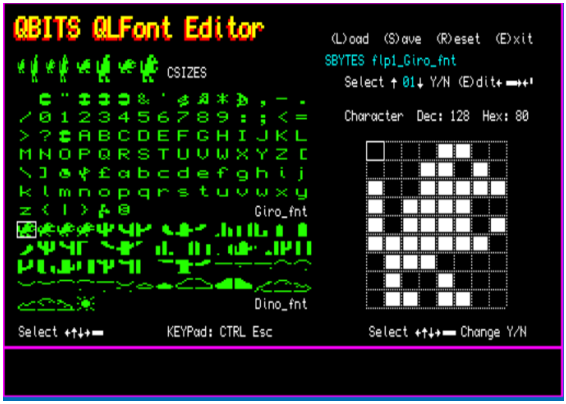


Introduction

Desktop publishing, created the need to digitised formats of letterforms, this resulted in "bitmap" Fonts. The disadvantage was in having separate fonts for each size and resolution. The current, generation of digital font technology provides for "scalable" outline fonts. They are smaller in memory size and faster to process. Analog drawings of letters are used to create an outline these are then digitized. Computer Applications are used to scale the outline for requested sizes and screen resolution. The outline scan converted to pixels and the resulting image written to screen. In today's world development of True Type technology allows for even greater degree of control over which pixels are turned on or left off.

Sinclair QL Fonts

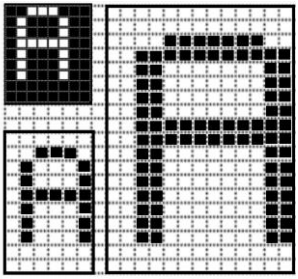
For 1980's home computing the O/S used ASCII Codes for Key Control, most of which produce printable Characters. The QDOS Bitmaps for these are stored in two Code sets 32 to 127, are the common alphanumeric, maths signs and backets etc. the second are Codes 128 to 191 the extended character set. The Format is the first two Bytes hold the Lowest Code number followed by Total number of Fonts. Then 9Bytes for each Font gives a 9x8 matrix used by QDOS to scale in two widths and two heights.

In addition, the spacing between characters is entirely flexible, so two additional spacings are added.

QDOS takes the Bitmap to generate a CSIZE range of

6 8 12 16 pixel's in width (spacing)

and two heights of 10 and 20 pixel's.



QDOS when creating different CSIZEs does not always display all the encoded Bits. This can lead to some interesting and at time 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 the 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 Font file for use with other Programs.

QBITS QLFont Editor Display

The screen display had to be easy and intuitive to use. At start-up the Default QL Character Fonts are displayed to screen. These are accessed from the default storage device which needs to be set before running the program (see Program line 1003 Dev$=).

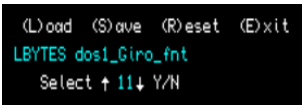
QBITS QLFont Editor Navigation

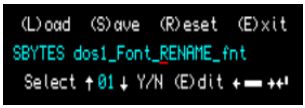
For the two Font Sets Navigate using the cursor keys and Select the Highlighted Character with Spacebar. This actions the Bitmap Grid, again navigate with cursor keys to highlight a Grid Cell and toggle the binary bit between 0 and 1 with Spacebar. Select 'N' to return without changing the existing Bitmap, 'Y' to write the changed bit pattern into memory.

QBITS (L)oad (S)ave (R)eset (E)xit

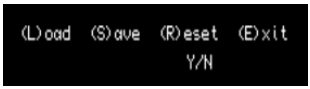
Access by pressing the bracketed Character. Information is displayed relative to the action requested. For Load and Save first select storage device with Up/Down cursor keys and Y/N.

For **Load** a search is made of available '_fnt' front files. A 'File Not found' will be returned if none are available. Use the Up/Down cursors to scroll through and make your choice then Y/N to load or abort. A program check is made to select the relevant Character Set Memory address, before the File is loaded and Fonts displayed to screen.





To **Save** use Up/Down cursors to select which Character Font set to Save. Included is a Line Editor to Rename the Font file. When ready to save a check is made and if device unavailable a 'DEVICE ERROR' is given. An 'Overwrite Y/N' is given if the file is detected as already existing. If good to go or an answer Y 'Saving…' is displayed.

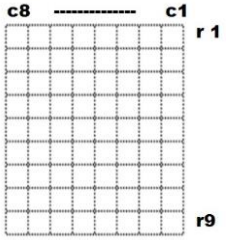


Press 'R' for **Reset** which prompt with 'Y/N', 'N' aborts and 'Y' reloads default Fonts. Pressing 'E' for **Exit** will again prompt with Y/N, 'N' returns to program, 'Y' will close opened channels and free Memory before halting the Program. If desired an LRUN can be added to start a Boot or Menu Program.

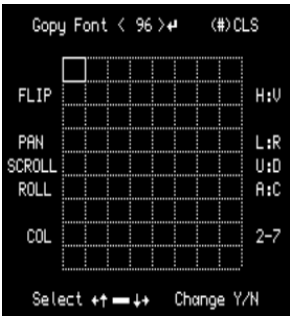
QBITS QLFont Editor

The heart of the editor is the Bitmap and what changes you can make to the character Fonts. It maybe changes to alphanumerical fonts to reflect Bold, Italics, your own stylised Fonts, futuristic or to depict some ancient language such as Cuneiform or Runes. For Retro Games transforming Fonts into game pieces might also be desirable.

