A Simple GUI pack

A purely MMBasic GUI pack to provide a WindowsCE/95 look-a-like LCD panel interface for your projects.

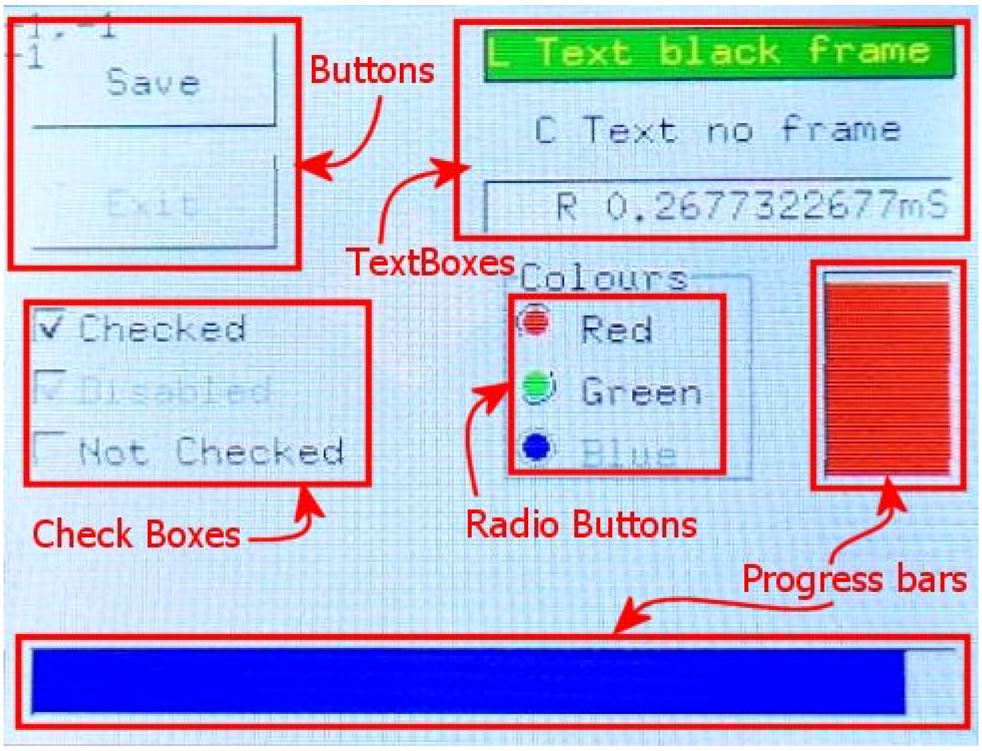
This is a work in progress in my spare time. This document has lots of improvements required - and I'll do them… just need a bit more time. Still to come: Lists, Drop-down Lists, tab strips, pop-up windows (if I can work out a way to do it). As it is now, it provides a lot of functionality and is usable. If you only need the simpler end of the GUI - buttons, checkboxes, radio buttons and text boxes etc, it is good to go. Let's face it; they probably cover 100% of simple interfaces.

Enjoy

A Universal, Simple GUI pack

The larger MicroMites and ARMMites have a built –in GUI that allows your code to have an interactive LCD panel with all the usual gadgets (buttons, check boxes etc.) that we have become used to in the last 30 years. The MicroMite Mk2 is constrained in that there simply isn't room in flash to add such luxuries to the firmware for these small but powerful microcontrollers. The following is a pure MMBasic solution so it can easily be deployed on any 'Mite platform and thanks to the efforts of others is platform independent across the whole MicroMite/ColourMaxiMite2 eco system.

GUI



Conventions

Throughout this article, things that get put on the screen using this GUI pack are called "Gadgets" (I was an Amiga boy – sue me). In the GUI software pack, anywhere you see "Obj" (short for object) it is the same as a gadget – the two terms are interchangeable here.

Variable and Constant name considerations

- Starting with a capital are global (defined in the main body of the program).
- Constants start with upper case.
- Local variables start with lower case (defined inside a Sub or Function).
- All arrays must start at 0.

