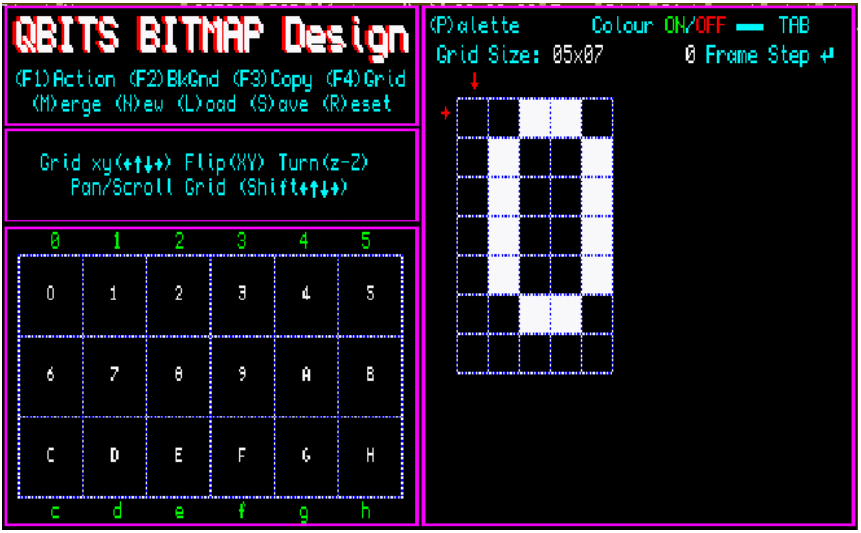
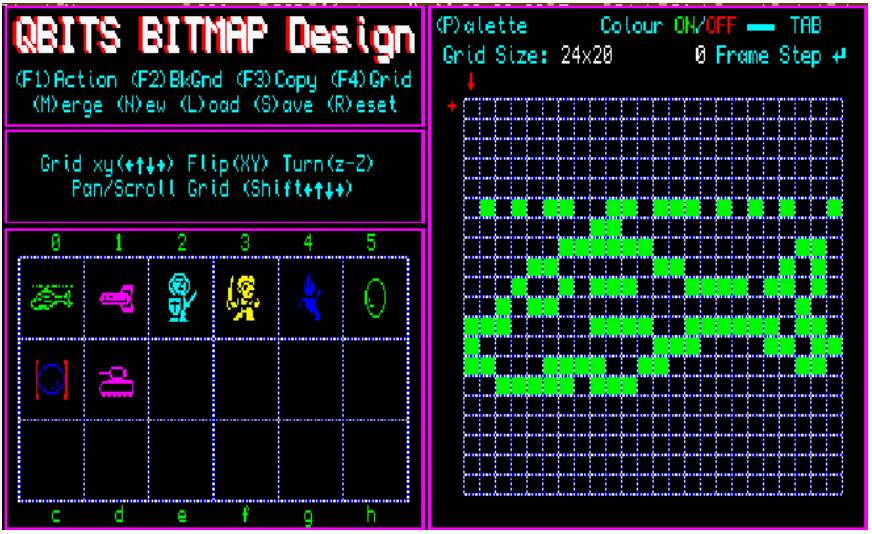
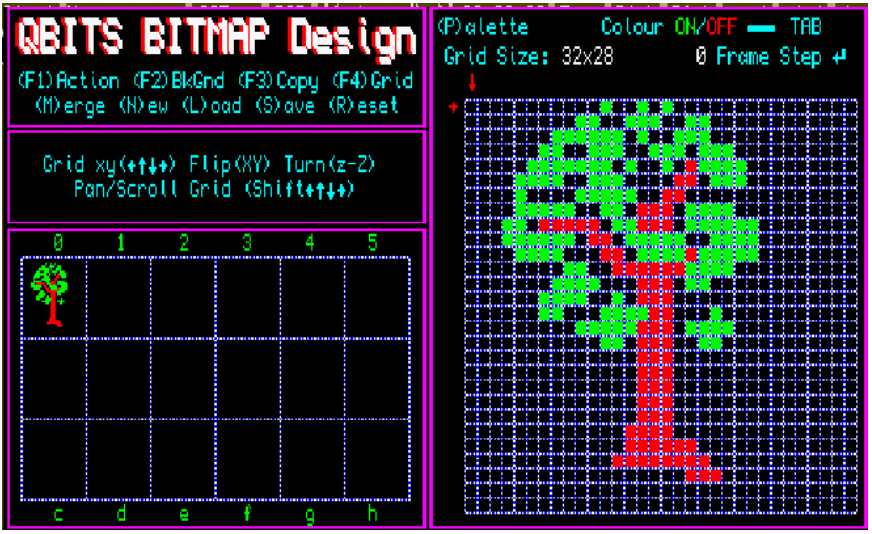
Sinclair QL Retro Gaming



Sinclair QL Retro Gaming


QBITS BITMAP Design
(F1)Action (F2)BkGnd (F3)Copy (F4)Grid
(M)erge (N)ew (L)oad (S)ave (R)eset
Grid xy(←↑↓→) Flip(XY) Turn(z-Z)
Pan/Scroll Grid (Shift←↑↓→)
0 1 2 3 4 5
c d e f g h
(P)alette Colour ON/OFF TAB
Grid Size: 24x20 0 Frame Step ↵


QBITS BITMAP Design
(F1)Action (F2)BkGnd (F3)Copy (F4)Grid
(M)erge (N)ew (L)oad (S)ave (R)eset
Grid xy(←↑↓→) Flip(XY) Turn(z-Z)
Pan/Scroll Grid (Shift←↑↓→)
0 1 2 3 4 5
c d e f g h
(P)alette Colour ON/OFF TAB
Grid Size: 32x28 0 Frame Step ↵

Introduction

Introduction of home computers in the 1980's with extended graphics capabilities opened the door to improved Character Fonts, whether you call them **Glyphs**, **Bitmaps** or created as **Vectored Images. The g**raphic capabilities soon created small 2D images named **Sprites.** In the 1980s and for most of the 1990s, **Sprites** became the standard way to integrate the graphics images used in what has became Classic Computer Games.

A **Sprite Bitmap** is designed to be part of a larger scene; it is made up of tiny squares of colour that represent the **pixels** of the 2D image displayed to screen. For computer programming purposes a **Bitmap** is a collection of Bytes or an Array stored in memory identifying the Pixels of an image arrangement in columns and rows. Initially used to handle graphical objects separate from the video memory the term has since been loosely applied to various graphical overlays.

Since modern computers and gaming consoles now have dedicated **3D Video chips,** they can actually render **3D objects** more efficiently than **2D Sprites**. The **Sprites** have therefore become less common in modern video games. However, they are still used for other purposes such as to add navigation buttons, click on symbols to enhance the user interface and add visual appeal.

Today's computer monitors screens have millions of **pixels** compared to early video consoles which only had a few thousand. Therefore the characters and objects in early games can look very pixelated. In the eighties computer platforms could only keep track of few moving **Sprites**. Moving images were a sequence of single **Sprites** typically 8x16 pixels with four colours, one being a transparency. Yet even within these restricted limitations programmers soon learned how to draw S**prites** that looked like vivid animated characters.

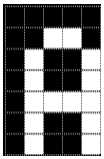
Drawing a **Sprite** with a smiley face needs to be six, seven, eight pixels across. At the beginning not having the space for a character's head, the best **Sprite Designers** would imply a face, without having to actually draw them.

The creating and modifying of pixel characters or objects soon became a recognised art form.

Character Bitmaps.

As Digital representation of data in computer systems gathered pace and more elaborate character codes were introduced, Internationally accepted standards permitted worldwide interchange of text in electronic form. Character sets such **ASCII** uses 8-bit encoding whereas **Unicode** uses variable bit encoding. **Unicode** represents most of the characters used in many written languages in today's world while **ASCII** still widely used, is based around the West's Latin script.

The **Bitmap** character Generation is also known as **dot matrix** because in this method characters are represented by an array of dots in the matrix form. The Array is two dimensional having columns and rows as in the 5x7 shown below. Modern higher resolution devices may use larger arrays of 100x100.



```
DATA 0,0,0,0,0
DATA 0,0,1,1,0
DATA 0,1,0,0,1
DATA 0,1,0,0,1
DATA 0,1,1,1,1
DATA 0,1,0,0,1
DATA 0,1,0,0,1
```

In this example the DATA fields shows 0's as Black and 1's as White, as a 1-bit Colour scheme.

Grid 5x7

Characters Fonts represent different styles and sized characters and aspects such as **Bold,** *Italics* etc. **Character Generation** accepts address for the character and from the **Fonts** selected gives the relative bit pattern for that character as an output. Here the size of the pixel is fixed and hence the size of the dot, therefore the more complicated a Character set, the higher number of pixels and therefore larger arrays.

Each **Pixel** is represented by a **Cell** in an Array and a character is placed on the screen by copying **Pixel values** from the character array into some position of the screen's frame buffer. Early values for a Pixel represented a **fixed colour** as with the original QL QDOS screen arrangement, however with modern computers a **Pixel value** also controls the **intensity** as well.

Bitmap Storage

There are many different bitmap file formats, but most are based around the simple premise of Pixel columns and rows. Today's standards involve colour depths and in some cases complicated encoding.

Hobbyist programs in the eighties employed simple methods for storage. The Bitmap saved as a string of Bytes literally dumped from an array or straight from memory. To LOAD they reversed the process, reading the file contents directly into memory or to an Array. The second method was to SAVE as DATA Lines to MERGE within other programs.

Bit Depth	Number of Colours
8	256
12	4,096
16	65,536
24	16,777,216
32	4,294,967,296

Resolution of Digital Images

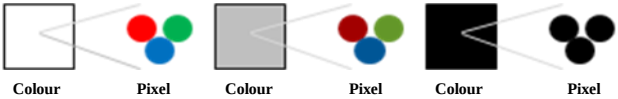
The higher the Resolution comes the greater the number of Pixels which leads to a better quality picture. Resolution is the measure of **Pixel density**, usually measured in **dots per square inch (dpi)**. For example Images with a Resolution of 72 dpi, such as those generally used on websites means that a 1-inch square contains a matrix of Pixels that is 72 pixels wide by 72 pixels high or 5184 **Pixels per square inch**. Each **Pixel** also has colour depth: **hue** (its quality of colour or wave length) and **value** (relative darkness or lightness its energy) and represented by a binary number. The combination of higher resolution and representation of colour depth requires a much larger screen memory.

Computer Screen Colour

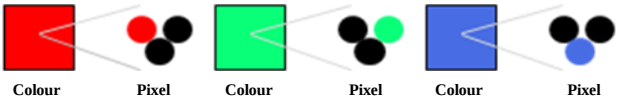
The number of bits indicates how many colours are available for each Pixel. For a Black and White image, only 1-bit is needed either 0 OFF or 1 ON, for 2-bit colour: 00, 01, 10, 11. Increasing the bits per pixel then the greater colour depth is achievable. A Monitor or TV screen **Pixel** is generated by three colours (**Red**, **Green**, and **Blue**) and the different colours seen are due to different combinations and light intensities of these primary colours.

White Grey Black

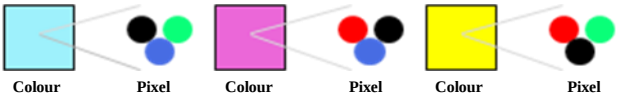
For a White section of a screen all three colours are active with about the same relative intensities as in sunlight. Gray parts of the screen have all three producing light, but at a much lower intensity. Black is the lack of any emitted light.



Primary Colour: Red Green Blue

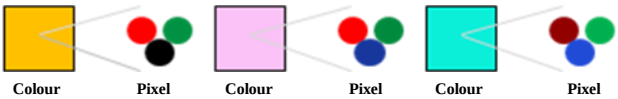


Colour Combinations: Cyan Magenta Yellow



Other Mixtures of Colours

Mixtures of two or three primary colours with different intensities give the other colours. The combinations for Orange (Red with a little Green) - Pink (Red with a little Green and a little Blue) and Turquoise (Blue and Green with a little Red) as shown here.



