,'Oh dear! ';
1815     END SElect
1816     EXIT Dino_GAME
1817 END IF
1818 END IF
1819 IF jumping%<>0
1820     IF jumping%=-1
1821         dinoy%=dinobasey%-72*(SIN(RAD(dy%)))
1822         dy%=dy%+18:IF dy%=90:jumping%=1 :REMark change direction
1823     ELSE
1824         dinoy%=dinobasey%-72*(SIN(RAD(dy%))) :REMark Upward
1825         dy%=dy%-18:IF dy%=0:jumping%=0:dinoy%=dinobasey%
1826     END IF
1827 ELSE
1828     IF demo_mode%=1
1829         IF backg$(2,10)<>' '
1830             jumping%=-1:dinoy%=dinobasey%:dy%=18
1831             IF sound_on% : BEEP 100,10,0,0,0,0,0
1832         END IF
1833     END IF
1834 END IF

```

PAUSED

M
E
N
U

C
H
O
I
S
E

D
E
T
E
C
T

O
B
S
T
A
C
L
E

D
I
N
O

J
U
M
P

```

1835 REMark *** PAN Cactus Display to Left - Height 4x18 Pixels -4(overlap)
1836 INK#3,cactus_colour%:CURSOR#3,0,pany%
1837 CHAR_INC#3,8,68: PAN#3,-8,3: CHAR_INC#3,8,18
1838 FOR y=2 TO 0 STEP -1:backg$(y)=backg$(y,2 TO 34)&' '
1839 REMark *** Add any extra Objects to right of background array
1840 IF cactus_gap%>0
1841   cactus_gap%=cactus_gap%-1 :REMark Decrement count to next cactii
1842   IF cactus_gap%=0
1843     cactus_run%=1 :REMark how far are we across a cactus?
1844     cactus_wide=RND(1 TO 3) :REMark cactus width 1-3
1845     cactus_high=RND(1 TO 3) :REMark cactus height 1-3
1846   END IF
1847 END IF
1848 IF cactus_run%>0
1849   SElect ON cactus_high
1850   =1:REMark cactus 1 high
1851     base_char =131+(cactus_wide=2)+(3*(cactus_wide=3))
1852     backg$(2,34)=CHR$(base_char+cactus_run%)
1853   =2:REMark cactus 2 medium
1854     base_char =137+(cactus_wide=2)+(3*(cactus_wide=3))
1855     backg$(1,34)= CHR$(base_char+cactus_run%)
1856     backg$(2,34)= CHR$(base_char+6+cactus_run%)
1857   =3:REMark cactus 3 low
1858     base_char =149+(cactus_wide=2)+(3*(cactus_wide=3))
1859     backg$(0,34)=CHR$(base_char+cactus_run%)
1860     backg$(1,34)=CHR$(base_char+6+cactus_run%)
1861     backg$(2,34)=CHR$(base_char+12+cactus_run%)
1862   END SElect
1863   cactus_run%=cactus_run% + 1
1864   IF cactus_run%>cactus_wide
1865     cactus_gap%=RND(10 TO 20):REMark start a new gap between cacti
1866     cactus_run%=0:cactus_high=0:cactus_wide=0 :REMark RESet
1867   END IF
1868 END IF
1869 :
1870 REMark *** Move Ground : Update Cacti
1871 OVER#3,1:CSIZE#3,1,0:INK#3,ground_colour%
1872 CURSOR#3,260,pany%+50:PRINT#3,CHR$(RND(168 TO 178));
1873 CSIZE#3,1,1:INK#3,cactus_colour%
1874 FOR y=2 TO 0 STEP -1:CURSOR#3,260,pany%+(18*y):PRINT#3,backg$(y,34);
1875 OVER#3,0
1876 :
1877 REMark *** Handle the score IF >99999 wrap around
1878 STRIP#3,80:INK#3,7:score=score+1:IF score>99999:score=0
1879 CURSOR#3,180,2:PRINT#3,FILL$(0,5-LEN(score))&score:STRIP#3,0
1880 :
1881 REMark *** Speed up Slightly as Score gets Higher (if set to do so)
1882 IF every%>0
1883   IF (score MOD every%)=0
1884     slowdown=slowdown-(slowdown*byPerCent/100)
1885   END IF
1886 END IF
1887 :

```

U
P
D
A
T
E

C
A
C
T
I

Note: UPDATE SCORE

Note: Adjust Speed

```

1888 REMark *** Increment Dino Frame Counter
1889 counter%=counter%+1:IF counter% >4:counter%=0
1890 :
1891 REMark *** Handle Clouds Once Every 4 Frames
1892 IF (counter% MOD 4)=0
1893     CURSOR#3,0,cloudy%:CHAR_INC#3,8,36:PAN#3,-8,3:CHAR_INC#3,8,18
1894     cloud_gap%=cloud_gap%-1
1895     IF cloud_gap%=0
1896         cloud_run%=1 :REMark how far across
1897         cloud_wide=RND(2 TO 3) :REMark cloud width 1-3
1898         cloud_high=RND(0 TO 1) :REMark cloud height 0-1
1899         cloudyy%=cloudy% + RND(0 TO 26-(9*cloud_high))
1900         rn=RND(1 TO 5) :REMark Cloud - Random Colour/Stipple
1901         SELECT ON rn
1902             =1:cloud_col%=7
1903             =2:cloud_col%=56
1904             =3:cloud_col%=63
1905             =4:cloud_col%=248
1906             =5:cloud_col%=255
1907         END SELECT
1908         SELECT ON cloud_wide
1909             =1:IF RND(1 TO 10)=10
1910                 cloud_base_char=191:cloud_high=0:cloud_col%=6
1911             ELSE
1912                 cloud_base_char=179+(RND>.5) :REMark small cloud
1913             END IF
1914             =2:cloud_base_char=180+(2*(RND>.5))
1915             =3:cloud_base_char=184+(3*(RND>.5))
1916         END SELECT
1917     END IF
1918     IF cloud_run%>0
1919         CSIZE#3,1,cloud_high:INK#3,cloud_col%
1920         CURSOR#3,33*8,cloudy%:PRINT#3,CHR$(cloud_base_char+cloud_run%);
1921         INK#3,7:cloud_run%=cloud_run%+1
1922     IF cloud_run%>cloud_wide
1923         cloud_gap%=RND(10 TO 20):cloud_run%=0:cloud_wide=0
1924     END IF
1925     END IF
1926 END IF
1927 END REPEAT Dino_GAME
1928 CSIZE#3,1,1:PRINT#3,'< Another Game Y/N >'
1929 REPEAT Ans_ip
1930     IF KEYROW(7)=64:EXIT Dino_program
1931     IF KEYROW(5)=64:EXIT Ans_ip
1932 END REPEAT Ans_ip
1933 IF score>hiscore:hiscore=score
1934 END REPEAT Dino_program
1935 CSIZE#3,0,0:CLS#3:FontSets
1936 END DEFine

```

U
P
D
A
T
E

C
L
O
U
D

< Another Game Y/N >

1938 DEFINE PROCEDURE DSpeed