Apart from toggling individual bites between 0 &1 what aids to Font design would be useful. When slight pattern changes are desired to Copy an existing Character into the Font Grid might be one. For Copy use keys **',< .>**' and Enter. Then being able to Flip **H**orizontally, or **V**ertically, Pan **L**eft or **R**ight, Scroll **U**p & **D**own and Rotate **A**nti-clockwise or **C**lockwise by 90*. For new designs (#) CLS clears the Bitmap setting to all '0'. Changing the Font INK colour use '2-7'.



Flip c1=c8 to c8=c1 Horizontal
r1=r9 to r9=r1 Vertical
Pan c1=c8 to c2=c1 c8=c7 Left
c1=c2 to c7=c8 c8=c1 Right
Scroll r1=r2 to r8=r9 r9=r1 Up
r2=r1 to r9=r8 r1=r9 Down
Roll r1,c1=r1,c8 r9,c1=r1,c1
r1,c1=r9,c1 r9,c8=r1,c8



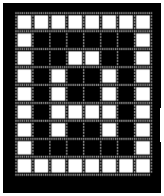
Showing the Fonts in their various CSIZES and different colours could be helpful in identifying any display problems.

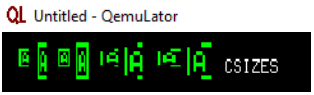
The Bitmap shows bits set to 0's & 1's for chosen Font. Returning from the Bitmap the Character under review is displayed in its different **CSIZES** top left with **KEYPad** below the Font charts showing the key(s) needed for printing the Character code.

QL Fonts & CSIZEs

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

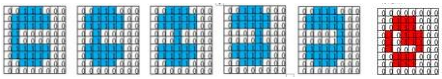
As an example, the bit Pattern 'A' shown was Surrounded with a box. Not all CSIZES showed the whole of the box and some QL O/S had different outcomes.





Program use of Font Designs

I chose two Programs, Giro Rescue and Dino that use redefined Fonts. Here Giro Rescue modifies the QLFont1 and Dino replaces the QLFont2 extended set.

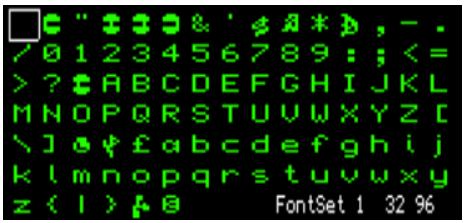


Giro Rescue

'! # $ % @' replaced for Rescue Pod

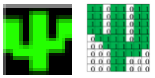
For Escape Pod '^'

'( ) * + _' for Ship Debry



Dino

Uses all of the extended Fonts.



QPC11 SMSQ\E

Qemulator

Conclusion, when using Fonts to create Retro Game features be aware of the bit pattern displays generated by the O/S for the different CSIZEs.

QBITS Program Code

To maintain compatibility across the various QL Platforms can have its drawbacks. The original QL came with only 128K of Ram after screen memory and other uses for heap memory etc. some progs were too large to load and run. QBITS QLFont Editor should load and run on a BBQL but requires TK2 to be present. The use of Arrays for holding '_Fnt' Filenames can overload available memory. It is therefore recommended to fit some extended memory.

For higher speed QL platforms as you scan through the Font Charts the relative Bitmap is displayed, for the BBQL this aspect is disabled and the Bitmap shown only when a Font is selected. The BBQL version is for standalone, but the code variances for the QPC2 are set for use with a Config file and QBITS Menu program.