QL Screen Organisation

The original QL used 32k of **Screen RAM** and the **Pixel** coordinate system to define the position and size of windows. The **Screen RAM** is organised as a series of **16 Bit words** starting from address **Hex 2000** and progressing in the order of the raster scan.

Hex 2000 - 2800 (Dec 131072 - 163840) 128 Bytes x 256 rows

High Byte								**Low Bite**								
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**Bit**
G	G	G	G	G	G	G	G	R	R	R	R	R	R	R	R	**512**
G	F	G	F	G	F	G	F	R	B	R	B	R	B	R	B	**256**

QDOS Screen MODE

The colour of each pixel on screen is a combination of Green and Red in **MODE 4** (**512**) plus **Blue** in **MODE 8** (**256**). Using certain bit values and setting the bits **ON** or **OFF**, the result is a **Primary** and /or **Contrast Colour** making a **Combined Colour**.

Bit	**Value**	**Colour**	**Bit Pattern**	**Colour Combination**	**Final Colour**	**MODE 8**	**MODE 4**
0	1	**Blue**	000	No colour	**Black**	0	0
1	2	**Red**	001	**Blue**	**Blue**	1	
2	4	**Green**	010	**Red**	**Red**	2	2
3	8	**Blue**	011	**Red+Blue**	**Magenta**	3	
4	16	**Red**	100	**Green**	**Green**	4	4
5	32	**Green**	101	**Green+Blue**	**Cyan**	5	
6	64	Stipple	110	**Green+Red**	**Yellow**	6	
7	128	Stipple	111	**Green+Red+Blue**	**White**	7	7

Stipple Contrast XOR Green Red Blue

Bit 00 Bit 01 Bit 10 Bit 11

QDOS Colour Components

For a 2x2 block of Pixels the components of a **QDOS Colour** are **Primary Colour, Contrast Colour** and **Stipple Pattern** of which there are four.

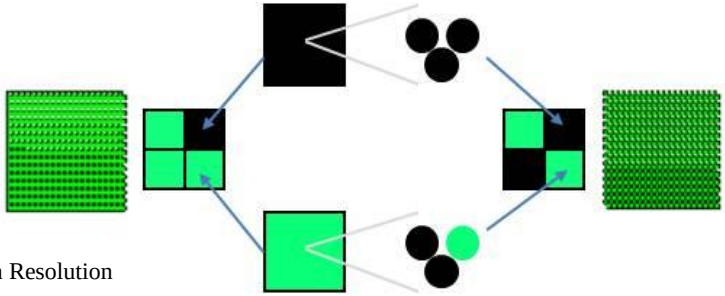
SuperBASIC Colour Combination

When **SuperBASIC** requests a colour parameter, the **Default Contrast** is the same as the **Primary** colour and the default pattern (or **stipple**) is a **checkerboard**. **SuperBASIC** then combines the three components into a one byte composite colour passed to **QDOS**.

To Calculate a composite colour from zero:

add +1 if main colour is Blue, +2 if Red, +4 if Green
then +8 if contrast is different by Blue, +16 if Red, +32 if Green
then 0 for dots, +64 if horizontal stripes, +128 for vertical, +192 for checkerboard.

QBITS QL Colour Exploration
To maximise screen usage, MODE 4 High Resolution become something of a necessity in developing a QBITS BITMap Designer, but has only a four colour scheme Black, Red, Green, White. Exploring the use of **Stipple effects** as in the example shown below, Green and Black gives a simplistic (and by today's standards very pixelated) change in intensity (**Light** to **Darkness**) adding a partial **value** to the limited colour range.

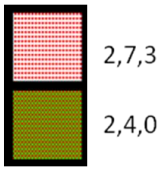


Mode 4 High Resolution

In using QDOS High Resolution MODE and Stipple combinations it is also possible to create the semblance of some additional colours:

Red and White to form a possible Pink.

Red and Green to form a brackish Yellow.



Original QL Limitations
Due to the expense of memory, early computers used 1-bit (2-color), 2-bit (4-color), 4-bit (16-color) or in case of the QL 8 Mode eight Colours and Flash. The problem with such small colour depths is that a full range of colours cannot be produced. Following 8-bit there are now 16-bit high colour and 24-bit true colour formats.

Extending the QL Colour Palette.
The original QL hardware and software limitations have been fortunately overcome as the QL design concept was ported across to faster machines with expanded memory. The resultant Higher Resolution Graphics gave better handling of the colour values for each screen pixel. The operating system **QDOS** and **SuperBASIC** having evolved into **SMSQ\E** and **SBASIC** provide improved performance and added commands to address issues with Colour Palettes.

SBASIC includes **COLOUR_NATIVE** (machine dependant), **COLOUR_QL** (standard QL colours), **COLOUR_PAL** (8-bit 256 colour palette) and **COLOUR_24** (true colour 24-bit palette). Further commands **PALETTE_QL** and **PALETTE_8** allow changing the colours by mapping then to alternative ones.

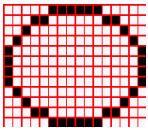
QBITS BITMap Concepts

To start, a matrix of **columns** (width) and **rows** (height) from which to construct a **Grid** displaying a number of **Cells** each representing a **Pixel**. Then any image developed to be displayed at **Pixel** level (**Frame**). If creating a number of **Bitmap Frames** then it would be useful to identify each for example with a Group name followed by a Frame number.

Create Bitmaps identity (**Group name**, Frame **num**ber**, Columns, Rows**)

As Monitor screens tend to be rectangular not square in shape with a different number of horizontal pixels to vertical ones, for this reason sometimes to obtain graphical balance with say drawing a circle the number of columns may exceed the number of rows.

To represent this in a **Bitmap** the horizontal is offset making it oval in shape. But when displayed as **Pixels**, appears circular.

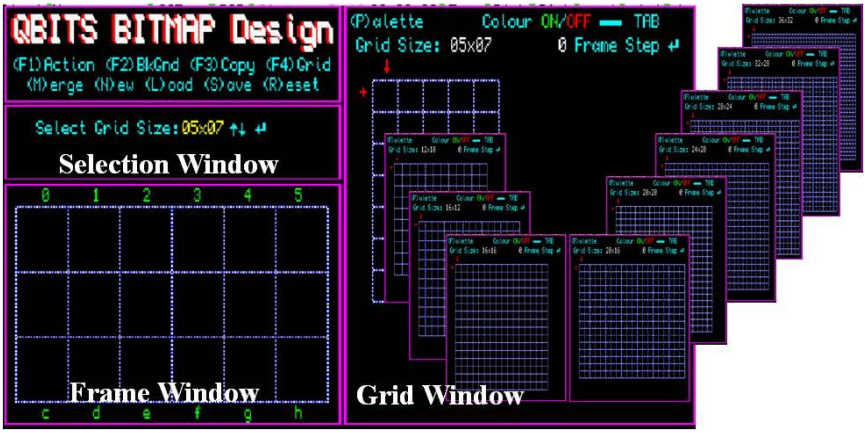


Grid 16x12

In constructing a **Bitmap** image the number of columns and rows (**Resolution**) will determine the quality of the display.

QBITS BITMap Screen Display

The display is divided into four areas: a Title window which also displays the main Commands, windows for the Grid and display of Pixel Frames, then an area for displaying the various options of Commands and Grid/Frame Functions.



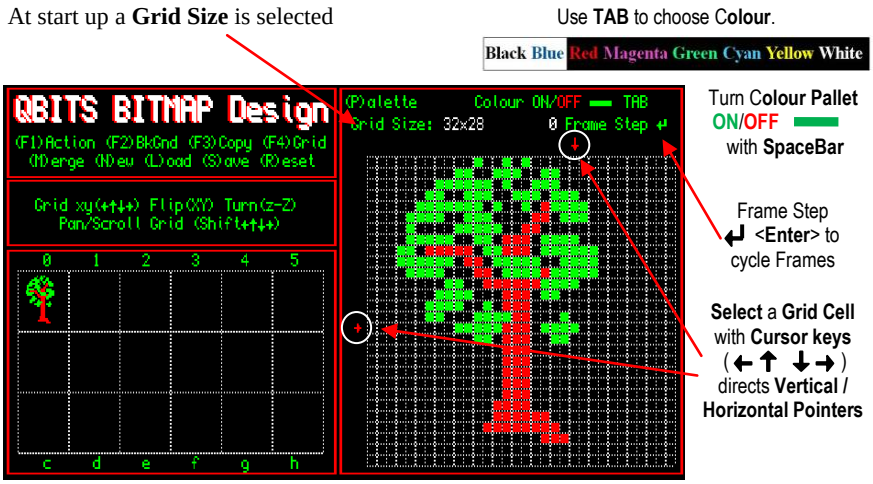
QBTS BITMap Grid Sizes

The consideration for Grid Sizes began with a **5x7** for a Character Set smaller than the 6x10 SuperBASIC CSIZE 0,0 The next **12x10** is equivalent to the minimum Mode 8 CSIZE 2,0. The rest ranking up to **36x32** are stepped increases using multiples of 4. As the **Grid Resolution** increases the **Cell width** and **height** are calculated to best utilise the space within the Grid Window.

QBITS BITMap Layout

An individual **Grid Cell Size** dictates to some extent what is reasonable as a screen display. This is especially true when trying at the same time to leave enough space to show a meaningful set of frames and all fitted within a **QL Mode 4** (512x256) Pixel range. Choosing a **Grid Cell Size 7x6** Pixels, for a **Grid** to display **36x32** (columns and rows), leaves just enough screen width at pixel level, to show six **Frames** in a row.

The screen layout therefore decided upon provides a **Design Grid** for creating a **Bitmap** object and a number of **Pixel Frames** for showing multiple results. The Frame area has three rows of six identified as 0 to 9 - a to h, allowing 18 Frames to be displayed. This can lead to some interesting usage, which will be viewed later.



Flip(XY) Swaps Left-Right/Top-Bottom **Rotate** (**zZ**). Turn Grid 90^0 Clockwise or Anti-clockwise

Pan/Scroll (Shift←↑↓→) moves the columns horizontally and rows vertically

QBITS BITMap Menu (M)erge (N)ew (L)oad (S)ave (R)eset

Select **Grid Size** at start up or later by use of the **(N)EW** command, which presents the opportunity to **(S)ave** current Frames (see BITMap Storage). When using the (**L)oad** command only Grid Files of the same **Grid Size** can be selected and loaded.

(M)erge This option allows two **Grid files** of the same or different **Grid Sizes** to be merged. If of different Grid sizes (**L)oad** the smaller first and **Resize** as required see (**F4)Grid** command. Then select **(M)erge** which allows a second file to be loaded where Frames **[0 to 9]** are merged with the first file's Frames **[a to h]**.

(**R)eset** clears all the Cells of the current selected **Frame** back to **Black** (**Colour=0**).To Clear all Frames use the **(N)ew** command.

QBITS BITMap Storage

The **Grid Size** determined, **Cell** squares can be selected and coloured in. As the image is created the changes are linked to an **Array** for storage and updated. Having created a number of designs displayed in the **Pixel Frames**, there are two formats to **(S)ave**. One as a **Grid/Frame Dump** from the **Grid Array**, the second as numbered lines generated as **DATA Statements**. The opening information saved is the maximum number of **Frames** (**pm**), **Columns** (**cm**) and **Rows** (**rm**) followed by the individual **Cell/Pixel Colour** by **Frame 0 to 17** and **column/row** of selected **Grid Size**. This identifies the File Grid Size for later loading and modifying.

Filenames: **QBPGridcmxrm_num** (**num** = 0 to 9)
QBPDatacmxrm_num (**num** = 0 to 9)

QBITS BITMAp (F) Keys

(F1) Action - see Page10.