```
1939 CSIZE#3,0,0 :CURSOR#3,232, 2:PRINT#3,'Speed'
1940 sl=slowdown_steps:CURSOR#3,224,12:PRINT #3,'- +'
```

1941 REPEAT Sp_lp

```
1942 CURSOR#3,232,12:PRINT#3,FILL$(0',5-LEN(sl));sl:K=CODE(INKEY$(-1))
```

```
1943 IF K=43 OR K=61:IF sl<50000:sl=sl+500
```

```
1944 IF K=45 OR K=95:IF sl>= 500:sl=sl-500
```

```
1945 IF K=10:slowdown_steps=sl:EXIT Sp_lp
```

1946 END REPEAT Sp_lp

```
1947 CURSOR#3,220,12:PRINT#3,' ';FILL$(0',5-LEN(sl));sl;' '
```

1948 END DEFINE DSpeed

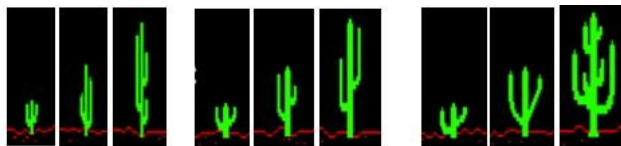
DINO Run HI 00000 Speed
~25000 +

Note: Adjust with +/- then Enter

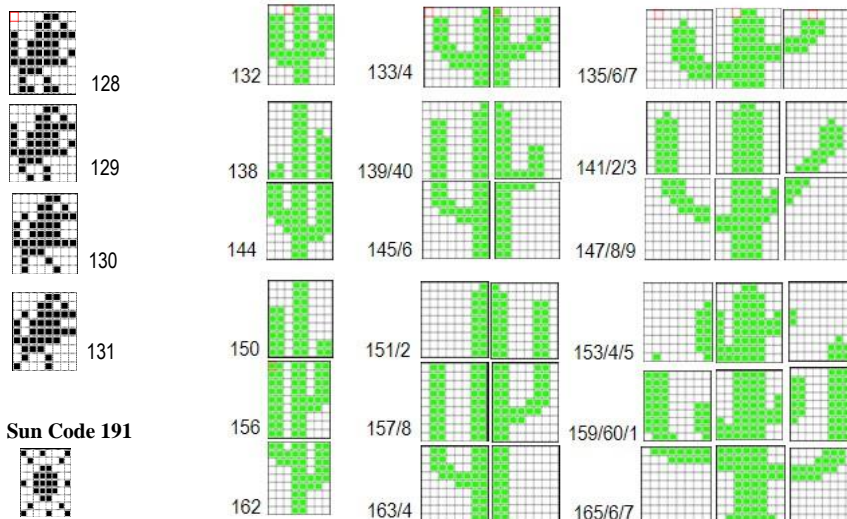
DINO_fnt uses the extended character code set 128 to 191

Cacti Codes 132 to 167

[3 widths 3 heights]



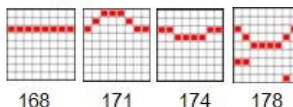
Dino Codes 128 to 131



Sun Code 191

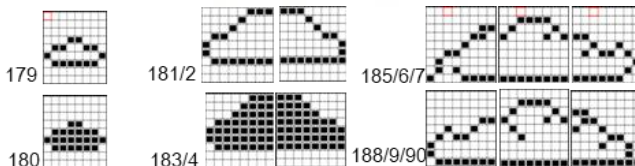


Ground Level Codes 168 to 178



Clouds Codes 179 to 190

[Three widths White/Black]



QBITS ARCADE - Giro Rescue

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

1950 REMark **ARCADE - Giro_Rescue** based on version by Henry Wrighton Aug 1989

```

1952 DEFine PROCEDURE Giro_Rescue
1953 Init_Giro
1954 REPEAT Giro
1955   set_level :REMark New Game Setup
1956   REPEAT Rescue
1957     tcap=tcap+1:cap=tcap:fu=90:sh=90:Giroscore sc
1958     INK#3,4:set_junk:INK#3,7:set_pod:main
1959     IF fin=1 OR tcap>15:EXIT Rescue :ELSE fin=0
1960     REPEAT Bonus
1961       fuel 2:sc=sc+(2*(tcap-9)):Giroscore sc:beeps 1
1962       PAUSE 5:IF fin:EXIT Bonus
1963     END REPEAT Bonus
1964   END REPEAT Rescue
1965   OVER#3,0:fin=0:i=0:CUSOR#3,32,80:PRINT#3,'Another Game y/n'
1966   REPEAT Key_Ip
1967     i=(i+1) MOD 7:INK#3,i:AT#3,4,6:PAUSE 5:PRINT#3,'GAME OVER'
1968     IF KEYROW(5)=64 THEN EXIT Key_Ip
1969     IF KEYROW(7)=64 THEN EXIT Giro
1970   END REPEAT Key_Ip
1971 END REPEAT Giro
1972 CLS#3:CSIZE#2,0,0:FontSets
1973 END DEFine

```

Note: Load Giro_fnt and Press (G)ame



```

1975 DEFine PROCEDURE Giroscore(sc)
1976 CSIZE#2,1,1:CUSOR#2,200,42:PRINT#2,FILL$('0',5-LEN(sc))&sc Note: sc score
1977 CUSOR#2,282,42:PRINT#2,tcap-9:CSIZE#2,0,0 Note: tcap levels
1978 END DEFine

```