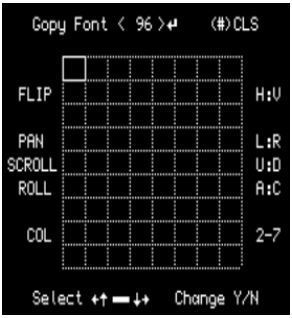
```
1000 REMark QBITS_FontEdit_v1 BBQL _v2 QL _v3 QPC2 (QBITS Font Editor 1991 release 2022)

1002 WMON:gx=0:gy=0                                                    BBQL
1003 DIM drv$(7,5):Dev$='flp1_':dm%=7  :REMark Device Drivers          BBQL

1002 OPEN_IN#9,'ram2_QBITSConfig':INPUT#9,gx\gy\dn$\Dev$\dn%\dm%       QPC2
1003 DIM drv$(15,5):FOR d=0 TO 15:INPUT#9,drv$(d):END FOR d:CLOSE#9    QPC2

1005 DIM Fnt$(9,8),Tmp$(9,8),File$(50,20):fm%=50   :REMark Font Arrays
1006 FBase1=ALCHP(875)             :REMark Memory for QLFont1_fnt
1007 FBase2=ALCHP(587)             :REMark Memory for QLFont2_fnt

1009 WHEN ERRor
1010  eck=1:CONTINUE
1011 END WHEN

1013 Init_Screen:FontMain

1015 DEFine PROCedure Init_Screen
1016 WINDOW#0,512, 32,gx,gy+224  :PAPER#0,0      :CSIZE#0,0,0:BORDER#0,1,3:CLS#0
1017 WINDOW#1,512,224,gx,gy      :PAPER#1,0      :CSIZE#1,0,0:BORDER#1,1,3:CLS#1
1018 WINDOW#2,512,224,gx,gy      :PAPER#2,0      :CSIZE#2,0,0:BORDER#2,1,3:CLS#2
1019 OPEN#3,scr_:WINDOW#3,276,178,gx+ 12,gy+32 :CSIZE#3,3,0:INK#3,4
1020 OPEN#4,scr_:WINDOW#4,136,112,gx+332,gy+88:CSIZE#4,3,0:INK#4,4
1021 CSIZE#2,2,1:OVER#2,1
1022 INK#2,2:FOR i=0 TO 1:CURSOR#2,6+i,8:PRINT#2,'QBITS QLFont Editor°'
1023 INK#2,6:FOR i=0 TO 1:CURSOR#2,8+i,6:PRINT#2,'QBITS QLFont Editor°'
1024 CSIZE#2,0,0:OVER#2,0:INK#1,7
1025 CURSOR#1,296, 18:PRINT#1,'(L)oad  (S)ave  (R)eset  (E)xit'
1026 CURSOR#1, 12,208:PRINT#1,'Select ⬅⬆  ⬇➡':BLOCK#1,12,3,70,212,7

1027 RESTORE 1028:FOR d=0 TO 7:READ str$:drv$(d)=str$                  BBQL
1028 DATA 'mdv1_','mdv2_','flp1_','flp2_','win1_','win2_','win3_','win4_'   BBQL

1027 :                                                                 QPC2 not used
1028 :                                                                 QPC2 not used
1029 END DEFine

1031 DEFine PROCedure FontReset(ck)
1032 IF ck=1:CURSOR#1,412,32:PRINT#1,'Y/N':PAUSE:IF KEYROW(5)<>64:RETurn
1033 LBYTES Dev$&'QLFont1_fnt',FBase1:cn1=96:cn1$='QLFont1_fnt'
1034 LBYTES Dev$&'QLFont2_fnt',FBase2:cn2=64:cn2$='QLFont2_fnt'
1035 FontSets cn1,cn2:FontName cn1$,cn2$:FontGrid:cn=32:col=7
1036 END DEFine

1038 DEFine PROCedure FontExit
1039 CURSOR#1,460,32:PRINT#1,'Y/N':PAUSE:IF KEYROW(5)<>64:RETurn
1040 CLOSE#4:CLOSE#3:CLS#1:RECHP FBase1:RECHP FBase2:LRUN Dev$&'QBITSProgs_v3'
1041 END DEFine
```

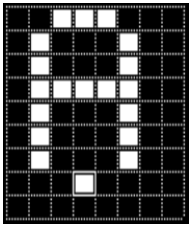
Note: Exit closes open channels and releases heap memory. STOP can be replaced with an LRUN command to return to another program such as LRUN flp1_Boot or win1_Progs_Menu etc.


```
1043 DEFine PROCedure FontMain
1044 dn%=6:cx=0:cy=2:x=1:y=1:FontReset 0:chk=0:eck=0 :REMark dn% default drive
1045 REPeat Main_lp
1046   IF cn=128:cy=9:cx=0
1047   IF cy=8  AND cx>5:cx=5:cn=127
1048   IF cy=13 AND cx>3:cx=3:cn=191
1049   cn=32+cx+(cy-2)*15:BLOCK#1,220,24,280,32,0:INK 7
1050   IF cn>127:cn=cn-9
1051   CURSOR 310,68:PRINT 'Font Code <    > Dec  ';HEX$(cn,8);' Hex'
1052   CURSOR 378,68:PRINT FILL$(' ',3-LEN(cn))&cn
1053   FontChar cx,cy,7                                    BBQL
1053   FontChar cx,cy,7          :FontPeek
1054   K=CODE(INKEY$(-1))    :BLOCK#3,260,22,0,0,0:BLOCK 140,10,140,208,0
1055   FontChar cx,cy,0                                    BBQL
1055   FontChar cx,cy,0          :IF K<>32:CLS#4:FontGrid
1056   SELect ON K
1057    =192:cx=cx-1:cn=cn-1 :IF cx< 0:cx=14:cy=cy-1:IF cy<2:cy=2:cx=0
1058    =200:cx=cx+1:cn=cn+1 :IF cx>14:cx= 0:cy=cy+1
1059    =208:cy=cy-1:cn=cn-15:IF cy< 2:cy= 2
1060    =216:cy=cy+1:cn=cn+15:IF cy>13:cy=13
1061    = 32:FontPeek:FontMod     :REMark Modify Char    BBQL
1061    = 32:FontMod              :REMark Modify Char    QPC2
1062    =108,76:FontLoad          :REMark (L)oad
1063    =115,83:FontSave          :REMark (S)ave
1064    =114,82:FontReset 1       :REMark (R)eset
1065    =101,69:FontExit          :REMark (E)xit
1066   END SELect
1067 END REPeat Main_lp
1068 END DEFine
```

Note: Switch from Main Character display to Bitmap display. To switch back press' N' for no change or 'Y' to update Character. Once Changes to a Font set are complete the updated Fonts can be saved.