Background

Many of my own projects use the small Micromite Mk2, a diminutive yet powerful PIX32MX170-based microcontroller with 60K of program flash and 50K RAM and have been, in the past, console only due to the pain of making a good interactive screen (on an LCD panel). In later projects, I have been building my own GUIs based very strongly around a WindowsCE/95 look-a-like. I just happen to think the nice clean lines of the GUI with its default grey background and relief style buttons etc is so much nicer than the "Metro" or flat look of Windows 10.

Not having a GUI subsystem, any such project that has buttons etc. had to be built from the ground up each time and were specifically managed with fixed co-ordinates and code to handle the presses of each button. It was a very inflexible solution. Modifications were a nightmare because the co-ordinates where specifically tested for presses etc. so if you moved a gadget, you had to hunt-down and tweak the code that looked for presses on it… annoying horrible stuff like that.

It had been on the back-burner of my mind to build (in MMBasic) a complete subsystem that got rid of that and just let me design the GUI then concentrate on the application code and not have to worry about managing the screen a-fresh each time. Other people had shown an interest in the displays on some projects and it was a bit embarrassing to reveal it was bespoke each time. The following is the fruits of my labours. It is currently about 12K in size – which might come across as a lot of space at first glance, but when you consider I would often have 5 or 6K of very inflexible code to handle a screenful of buttons, it isn't that much of a sacrifice. You could trim it down to get rid of the bits you don't want if size becomes a problem - as you will see it provides quite an array of gadgets. Even drop-down lists and pop-up windows (eventually)!

A problem with driving LCD panels from MMBasic on smaller beasts is the one-directional nature of the communication with the LCD panel… you can throw stuff on them, but it is hard work (and slow in Basic) to pull back what is on the screen and buffer it so you can repair "damage". In my approach here (confession: I have stolen a few ideas from my years as a VB programmer) is structured items that can be quickly re-drawn when scribbled over. An example here is the text box. At first thought you might wonder why you would use it instead of simply throwing text on the panel as normal. If you now think of a drop down menu drawing all over the screen contents, unless you know what it has damaged, how do you repair your lovely GUI? With simple Text, that isn't possible unless you have buffered the memory of the affected screen area (this is what Windows does). With a structured text box, the list or pop-up can tell which items it has drawn over and so when it goes away, it can easily initiate re-drawing just those items.

Structure Mechanism

Two shared arrays hold the attributes of each gadget; its type, its place on the screen, the colours, its properties etc. The limit to the number of gadgets is available RAM. The values stored here are then used to draw the gadgets using the in-built primitives of MMBasic (Box, Line, Text etc…)

Each gadget has a numerical “name” that can be whatever you like. It is a number you decide in your program and is entirely defined by you - any 31 bit positive integer (I have plans for the top 32 bits). Internally, all objects are identified by a pointer, so where you see “n” in the code, that is usually it - you never need to concern yourself with it. The codepack will translate back and forth between ID and pointer as needed. Whenever you want to talk to one of your gadgets, you use the ID you gave it, never the internal pointer. You can use the ID to apply logical grouping to your gadgets.

I have an idea to allow pages of gadgets which would give a multi-screen effect simply by hiding everything not in a certain range… perhaps tabs along the top to select between the pages. You could even have several “layers” of them and simply switch them on and off as you see fit making context-sensitive “screens” a breeze.

There are lots of areas for improvement - speed-wise, I could provide a short-cut sub for drawing an object in places where we already have its pointer, but that's two Subs and then the size starts to creep up and we are impacting the precious 60K of Flash space. Always a trade-off.