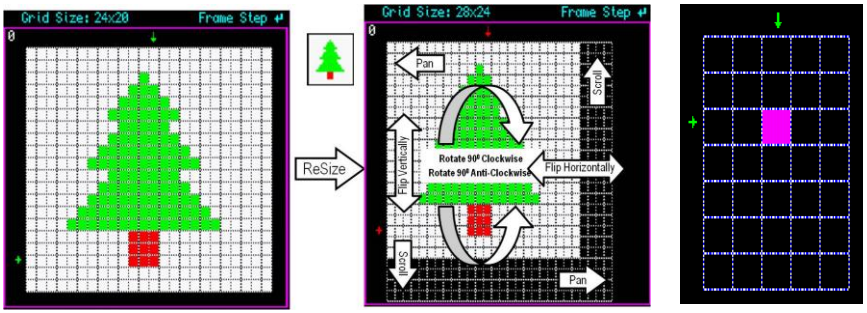
(F2) BkGnd change background colour from default Black to another colour.

(F3) Copy Frame is useful for developing a number of similar Fames.

(F4) Grid Whereby the current Grid Size can be **Resized** to a higher **Grid Size**.

QBITS BITMap Grid

The Pointers highlighted in **Green**/**Red** are moved to identify a particular **Grid Cell.** The **TAB** colour shown with Colour Palette **ON** will be shown in the selected **Cell.** Pressing the **TAB** key will cycle through the Colour Palette. Moving Pointers in **OFF** Mode does not write the colour to the Cell. Other actions to rearrange and position the Grid Object can be carried out by use of **Flip**(**XY**), **Rotate**(**zZ**), **Pan/Scroll**.



QBITS BITMap Colour

Screen output used in personal and home computing often had sets of defined Modes. The Original QL MODE 4 has Black(no colour) Red/Green/White (Green+Red). Mode 8 added Blue and with combinations produces Magenta/Cyan/Yellow. A further option is to set a Primary and Contrast Colour with a Stipple pattern.

QBITS QL Colour (P)alette

Each of the 14 Colours can be set with different combinations of **Colour** /**Contrast** and **Stipple.** Highlight a Palette Colour from those displayed using Left & Right cursor keys. Use Spacebar to Switch to Palette Mix, where Colour and Contrast can be switched **ON/OFF** with **R,r**,**G,g**,**B,b** keys and the Stipple pattern with S.

Spacebar returns to Palette Select showing any change. Use **TAB** to exit back to the **Grid.** Spacebar now turns Colour Mode ON/OFF. TAB will cycle through the Palette displaying the relevant colour combinations.

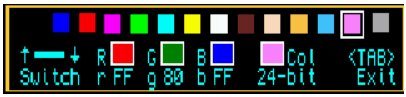
Colours displayed with any Contrast and Stipple combination is in the range 0 to 255. The bits are identified using DIV and MOD functions: **Stipple** (num DIV 64); **Contrast** (num MOD 64 DIV 8); **Colour** MOD 64 MOD 8). Then the Colour and any Contrast colours are checked against a table to reconstruct the ON/OFF switch settings of **Red Green** and **Blue** bits.

QBITS QPC 24 Colour (P)alette

The 14 Colours are set with the 24-bit true colour range. Here **Red**, **Green** & **Blue** each have a range of 0 - 255 or in Hex 00 - FF so the set up is slightly different. Here **Red** occupies the highest or most significant Byte, **Green** occupies the Middle Byte and **Blue** the lowest or least significant Byte. The number representing a colour is therefore between 0 and 16,77,215 or more easily written in Hex 000000 to FFFFFF.

Not requiring a stipple with 24-bit true colours the Palette display is slightly different.

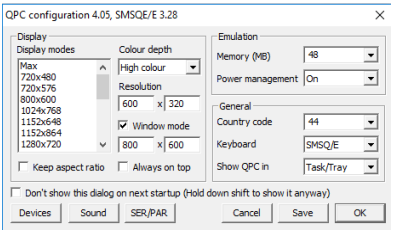
Red = num DIV 65536
Green = num MOD 65536 DIV 256
Blue = num MOD 65536 MOD 256



As before Spacebar switches between Palette and colour mixing of selected Palette colour. The colour mix is obtained using **R, r**, **G, g, B, b** keys to Increase/Decrease the value for each of the **Red Green Blue** colours.

Note: QPC2 Windows Settings

Set QPC configuration to something like as shown, WINDOWl#0,#1,#2 x,y coordinates to 20,50. Then to use 24-Bit true colours you need to activate **Colour_24**.



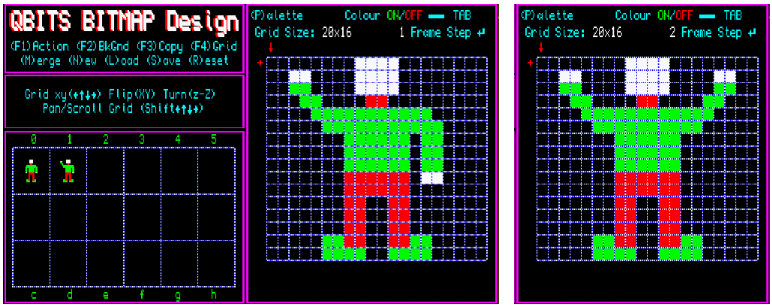
Make sure any WINDOW's; PAPER, INK, BLOCK or STRIP are set with an appropriate colour number. These are more easily written in Hex ($num).

ie. Black=0, **Red**=$FF0000, **Green**=$FF00 **Blue**=$FF and **White**=$FFFFFF

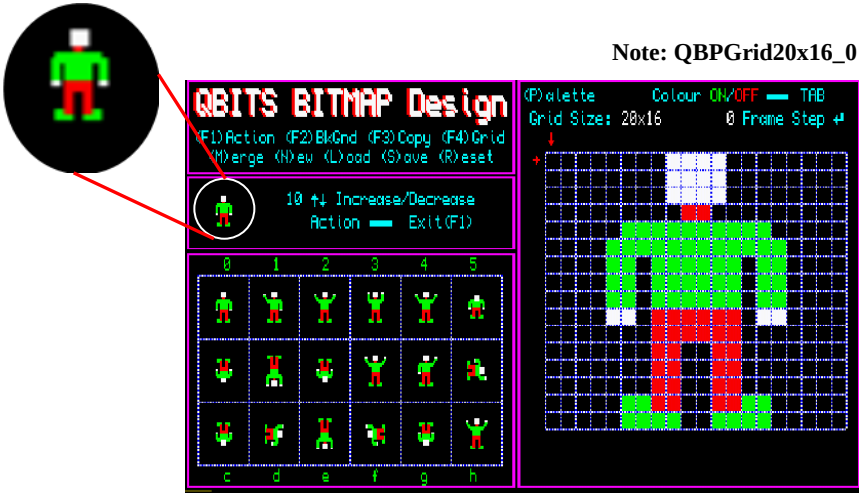
QBITS BITMap Design in Motion

Early Games took **BITMaps** design to a new level creating those **Sprites** as in **Aliens Invaders** shot down with your laser gun, or the **Ghostly** figures chasing you around those two dimensional mazes. Then the **action Sprites** with **Fisticuffs** and **Sword fights**.

As I was putting together the code for **QBITS BITMap Design**, one of my thoughts was to add the means of creating a short action sequence. Below is the design screen showing Frames that sequence a simple Sprite in various positions. From the basic image seen in **Frame(0)** using **(F3) Copy** I copied to **Frame(1)** and then changed the left arm position. Copying this to **Frame(2)**, I then created the next change. In copying Frames, I used the **Flip (XY), Rotate (zZ)** and **Pan/Scroll** commands plus small changes to quickly build up a sequence of images.



(F1)Action - runs the sequence of Frames with PAUSE Times set between -1 to 20. Use the SpaceBar to start the Action or step through each frame at your own pace.

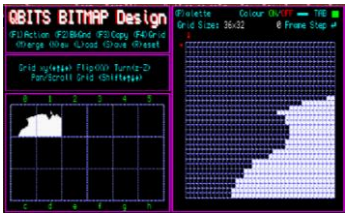


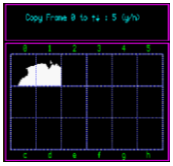
Note: QBPGrid20x16_0

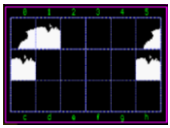
Note: Using Frame Step <**ENTER**> takes you through the sequence of Frames, but using the Grid.

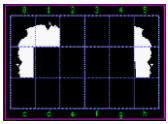
QBITS BITMap Background Scenes

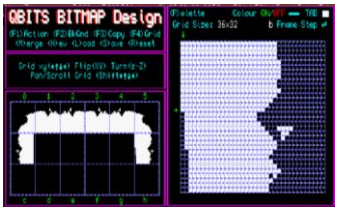
The three rows of six Pixel Frames is now revealed as a means to explore creating backgrounds. Taking **Frame(0)** the bottom right squares are coloured White. Moving to **Frame(1)** and use of (**F2**)**BkGnd** to select **White** and press **SpaceBar**. Revert the top area to black with a jagged pattern. Then Copy **Frames(0)**&(**1**) to **Frames(5), (6), (b)** as shown, use the **Flip**(**X**) and **Rotate** (**zZ**) to build the outer edges to a background scene.



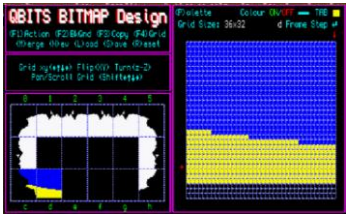






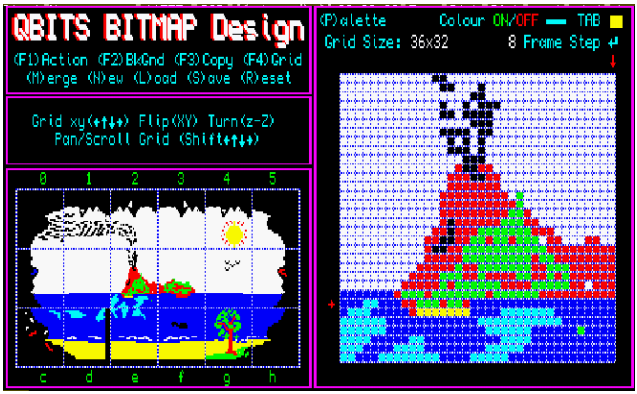


Using a **36x32 Grid** when you Rotate a Frame the two columns to left and right are filled with the background colour (the colour contained in cell 0,0). Therefore at sometime these cells will need to be coloured in.



Selecting **Frame(d)** we add a blue sea and yellow sandy beach then continue with this theme. Above the seas we have sky so for **Frames(7)** to **(a)** use (**F2**)**BkGnd** and set to **White**.

Note: QBPGrid36x32_0



It's now a question of finishing the remaining Frames, filling in sky, sea and sand. Then adding a few extras, a hot Sun, a group of birds flying, a shady tree, then just to finish things off, out at sea an island with an erupting volcano.

QBITS BITMap Design Procedures

The Program comes in two versions **QBBMDesign_QL** with a **QL Colour Palette** and the **QBBMDesign_QPC v**ersion using a **24-bit Palette**.