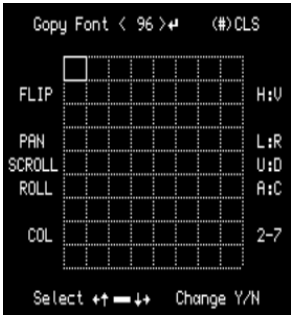
```
1070 DEFine PROCedure FontMod
1071 CURSOR#1,310, 68:PRINT#1,'Gopy Font <   >←  (#)CLS':CLS 4
1072 CURSOR#1,332,208:PRINT#1,'Select ←↑  ↓→  Change Y/N';cs=8:co=cn
1073 BLOCK 2,4,412,70,7:BLOCK 12,3,368,212,7:REMark Spacebar & Enter Tail
1074 RESTORE 1075:FOR i=1 TO 10:READ a,b,c$:CURSOR a,b:PRINT c$
1075 DATA 300,104,'FLIP',470,104,'H:V',300,128,'PAN',470,128,'L:R'
1076 DATA 292,140,'SCROLL',470,140,'U:D',300,152,'ROLL',470,152,'A:C'
1077 DATA 306,176,'COL',470,176,'2-7'
1078 FontChar cx,cy,7:rm=9:cs=8:co=cn
1079 REPeat Chg_lp
1080   CURSOR 378,68:PRINT FILL$(' ',3-LEN(cn))&cn
1081   FontBit x,y,7:K=CODE(INKEY$(-1)):FontBit x,y,248
1082   SELect ON K
1083     =192:x=x -1:IF x<1:x=1
1084     =200:x=x+1:IF x>8:x=8
1085     =208:y=y -1:IF y<1:y=1
1086     =216:y=y+1:IF y>9:y=9
1087     = 60,44:cn=cn-1:IF cn<32:cn=191        :REMark <, lower
1088     = 62,46:cn=cn+1:IF cn>191:cn=32        :REMark >. higher
1089     =104,72:FontFlip 9,-1,0,1              :REMark (H)orizontal Flip
1090     =118,86:FontFlip 0,1,10,-1             :REMark (V)ertical  Flip
1091     =108,76:FontSlid 0,8,1, 0,1            :REMark (L)eft  PAN  Grid
1092     =114,82:FontSlid 0,1,8,-1 0            :REMark (R)ight PAN  Grid
1093     =117,85:FontSlid 1,9,1, 1              :REMark (U)p SCROLL  Grid
1094     =100,86:FontSlid 1,1,9,-1              :REMark (D)n SCROLL  Grid
1095     =  97,65:FontRoll 1                    :REMark (A) Roll Anti-Clockwise
1096     =  99,67:FontRoll 2                    :REMark (C) Roll Clockwise
1097     =50 TO 55:col=K-48:FontDraw            :REMark  Colour change
1098     =110,78:EXIT Chg_lp                    :REMark (N)o Return
1099     =121,89:FontPoke:EXIT Chg_lp           :REMark (Y)es Change Font
1100     =    35:FontNew                        :REMark (#) New
1101     =    10:CLS#4:FontPeek                 :REMark Change Font Pattern
1102     =    32:Bitswap                        :REMark Bit Swap 0<>1
1103   END SELect
1104 END REPeat Chg_lp
1105 BLOCK 40,98,292,94,0:BLOCK 170,10,310,208,0:BLOCK 30,98,464,94,0
1106 INK#3,col:cn=co:FontSize:CLS#4:FontGrid:KEYPad
1107 END DEFine

1109 DEFine PROCedure FontGrid
1110 FOR i=0 TO 8:BLOCK#4,1,108,2+i*16,2,248
1111 FOR i=0 TO 9:BLOCK#4,128,1,2,2+i*12,248
1112 END DEFine

1114 DEFine PROCedure FontDraw
1115 FOR r=0 TO 8
1116   FOR c=0 TO 7:IF Fnt$(r+1,c+1)<>'0':BLOCK#4,13,9,4+c*16,4+r*12,col
1117 END FOR r
1118 END DEFine
```





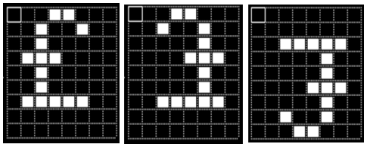
```
1120 DEFine PROCedure FontNew
1121 FOR r=1 TO 9:Fnt$(r)='000000000':END FOR r:CLS#4:FontGrid
1122 END DEFine
```

Note: (**#**) to Clear font grid - (**Spacebar**) to Toggle O &1 bites




```
1124 DEFine PROCedure Bitswap
1125 IF Fnt$(y,x)='0':Fnt$(y,x)='1':ELSE Fnt$(y,x)='0'
1126 IF Fnt$(y,x)='0':BLOCK#4,13,9,4+(x-1)*16,4+(y-1)*12,0
1127 IF Fnt$(y,x)='1':BLOCK#4,13,9,4+(x-1)*16,4+(y-1)*12,col
1128 END DEFine
```