Code Portability

The MicroMite range of controllers is very broad for a single concept. Tiny 28pin devices, all the way up through to the mighty ARM based CMM2. The astonishing thing here is the degree of code compatibility - certainly backwardly. I passionately believe in making my code as universal as possible and some tests done recently on TheBackShed forum have shown that the original code as first published was subject to “lift 'n' shift” onto several other 'mites and ran without any tweaks or issues. The CMM2 is a leap in what is technologically available on the platform and provides all the HID features of modern PCs - inevitably is is difficult to keep things 100% compatible as the language has to evolve to make these features available to the user. TassyJim (also from TBH) has done some Stirling work to maintain platform independence. This results in some small modifications to the code and where as these do increase its size marginally, I have decided to include them in the belief that compatibility is king across the various 'mite systems and that they are easily enough identified and removed if size on the smaller platforms becomes an issue. The pros of having the GUI work on everything out-weigh the cons of having to remove the auto-configure if you don't want it. If you have space - I strongly recommend keeping them as future releases of this code here will as well and you'll be saving yourself hassle in filleting the code when you upgrade for new/improved features.

Stuff you need to know (in no particular order)

- Not all of this is working yet. Gadgets that have yet to be finished are shown.
- There are nine different gadget types – we’ll discuss them all in detail further on:
  - Button, Radio, CheckBox, Slider, TextBox, List Static, List DropDown, Progress Bar, Frame
  - Every gadget has a set of attributes which are defined in the preamble of your code – that is just a convenient place to put them; you can actually define and redefine all aspects of a gadget at anytime. A function, **GUIObjDef** will format them and correctly fill the O() array to keep track without any further intervention by your own program.
- Gadgets have properties and values that you can read and write to influence their operation.
- After a gadget is defined and properties set etc., nothing happens on screen until you issue a

Draw command for it (an exception here is animations when you touch certain types of gadget).
- All gadgets are touchable and you can find out which really easily. Even gadgets that don't generally do anything when touched (like Progress Bars) will report back their number if you touch them. It's entirely down to you how you "run" your GUI.
- Multiple fonts are not directly supported; if you set the font when you define affected gadgets and when you draw them, it *may* work. It will come but I just need to get it all working first.

Actually doing it – Getting usable Gadgets on the screen

There is some unavoidable setup required in the preamble of your program besides simply placing the GUI pack in your program. The following fragment of code should be placed near the start and executed before your program gets underway with its main tasks.

Preamble

```
'{mandatory GUI config
    Option Base 0
    Const cGy=&hdddddd'Pale grey screen background
    Dim Integer Objs=20,P=0,Flags,Xt,Yt,CMM2
    Dim Integer O(Objs,10)'GUI object settings
    Dim String Ot(Objs) Length 64'GUI object text
    Colour 0,cGy:Cls

    'CMM2 Compatibility - and any others if necessary
    Select Case MM.Device$
        Case "Colour Maximite 2"
            mp=MM.info(option mouse) ' mouse port
            CMM2=1
            Option Console Serial
            GUI CURSOR ON 0, 100,100,rgb(red)
            CONTROLLER MOUSE OPEN mp
        Case Else
    End Select

'}
```

- A constant is defined for the background colour.
- Some global variables are defined:
    - Objs is the total number of gadgets usable, increase or decrease as your application requires.
    - P is the pointer to the next available gadget number – this is an internal number and points to a position in the array – **it is not your object ID**.
    - Xt & Yt are the co-ordinates of touches detected on the LCD panel and CMM2 is a flag for Colour MaxiMite 2 compatibility.
    - O() is a two-dimensional array that holds the specific properties of each gadget e.g. its type, its place on the screen and is state etc.
    - Ot() is a single dimensioned string array which holds any text associated with a gadget. Not all gadgets have a text property.

- Clear the panel to the default state of black text on a grey background.
- Finally we configure the system for CMM2 if required.

All objects are described to the GUI system using the **GUIObjDef()** function. This handles all types of gadget.

Key Functions and Subs

Functions return interesting values and are signified by a preceding =

GUIObjDef(a,b,c,d,e,f,g,h,i,j,k)

Has 11 arguments, some of which are optional and others which are redundant (meaningless in the context).

Returns -1 if there is no space to store the gadget attributes (increase **Objs** in the preamble). Other values can be ignored.