```

1980 DEFine PROCEDURE set_level
1981 DIM al(36,19):BLOCK#3,276,140,0,22,0:INK#3,7
1982 sh=90:sc=0:tcap=9:pod$='@ @ @ @ @':pm=1:junk$='è è ì ì ì'
1983 FOR i=1 TO 80:INK RND(1 TO 5):POINT#3,RND(5 TO 110),RND(12 TO 84)
1984 INK#3,4:FOR K=1 TO 12:set_junk
1985 END DEFine

```

Note: pod\$ CHR\$(139 to 143)



Note: junk\$ CHR\$(144 to 148)

```

1987 DEFine PROCEDURE set_pod
1988 FOR i=1 TO tcap
1989   REPEAT pp
1990     ay=RND(3 TO 14):ax=RND(2 TO 20)
1991     IF ax<>11 AND ay<>8 AND al(ax,ay)=0:EXIT pp
1992   END REPEAT pp
1993   al(ax,ay)=1:AT#3,ay,ax:PRINT#3,"":beeps 5
1994 END FOR i
1995 END DEFine

```

Note: Escape Pod CHR\$(149)



```

1997 DEFine PROCEDURE set_junk
1998 REPEAT junk
1999   x=RND(2 TO 20):y=RND(3 TO 14):IF x<>10 AND y<>8:EXIT junk
2000 END REPEAT junk
2001 al(x,y)=2:AT#3,y,x:PRINT#3,junk$(RND(1 TO 5)):beeps 6
2002 END DEFine

```




```

2004 DEFine PROCEDURE main
2005 yy=0:xx=0:x=120:y=80:ix=2:iy=1:fu=90:fin=0:pm=1
2006 OVER#3,0:BLOCK#3,90,4,52,170,5:BLOCK#3,90,4,180,170,5
2007 INK#3,7:CURSOR#3,x,y:PRINT#3,pod$(pm):OVER#3,-1
2008 REPEAT loop
2009   get_keys:PAUSE 5
2010   IF x+xx< 0 OR x+xx>260:xx=-xx :beeps 2
2011   IF y+yy<20 OR y+yy>150:yy=-yy :beeps 2
2012   IF al((x)/12,(y)/10)=1:pod_clear :beeps 3
2013   IF al((x)/12,(y)/10)=2:shield 1 :beeps 4
2014   CURSOR#3,x,y:PRINT#3,pod$(pm):IF fin :PAUSE 30:EXIT loop
2015   x=x+xx:y=y+yy:pm=(pm MOD 5)+1:CURSOR#3,x,y:PRINT#3,pod$(pm)
2016 END REPEAT loop
2017 END DEFine

```

```

2019 DEFine PROCEDURE get_keys
2020 K=KEYROW(1):tx=xx:ty=yy
2021 SElect ON K
2022   =128:yy=yy+iy
2023   =144:yy=yy+iy:xx=xx+ix
2024   =16 :xx=xx+ix
2025   =20 :yy=yy-iy:xx=xx+ix
2026   =4 :yy=yy-iy
2027   =6 :yy=yy-iy:xx=xx-ix
2028   =2 :xx=xx-ix
2029   =130:yy=yy+iy:xx=xx-ix
2030 END SElect
2031 SElect ON K=128,144,16,20,4,6,2,130:fuel 1
2032 IF xx>11 OR xx<-11 THEN xx=tx
2033 IF yy> 8 OR yy<-8 THEN yy=ty
2034 END DEFine

```



```

2036 DEFine PROCEDURE pod_clear
2037 al(x/12,y/10)=0:AT#3,y/10,x/12:PRINT#3,'':sc=sc+10
2038 Giroscore sc,l:cap=cap-1:IF cap=0:fin=2
2039 END DEFine

```

Note: CHR\$(149)



```

2041 DEFine PROCEDURE fuel(f)
2042 fu=f-f:BLOCK#2,f,4,196+fu,211,0:IF fu<=0:fin=1
2043 END DEFine

```



```

2045 DEFine PROCEDURE shield(s)
2046 sh=sh-s:BLOCK#2,s,4, 70+sh,211,0:IF sh<=0:fin=1
2047 END DEFine

```



```

2049 DEFine PROCEDURE beeps(b)
2050 SElect ON b
2051   =1:BEEP 100,0
2052   =2:BEEP 3000,0,200,2,3
2053   =3:BEEP 500,0
2054   =4:BEEP 1000,10,20,1,1
2055   =5:BEEP 1000,0,150,100,1
2056   =6:BEEP 1000,100
2057 END SElect
2058 END DEFine

```

```

2060 DEFine PROCEDURE Init_Giro
2061 CLS#3:FOR i=1 TO 100:INK#3,(RND(1 TO 5)):POINT#3,RND(5 TO 110),RND(10 TO 85)
2062 Star_Ship 40,50:PAUSE 20:CSIZE#3,2,1:OVER#3,1
2063 INK#3,6:FOR i=0 TO 1:CURLOR#3,1+i,1:PRINT#3,'GIRO RESCUE' Note: CHR$(128 to 138)
2064 INK#3,2:FOR i=0 TO 2:CURLOR#3,90+i,22:PRINT#3,'RED ALERT'
2065 CSIZE#3,2,0:OVER#3,0:INK#3,5
2066 RED_Alert 40,50:PAUSE 20:BEEP:CSIZE#3,0,0:INK#3,5
2067 CURSOR#3,58,132:PRINT#3,'Press Any Key to begin RESCUE'
2068 CURSOR#3,36,144:PRINT#3,'Use ←↑→ keys to Guide your Giro Ship'
2069 CURSOR#3,86,154:PRINT#3,'to pick up Survivors'
2070 PAUSE:FOR i=1 TO 9:PAUSE 3:BLOCK#3,276,i*16,0,90-i*8,0
2071 OVER#3,1:INK#3,7:RESTORE 1975
2072 FOR a=1 TO 4
2073 READ x,y,str$:FOR b=0 TO 1:CURLOR#3,x+b,y:PRINT#3,str$:END FOR b
2074 END FOR a
2075 DATA 146,9,'Score:',228,9,'Level:',2,166,'Shields:',148,166,'Fuel:'
2076 OVER#3,0:CURLOR#3,0,0:CSIZE#3,2,0
2077 END DEFine

```