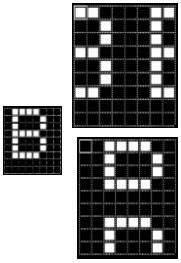
```
1130 DEFine PROCedure FontFlip(cf,cz,rf,rz)
1131 FOR r=1 TO 9:Tmp$(r)=Fnt$(r)
1132 FOR r=1 TO 9
1133   FOR c=1 TO 8:Fnt$(r,c)=Tmp$(rf+r*rz,cf+c*cz)
1134 END FOR r
1135 CLS#4:FontGrid:FontDraw
1136 END DEFine
```



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

```
1138 DEFine PROCedure FontSlid(md,a,b,d,e)
1139 FOR r=1 TO 9:Tmp$(r)=Fnt$(r)
1140 FOR r=1 TO 9
1141   IF md=0:Fnt$(r,a)=Tmp$(r,b:FOR c=1 TO 8:Fnt$(r,c+d)=Tmp$(r,c+e)
1142   IF md=1: Fnt$(a)=Tmp$(b)  :FOR c=1 TO 8: Fnt$(r,c)=Tmp$(r+d,c)
1143 END FOR r
1144 CLS#4:FontGrid:FontDraw
1145 END DEFine
```

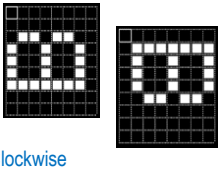
Note: Slide **L**eft **R**ight



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

```
1147 DEFine PROCedure FontRoll(rd)
1148 FOR r=1 TO 9:Tmp$(r)=Fnt$(r)
1149 FOR r=1 TO 8
1150   IF rd=1:rx=r   :ry=8:FOR c=1 TO 8:Fnt$(ry,rx)=Tmp$(r,c):ry=ry-1
1151   IF rd=2:rx=9-r:ry=1:FOR c=1 TO 8:Fnt$(ry,rx)=Tmp$(r,c):ry=ry+1
1152 END FOR r
1153 CLS#4:FontGrid:FontDraw
1154 END DEFine
```

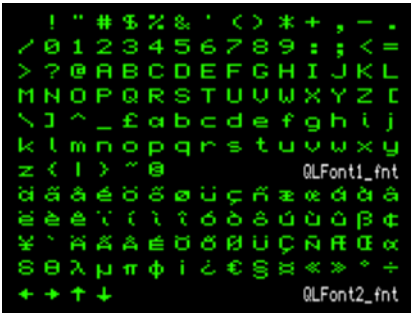
Note: Rotate 90* **A**nticlockwise



Note: Rotate 90* **C**lockwise

```
1156 DEFine PROCedure FontSize
1157 CURSOR#1,148,40:PRINT#1,'CSIZES':RESTORE 1160:INK#3,col
1158 FOR i=1 TO 8:READ a,b,c,d:CSIZE#3,a,b:CURSOR#3,c,d:PRINT#3,CHR$(cn)
1159 DATA 0,0,2,4, 0,1,14,0, 1,0,28,4, 1,1,40,0
1160 DATA 2,0,56,4, 2,1,72,0, 3,0,94,4, 3,1,114,0
1161 END DEFine
:
```


```
1163 DEFine PROCedure FontSets(cn1,cn2)
1164 fx=2:fy=26:CHAR_USE#3,FBase1,FBase2
1165 CLS#3:CSIZE#3,3,0:INK#3,4:CLS#4:INK#1,7
1166 FOR c=32 TO 31+cn1
1167   CURSOR#3,fx,fy:PRINT#3,CHR$(c)
1168   fx=fx+18:IF fx>260:fx=2:fy=fy+12
1169 END FOR c
1170 fx=2:fy=110
1171 FOR c=128 TO 127+cn2
1172   CURSOR#3,fx,fy:PRINT#3,CHR$(c)
1173   fx=fx+18:IF fx>260:fx=2:fy=fy+12
1174 END FOR c
1175 END DEFine
```



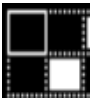

```
1177 DEFine PROCedure FontName(cn1$,cn2$)
1178 CURSOR#1,156,130:PRINT#1,FILL$(' ',20-LEN(cn1$))&cn1$
1179 CURSOR#1,156,190:PRINT#1,FILL$(' ',20-LEN(cn2$))&cn2$
1180 END DEFine
```

Note: Shows 'Filename_fnt' of Fonts displayed by Chart

```
1182 DEFine PROCedure FontChar(cx,cy,ci)
1183 BLOCK#3,17,1,1+cx*18,2+cy*12,ci:BLOCK#3,17,1, 1+cx*18,12+cy*12,ci
1184 BLOCK#3,1,11,1+cx*18,2+cy*12,ci:BLOCK#3,1,11,18+cx*18, 2+cy*12,ci
1185 END DEFine
```

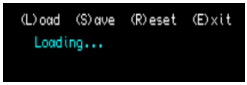
Note: Highlights a Character within the Font Charts

```
1187 DEFine PROCedure FontBit(x,y,ci)
1188 BLOCK#4,16,1,-14+x*16,-10+y*12,ci:BLOCK#4,16,1,-14+x*16,2+y*12,ci
1189 BLOCK#4,1,12,-14+x*16,-10+y*12,ci:BLOCK#4,1,12,2+x*16,-10+y*12,ci
1190 END DEFine
```