Argument	Attribute	Definition
a	ID	The numeric name your program wants to use for this gadget. You define this, if you provide the same ID to multiple gadgets, the definition will be over-written. The ID allows you to group your gadgets sensibly. This is not the internal reference for the gadget.
b	Type	The type number of the gadget - see Gadgets and their Attributes below.
c	X	The X & Y co-ordinates of the top left corner of the gadget on screen. These are given in “native” LCD panel co-ordinates i.e. 0,0 is the top left corner of the screen and 219,239 (for example) is the bottom right corner.
d	Y	\\
f	W	The width and height of the gadget in pixels which go on to define the bottom right corner of the gadget. Remember these are relative values to the X & Y – they are not physical co-ordinates themselves.
g	H	\\
h	Text	Any text which should be displayed with the gadget. This might be the word inside a button, the text in a TextBox etc… Not all types of gadgets will use this attribute.
i	Type Specific	Optional. This value is interpreted depending on the type of gadget. Consult the relevant section for the gadget type below
j	Type Specific	Optional. This value is interpreted depending on the type of gadget. Consult the relevant section for the gadget type below
k	Type Specific	Optional. This value is interpreted depending on the type of gadget. Consult the relevant section for the gadget type below

Each of the arguments is mapped generally onto the O(n,1 to 10) elements, with some processing where required.

Examples:

A simple Save button

```
=GUIObjDef(21,1,10,10,80,30,"Save",,,1)
```

Red RadioButton in group 4800

```
=GUIObjDef(40,2,170,100,0,0,"Red",4800,&hff0000)
```

Blue progress bar ranged 0-5000

```
=GUIObjDef(61,8,10,210,300,20,"",0,5000,&h0000ff02)
```

GUIObjPSet(a,b)

Sets the Properties of the gadget (these are values that are useful to the functions of your program).

Has two arguments:

Argument	**Attribute**	**Definition**
a	ID	The numeric name your program wants to use for this gadget. You define this, if you provide the same ID to multiple gadgets, the definition will be over-written. The ID allows you to group your gadgets sensibly. This is not the internal reference for the gadget.
b	status byte	Value to set in the status. This is a bit-significant value. The meaning of each bit is as follows

The Properties of the gadget (these are values that are useful to the running of your code) are stored on O(n,0). This status byte is a bit-field. **Important**: Some the properties require **GUIObjDraw()** to see.

Status bit meanings (when set)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
\\	Reserved	CleanUp	Set	Touched	Visible	Enabled	Reserved	Reserved

Note: All reserved bits should be set to zero

- 2: Enable. Says whether the gadget is active or “ghosted”. Disabled (bit 2=0) gadgets are visible on the screen (if bit 3=1) but do not respond to touches. When touched the touch processor will continue to return -1 as if the gadget were not there.
- 3: Visible. The gadget is displayed. If bit 3=0 the gadget will have a grey box drawn over it to erase it from the screen
- 4: The object has been touched. If the object is not enabled or not visible (bits 2 or 3=0), this bit is not set by the touch processor. This bit must be cleared by your code. Use of this bit can be avoided if you are polling for touched by regular calls to the touch process Sub, i.e. in the main loop of your program. If however you are using interrupts (maybe your code spends long periods away from the main loop), this bit records touches for later use(**Warning**: the order in which touches occurred cannot be determined).
- 5: Records the value of a gadget if applicable. Mainly used for Checkboxes and RadioButtons e.g. the state of a checkbox is: checked=1, not checked=0.
- 6: Indicates the gadget may have been damaged by other gadgets and should be re-drawn. For

drop-down lists, any gadgets displayed below the top line of the list will be over-written when the menu drops over them. Such gadgets will have bit 6 set in their status. The menu close sub will largely take care of this automatically.

Examples:

Our save Button from above is visible and enabled

```
=GUIObjPSet(21,&b00001100)
```

Our red RadioButton is disabled

```
=GUIObjPSet(40,&b00001000)
```

For flipping individual bits, here are two examples using a combination of GUIObjPGet and GUIObjPSet…

… to enable a gadget

```
Result=GUIObjPSet(myObj,GUIObjPGet(myObj) OR &b00000100)
```

… to disable a gadget

```
Result=GUIObjPSet(myObj,GUIObjPGet(myObj) AND &b11111011)
```

GUIObjPGet(ID)

Returns the Properties of the gadget. See above for the meaning of each bit. The single argument is the numeric name ID assigned to the gadget by your program. The meaning of the bits in the returned value are as shown in **GUIObjPSet()** above.