BMDesign	Access to Grid Functions and Main Commands
FCalc	Calculate position [0,0] x,y of Pixels Frame position
SDraw	Draws a whole Frame and/or Grid display
GDraw	**P**rints individual Pixel and/or Grid blocks
GReset	Reset current Frame Pixels & Grid Cells all to zero
Gflip	Flip horizontally or vertically the Pixel Frame & Grid
GRoll	Rotate 90^0 Clockwise or Anticlockwise Pixel Frame & Grid
GSlid	Pan / Scroll pixel frame & Grid
FCopy	Copy current Frame and Overwrite in another Frame
FAction	Step through sequence of Frames simulating Animation
GBkGnd	Select an alternative Background colour
CBkGnd	Changes the background with new colour
GMerge	Select and load new Grid Merge with current Grid
GNew	Option to save current Grid then Select new Grid
GSize	Selection of Grid Size (N)ew or (R)esize
DGrid	Sets Grid Attributes and draws Grid lines
GMerge	Loads and Merges previous and new File Frames
GTemp	Copy **SGrid(p,c,r)** to **TGrid(p,c,r)**
GLive	Copy **TGrid(p,c,r)** to **SGrid(p,c,r)**
InitGrid	Gets Grid attributes, opens Grid & Frame Windows
Ghelp	Displays keys used for Grid Functions
CPQL	Colour Palette for QL Colour/ Contrast/Stipple mix
CP24	Colour Palette for QPC 24-Bit true colour
CPSel	Highlight a Palette Colour
CMode	Switch between palette selection and colour mix
CRead	Reads and identifies Colour/Contrast/Stipple
CRead	Reads and identifies 24-bit Colour mix
CPrnt	Prints colour components of colour/contrast/stipple
CPrnt	Prints colour components of 24-bit colour mix
InitCPQL	Sets the Colour Palette/Contrast/Stipple parameters
InitCP24	Sets the Colour Palette
GFSel	Identifies Load/Merge/Save the latter for Grid or Data
GFChk	Checks file exists for Load or overwrite (y/n) for Save
GLoad	Loads selected file into the **SGrid(p,c,r)**
GSave	Saves Array dump or DATA statements to selected file
InitDrive	Set available Device Drives
InitWin	Set WINDOWS#0,#1,#2 and call other Init code
InitTitle	Display QBITS Title and main Commands New/Load/Save etc.

QBBMDesign_QL & QBBMDesign_QPC
The main code differences being driven by the **Colour Palettes** and in most cases are the change of colour allocation from Decimal (0-255) to Hex ($000000 to $FFFFFF) for use with 24-bit true colour in setting a Windows BORDER, PAPER and INK.

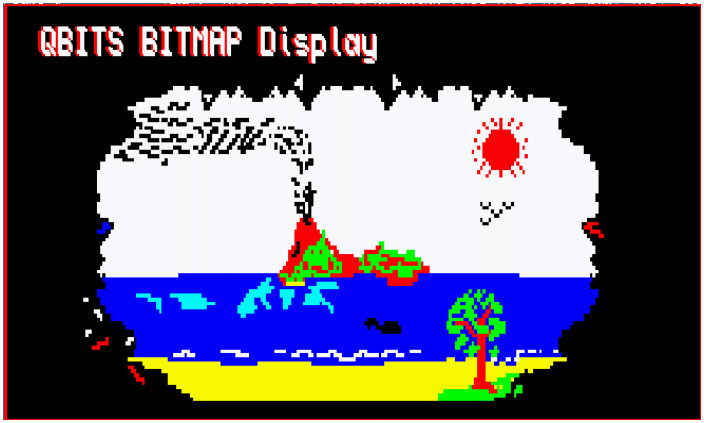
Note: They can be the change of a single INK or BLOCK colour attribute to a full line or group of lines of code shown with the same line numbers. This especially applies to code lines of **CPQL** & **CP24** and **Init** for Colour Palettes. Lines **619, 622, 623** are specific to only the **QL version** and these are shown enclosed.

The Code is set out in groups:
Lines 100 plus are BMDesign the Main Menu of Commands and Grid/Frame Functions
Lines 400 plus are Selecting and setting up the Grid Size for New, Merge and Resize.
Lines 600 plus are Selecting and setting the Colour Palette
Lines 800 plus are File Select, File Check, Load & Save
Lines 900 plus the Initial WINDOW's set up and their parameters etc.
Lines 1000 plus A QL to QPC Grid Colour Conversion.

QBITS BITMap Display
As an added program **QBBMDisplay** Merges a Frame DATA file and displays the 18 Frames as a BITMap picture which can be enlarged to fill more of the screen. First load the Prog, change Line 103 to the Data Filename required, then decide the Mode to run. If there is space, increase the (en) enlarge value. Then RUN the program. After loading the DATA file the program should display the result to screen.

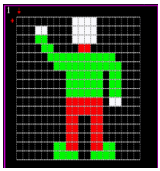
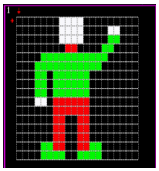
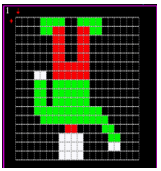
QBBMDisplay Procedures

Init	Sets the Screen Windows and Title
DLoad	Reads Data lines of MERGE File and Displays to screen
DCalc	Calculates the position offsets for each Frame
DDraw	Draws the Pixel BLOCKs to screen



```
100 REMark QBBMDesign_QL (QBITS BITMap Design 2019 v01)
100 REMark QBBMDesign_QPC    (QBITS BITMap Design 2019 v02)

102 MODE 4      :InitWin:BMDesign
102 COLOUR_24:InitWin:BMDesign

104 DEFine PROCedure BMDesign
105 dn=5:gm=0:sm=0:GSize:fp=1:p=0:FCalc:CLS#0          :REMark dn Sets Default Drive

106 REPeat Des_lp
107   IF cur=4:SGrid(p,x,y)=CP(cs%):c=x:r=y:fp=1:gp=1:FCalc:GDraw
108   BLOCK#4,236   8,8,26,0:INK#4,cur:CURSOR#4,cx+x*cw,24  :PRINT#4,'↑ '
109   BLOCK#4,12,170,4,34,0:INK#4,cur:CURSOR#4,8,24+cy+y*rh:PRINT#4,'↪ '
110   BLOCK#4,10,7,230,4,CP(cs%):k=CODE(INKEY$(-1))

111   SELect ON k
112     =48 TO  57:p=k-48:fp=1:gp=1 :FCalc:SDraw          :REMark 0 to 9
113     =97 TO 104:p=k-87:fp=1:gp=1:FCalc:SDraw           :REMark a to h
121     =10:IF p=pm:p=0:gp=1:FCalc:SDraw:ELSE p=p+1:gp=1:FCalc:SDraw

118     =80,112:CLS#7: CPQL                 :WHelp    REMark (P)alette  QL
118     =80,112:CLS#7: CP24                 :WHelp    REMark (P)alette  QPC

119     = 9:IF cur=4:cs%=cs%+1:IF cs%>13:cs%=0          :REMark <TAB>Change Colour
120     =32:IF cur=2:cur=4:ELSE cur=2                   :REMark <SBar> Colour ON/OFF

119     = 9:IF cur=$FF00:cs%=cs%+1:IF cs%>13:cs%=0      :REMark <TAB>Change Colour
120     =32:IF cur=$FF0000:cur=$FF00:ELSE cur=$FF0000   :REMark <SBar>Colour ON/OFF

122     =192:x=x-1:IF x< 0:x=0
123     =200:x=x+1:IF x=cm:x=cm-1                       :REMark (← ↑ ↓ →) x,y Grid Pointers
124     =208:y=y-1:IF y< 0:y=0
125     =216:y=y+1:IF y=rm:y=rm-1

114     =232:CLS#7                  :FAction  :WHelp    :REMark (F1)Frame Action
115     =236:CLS#7                  :GBkGnd :WHelp      :REMark (F2)Grid BackGnd
116     =240:CLS#7:n=p              :GCopy    :WHelp    :REMark (F3)Copy
117     =244:CLS#7:IF gm<9 :sm=2    :GSize    :WHelp    :REMark (F4)Grid Resize
126     =88,120:xf=cm-1:yf=0:zx=-1:zy=1             :GFlip    :REMark (X)Flip Grid
127     =89,121:yf=rm-1:xf=0:zy=-1:zx=1             :GFlip    :REMark (Y)Flip Grid
128     = 90:rxm=rm-1+cs:rt=-1:rym=0:yt=1           :GRoll    :REMark (z)Roll ClockWise
129     =122:rxm=cs:rt=1 :rym=rm-1:yt=-1            :GRoll    :REMark (Z)Roll Anti-CW
130     =196:pa=cm-1:pb=0:pc=-1:pd=0:md=0           :GSlid    :REMark <Shift ← → >Pan Grid
131     =204:pa=0:pb=cm-1:pc=0:pd=-1:md=0           :GSlid
132     =212:sa=rm-1:sb=0:sc=-1:sd=0:md=1           :GSlid    :REMark <Shift ↑ ↓>Scroll Grid
133     =220:sa=0:sb=rm-1:sc=0:sd=-1:md=1           :GSlid

135     =77,109:ck=0:dg=2 :CLS#7    :GMerge   :WHelp    :REMark (M)erge Grid
136     =78,110:ck=0:dg=3:sm=1      :GSize    :WHelp    :REMark (N)ew
137     =76,108:ck=0:dg=1 :CLS#7    :GFSel    :WHelp    :REMark (L)oad
138     =83,115:ck=0:dg=0 :CLS#7    :GFSel    :WHelp    :REMark (S)ave Data/Grid
139     =83,114:col=0       :CLS#7  :GReset   :WHelp    :REMark (R)eset Grid
134     =27:CLS#0:CLS#1:STOP                            :REMark Exit
140   END SELect
141 END REPeat Des_lp
142 END DEFine
```

```
144 DEFine PROCedure FCalc
145 IF p<10:p$=p:ELSE p$=CHR$(p+87)
146 INK#4,7:CURSOR#4,150,14:PRINT#4,p$                    Note: CP24  INK#4,$FFFFFF
147 IF fp=1
148    ch=5:pc=p:pr=0
149    IF p>  5:pc=p  -6:pr=33
150    IF p>11:pc=p-12:pr=66
151    px=7+INT((36-cm)/2)+pc*37:py=11+INT((32-rm)/2)+pr     Note: Sets Frame position px,py
152 END IF
153 END DEFine

155 DEFine PROCedure SDraw                                  Note: Draw Single Frame/Grid
156 FOR r=0 TO rm-1:FOR c=0 TO cm-1:GDraw:END FOR c:END FOR r
157 fp=0:gp=0:cur=2
158 END DEFine

160 DEFine PROCedure GDraw                                  Note: Draw Single Pixel/Cell
161 IF gp=1:BLOCK#4,cw-1,rh-1,19+c*cw,37+r*rh,SGrid(p,c,r)
162 IF fp=1 OR fp=3:BLOCK#ch,1,1,px+c,py+r,SGrid(p,c,r)
163 END DEFine

165 DEFine PROCedure GReset
166 CLS#7:CURSOR#7,20,6:PRINT#7,'Reset/Clear Grid: ';p$;' (y/n)'
167 IF INKEY$(-1)<>'y':CLS#7:RETurn
168 FOR r=0 TO rm-1:FOR c=0 TO cm-1:SGrid(p,c,r)=col:END FOR c:END FOR r
169 fp=1:FCalc:BLOCK#5,36,32,7+pc*37,11+pr,0:fp=0:gp=1:SDraw:cur=2:CLS#7
170 END DEFine                                            Note: CP24 cur=$FF0000
```

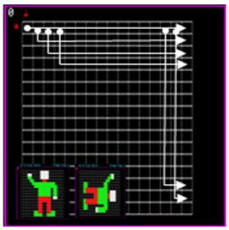




Note: Flip Column's or Rows

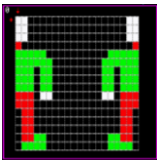
```
172 DEFine PROCedure GFlip
173 FOR r=0 TO rm-1
174     FOR c=0 TO cm-1:TGrid(p,xf+c*zx,yf+r*zy)=SGrid(p,c,r)
175 END FOR r
176 GLive:fp=1:gp=1:FCalc:SDraw
177 END DEFine
```