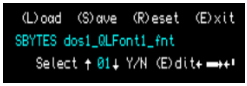
Note: Highlights a Bit square within the Font BitMap

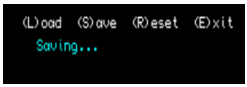
```
1192 DEFine PROCedure FontPeek
1193 a1%=PEEK(FBase1):a2%=PEEK(FBase2)              Note: Calculates Font address
1194 IF cn<128:addr=FBase1+2+(cn-a1%)*9:ELSE addr=FBase2+2+(cn-a2%)*9
1195 FOR r=0 TO 8
1196   Fnt$(r+1)=BIN$(PEEK(addr+r),8)                writes bits info to Fnt$(9,8) array
1197 END FOR r
1198FontGrid:FontDraw
1199 END DEFine

1201 DEFine PROCedure FontPoke
1202 a1%=PEEK(FBase1):a2%=PEEK(FBase2):cn=co        Note: Calculates Font address
1203 IF cn<128:addr=FBase1+2+(cn-a1%)*9:ELSE addr=FBase2+2+(cn-a2%)*9
1204 FOR r=0 TO 8:POKE(addr+r),BIN(Fnt$(r+1))      writes FnT$(9,8) array to memory
1205 FontSets cn1,cn2:FontName cn1$,cn2$
1206 END DEFine

1208 DEFine PROCedure FontLoad
1209 chk=0:eck=0:Lchk=0:sf%=0:ft%=0:fm%=50:FOR i=1 TO 50:File$(i)=''
1210 INK 7:CURSOR 300,46:PRINT 'Select ↑  ↓ Y/N':INK 5:SelDrv
1211 FntList 1:SelFont 1,'LBYTES ':IF Lchk=1:GrpChk:ELSE RETurn
1212 BLOCK 200,24,292,32,0:CURSOR 310,32:PRINT#1,'Loading...':PAUSE 30
1213 IF sg%<127 :LBYTES drv$(dn%)&File$(sf%),FBase1:cn= 32:cn1=96:cn1$=File$(sf%)
1214 IF sg%>=127:LBYTES drv$(dn%)&File$(sf%),FBase2:cn=128:cn2=64:cn2$=File$(sf%)
1215 FontSets cn1,cn2:FontName cn1$,cn2$:FontGrid:eck=0:INK 7
1216 END DEFine
```



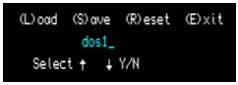
Note **LOAD** returns '**No Files Found**' if FontList fails to find any on selected device. If t'_fnt' Files are found Scroll Up/Down with cursor keys and select one of choice. For **SAVE** select one of the two Font files. You can save with exiting name to a new device or with a new name with use of the Filename Editor (**E**)dit. If device not available **DEVICE ERROR**' will be displayed. If file already exist you are prompted to '**Overwrite Y/N'**.

```
1218 DEFine PROCedure FontSave
1219 chk=0:eck=0:Lchk=0:sf%=1:ft%=2:File$(1)=cn1$:File$(2)=cn2$
1220 INK 7:CURSOR 300,46:PRINT 'Select ↑  ↓ Y/N  (E)dit ←  →←':
1221 BLOCK 12,3,453,50,7:BLOCK 2,4,481,48,7:INK 5:SelDrv
1222 SelFont 2,'SBYTES ':IF Lchk=0:RETurn
1223 FntList 2:BLOCK 200,10,300,46,0
1224 INK 7:CURSOR 312,46
1225 IF eck=1:PRINT#1,'DEVICE ERROR...':PAUSE 50:RETurn
1226 IF chk=1:PRINT#1,'Overwrite Y/N'  :PAUSE:IF KEYROW(5)<>64:RETurn
1227 INK 5:DELETE drv$(dn%)&File$(sf%)
1228 BLOCK 200,24,292,32,0:CURSOR 310,32:PRINT#1,'Saving...':PAUSE 30
1229 IF sf%=1:SBYTES drv$(dn%)&File$(1),FBase1,875:cn1$=File$(1)
1230 IF sf%=2:SBYTES drv$(dn%)&File$(2),FBase2,587:cn2$=File$(2)
1231 FontName cn1$,cn2$:INK 7
1232 END DEFine
```



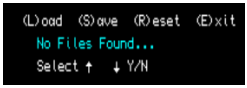


```
1234 DEFine PROCedure SelDrv
1235 REPeat drv_lp
1236  CURSOR 342,32:PRINT drv$(dn%)
1237  K=CODE(INKEY$(-1))
1238  SELect ON K
1239    =208:dn%=dn%-1:IF dn%<0:dn%=dm%
1240    =216:dn%=dn%+1:IF dn%>dm%:dn%=0
1241    =121,89,110,78:EXIT drv_lp
1242  END SELect
1243 END REPeat drv_lp
1244 END DEFine
```