Example:

Has our save button been touched?

```
=GUIObjPGet(21) AND &b00010000
```

A non-zero result says yes, otherwise no. As an aside, there are two approaches for detecting touches which are discussed in the section “Interrupt or not?” below

GUIObjVSet(ID,x)

Set the numeric value of a slider or progress bar. The value will be constrained to within the limits specified for the gadget in the corresponding GUIObjDef()

GUIObjVGet(ID)

Returns the numeric value of a Slider,ProgressBar, CheckBox or RadioButton. For Sliders & ProgressBars the number returned is the actual value within its range. The value will be constrained to within the limits specified for the gadget in the corresponding GUIObjDef()

For CheckBoxes and RadioButtons, the value returned is 0=not selected, <>0=selected.

The Gadgets and their Attributes

We have discussed the functions to get gadgets on the screen, what their properties mean and how to manipulate them. There now follows a detailed discussion of each gadget type.

Buttons

Buttons are the “traditional” command button. Rectangular, bordered areas (it is there even if you don’t show it) of the screen. The entire area is active, i.e. it causes an action if you click anywhere inside it. Usually there is some text which is displayed centrally both horizontally and vertically. It is usual for the border to have a raised appearance and to be animated to a depressed appearance when clicked, returning to its former when released.

GUIObjDef Attributes	
ID	The numeric name your program will use for this object
Type	1
X	Location co-ordinates on screen X,Y
Y	\\
&lt;null&gt;	Argument is ignored
&lt;null&gt;	Argument is ignored
Text	Button text
Background Colour	The colour of the face of the button
Text Colour	The colour of the text of the button. When a button is disabled, this value is over-ridden by the “ghost” colour
Border Type	Usually 1, consult the section on border types for options

Buttons do not have numeric values that can be altered.

GUIObjVSet	
ID	The numeric name your program will use for this object
a	Argument is ignored

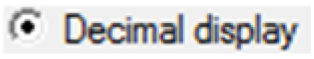
To change the text of a Button, set the Ot(n) value directly

Example:

```
Ot(GUIObjFind(id))="New Text"
```

```
GUIObjDraw id
```

RadioButtons



RadioButtons are an option button. A circular area bounded by a circular border in relief. Once activated, they can only be de-activated by touching another RadioButton in the same group. Its state can be set or unset by program control manipulating the SET bit in the gadget's properties. Usually there is some text which is displayed to the right of the RadioButton. The entire area is active – including the text, i.e. it causes an action if you click anywhere on the button or its label. This makes it easier to activate on smaller displays as precise touch is not so critical. An activated RadioButton has a depressed circular appearance. A de-activated RadioButton has a raised circular appearance. Any change is animated when touched or under program control.

The Height of a RadioButton is fixed at 11 pixels, Width is determined by the length of the text+15 pixels.

GUIObjDef Attributes	
ID	The numeric name your program will use for this object
Type	2
X	Location co-ordinates on screen X,Y
Y	\\
&lt;null&gt;	Argument is ignored
&lt;null&gt;	Argument is ignored
Text	RadioButton text
Group	The group to which this button belongs. Buttons in the same group are affected by the operation of their members e.g. activating a button will deactivate all others in the group
Colour	The colour for the central "dot" of the button as &hRRGGBB
&lt;null&gt;	Argument is ignored

RadioButtons do not have numeric values that can be altered.

GUIObjVSet	
ID	The numeric name your program will use for this object
a	Argument is ignored

Retrieving a RadioButton value. Besides reading the Set bit in the status register of a RadioButton, the state can also be found as follows

=GUIObjVGet	
ID	The numeric name your program uses for this object

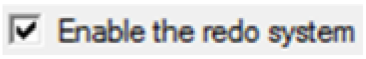
Return: 0=Not selected, <>0=selected.