Note: Rotate Cells Column's to Rows

```
179 DEFine PROCedure GRoll
180 IF cm=5 OR rm=10:RETurn
181 FOR r=0 TO rm-1
182    rx=rxm+(r*rt):ry=rym
183    FOR c=0 TO cm-1:TGrid(p,c,r)=SGrid(p,0,0)
184    FOR c=cs TO cs+rm-1:TGrid(p,c,r)=SGrid(p,rx,ry):ry=ry+yt
185 END FOR r
186 GLive:fp=1:gp=1:FCalc:SDraw
187 END DEFine
```



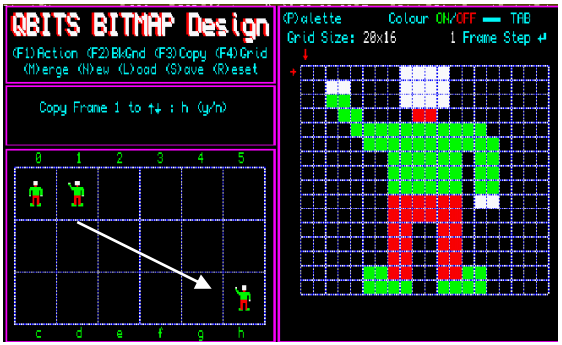
Note: Shift ← ↑ ↓→ Pan/Scroll Slid Column's left or right or Rows Up or Down



```
189 DEFine PROCedure GSlid
190 IF md=0
191   FOR r=0 TO rm-1
192     FOR c=1 TO cm-1
193       TGrid(p,pa,r)=SGrid(p,pb,r):TGrid(p,c+pc,r)=SGrid(p,c+pd,r)
194     END FOR c
195   END FOR r
196 ELSE
197   FOR r=1 TO rm-1
198     FOR c=0 TO cm-1
199       TGrid(p,c,sa)=SGrid(p,c,sb):TGrid(p,c,r+sc)=SGrid(p,c,r+sd)
200     END FOR c
201   END FOR r
202 END IF
203 GLive:fp=1:gp=1:FCalc:SDraw
204 END DEFine
```

Note: Copy a Frame and overwrite to another Frame

```
206 DEFine PROCedure GCopy
207 CURSOR#7,28,8:PRINT#7,'Copy Frame ';p$;' to ↑ ↓ : ';n$;' (y/n)'
208 REPeat C_lp
209   IF n<10:n$=n:ELSE n$=CHR$(n+87)
210   CURSOR#7,154,8:PRINT#7,n$
211   k=CODE(INKEY$(-1))
212   SELect ON k
213     =208:n=n -1:IF n<0:n=0
214     =216:n=n+1:IF n>pm:n=pm
215     =89,121:IF n=p:CLS#7:RETurn :ELSE EXIT C_lp
216     =78,110,240:CLS#7:RETurn
217   END SELect
218 END REPeat C_lp
219 FOR r=0 TO rm-1
220   FOR c=0 TO cm-1:SGrid(n,c,r)=SGrid(p,c,r)
221 END FOR r
222 p=n:fp=1:gp=1:FCalc:SDraw
223 END DEFine
```



Note: (F1) Action see Animation Page 10

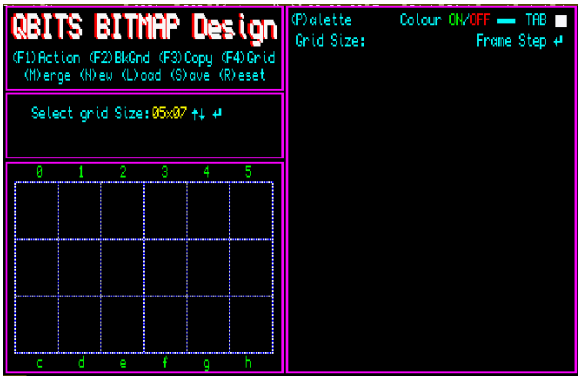
```
225 DEFine PROCedure FAction
226 ch=7:CLS#7:del=10:px=9+INT(32-cm)/2:py=3+INT(28-rm)/2
227 CURSOR#7,88,6:PRINT#7,'↓ ↑ Increase/Decrease'
228 CURSOR#7,88,18:PRINT#7,'Action      Exit(F1)'
229 BLOCK#7,16,3,132,22,5:p=0:fp=3:FCalc:SDraw
229 BLOCK#7,16,3,132,22,$FFFF:p=0:fp=3:FCalc:SDraw                    Note: CP24
230 REPeat Ani_lp
231   CURSOR#7,70,6:PRINT#7,FILL$(' ',2-LEN(del))&del
232   k=CODE(INKEY$(-1))
233   SELect ON k
234     =208:del=del+2:IF del>20:del=20
235     =216:del=del-2:IF del< 0:del=-1
236     = 32:FOR p=0 TO pm:PAUSE del:fp=3:FCalc:SDraw
237     =232:p=0:CLS#7:EXIT Ani_lp
238   END SELect
239 END REPeat Ani_lp
240 END DEFine
```

Note: Use the Colour Palette to change Background Colour.

```
242 DEFine PROCedure GBkGnd
243 CLS#7:FOR i=0 TO 13:BLOCK#7,10,8,10+i*16,3,CP(i)
244 CURSOR#7,42,14:PRINT#7,'↑ ↓ Set      Exit(F2)'
245 BLOCK#7,16,3,90,17,5:ch=7:xg%=7:x1%=cs%:y1%=1:p1=CP(cs%)
246 REPeat Bk_lp
247   CPSel:k=CODE(INKEY$(-1))
248   SELect ON k
249     =192:cs%=cs% -1:IF cs%< 0:cs%=13
250     =200:cs%=cs%+1:IF cs%>13:cs%=0
251     =236:CLS#7:ch=5:EXIT Bk_lp
252     = 32:Bk=CP(cs%):CSwap:fp=1:gp=1:FCalc:SDraw:ch=7
253   END SELect
254 END REPeat Bk_lp
255 END DEFine
```

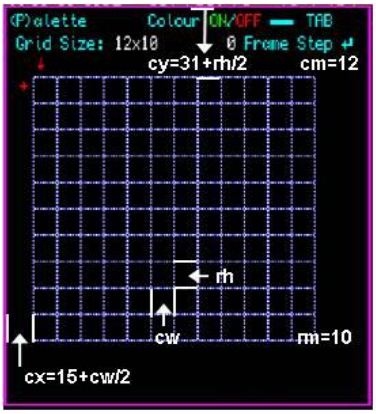

```
257 DEFine PROCedure CSwap
258 FOR r=0 TO rm-1
259   FOR c=0 TO cm-1
260     IF SGrid(p,c,r)= 0 :SGrid(p,c,r)=16
261     IF SGrid(p,c,r)=Bk:SGrid(p,c,r)=32
262     IF SGrid(p,c,r)=16:SGrid(p,c,r)=Bk
263     IF SGrid(p,c,r)=32:SGrid(p,c,r)=0
264   END FOR c
265 END FOR r
266 END DEFine
```

Note: Select Grid Size:




```
401 DEFine PROCedure GSize
{404 CLS#7:INK#7,5:IF sm=1:GFSel                    Note:sm=0 Resize Grid size sm=1 New grid
 404 CLS#7:INK#7,$FFFF:IF sm=1:GFSel                Note: CP24
405 CURSOR#7,20,6:PRINT#7,'Select Grid Size:      ↑↓ '
{406 BLOCK#7,2,4,182,8,5:INK#7,6
 406 BLOCK#7,2,4,182,8,$FFFF:INK#7,$FFFF00          Note: CP24
407 REPeat Glp
408   IF gm=0:c1$='05':r1$='07':ELSE c1$=GA(gm,0):r1$=GA(gm,1)
409   CURSOR#7,124,6:PRINT#7,c1$;'x';r1$:k=CODE(INKEY$(-1))
410   SELect ON k
411     =208:gm=gm+1:IF gm>9:gm=9
412     =216:gm=gm-1:IF gm<0:gm=0
413     = 10:c$=c1$:r$=r1$:EXIT Glp
414     = 32,78,110,244:IF sm>0:CLS#7:RETurn
415   END SELect
416 END REPeat Glp
417 IF sm=2                                         Note:sm=2 Grid Resize
418   FOR p=0 TO pm:GTemp
419   DGrid:DIM SGrid(p,cm-1,rm-1)
420   FOR p=0 TO pm:GLive:fp=1:FCalc:SDraw
421   p=0:fg=0:gp=1:FCalc:SDraw
422 END IF
423 IF sm=1                                         Note:sm=1 New Grid Size
424   DGrid:DIM SGrid(pm,cm-1,rm-1),TGrid(17,36,32)
425   FOR pr=0 TO 2
426     FOR pc=0 TO 5:BLOCK#5,36,32,7+pc*37,11+pr*33,0   Note :Reset/CLS Frames
427   END FOR pr
{428   obg=0:nbg=0:col=7:cur=2:pn=0:fp=0:gp=0          :REMark Set Variables
 428   obg=0:nbg=0:col=$FFFFFF:cur=$FF0000:pn=0:fp=0:gp=0   Note: CP24
429 END IF
430 IF sm=0:sm=1:DGrid:DIM SGrid(pm,cm-1,rm-1),TGrid(17,36,32)   Note:sm=0 Sel Grid Size
431 CLS#7:WHelp
432 END DEFine
```

Note: GA Grid Attributes Identify the Grid Size column **cm** & rows **rm** with cell width **cw** and cell height **rh**, which are used to calculate the offsets for the Pointers **cx** & **cy**.




```
434 DEFine PROCedure DGrid
435 pm=17:cm=GA(gm,0):rm=GA(gm,1)
436 cw=GA(gm,2):rh=GA(gm,3):cx=15+cw/2:cy=31+rh/2
437 cur=2:BLOCK#4,228,172,18,36,0:BLOCK#4,30,10,72,14,0        Note: CP24 cur=$FF0000
438 IF cm>rm:cs=2:ELSE cs=0
439 IF cm=5:c$='05':r$='07':ELSE c$=cm:r$=rm
440 INK#4,7:CURSOR#4,73,14:PRINT#4,c$;'x';r$                  Note: CP24 INK#4,$FF0000
441 FOR c=0 TO cm:BLOCK#4,1,rm*rh,18+c*cw,36,241              Note:CP24  241-$B6B6B6
442 FOR r=0 TO rm:BLOCK#4,cm*cw,1,18,36+r*rh,241              Note:CP24  241-$B6B6B6
443 END DEFine
.
```

Note: Merge a second Frame file with one already loaded

```
445 DEFine PROCedure GMerge
446 FOR p=10 TO 17:GTemp
447 GFSel:IF ck=0:RETurn
448 FOR p=10 TO 17:GLive
449 p=0:fp=0:gp=1:FCalc:SDraw
450 END DEFine

452 DEFine PROCedure GTemp
453 FOR r=0 TO rm-1
454   FOR c=0 TO cm-1:TGrid(p,c,r)=SGrid(p,c,r)               Note: Temporary Copy
455 END FOR r
456 END DEFine

458 DEFine PROCedure GLive
459 FOR r=0 TO rm-1
460   FOR c=0 TO cm-1:SGrid(p,c,r)=TGrid(p,c,r)               Note: Restore Frames
461 END FOR r
462 END DEFine
```