```

2079 DEFine PROCEDURE Star_Ship(sx,sy)
2080 FILL#3,1:INK#3,7:CIRCLE#3,sx+4,sy-3,8:FILL#3,0
2081 FILL#3,1:OVER#3,1:LINE#3,sx+40,sy+8
2082 RESTORE 1988:FOR i=1 TO 12:READ x,y:LINE#3 TO sx+x,sy+y
2083 FILL#3,0:OVER#3,0:INK#3,0
2084 LINE#3,sx+12,sy TO sx+26,sy+4 TO sx+28,sy+6 TO sx+40,sy+8
2085 LINE#3,sx+44,sy+4 TO sx+32,sy+1 TO sx+30,sy+1.2
2086 CIRCLE#3,sx+4,sy-3,8:ARC#3,sx-4,sy-2 TO sx+6,sy-4,PI/2
2087 FOR i=1 TO 6:CIRCLE#3,sx+12.5+i*2.2,sy-4.2+i*.6,1
2088 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
2089 END DEFine

```

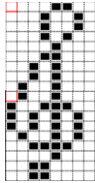
Note: The QBITS Intro Screen to Giro Rescue



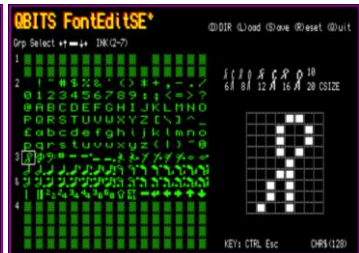
```

2091 DEFine PROCEDURE RED_Alert(sx,sy)
2092 RESTORE 2000:BEEP 0,10,100,5,-5,10,10,15
2093 FOR i=1 TO 8
2094 INK#3,RND(2 TO 5):CURSOR#3,70,42:PRINT#3,'Abandon Ship'
2095 READ x,y:INK#3,248:LINE#3,sx+12.5+i*2.2,sy-5+i*.6 TO sx+x,sy+y
2096 FILL#3,1:INK#3,7:CIRCLE#3,sx+x,sy+y,1:FILL#3,0:PAUSE 5
2097 READ x,y:INK#3,248:LINE#3,sx+6+i*2.2,sy+5+i*.6 TO sx+x,sy+y
2098 FILL#3,1:INK#3,7:CIRCLE#3,sx+x,sy+y,1:FILL#3,0:PAUSE 9
2099 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

```


QBITS QLXMUSIC_fnt

All eight columns of QL 8x9 Bitmap are required [CSIZE 1,0 - 3.1] The most difficult was the **GCleft** which has an upper and lower Font. For **Semibreve** a horizontal Ellipse, then an angled Ellipse Style Head for **Minim**, filled for **Crotchet** with Flags 1 & 2 added for **Quaver** and **Semiquaver**. That left **Dots** and **Sharps**. **RESTS** took a bit of experimentation, leaving just the Bar, End Bar and Beat Signatures to finish off.



2250 REMark Character Fonts for MUSIC Symbols

2252 DEFine PROCedure Music_Fonts

```

2225 CLS#6:INK#6,0:OVER#6,1:OVER#5,1:CHAR_USE#6,0,baseM:KeyBrd
2254 CSIZE#6,2,1:FOR i=0 TO 1:CURSOR#6,6,i+1:PRINT#6,'QL MUSIC Fonts'
2255 CSIZE#6,0,0:CURSOR#6,230,12:PRINT#6,'QLXMUSIC.fnt Musical SCORE Fonts'
2256 FOR i=1 TO 25 STEP 5:BLOCK#6,474,1,6,29+i,7:BLOCK#6,474,1,6,64+i,7
2257 FOR i=1 TO 15 STEP 3:BLOCK#6,474,1,6,100+i,7:BLOCK#6,474,1,6,120+i,7
2258 CSIZE#6,1,1:INK#6,0:STRIP#6,4:OVER#6,1:CURSOR#6,22,28:PRINT#6,CHR$(130)
2259 CURSOR#6,8,20:PRINT#6,CHR$(128):CURSOR#6,8,38:PRINT#6,CHR$(129)
2260 n=0:FOR i=132 TO 175:CURSOR#6,36+n,30:PRINT#6,CHR$(i):n=n+10:END FOR i
2261 CSIZE#6,0,0:REMARK CSIZE 1,1 RESTS      NOTE & DOTTED NOTE
2262 QBPrnt 5,0,6,20,52,'εΙΝ'θ εδδ εφδζ      θλδς & θλζδδθ θλζδ'
2263 QBPrnt 6,0,6,360,50,'+':QBPrnt 5,0,6,366,52,'ιCαCμi':REMARK Sharp
2264 QBPrnt 6,0,6,32,67,'Cclef'      Fclef'
2265 QBPrnt 5,0,5,68,68,'ζαμ εCφ+υεδδου      εCφ+υεδδου'
2266 QBPrnt 5,0,5,52,76,'ΑΛΛΙCΙC εCφ+υεδδου      εΙΝ'θ (δε)'
2267 CURSOR#6,0,0:CURSOR#6,6,3,1      :CURSOR#6,166,63:PRINT#6,CHR$(130)
2268 CURSOR#6,136,55:PRINT#6,CHR$(128):CURSOR#6,136,73:PRINT#6,CHR$(129)
2269 CSIZE#6,1,0
2270 QBPrnt 6,0,6,296,66,'Sharp ε Shift « CTRL »'
2271 QBPrnt 5,0,5,296,76,'εCφ+υεδδου      εCφ+υεδδδδδδδδδδδδδδ'
2272 REMARK      CHR$(131) CHR$(184-----185)
2273 CURSOR#6,10,97:PRINT#6,CHR$(128):CURSOR#6,10,106:PRINT#6,CHR$(129)
2274 CURSOR#6,22,102:PRINT#6,CHR$(181):CURSOR#6,10,124:PRINT#6,CHR$(130)
2275 n=0:FOR i=132 TO 175:CURSOR#6,36+n,103:PRINT#6,CHR$(i):n=n+10:END FOR i
2276 QBPrnt 5,0,6,20,92,'εΙΝ'θ εδδ      εCφ+υεδδδδδδδδδδδδδδ'
2277 QBPrnt 6,0,6,28,123,'Bar μ End Bar'
2278 QBPrnt 5,0,5,118,124,'εCφ+υεδδδδδδδδδδ'      :REMARK CHR$(176-177)
2279 CURSOR#6,200,117:PRINT#6,CHR$(128):CURSOR#6,200,126:PRINT#6,CHR$(129)
2280 QBPrnt 6,0,8,210,123,'φιCεCε'      :REMARK 2/2 2/4 3/44/4 3/8 6/8
2281 QBPrnt 5,0,5,264,124,'εCφ+υεδδδδδδδδδδ'      :REMARK CHR$(178-183)
2282 QBPrnt 6,0,9,3,346,123,'C++t++'
2283 QBPrnt 5,0,5,404,124,'εCφ+υεδδδδδδδδδδ'      :REMARK CHR$(186-191)
2284 END DEFine

```

2286 DEFine PROCedure KeyBrd

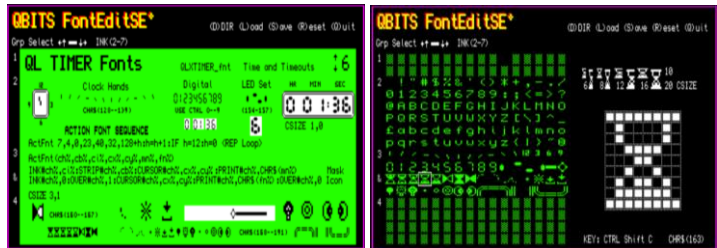
```