To change the text of a RadioButton, set the Ot(n) value directly

Example:

```
Ot(GUIObjFind(id))="New Text"
GUIObjDraw(id)
```

CheckBoxes



Checkboxes are a toggling option button. A small area bounded by a square border in relief. Once activated, they can be de-activated by touching again and vice versa. Its state can be set or unset by program control manipulating the SET bit in the gadget's properties. Usually there is some text which is displayed to the right of the Checkbox. The entire area is active – including the text, i.e. it causes an action if you click anywhere on the Checkboxes or its label. This makes it easier to activate on smaller displays as precise touch is not so critical. An activated Checkbox has a check-mark in the small box area to the left of the text. A de-activated Checkbox has a blank box area. Any change is animated when touched or under program control.

The Height of a Checkbox is fixed at 11 pixels, Width is determined by the length of the text+15 pixels.

GUIObjDef Attributes	
ID	The numeric name your program will use for this Gadget
Type	3
X	Location co-ordinates on screen X,Y
Y	\\
&lt;null&gt;	Argument is ignored
&lt;null&gt;	Argument is ignored
Text	Checkbox text
&lt;null&gt;	Argument is ignored
&lt;null&gt;	Argument is ignored
&lt;null&gt;	Argument is ignored

CheckBoxes do not have numeric values that can be altered.

GUIObjVSet	
ID	The numeric name your program will use for this Gadget
a	Argument is ignored
b	Argument is ignored
c	Argument is ignored

Retrieving a CheckBox value. Besides reading the Set bit in the status register of a CheckBox, the state can be found as follows

=GUIObjVGet	
ID	The numeric name your program uses for this Gadget

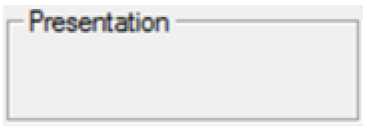
Return: 0=Not selected, <>0=selected.

To change the text of a CheckBox, set the Ot(n) value directly

Example:

```
Ot(GUIObjFind(id))="New Text"
GUIObjDraw id
```

Frames



A simple rectangular box with a text label. Used for grouping related gadgets more for aesthetics than functionality. Beware defining Frames before the gadgets they contain. They can prevent gadgets from being identified when touched in favour of the enclosing frame. Best to places gadgets on the screen then place Frames last of all.

Unlike desktop GUIs, disabling a frame will not automatically disable all items within it.

GUIObjDef Attributes	
ID	The numeric name your program will use for this Gadget
Type	9
X	Location co-ordinates on screen X,Y
Y	\\
W	Dimensions Width and Height
H	\\
Text	Frame text
&lt;null&gt;	Argument is ignored
&lt;null&gt;	Argument is ignored
Border Type	Usually 5, consult the section on border types for options

Frames do not have numeric values that can be altered.

GUIObjVSet	
ID	The numeric name your program will use for this Gadget
a	Argument is ignored
b	Argument is ignored
c	Argument is ignored

To change the text of a Frame, set the Ot(n) value directly

Example:

```
Ot(GUIObjFind(id))="New Text"
```

```
GUIObjDraw id
```

TextBoxes

A rectangular area in which text is written. May have any style of border

GUIObjDef Attributes	
ID	The numeric name your program will use for this Gadget
Type	5
X	Location co-ordinates on screen X,Y
Y	\\
W	Dimensions Width and Height
H	\\
Text	Text value of the text field
Background colour (paper)	Background colour as &hRRGGBB
Foreground colour (pen)	Foreground colour as &hRRGGBB
Border Type and Alignment	Border type and Alignment are stuffed in a single byte. Consult the section on border types for options

TextBoxes do not have numeric values that can be altered.

GUIObjVSet	
ID	The numeric name your program will use for this Gadget
a	Argument is ignored
b	Argument is ignored
c	Argument is ignored

To change the text of a TextBoxes, set the Ot(n) value directly

Example:

```
Ot(GUIObjFind(id))="New Text"
GUIObjDraw id
```

Progress bars

A progress bar is a rectangular area with increasing/decreasing coverage depending on the current set value. The minimum and maximum extremes of the range are settable to any value and the physical size of the bar on the screen will be scaled accordingly. The border of the bar may be set to any frame style although the depressed button appearance is most usual.