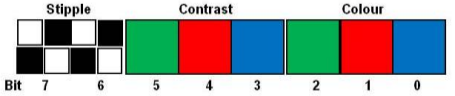
```
464 DEFine PROCedure InitGrid
465 LOCal a,b,c:cs%=0:x=0:cx=0:y=0:cy=0:pn=0:fp=0:gp=0
466 DIM a$(4),b$(5),c$(2),r$(2),p$(2),n$(2)
467 DIM TGrid(17,32,28),SFile$(20),CFile$(20)

469 REMark Grid Sizes
470 DIM GA(9,3):RESTORE 472
471  FOR a=0 TO 9:FOR b=0 TO 3:READ c:GA(a,b)=c:END FOR b:END FOR a
472  DATA  5,  7,18,16
473  DATA 12,10,16,14
474  DATA 16,12,13,12
475  DATA 16,16,12,10
476  DATA 20,16,11,  9                    Note: variables  cm, rm, cw, rh
477  DATA 20,20,10,  8
478  DATA 24,20,  9,  8
479  DATA 28,24,  8,  7
480  DATA 32,28,  7,  6
481  DATA 36,32,  6,  5

483 REMark Grid Win
484 ch=4:OPEN#ch,scr_251x212a250x4:BORDER#ch,1,3:PAPER#ch,0:CLS#ch
485 INK#ch,5:CURSOR#ch,0,2:PRINT#ch,'(P)alette      Colour ON/OFF     TAB'
486 INK#ch,4:CURSOR#ch,138,2:PRINT#ch,'ON'  :BLOCK#ch,16,3,180,6,5
487 INK#ch,2:CURSOR#ch,156,2:PRINT#ch,'OFF':BLOCK#ch,10,7,230,4,7
488 INK#ch,5:CURSOR#ch,6,14:PRINT#ch,'Grid Size:              Frame Step← '
489 BLOCK#ch,2,4,234,16,5:c$='05':r$='07'

Note: QPC
484 ch=4:OPEN#ch,scr_251x210a270x54:BORDER#ch,1,$FFC82D:PAPER#ch,0:CLS#ch
485 INK#ch,$FFFF:CURSOR#ch,0,2:PRINT#ch,'(P)alette      Colour ON/OFF     TAB'
486 INK#ch,$FF00:CURSOR#ch,138,2:PRINT#ch,'ON'  :BLOCK#ch,16,3,180,6,$FFFF
487 INK#ch,$FF0000:CURSOR#ch,156,2:PRINT#ch,'OFF':BLOCK#ch,10,7,230,4,$FFFFFF
488 INK#ch,$FFFF:CURSOR#ch,6,14:PRINT#ch,'Grid Size:              Frame Step← '
489 BLOCK#ch,2,4,234,16,$FFFF

491 REMark Frame Win
492 ch=5:OPEN#ch,scr_240x122a8x94:BORDER#ch,1,3:PAPER#ch,0:CLS#ch
493 FOR i=0 TO 3:BLOCK#ch,222,1,6,10+i*33,241
494 FOR i=0 TO 6:BLOCK#ch,1,100,6+i*37,10,241

Note: QPC
492 ch=5:OPEN#ch,scr_240x122a8x94:BORDER#ch,1,$FF00FF:PAPER#ch,0:CLS#ch
493 FOR i=0 TO 3:BLOCK#ch,222,1,6,10+i*33,$B6B6B6
494 FOR i=0 TO 6:BLOCK#ch,1,100,6+i*37,10,$B6B6B6

495 FOR i=0 TO 5:CURSOR#ch,24+i*36,0:PRINT#ch,i
496 FOR i=0 TO 5:CURSOR#ch,24+i*36,110:PRINT#ch,CHR$(i+99)
497 END DEFine

499 DEFine PROCedure GHelp
500 ch=7:CLS#ch:INK#ch,5
501 CURSOR#ch,18, 8:PRINT#ch,'Grid xy(← ↑ ↓ → ) Flip(XY) Turn(z-Z)'
502 CURSOR#ch,36,18:PRINT#ch,'Pan/Scroll Grid (Shift← ↑ ↓ → )':ch=5
503 END DEFine
```

```
600 DEFine PROCedure CPQL                          Note: QL
601 ch=7:CLS#ch:INK#ch,7
602 CURSOR#ch,76,16:PRINT#ch,'R  G  B   Col      <TAB>'
603 CURSOR#ch,46,24:PRINT#ch,'S   r   g   b    XOR       Exit'

600 DEFine PROCedure CP24                          Note: QPC
601 ch=7:CLS#ch:INK#ch,$FFFFF
602 CURSOR#ch,52,16:PRINT#ch,'R   G   B      Col   <TAB>'
603 CURSOR#ch,52,24:PRINT#ch,'r   g   b    24-bit   Exit'
604 FOR i=0 TO 3:BLOCK#ch,14,10,61+i*30,14,$FFFFFF

605 CURSOR#ch,4,24:PRINT#ch,'Switch'
606 FOR i=0 TO 13:BLOCK#ch,10,8,10+i*16,2,CP(i)
607 xg%=7:x1%=cs%:y1%=1:p1=CP(cs%):cp1=p1:col=cp1
608 CRead:CPrnt:pck=1:CMode
609 REPeat CP_lp
610   IF pck=0:CPSel:ELSE CPrnt:cp1=col
611   k=CODE(INKEY$(-1))
612   SELect ON k

613     =  66:  IF BP%=1   :BP% =0 :ELSE BP% =1
614     =  98:  IF bc% =1  :bc%  =0 :ELSE bc%  =1
615     =  71:  IF GP%=4   :GP%=0  :ELSE GP% =4
616     =103:  IF gc% =4   :gc%  =0 :ELSE gc%  =4
617     =  82:  IF RP%=2   :RP% =0 :ELSE RP% =2
618     =114:  IF rc%  =2  :rc%   =0 :ELSE rc%   =2

613     =  66:  BP%=BP%+1 :IF BP%>255:BP%=0
614     =  98:  BP%=BP% -1 :IF BP%<0:BP%=255
615     =  71:  GP%=GP%+1:IF GP%>255:GP%=0
616     =103:  GP%=GP% -1:IF GP%<0:GP%=255
617     =  82:  RP%=RP%+1 :IF RP%>255:RP%=0
618     =114:  RP%=RP% -1 :IF RP%<0:RP%=255

619     = 83,115:stn%=stn%+1:IF stn%>3:stn%=0          Note: QL Stipple

620     =192:IF pck=0:cs%=cs% -1:IF cs%<0  :cs%=13
621     =200:IF pck=0:cs%=cs%+1:IF cs%>13:cs%=  0

622     =208:IF pck=1:cp1=cp1+1:CRead                  Note: QL Colour Palette 0-255
623     =216:IF pck=1:cp1=cp1 -1:CRead                 Note: QL Colour Palette 0-255

624     =  32:CMode
625     =  9,80,112:FCalc:EXIT CP_lp
626   END SELect
627 END REPeat CP_lp
628 END DEFine                                        Note x1%=cs%

630 DEFine PROCedure CPSel
631 BLOCK#7,14,12,xg%+x1%*16,y1%,0:BLOCK#7,10,8,xg%+2+x1%*16,y1%+2,p1
632 p1=CP(cs%):x1%=cs%:BLOCK#7,14,12,xg%+x1%*16,y1%,7
633 BLOCK#7,12,10,xg%+1+x1%*16,y1%+1,0:BLOCK#7,10,8,xg%+2+x1%*16,y1%+2,p1
634 END DEFine
```

```
636 DEFine PROCedure CMode
637 IF pck=0                                  Note: Colour CPQL   CP24
638    pck=1:cp1=CP(cs%):CRead                                ↓      ↓
639    CURSOR#7,6,14:PRINT#7,' ↑   ↓ ':BLOCK#7,16,3,16,17,5      [ $FFFF ]
640 ELSE
641    pck=0:CP(cs%)=cp1:CRead
642    CURSOR#7,6,14:PRINT#7,' ←   → ':BLOCK#7,16,3,16,17,5      [ $FFFF ]
643 END IF
644 END DEFine
```

```
646 DEFine PROCedure CRead                          Note: QL
647 IF cp1>255:cp1=0
648 IF cp1<0:cp1=255
649 stn% =cp1 DIV 64
650 Con%=cp1 MOD 64 DIV 8  :rc%=Pal(Con%,0):gc%=Pal(Con%,1):bc%=Pal(Con%,2)
651 Maj% =cp1 MOD 64 MOD 8:RP%=Pal(Maj%,0):GP%=Pal(Maj%,1):BP%=Pal(Maj%,2)
652 END DEFine
```

Note: QL Stipple/Contrast/ Colour



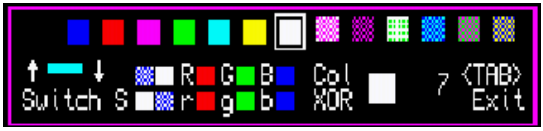
Note: QPC 24Bit Colour


```
646 DEFine PROCedure CRead
647 RP%=cp1 DIV 65536
648 GP%=cp1 MOD 65536 DIV 256
649 BP%=cp1 MOD 65536 MOD 256
650 END DEFine
```