2287 BLOCK 281,32,123,181,0:FOR i=0 TO 27:BLOCK 9,30,124+i*10,182,7
2288 FOR i=0 TO 27
2289 IF i=0 OR i=3 OR i=7 OR i=10 OR i=14 OR i=17 OR i=21 OR i=24:NEXT i
2290 BLOCK 9,18,120+(i*10)-1,182,0:BLOCK 7,17,120+(i*10),182,2
2291 END FOR i
2292 END DEFINE

```

QBITS QLXTIMER_fnt

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 **ActFnt**. [Action Font]



2300 REMark Character Fonts for Timeouts and Clocks

2302 DEFine PROCEDURE Timer_Fnts

```

2303 STRIP#5,4:CLS#7:OVER#7,1:INK#7,0:CHAR_USE#7,0:baseT
2304 CSIZE#7,2,1:FOR i=0 TO 1:CUSOR#7,6+i,4:PRINT#7,'QL TIMER Fnts'
2305 CSIZE#7,0,0:CUSOR#7,230,12:PRINT#7,'QLXTIMER_fnt Time and Timeouts'
2306 REMark *** Clock Display
2307 CSIZE#7,1,0:FOR a=1 TO 12:CUSOR#7,44+a*12,40:PRINT#7,CHR$(127+a)
2308 QBPrnt 7,0,6,90,30,'Clock Hands'
2309 QBPrnt 5,0,5,90,52,'éÇþ÷ûëïðððéúú' :REMark CHR$(128-139)
2310 cx%=18:cy%=36:BLOCK#7,22,25,cx%+1,cy%+2,7
2311 CUSOR#7,cx%,cy% :PRINT#7,CHR$(184);CHR$(185);CHR$(186) :REMark --
2312 CUSOR#7,cx%,cy%+9 :PRINT#7,CHR$(187);CHR$(32);CHR$(188) :REMark |
2313 CUSOR#7,cx%,cy%+18:PRINT#7,CHR$(189);CHR$(190);CHR$(191) :REMark --
2314 CUSOR#7,28,30:PRINT#7,CHR$(140) :REMark 12
2315 CUSOR#7,11,48:PRINT#7,CHR$(143); CHR$(141) :REMark 9 3
2316 CUSOR#7,28,64:PRINT#7,CHR$(142) :REMark 6
2317 REMark *** Digital Display
2318 QBPrnt 7,0,6,234,28,'Digital LED Set'
2319 QBPrnt 5,0,5,386,28,'Ç Sñ' :REMark HR MIN SEC
2320 QBPrnt 7,0,7,228,40,'éëïïïðððúúþ' :REMark 0123456789
2321 BLOCK#7,102,23,377,40,7:STRIP#7,7:CUSOR#7,438,46:PRINT#7,':'
2322 CUSOR#7,376,38:PRINT#7,':<>>>>>>>>>>' :REMark ----- Top
2323 CUSOR#7,376,47:PRINT#7,',' :REMark | Left
2324 CUSOR#7,472,47:PRINT#7,',' :REMark Right |
2325 CUSOR#7,376,54:PRINT#7,':+++++++' :REMark ----- Bottom
2326 QBPrnt 5,0,5,222,52,'éï ðÇþ éððú ùëïððéúú' :REMark CTRL 0--9 134-
2327 QBPrnt 7,0,5,12,130;CSIZE 3,1':QBPrnt 7,0,6,380,66,'CSIZE 1,0'
2328 REMark *** Active Timeouts
2329 OVER#7,0:STRIP#7,4:QBBD 5,0,64,70,'ACTION FONT SEQUENCE'
2330 QBPrnt 7,0,6,12,82,'ActFnt 7,4,0,23,40,32,128+h:h=h+1:IF h=12:h=0 <REP Loop>'
2331 QBPrnt 7,0,6,12,96,'ActFnt(ch%,cb%,ci%,cx%,cy%,mn%,Fg%)'
2332 QBPrnt 7,0,6,12,107,'INK#ch%,ci%;STRIP#ch%,cb%;CUSOR#ch%,cx%,cy%'
2333 QBPrnt 7,0,6,278,107,':PRINT#ch%,CHR$(mn%) Mask'
2334 QBPrnt 7,0,6,12,116,'INK#ch%,0:OVER#ch%,1:CUSOR#ch%,cx%,cy%'
2335 QBPrnt 7,0,6,246,116,':PRINT#ch%,CHR$(Fg%):OVER#ch%,0 Icon'
2336 CUSOR#5,450,4:PRINT#5,':↑':CUSOR#5,450,14:PRINT#5,':↓'
2337 FOR a=0 TO 7:CUSOR#7,40+a*9,163:PRINT#7,CHR$(160+a)
2338 QBPrnt 5,0,5,42,148,'éÇþ÷ûëðððððð' :REMark CHR$(160-167)
2339 FOR a=0 TO 15:CUSOR#7,144+a*10,163:PRINT#7,CHR$(168+a)
2340 QBPrnt 5,0,5,314,164,'éÇþ÷ûëððððððú' :REMark Chr$(184-191)
2341 FOR a=0 TO 7:CUSOR#7,394+a*10,163:PRINT#7,CHR$(184+a)
2342 ci=0:c2=0:c3=0:c4=0:c5=0:c6=0:c7=1:c8=0:fx%=250:xx%=4:cnt%=0:td%=5
2343 h=0:hr%=0:min%=0:sec%=0:ic%=0:bc%=4:mc%=0:BLOCK#7,130,14,236,144,7
2344 END DEFINE

```