Progress bars may be horizontal or vertical. They always progress low to high as left to right or bottom to top respectively

GUIObjDef Attributes	
ID	The numeric name your program will use for this Gadget
Type	8
X	Location co-ordinates on screen X,Y
Y	\\
W	Dimensions Width and Height
H	\\
&lt;null&gt;	Progress bars do not have a text attribute, Set to ""
Bottom of range	Any numeric value &lt; top of range
Top of range	Any numeric value &gt; bottom of range
Colour, Border Type and Alignment	Border type and Alignment are stuffed in a single byte. Consult the section on border types for options

Setting ProgressBar value.

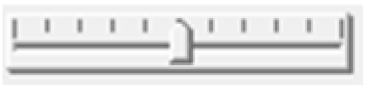
GUIObjVSet	
ID	The numeric name your program uses for this Gadget
a	Current value to set the slider. This will be constrained to the relevant range given in GUIObjDef()

Retrieving the ProgressBar value.

=GUIObjVGet	
ID	The numeric name your program uses for this Gadget

Returns the current value of the ProgressBar.

Sliders



A moving button that can be slid up and down a scale to indicate a value. The minimum and maximum extremes of the range are settable to any value and the physical size of the slider on the screen scales accordingly.

GUIObjDef Attributes	
ID	The numeric name your program will use for this Gadget
Type	4
X	Location co-ordinates on screen X,Y
Y	\\

GUIObjDef Attributes	
W	Width of the slider
&lt;null&gt;	Argument is ignored
&lt;null&gt;	Sliders do not have a text attribute, Set to ”“
Bottom of range	Any numeric value &lt; top of range
Top of range	Any numeric value &gt; bottom of range
Paired Gadget ID	The ID of a linked item to have its text attribute mirror the Slider value. See the discussion below on Object Pairing.

Setting the Slider value. Besides operating the slider using the touch interface, slider values can be set program control.

GUIObjVSet	
ID	The numeric name your program uses for this Gadget
a	Current value to set the slider. This will be constrained to the relevant range given in GUIObjDef()

Retrieving the Slider value.

=GUIObjVGet	
ID	The numeric name your program uses for this Gadget

Returns the current value of the Slider.

Gadget Pairing

When moving a Slider, it is usual to need to see some feedback on the value as it varies. Within the Slider this is not possible for three reasons:

1. While moving the slider your code is not running (ProcessTouch() is) and the value the slider settled at can only be reported once the Slider is released.
2. ProcessTouch() has no idea what you want to do with the values your set and so makes no assumptions.
3. Putting the value in the slider button is impractical because the width of the text could destroy the aesthetics of the slider.

To provide some indication, the ID of another gadget can be set to have its text attribute linked to the value of the Slider. As the slider is moved, ProcessTouch() will update the paired gadget (perhaps a TextBox).

List Static

Similar in appearance to the text box, the Static list has direction arrows at each end and the list can be navigated through with the current selected item displaying in the text area.

GUIObjDef Attributes	
ID	The numeric name your program will use for this Gadget
Type	7
X	Location co-ordinates on screen X,Y
Y	\\
W	Dimensions Width and Height
H	\\
Text	Frame text
Background colour (paper)	Background colour as &hRRGGBB
Foreground colour (pen)	Foreground colour as &hRRGGBB
Border Type and Alignment	Border type and Alignment are stuffed in a single byte. Consult the section on border types for options

---

List Drop-Down

Similar in appearance to the text box, the Drop-down list displays a rectangular area containing lines of text when touched. An item may be selected by touching it. If more items exist than the box can contain, the up and down arrows on the right of the drop-down can be used to navigate through the list. When an item is selected, the drop-down box is erased, the text is set to the selected item and any damaged items (where the drop-down obliterated them) are set for cleanup.