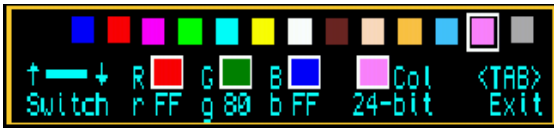
```
654 DEFine PROCedure CPrnt                          Note: QL
655 col=(RP%+GP%+BP%)+(rc%+gc%+bc%)*8+stn%*64
656 st%=stn%:r1%=RP%:r2%=rc%:g1%=GP%:g2%=gc%:b1%=BP%:b2%=bc%
657 IF col<8:r2%=r1%:g2%=g1%:b2%=b1%:st%=3
658 BLOCK#7,8,6,56,18,Stp(st%,0):BLOCK#7,8,6,65,18,Stp(st%,1)
659 BLOCK#7,8,6,56,26,Stp(st%,2):BLOCK#7,8,6,65,26,Stp(st%,3)
660 BLOCK#7,8,6,   84,18,r1%:BLOCK#7,8,6,  84,26,r2%
661 BLOCK#7,8,6,102,18,g1%:BLOCK#7,8,6,102,26,g2%
662 BLOCK#7,8,6,120,18,b1%:BLOCK#7,8,6,120,26,b2%
663 BLOCK#7,12,10,162,20,col
664 CURSOR#7,180,19:PRINT#7,FILL$(' ',3-LEN(col))&col
665 END DEFine
```

```
652 DEFine PROCedure CPrnt                          Note: QPC
653 r1=RP%*65536:g1=GP%*256:b1=BP%:col=r1+g1+b1
654 BLOCK#ch,12,8,62,15,r1
655 CURSOR#ch,  62,24:PRINT#ch,HEX$(RP%,8)
656 BLOCK#ch,12,8,92,15,g1
657 CURSOR#ch,  92,24:PRINT#ch,HEX$(GP%,8)
658 BLOCK#ch,12,8,122,15,b1
659 CURSOR#ch,122,24:PRINT#ch,HEX$(BP%,8)
660 BLOCK#ch,12,8,152,15,col
661 END DEFine
```

```
667 DEFine PROCedure InitCPQL                    Note: CPQL
668 LOCal col, cs, s1, s2, r, g, b
669 DIM CP(13):RESTORE 671        :REMark Colour Palette 0-11
670  FOR cs=0 TO 13:READ col:CP(cs)=col
671  DATA   0,1,2,3,4,5,6,7,227,216,31,225,251,254

673 DIM Pal(7,3):RESTORE 676        :REMark Palette Colour/Contrast
674  FOR col=0 TO 7:READ r,g,b:Pal(col,0)=r:Pal(col,1)=g:Pal(col,2)=b
675  REMark 0 to 7 Colour RP%,GP%,BP% or rc%,gc%,bc%
676  DATA 0,0,0,0,0,1,2,0,0,2,0,1,0,4,0,0,4,1,2,4,0,2,4,1

678 DIM Stp(3,3):RESTORE 682        :REMark Stipple 0-3
679  FOR s1=0 TO 3
680    FOR s2=0 TO 3:READ s:Stp(s1,s2)=s
681  END FOR s1
682  DATA 7,241,7,7,7,7,241,241,241,7,241,7,241,7,7,241
683 END DEFine
```




```
663 DEFine PROCedure InitCP24                    Note: CP24
664 LOCal col,r,g,b :cs%=7          :REMark  colour select%
665 DIM CP(13):RESTORE 670          :REMark Colour Palette 0 to 11
666  FOR col=0 TO 13
667     READ r,g,b:CP(col)=r*65536+g*256+b
668  END FOR col
669  REMark 0 to 13 Colour RP%,GP%,BP%
670  DATA 0,0,0                     :REMark Black
671  DATA 0,0,$FF                   :REMark Blue
672  DATA $FF,0,0                   :REMark Red
673  DATA $FF,0,$FF                 :REMark Magenta
674  DATA 0,$FF,0                   :REMark Green
675 DATA 0,$FF,$FF                  :REMark Cyan
676  DATA $FF,$FF,0                 :REMark Yellow
677  DATA $FF,$FF,$FF               :REMark White
678  DATA $6D,$24,$24               :REMark Brown
679  DATA $FF,$D8,$B8               :REMark Beige
680 DATA $FF,$C0,$40                :REMark Orange
681  DATA $40,$C0,$FF               :REMark Light Blue
682  DATA $FF,$80,$FF               :REMark Pink
683  DATA $AA,$AA,$AA               :REMark Slate
684 END DEFine
```



```
800 DEFine PROCedure GFSel                          Note: Grid File Select
801 IF dg%=0
802   CURSOR#7,26,14:PRINT#7,'Save as (D)ata or (G)rid file'
803   IF KEYROW(3)=64:dg%=3:CLS#7:GO TO 809          :REMark (G)rid
804   IF KEYROW(4)=64:dg%=4:CLS#7:GO TO 810          :REMark (D)ata
805   GO TO 802
806 END IF
807 IF dg%=1:a$='Grid':b$='Load '
808 IF dg%=2:a$='Grid':b$='Merge'
809 IF dg%=3:a$='Grid':b$='Save '
810 IF dg%=4:a$='Data':b$='Save'
811 CURSOR#7,38,  2:PRINT#7,'Select ';a$;' file to ';b$;
812 CURSOR#7,20,14:PRINT#7,'↑ ↓       QBP';a$;c$;'x';r$;'_ ← → (y/n)'
813 INK#7,6
814 REPeat Sel_lp
815   CURSOR#7,38,14:PRINT#7,Drv$(dn%):CURSOR#7,146,14:PRINT#7,pn%
816   k=CODE(INKEY$(-1))
817   SELect ON k
818     =192:pn%=pn% -1:IF pn%<0:pn%=0               :REMark File  0-9
819     =200:pn%=pn%+1:IF pn%>9:pn%=9
820     =208:dn%=dn%+1:IF dn%>8:dn%=8                :REMark Device mdv1_ to Dos1_
821     =216:dn%=dn% -1:IF dn%<1:dn%=1
822     =89,121:ck%=1:SFile$='QBP'&a$&c$&'x'&r$&'_'&pn%:EXIT Sel_lp
823     =78,110:INK#7,5:CLS#7:RETurn
823     =78,110:INK#7,$FFFF:CLS#7:RETurn             Note: Hex Numbers for CP24
824   END SELect
825 END REPeat Sel_lp
826 IF dg%=1 OR dg%=2:GFChk:IF ck%=1:GLoad:CLS#7:ELSE CLS#7:RETurn
827 IF dg%=3 OR dg%=4:GFChk:GSave:CLS#7
828 END DEFine

830 DEFine PROCedure GFChk
831 INK#7,5:CURSOR#7,20,24:PRINT#7,'  Searching...  ':PAUSE 20
832 DELETE Drv$(dn%)&'FList'
833 OPEN_NEW#9,Drv$(dn%)&'FList':DIR#9,Drv$(dn%):CLOSE#9
834 OPEN_IN#9,Drv$(dn%)&'FList'
835 REPeat dir_lp
836   IF EOF(#9)
837     CLOSE#9:CURSOR#7,20,24:PRINT#7,'  File NOT Found'
838     PAUSE 20:ck%=0:EXIT dir_lp
839   END IF
840   INPUT#9,CFile$:IF CFile$==SFile$:CLOSE#9:ck%=1:EXIT dir_lp
841 END REPeat dir_lp
842 END DEFine
```

```
844 DEFine PROCedure GLoad
845 CURSOR#7,20,24:PRINT#7,'  Loading...    ':PAUSE 20
846 fp%=1:gp%=0:OPEN_IN#9,Drv$(dn%)&SFile$:INPUT#9,pm%\rm%\cm%
847 pt%=pm%:IF dg%=2:pm%=9
848 FOR p=0 TO pm%
849   FCalc
850   FOR r=0 TO rm%-1
851     FOR c=0 TO cm%-1:INPUT#9,SGrid(p,c,r):GDraw
852   END FOR r
853 END FOR p
854 CLOSE#9:pm%=pt%:p%=0:FCalc:fp%=0:gp%=1:SDraw
855 END DEFine

857 DEFine PROCedure GSave
858 IF ck=1
859    CURSOR#7,20,24:PRINT#7,'   Overwrite (y/n)'
860    IF INKEY$(-1)=='y':DELETE Drv$(dn%)&SFile$:ELSE RETurn
861 END IF
862 num=2000:CURSOR#7,20,24:PRINT#7,'  Saving...       ':PAUSE 20
863 OPEN_NEW#9,Drv$(dn%)&SFile$
864 IF dg%=4:PRINT#9,num&' DATA '&pm%&','&rm%&','&cm%:ELSE PRINT#9,pm%\rm%\cm%
865 FOR p=0 TO pm%
866   IF dg%=4:num=num+1:PRINT#9,num&' :'
867   FOR r=0 TO rm%-1
868     IF dg%=4
869       num=num+1:PRINT#9,num&' DATA ';
870       FOR c=0 TO cm%-1:PRINT#9,SGrid(p,c,r);',';
871       PRINT#9,SGrid(p,cm%-1,r)
872     ELSE
873       FOR c=0 TO cm%-1:PRINT#9,SGrid(p,c,r)
874     END IF
875   END FOR r
876 END FOR p
877 CLOSE#9:p%=0
878 END DEFine

880 DEFine PROCedure InitDrive
881 DIM Drv$(8,5):RESTORE 883
882 FOR dn=1 TO 8:READ d$:Drv$(dn)=d$
883 DATA 'mdv1_','mdv2_','flp1_','flp2_','win1_','win2_','dos1_','dos2_'
884 END DEFine
```

QBBMDesign_QL - QBBMDesign_QPC

```
900 DEFine PROCedure InitWin
901 WINDOW#0,496,32,6,218  :PAPER#0,0:INK#0,7:CLS#0
902 WINDOW#1,496,212,6,4   :PAPER#1,0:INK#1,7:CLS#1
903 WINDOW#2,496,213,6,4   :PAPER#2,0:INK~2,7:CLS#2
904 ch=7:OPEN#ch,scr_240x38a8x54:BORDER#ch,1,3:PAPER#ch,0:CLS#ch
905 BMTitle:InitDrive:InitGrid:InitPal
906 END DEFine
```

Note: The Window#0,#1,#2 xy coordinates are set to reflect the Higher resolution and larger Width/Depth measurements of the screen. The Border Colours are also changed for CP24 Mode and written in Hex.

```
900 DEFine PROCedure InitWin                                    Note:QPC
901 WINDOW#0,496,32,26,268   :PAPER#0,0:INK#0,$FFFFFF:CLS#0
902 WINDOW#1,496,212,26,54   :PAPER#1,0:INK#1,$FFFFFF:CLS#1
903 WINDOW#2,496,213,26,54   :PAPER#2,0:INK#2,$FFFFFF:CLS#2
904 ch=7:OPEN#ch,scr_240x36a28x104:BORDER#ch,1,$FFC82D:PAPER#ch,0:CLS#ch
905 BMTitle:InitDrive:InitGrid:InitCP24
906 END DEFine
```

```
950 DEFine PROCedure BMTitle
951 ch=6:OPEN#ch,scr_240x49a8x4:BORDER#ch,1,3:PAPER#ch,0:CLS#ch
952 CSIZE#ch,2,1:OVER#ch,1
953 INK#ch,2:FOR i=0 TO 1:CURSOR#ch,1+i,1+i:PRINT#ch,'QBITS BITMAP Design'
954 INK#ch,7:FOR i=0 TO 1:CURSOR#ch,3+i,1+i:PRINT#ch,'QBITS BITMAP Design'
955 CSIZE#ch,0,0:OVER#ch,0:INK#ch,5
956 CURSOR#ch,  2,24:PRINT#ch,'(F1)Action (F2)BkGnd (F3)Copy (F4)Grid'
957 CURSOR#ch,12,34:PRINT#ch,'(M)erge (N)ew (L)oad (S)ave (R)eset'
960 END DEFine
```

Note: The Border colour of the Title screen has changed and the Title uses Hex ($num) Style Numbering.

```
950 DEFine PROCedure BMTitle                                    Note:QPC
951 ch=6:OPEN#ch,scr_240x49a28x54:BORDER#ch,1,$FFC82D:PAPER#ch,0:CLS#ch
952 CSIZE#ch,2,1:OVER#ch,1
953 INK#ch,$FF0000:FOR i=0 TO 1:CURSOR#ch,1+i,1+i:PRINT#ch,'QBITS BITMAP Design'
954 INK#ch,$FFFFFF:FOR i=0 TO 1:CURSOR#ch,3+i,1+i:PRINT#ch,'QBITS BITMAP Design'
955 CSIZE#ch,0,0:OVER#ch,0:INK#ch,$FFFF
956 CURSOR#ch, 2,24:PRINT#ch,'(F1)Action (F2)BkGnd (F3)Copy (F4)Grid'
957 CURSOR#ch,12,34:PRINT#ch,'(M)erge (N)ew (L)oad (S)ave (R)eset'
960 END DEFine
```

QBITS Colour Palette Conversion
Having the **two program versions** created the need at times to load a Frame File created with the **QL Colour Palette** and convert to a **24-Bit Colour Palette** of the **QPC version**.

To accomplish this **PROC**edure **CPConversion** is attached to the **QBBMDesign_QPC** Prog version. **Line 1005 SFile$** contains the Frame File to be converted (this can be changed as required).

Calling **CPConversion:** The Frame File is loaded into the **SGrid** array and should pose no problems. By use of **GTemp** the contents are then copied to the **TGrid** Array. Then it was a question of reading the **TGrid** array back into the **SGrid** array and in such a way that each **QL Palette Colours were** converted into a **24-Bit Palette Hex** number.

Taking a simplistic approach reading each **TGrid** array entry and interpreting its **CPQL Palette** number as if from the **24-Bit Colour Palette** works to some extent if basic colours are used. However, this does not apply to more complex Colour, Contrast and Stipple combinations. If this is desired, the information contain within this PDF should help in writing your own code

```
1000 DEFine PROCedure CPConversion                    Note:(QL to QPC)
1001 REMark RUN & Select Grid: Exit Prog :Change SFile$ variable as required…
1002 CLS:
1005 SFile$='QBPGrid24x20_0':dn=6:ck=0:GLoad
1007 PRINT 'File Loaded...'
1010 FOR p=0 TO pm:GTemp
1020 FOR p=0 TO pm
1023   FOR r=0 TO rm-1
1024     FOR c=0 TO cm-1:SGrid(p,c,r)=CP(TGrid(p,c,r)):PRINT#0,;'.';
1034   END FOR r
1044 END FOR p
1046 PRINT 'Save File...'
1049 SFile$='QBTGrid24x20_0':dn=6:dg=3:ck=0:GSave
1054 END DEFine
```

QBITS BITMap Files
To differentiate between the two Palettes I use (**P**)al and (**T**)rue in their filenames.