Yes or No Selects Device

```
1246 DEFine PROCedure SelFont(act%,Act$)
1247 sf%=1:BLOCK 200,10,292,32,0
1248 IF ft%<1:CURSOR 310,32:PRINT 'No Files Found...':PAUSE 30:RETurn
1249 REPeat File_lp
1250  CURSOR 292,32:PRINT Act$&drv$(dn%)&File$(sf%):CLS 4
1251  CURSOR 351,46:PRINT FILL$('0',2-LEN(sf%))&sf%
1252  K=CODE(INKEY$(-1))
1253  SELect ON K
1254    =208:sf%=sf%-1:IF sf%<1:sf%=ft%
1255    =216:sf%=sf%+1:IF sf%>ft%:sf%=1
1256    =101,69:IF act%=2:EditName File$(sf%)
1257    =110,78:Lchk=0:EXIT File_lp
1258    =121,89:Lchk=1:EXIT File_lp
1259  END SELect
1260 END REPeat File_lp
1261 IF File$(sf%)='QLFont1_fnt'OR File$(sf%)='QLFont2_fnt':Lchk=0
1262 END DEFine
```

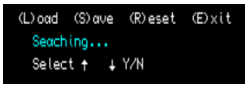


No abort action

Yes Load / Save Font File

Note: Checks default Font Files not overwritten

```
1264 DEFine PROCedure FntList(act%)
1265 IF act%=1:CURSOR 310,32:PRINT 'Seaching...':CLS 4:PAUSE 20
1266 DELETE drv$(dn%)&'FList'
1267 OPEN_NEW#9,drv$(dn%)&'FList':DIR#9,drv$(dn%):CLOSE#9
1268 OPEN_IN#9, drv$(dn%)&'FList'
1269 REPeat dir_lp
1270  IF act%=1
1271    IF EOF(#9) OR sf%>fm%:ft%=sf%-1:CLOSE#9:EXIT dir_lp
1272    INPUT#9,F$:fl%=LEN(F$)
1273    IF fl%<=20 AND '_fnt' INSTR F$>0:File$(sf%)=F$:sf%=sf%+1
1274  END IF
1275  IF act%=2
1276    IF EOF(#9):CLOSE#9:chk=0:EXIT dir_lp
1277    INPUT#9,F$:IF F$==File$(sf%):CLOSE#9:chk=1:EXIT dir_lp
1278  END IF
1279 END REPeat dir_lp
1280 END DEFine
```



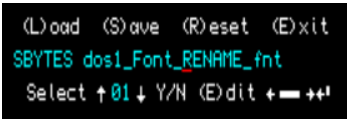
Note: Searches _nft' Files to LOAD

Note: Checks If SAVE File Exists

```
1282 DEFine PROCedure GrpChk
1283 OPEN_IN#9,drv$(dn%)&File$(sf%):sg%=CODE(INKEY$(#9)):CLOSE#9
1284 END DEFine
```

Note: Before Loading to FBase1 or FBase2 memory address **GrpChk** check Lowest Character held in first byte of file. If less than 127 file LBYTES to address FBase1 if greater to FBase2 memory address.

Note The Filename Editor restricts characters to those used for filenames. Position the underline 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 the right. The QL Delete uses CTRL Right cursor for character above underline to make thing easier I have utilised the **Spacebar** as a **Delete** key. CTRL Left Backspace still applies to Delete character to left of underline.




```
1286 DEFine PROCedure EditName(str$)
1287 sm%=16:sl%=LEN(str$):cp%=1
1288 temp$=str$:sl%=('_fnt' INSTR str$)-1:str$=str$(1 TO sl%)
1289 REPeat Ed_lp
1290   Ln_Prn:Ln_Cur:k$=INKEY$(#0,-1):K=CODE(k$)
1291   SELect ON K
1292     = 10:EXIT Ed_lp
1293     = 48 TO 57, 65 TO 90,95, 97 TO 122:Add_chr
1294     =194   :IF cp%>1:cp%=cp%-1:Del_chr
1295     =202,32:Del_chr
1296     =192   :IF cp%>1:cp%=cp%-1
1297     =200   :IF cp%<sl%+1:cp%=cp%+1
1298   END SELect
1299 END REPeat Ed_lp
1300 IF sl%=0:str$=temp$:ELSE str$=str$&'_fnt'
1301 END DEFine
```

Delete Character to left of cursor
Delete Character above cursor

```
1303 DEFine PROCedure Ln_Prn
1304 IF LEN(str$)>sm%:str$=str$(1 TO sm%):cp%=sm%
1305 INK 5:CURSOR 364,32:PRINT str$&'_fnt':CLS 4
1306 END DEFine

1308 DEFine PROCedure Ln_Cur
1309 BLOCK 6,1,364+cp%*6-6,41,2
1310 END DEFine

1312 DEFine PROCedure Add_chr
1313 IF cp% =1 AND sl%=0  :str$=str$&k$
1314 IF cp%>=1 AND cp%<sl%:str$=str$(1 TO cp%-1)&k$&str$(cp% TO sl%)
1315 IF cp%>=1 AND cp%=sl%:str$=str$(1 TO cp%-1)&k$&str$(cp%)
1316 IF cp%> 1 AND cp%>sl%:str$=str$&k$
1317 IF cp%=sm%:str$(cp%)=k$
1318 IF sl%<sm%:sl%=sl%+1:ELSE sl%=sm%
1319 IF cp%<sm%:cp%=cp%+1:ELSE cp%=sm%
1320 END DEFine

1322 DEFine PROCedure Del_chr
1323 IF cp%=sl%:str$=str$(1 TO sl%-1):sl%=sl%-1
1324 IF cp%>=1 AND cp%<sl%:str$=str$(1 TO cp%-1)&str$(cp%+1 TO sl%):sl%=sl%-1
1325 IF cp%=sm%:str$=str$(1 TO sm%-1):cp%=cp%-1:sl%=sm%-1
1326 IF cp%=1 AND sl%=1:str$="":sl%=0
1327 END DEFine
```