```

2346 DEFINE PROCEDURE Action_Fonts
2347 REPEAT Fnt_Ip
2348 IF KEYROW(1)=64 :EXIT Fnt_Ip:END IF
2349 CURSOR#7,460,4 :PRINT#7,td%
2350 IF td%>0:PAUSE td%*2:ELSE PAUSE :REMark Set Timer
2351 IF KEYROW(1)= 4:td%=td%+1 :IF td%>9:td%=9:PAUSE 1 :REMark Up/More
2352 IF KEYROW(1)=128:td%=td%-1:IF td%<1:td%=0:PAUSE 1 :REMark Down/Less
2353 CSIZE#7,1,0:sec%=sec%+1:Digits 7,0,4,236,64 :REMark Digital Clock
2354 cnt%=cnt%+1:fx%=fx%+xx%:BarFnt 7,250,350,fx%,146 :REMark Sliding Bar
2355 CSIZE#7,3,1:mn%=166:IF c1>4:mn%=167:END IF :REMark Mask check
2356 ActFnt 7,7,0,0, 23,40, 32,128+h :h=h+1 :IF h=12:h=0 :REMark HR Clock
2357 ActFnt 7,4,7,0, 18,140,mn%,160+c1:c1=c1+1 :IF c1>5:c1=0 :REMark Egg Timer
2358 ActFnt 7,4,0,0,144,138, 32,168+c2 :c2=c2+1 :IF c2=4:c2=0 :REMark Spin
2359 ActFnt 7,4,0,0,174,138, 32,172+c3 :c3=c3+1 :IF c3=2:c3=0 :REMark + *
2360 ActFnt 7,4,0,0,204,138, 32,174+c4 :c4=c4+1 :IF c4=2:c4=0 :REMark Arrow
2361 ActFnt 7,4,7,0,378,138,176,177+c5 :c5=c5+1 :IF c5=2:c5=0 :REMark Lamp
2362 ActFnt 7,4,0,0,404,138, 32,179+c6 :c6=c6+1 :IF c6=3:c6=0 :REMark . o O
2363 ActFnt 7,4,0,0,434,138, 32,182+c7 :c7=c7+1 :IF c7=2:c7=0 :REMark L Eye
2364 ActFnt 7,4,0,0,454,138, 32,182+c8 :c8=c8+c7:IF c8=2:c8=0 :REMark R Eye
2365 END REPEAT Fnt_Ip
2366 END DEFINE

```



REPEAT Fnt_Ip cycles the counters for each FONT sequence.

PROC - ActFnt Prints Font to Screen

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

Fonts Masks



Mask is printed First then Icon FONT is overlayed on top.

Counter incremented and Next Font sequence is Printed.

Once the sequence is complete the counter is Reset.

```

1176 DEFINE PROCEDURE ActFnt(bc%,mc%,fc%fx%,fy%,mn%,Fg%)
1177 REMark Action BackGnd STRIP bc% INK ic% Mask mn% Font Fg%
1178 INK#7,mc%:STRIP#7,bc%:CURSOR#7,fx%,fy%:PRINT#7,CHR$(mn%) :REMark Mask
1179 INK#7,fc%:OVER#7,1 :CURSOR#7,fx%,fy%:PRINT#7,CHR$(Fg%):OVER#7,0 :REMark Font
1180 END DEFINE

```



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

Lamp Mask

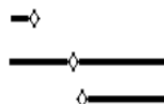


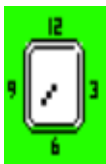
The Sliding Bar has two parts, the Bar and the Slider. In this example the Bar grows left to right with the Slider to max. The Slider then cycles across the Bar until it reaches a set number of loops. As the Slider moves the Bar length then diminishes left to right. The process repeats with counter Reset.

```

2368 DEFINE PROCEDURE BarFnt(ch%,lx%,rx%,fx%,fy%)
2369 IF fx%<lx% OR fx%>rx%:xx%=-xx% :BLOCK#7,8,10,rx%+4,fy%,7
2270 IF fx%>rx% AND cnt%<60:fx%=lx%
2271 IF fx%>rx% AND cnt%>60:mc%=7:fx%=lx%
2272 IF fx%<lx% AND cnt%>99:mc%=0:cnt%=0
2273 STRIP#ch%,7:INK#ch%,mc%:CURSOR#ch%,fx%,fy% :PRINT#ch%,CHR$(150)
2274 INK#ch%,0 :CURSOR#ch%,fx%+xx%,fy%:PRINT#ch%,CHR$(159)
2275 END DEFINE

```



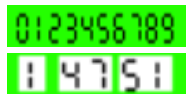


The simple Analogue Clock uses Fonts CHR\$(128 to 139) one for each of the twelve-hour hand positions. CHR\$(140 to 143) 5x5Bit Fonts **12 3 6 9**



Clock Border Fonts CHR\$(184 -191)

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

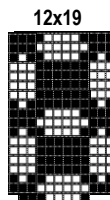


This continued with a more weighted Style of LED with numbers using just Four Fonts CHR\$(154-160). To create the numbers 0-9 an Array LED(9,7) is initialised and set with relative entries CHR\$(n) or zero if not used. For example: ONE would be LED(1,6 & 7) set to CHR\$(157) the rest set to zeros.

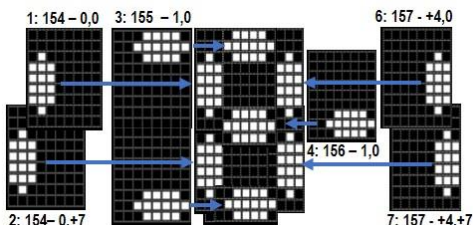