QBBMDesign_QL
QBPGrid05x07_0
QBPGrid20x16_0
QBPGrid24x20_0
QBPGrid36x32_0
QBBMDesign_QPC
QBTGrid20x16_0
QBTGrid24x20_0
QBTGrid32x28_0

QBBMDisplay
QBPData32x28_0

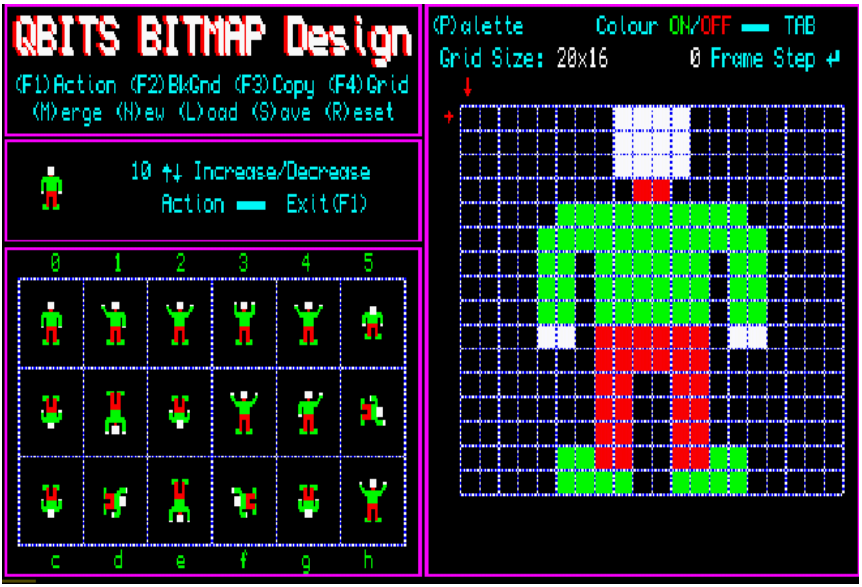
QBITS BITMap Display

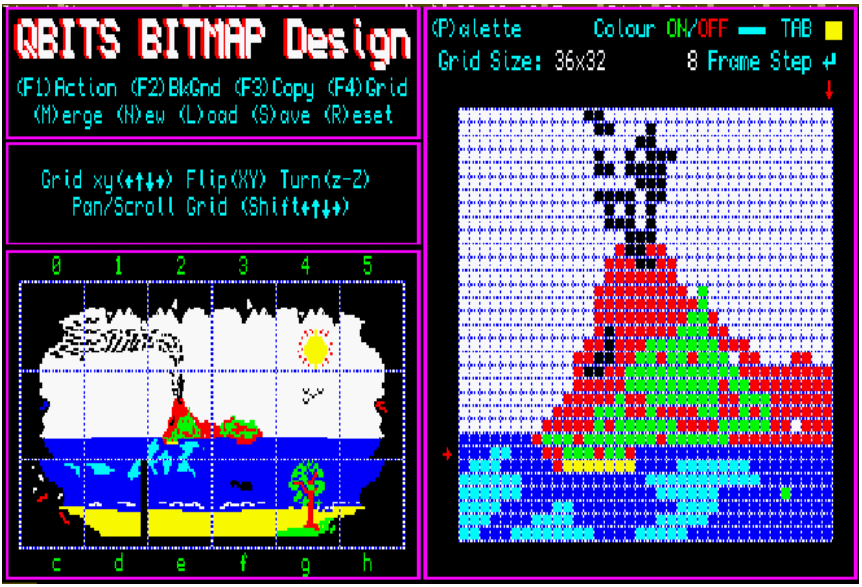
MERGE a Saved **QBPDatacmxrm_n** file with program.

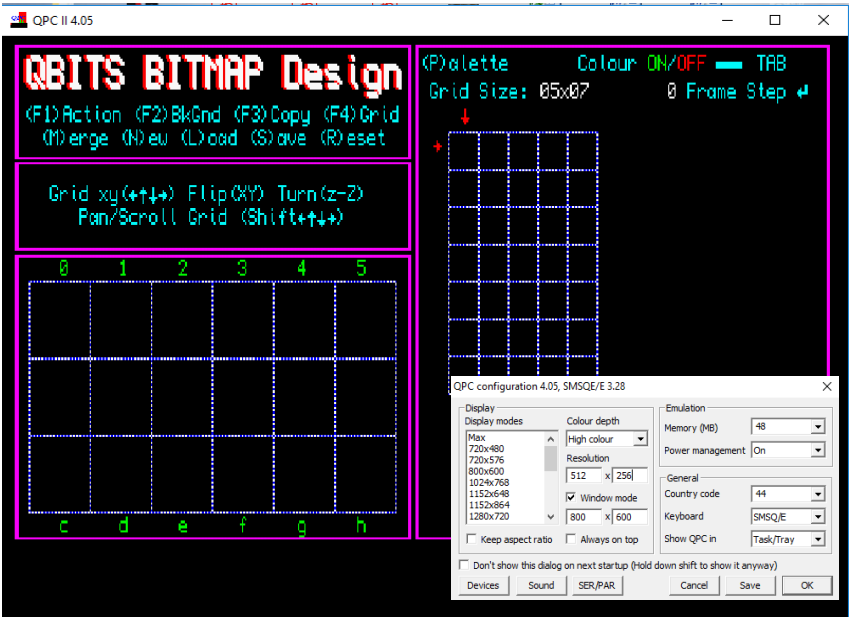
```
100 REMark QBBMDisplay (QBITS BITMap DATA Display v01 2019)

102 km=4:en=2:Init
103 MERGE win1_QBPData32x28_0              :REMark Collect DATA File info
104 DLoad

106 DEFine PROCedure Init
107 WINDOW#1,496/(km/4),212,6,6:BORDER#1,1,2:PAPER#1,0:CLS#1
108 CSIZE#1,2,1:OVER#1,1:INK#1,4
109 INK#1,2:FOR i=0 TO 1:CURSOR 20+i,8+i:PRINT 'QBITS BITMAP Display'
110 INK#1,7:FOR i=0 TO 1:CURSOR 22+i,8+i:PRINT 'QBITS BITMAP Display'
111 CSIZE#1,0,0:OVER#1,0
112 CLS#0:CURSOR#0,20,6:PRINT#0,'Reading DATA...'
113 END DEFine

115 DEFine PROCedure DLoad
116 num=2000:RESTORE num:READ pm,rm,cm:DIM DGrid(pm,cm-1,rm-1)
117 FOR p=0 TO pm
118   FOR r=0 TO rm-1
119     FOR c=0 TO cm-1:READ a$:DGrid(p,c,r)=a$
120   END FOR r
121 END FOR p
122 CLS#0:BLOCK 2+192*en,2+en*84,49,29,7:FOR p=0 TO pm:DCal:DDraw
123 END DEFine

125 DEFine PROCedure DCal
126 ch=1:pc=p:pr=0                         :REMark Enlarge Image en=1 or 2
127 IF p> 5:pc=p -6:pr=28*en
128 IF p>11:pc=p-12:pr=56*en
129 px=50+pc*32*en
130 py=30+pr
131 END DEFine

133 DEFine PROCedure DDraw
134 FOR r=0 TO rm-1
135   FOR c=0 TO cm-1:BLOCK#ch,en,en,px+c*en,py+r*en,DGrid(p,c,r)
136 END FOR r
137 END DEFine
```

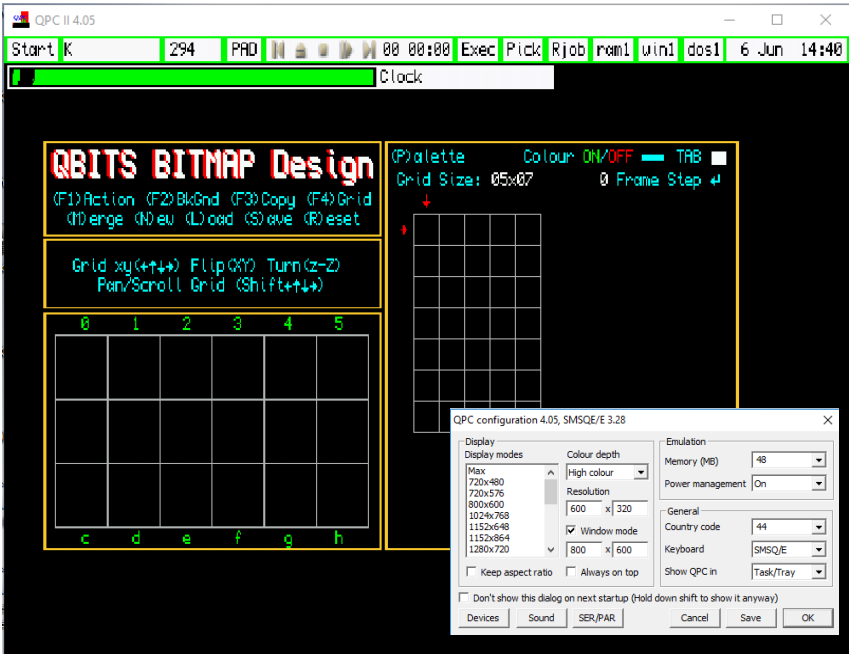
Merged file **win1_QBPData32x28_0**

```
2000 DATA 17,28,32                         REMark  pm, rm, cm
2001 :
2002 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2003 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2004 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
etc..
```


QBITS BITMAP Design
(F1)Action (F2)BkGnd (F3)Copy (F4)Grid
(M)erge (N)ew (L)oad (S)ave (R)eset
10 ↑↓ Increase/Decrease
Action ▬ Exit(F1)
0 1 2 3 4 5
c d e f g h
(P)alette Colour ON/OFF ▬ TAB
Grid Size: 20x16 0 Frame Step ↵


QBITS BITMAP Design
(F1)Action (F2)BkGnd (F3)Copy (F4)Grid
(M)erge (N)ew (L)oad (S)ave (R)eset
Grid xy(←↑↓→) Flip(XY) Turn(z-Z)
Pan/Scroll Grid (Shift←↑↓→)
0 1 2 3 4 5
c d e f g h
(P)alette Colour ON/OFF ▬ TAB
Grid Size: 36x32 8 Frame Step ↵


QPC II 4.05
QBITS BITMAP Design
(F1)Action (F2)BkGnd (F3)Copy (F4)Grid
(M)erge (N)ew (L)oad (S)ave (R)eset
Grid xy(←↑↓→) Flip(XY) Turn(z-Z)
Pan/Scroll Grid (Shift←↑↓→)
0 1 2 3 4 5
c d e f g h
(P)alette Colour ON/OFF TAB
Grid Size: 05x07 0 Frame Step
QPC configuration 4.05, SMSQE/E 3.28
Display
Display modes
Max
720x480
720x576
800x600
1024x768
1152x648
1152x864
1280x720
Colour depth
High colour
Resolution
512 x 256
Window mode
800 x 600
Keep aspect ratio
Always on top
Emulation
Memory (MB) 48
Power management On
General
Country code 44
Keyboard SMSQ/E
Show QPC in Task/Tray
Don't show this dialog on next startup (Hold down shift to show it anyway)
Devices
Sound
SER/PAR
Cancel
Save
OK


QPC II 4.05
Start K 294 PAD 00 00:00 Exec Pick Rjob ram1 win1 dos1 6 Jun 14:40
Clock
QBITS BITMAP Design
(F1)Action (F2)BkGnd (F3)Copy (F4)Grid
(M)erge (N)ew (L)oad (S)ave (R)eset
Grid xy(←↑↓→) Flip(XY) Turn(z-Z)
Pan/Scroll Grid (Shift←↑↓→)
0 1 2 3 4 5
c d e f g h
(P)alette Colour ON/OFF TAB
Grid Size: 05x07 0 Frame Step
QPC configuration 4.05, SMSQE/E 3.28
Display
Display modes
Max
720x480
720x576
800x600
1024x768
1152x648
1152x864
1280x720
Colour depth
High colour
Resolution
600 x 320
Window mode
800 x 600
Keep aspect ratio
Always on top
Emulation
Memory (MB) 48
Power management On
General
Country code 44
Keyboard SMSQ/E
Show QPC in Task/Tray
Don't show this dialog on next startup (Hold down shift to show it anyway)
Devices
Sound
SER/PAR
Cancel
Save
OK