Note: KEYPad Proc Shows the keying for selected printable Character

```
1329 DEFine PROCedure KeyPad
1330 SELect ON cn=32  TO 93       :Key$=CHR$(cn)
1331 SELect ON cn=32              :Key$='Spacebar'
1332 SELect ON cn=33,34           :Key$='Shift '&(cn-32)
1333 SELect ON cn=36,37           :Key$='Shift '&(cn-32)
1334 SELect ON cn=38              :Key$='Shift 7'
1335 SELect ON cn=40              :Key$='Shift 9'
1336 SELect ON cn=41              :Key$='Shift 0'
1337 SELect ON cn=42              :Key$='Shift 8'
1338 SELect ON cn=43              :Key$='Shift ='
1339 SELect ON cn=58              :Key$='Shift ;'
1340 SELect ON cn=60              :Key$='Shift ,'
1341 SELect ON cn=62              :Key$='Shift .'
1342 SELect ON cn=63              :Key$='Shift /'
1343 SELect ON cn=64              :Key$="Shift '"
1344 SELect ON cn=65  TO  90      :Key$='Shift '&CHR$(cn)
1345 SELect ON cn=94              :Key$="Shift 6"
1346 SELect ON cn=95              :Key$="Shift -"
1347 SELect ON cn=96              :Key$="Shift 3"
1348 SELect ON cn=97  TO 122      :Key$=CHR$(cn-32)
1349 SELect ON cn=123             :Key$='Shift ['
1350 SELect ON cn=124             :Key$='Shift \'
1351 SELect ON cn=125             :Key$='Shift ]'
1352 SELect ON cn=126             :Key$='Shift #'
1353 SELect ON cn=127             :Key$='Shift Esc'
1354 SELect ON cn=128             :Key$='CTRL Esc'
1355 SELect ON cn=129             :Key$='CTRL Shift 1'
1356 SELect ON cn=130             :Key$="CTRL Shift '"
1357 SELect ON cn=131 TO 133      :Key$='CTRL Shift '&CHR$(cn-80)
1358 SELect ON cn=134             :Key$='CTRL Shift 7'
1359 SELect ON cn=135             :Key$="CTRL '"
1360 SELect ON cn=136             :Key$='CTRL Shift 9'
1361 SELect ON cn=137             :Key$='CTRL Shift 0'
1362 SELect ON cn=138             :Key$='CTRL Shift 8'
1363 SELect ON cn=139             :Key$='CTRL Shift ='
1364 SELect ON cn=140 TO 153      :Key$='CTRL '&CHR$(cn-96)
1365 SELect ON cn=154             :Key$='CTRL Shift ;'
1366 SELect ON cn=155             :Key$='CTRL ;'
1367 SELect ON cn=156             :Key$='CTRL Shift ,'
1368 SELect ON cn=157             :Key$='CTRL ='
1369 SELect ON cn=158             :Key$='CTRL Shift .'
1370 SELect ON cn=159             :Key$='CTRL Shift /'
1371 SELect ON cn=160             :Key$='CTRL Shift 2'
1372 SELect ON cn=161 TO 186      :Key$='CTRL Shift '&CHR$(cn-96)
1373 SELect ON cn=187             :Key$='CTRL ['
1374 SELect ON cn=188             :Key$='CTRL \'
1375 SELect ON cn=189             :Key$='CTRL ]'
1376 SELect ON cn=190             :Key$='CTRL Shift 6'
1377 SELect ON cn=191             :Key$='CTRL Shift -'
1378
1379 CURSOR 148,208:PRINT 'KEYPad: '&Key$
1380 END DEFine
```

QL Font Installation

General information on installation of Fonts, first space is allocated in the common heap (RAM) for both sets of Fonts the Bytes required 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

Activate with keyword **CHAR_USE#ch,**FBase1,FBase2 and repeat for other **ch**annels where Font is needed. When finished release memory with the RECHP(FBase?). Use RESPR(8&6+588) for a more permanent installation, where Fonts remain until Switch Off or a System Reset.

Note: With the **QLFont1_fnt** & **QLFont2_fnt** assigned to RAM the Bitmap for each individual character can be Read and Overwritten with the PEEK and POKE commands. Each entry is identified by an offset from the **QLFont1** & **2** assigned Base address **FBase1** & **FBase2** given by **ALCHP.** The first 2 Bytes holding the Lowest code & Total number of Font Characters, followed by the Bitmaps each being multiples of 9 Bytes.