```
2377 DEFINE PROCEDURE Digits(ch%,bc%,ic%,fx%,fy%)
2378 IF sec%=60:sec%=0:min%=min%+1:END IF :st% =sec% DIV 10:su%=sec% MOD 10
2379 IF min%=60:min%=0:hr% =hr%+1 :END IF :mt%=min% DIV 10:mu%=min% MOD 10
2380 IF hr%=10:hr%=0:min%=0:sec%=0:END IF :STRIP#ch%,7
2381 sec$=CHR$(144+st%)&CHR$(144+su%):CURSOR#ch%,fx%+30,fy%:PRINT#ch%,sec$
2382 min$=CHR$(144+mt%)&CHR$(144+mu%):CURSOR#ch%,fx%+12,fy%:PRINT#ch%,min$
2383 hr$=CHR$(144+hr%):CURSOR#ch%,fx%,fy%:PRINT#ch%,hr$:STRIP#ch%,4
2384 Prt_LED 446,43,st% :Prt_LED 462,43,su% :Prt_LED 332,62,su%
2385 Prt_LED 406,43,mt%:Prt_LED 424,43,mu%:Prt_LED 382,43,hr%
2386 END DEFINE
```

```
2388 DEFine PROCEDURE Prt_LED(dx%,dy%,dn%)
2389 BLOCK#7,17,18,dx%-2,dy%,7
2390 CURSOR#7,dx% ,dy% :PRINT#7,CHR$(LED(dn%,1)):OVER#7,1
2391 CURSOR#7,dx% ,dy%+7:PRINT#7,CHR$(LED(dn%,2))
2392 CURSOR#7,dx%+1,dy% :PRINT#7,CHR$(LED(dn%,3))
2393 CURSOR#7,dx%+1,dy%+1:PRINT#7,CHR$(LED(dn%,4))
2394 CURSOR#7,dx%+1,dy%+8:PRINT#7,CHR$(LED(dn%,5))
2395 CURSOR#7,dx%+4,dy% :PRINT#7,CHR$(LED(dn%,6))
2396 CURSOR#7,dx%+4,dy%+7:PRINT#7,CHR$(LED(dn%,7)):OVER#7,0
2397 END DEFine
```

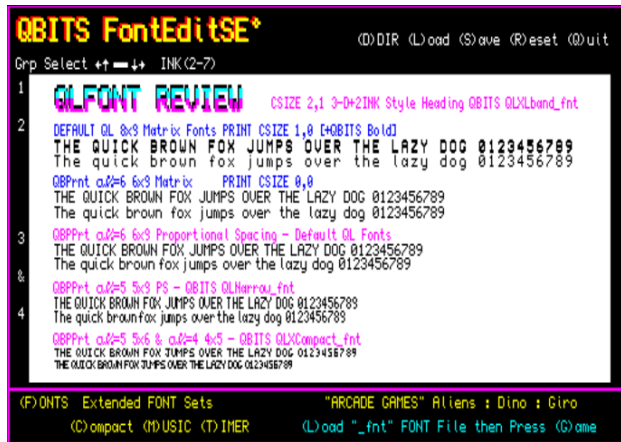


```
2399 DEFINE PROCEDURE Set_LED
2400 DIM LED(9,7):RESTORE 3192:FOR a=0 TO 9:FOR b=1 TO 7:READ LED(a,b)
2401 DATA 154,154,155, 0,156,157,157
2402 DATA 0, 0, 0, 0, 0,157,157
2403 DATA 0,154,155,156,156,157, 0
2404 DATA 0, 0,155,156,156,157,157
2405 DATA 154, 0, 0,156, 0,157,157
2406 DATA 154, 0,155,156,156, 0,157
2407 DATA 154,154,155,156,156, 0,157
2408 DATA 0, 0,155, 0, 0,157,157
2409 DATA 154,154,155,156,156,157,157
2410 DATA 154, 0,155,156,156,157,157
2411 END DEFINE
```



QBITS FontEditSE+ - FONT Review

The Review looks at various ways to display Character Fonts. **QLXLband_fnt** is used to **OVER PRINT** in a different INK colour the lower half of the Heading styled with a 3-D appearance. CSIZE and OVER are used with **QBITS PROCEDURE**'s to PRINT the QL Default or **QLXNarrow_fnt** Font Strings in **BOLD**, set to a **Character Width**, or **Proportional Spacing**. **QLXCompact_fnt** is a Small Font useful for labelling.



2500 REMark QBITS_FONT_Review

2501 DEFINE PROCEDURE FONT_Review

```

2502 CSIZE#5,2,1:OVER#5,1:PAPER#5,7:CLS#5:CHAR_USE#5,0,baseC
2503 INK#5,0:FOR i=1 TO 2:CUSOR#5,18+i,4:PRINT#5,'QLFONT REVIEW'
2504 INK#5,5:FOR i=2 TO 3:CUSOR#5,18+i,6:PRINT#5,'QLFONT REVIEW'
2505 INK#5,3:FOR i=2 TO 3:CUSOR#5,18+i,6:PRINT#5,'THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG 0123456789'
2507 CSIZE#5,1,0:CHAR_USE#5,0,baseI
2508 str1$='THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG 0123456789'
2509 str2$='The quick brown fox jumps over the lazy dog 0123456789'
2510 QBPrint 5,0,20,37,str1$ :QBPrint 5,0,8,20, 46,str2$
2511 QBPrint 5,0,6,20, 67,str1$ :QBPrint 5,0,6,20, 76,str2$
2512 QBPrint 5,0,6,20, 98,str1$ :QBPrint 5,0,6,20,106,str2$
2513 CHAR_USE#5,baseN,baseI
2514 QBPrint 5,3,5,200,12,'CSIZE 2,1 3D+2INK Style Heading QBITS QLXStripe_fnt'
2515 QBPrint 5,1,5,20, 28,'DEFAULT QL 8x9 Matrix Fonts PRINT CSIZE 1,0 [+QBITS Bold]'
2516 QBPrint 5,1,5,20, 58,'QBPrint cw%=6 6x9 Matrix Char width = CSIZE 0,0'
2517 QBPrint 5,3,5,20, 89,'QBPrint cw%=6 6x9 Proportional Spacing - Default QL Fonts'
2518 QBPrint 5,3,5,20,121,'QBPrint cw%=5 5x9 PS - QBITS QLXNarrow_fnt'
2519 QBPrint 5,3,5,20,150,'QBPrint cw%=5 5x6 & cw%=4 4x5 - QBITS QLXCompact_fnt'
2520 QBPrint 5,0,5,20,129,str1$ :QBPrint 5,0,5,20,137,str2$
2521 QBPrint 5,0,5,20,158,'THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG 0123456789'
2522 QBPrint 5,0,4,20,165,'THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG 0123456789'
2523 OVER#5,0:PAPER#5,0:CHAR_USE#5,0,baseI
2524 END DEFINE

```

Note: QL Edit screen shows Lines 2171 & 2172 with Default extended printable Characters these are replaced by the Loaded _fnt file Characters when printing to the **CHAR_USE#ch**, defined Window.

QBITS FontEditSE+ - Scalable Bitmaps

CSIZE 0,0 Prints the first six columns and CSIZE 1,0 Prints all eight columns CSIZE 2,0 and 3,0 doubles the Print character columns to 12 and 16. CSIZE can also double the hight from 10 to 20 rows. A Simple Scaling Up method is to use BLOCK to represent each PIXEL and keep increasing its size and spacing.

3000 REMark Scalable Fonts

3002 **DEFine PROCEDURE FontScale**

3003 FSize=6:sx=1:sy=.8

3004 **REPEAT lp**

3005 IF KEYROW(5)=32:IF sx>1:sx=sx-.25:sy=sy-.2:FSize=FSize-2

3006 IF KEYROW(3)=32:IF sx<9:sx=sx+.25:sy=sy+.2:FSize=FSize+2

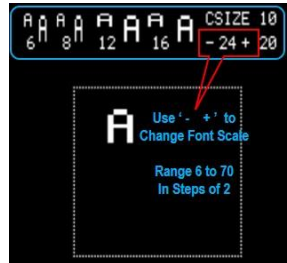
3007 BLOCK#4,111,89,47,41,0:IF KEYROW(1)=64:EXIT lp

3008 CURSOR 441,67:PRINT FILL\$(0',2-LEN(FSize))&FSize

3009 FOR r=0 TO 8:FontPrt 4,sx,sy,66,52 :END FOR r:PAUSE

3010 **END REPEAT lp**

3011 **END DEFine**



3013 **DEFine PROCEDURE FontPrt(ch%,sx,sy,posx,posy)**

3014 FOR c=0 TO 7

3015 IF Fnt\$(r+1,c+1)='1':pcol=col:ELSE pcol=bcol

3016 BLOCK#ch%,1+sx,1+sy,posx+c*sx,posy+r*sy,pcol

3017 END FOR c

3018 **END DEFine**

3020 **DEFine PROCEDURE ReviewPg2**

3021 PAPER#5,7:CLS#5 :CSIZE#5,2,1:OVER#5,1:

3022 INK#5,0:FOR i=1 TO 2:CURSOR#5,12+i,4:PRINT#5,'SCALABLE FONTS'

3023 INK#5,5:FOR i=2 TO 3:CURSOR#5,12+i,6:PRINT#5,'SCALABLE FONTS'

3024 CSIZE#5,1,0

3025 Test\$='The Quick Brown Fox':bl%=LEN(Test\$)-1

3026 bcol=7:col=0:cs=7:cd=7.3:posy=20

3027 FOR sx=1 TO 3 STEP .25

3028 posy=posy+sx*cd:sy=sx-.5E-2:PAUSE 5

3029 FOR i=0 TO bl%

3030 cn=CODE(Test\$(i+1)):posx=10+i*(cs*sx)

3031 IF cn<128:addr=FBaseA+2+cn*9

3032 IF cn>127:addr=FBaseB+2+(cn-127)*9

3033 FOR r=0 TO 8

3034 Fnt\$(r+1)=BIN\$(PEEK(addr+r),8)

3035 FontPrt 5,sx,sy,posx,posy

3036 END FOR r

3037 END FOR i

3038 END FOR sx

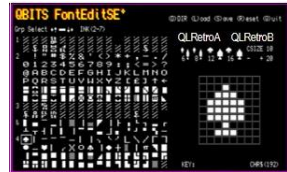
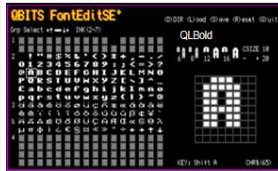
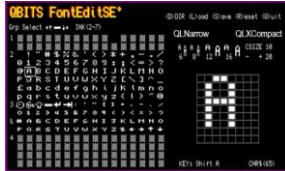
3039 PAUSE :PAPER#5,0:CLS#5:bcol=0:col=7:FontSets:FontGrid

3040 **END DEFine**



QBITS FontEditSE – FONT Sets

QLFontA_fnt	QL Default	32 - 127
QLFontB_fnt	QL Default Extended	128 - 191
QLRetroA_fnt	Retro Fonts	0 - 127
QLRetroB_fnt	Retro Fonts	128 - 225
QLNarrow_fnt	QBITS Narrow	32 - 127
QLBold_fnt	QBITS Bold	32 - 127
QLItalics_fnt	QBITS Italics	32 - 127



QLXCompact_fnt	QBITS 5x6 Fonts	128 - 191
QLXMUSIC_fnt	QBITS Notes etc	128 - 191
QLXTIMER_fnt	QBITS Time/Timeouts	128 - 191
QLXLband_fnt	QBITS Lower Capitals	128 - 191
QLXEmoticons_fnt	QBITS Facial Expressions	128 - 191



Aliens_fnt Game Fonts



Dino_fnt Game Fonts



Giro_fnt Game Fonts