GUIObjDef Attributes	
ID	The numeric name your program will use for this Gadget
Type	7
X	Location co-ordinates on screen X,Y
Y	\\
W	Dimensions Width and Height
H	\\
Text	Frame text
Background colour (paper)	Background colour as &hRRGGBB
Foreground colour (pen)	Foreground colour as &hRRGGBB
Border Type and Alignment	Border type and Alignment are stuffed in a single byte. Consult the section on border types for options

---

Border Types and alignment

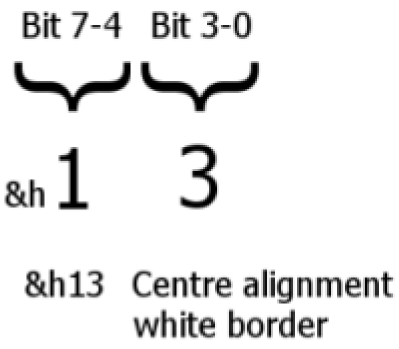
Not all gadgets support a border type but for those that do, the value contains several indicators in a single integer.

Alignment Value! Byte 3! Byte 2! Byte 1	**Byte 0**			
\\	Red for stuffed colour	Green for stuffed colour	Blue for stuffed colour	Border Style and Alignment

The rightmost byte is further interpreted by it's two nibbles thus:

Bit 7-4! Bits 3-0	Description	
x	0	No Border
x	1	Button style raised
x	2	Button style depressed
x	3	White
x	4	Black
x	5	Grey (disabled)
0	x	Align Left/Horizontal
1	x	Align Centre/Vertical
2	x	Align Right

Example:



Due to redundancy, not all border/alignment combinations are applicable to any given gadget

The Code

```
'V0.68

'{mandatory GUI config
    Option Base 0
    Const CGy=&hdddddd,Cgh=&h909090'Pale grey screen background
    Colour 0,cGy:Cls
    Dim Integer Objs=20,P=0,Flags,Xt,Yt,CMM2
    Dim Integer O(Objs,10)'GUI object settings
    Dim String Ot(Objs) Length 63'GUI object text

    'CMM2 Compatibility - and any others if necessary
    Select Case MM.Device$
        Case "Colour Maximite 2"
            mp=MM.info(Option Mouse) ' mouse port
            CMM2=1
            Option Console Serial
            GUI Cursor On 0, 100,100,rgb(red)
            Controller Mouse Open mp
        Case Else
    End Select
'}
```

```
    Backlight 80' 100 is off

MainInit:

    Dim Integer res

'Buttons
    res=GUIObjDef(21,1,10,10,80,30,"Save",&hff0000,&hffffff,1)
    res=GUIObjDef(23,1,10,50,80,30,"Exit",,,1)
'Checkboxes
    res=GUIObjDef(25,3,10,100,,,"Checked")
    res=GUIObjDef(27,3,10,120,,,"Disabled")
    res=GUIObjDef(29,3,10,140,,,"Not Checked")
'Radio boxes (in a frame)
    res=GUIObjDef(40,2,170,100,,,"Red",1,&hff0000)
    'res=GUIObjDef(40,2,170,100,,,"Red",1,0)
    res=GUIObjDef(41,2,170,120,,,"Green",1,&hff00)
    'res=GUIObjDef(41,2,170,120,,,"Green",1,0)
    res=GUIObjDef(42,2,170,140,,,"Blue",1,&hff)
    'res=GUIObjDef(42,2,170,140,,,"Blue",1,0)
    res=GUIObjDef(39,9,165,90,80,65,"Colours",,,&h05)
'text boxes
    res=GUIObjDef(50,5,160,10,150,15,"L Text black
frame",&h00aa00,&hffff00,&h04)
    res=GUIObjDef(51,5,160,35,150,15,"C Text no frame",cGy,0,&h10)
    res=GUIObjDef(52,5,160,60,150,15,"",&hffffff,0,&h22)
'progress bar
    res=GUIObjDef(61,8,10,210,300,20,"",0,500,&h0000ff02)
    res=GUIObjDef(62,8,270,90,40,65,"",0,500,&hff000012)
'Slider
    res=GUIObjDef(71,4,20,180,100,,"",0,500,51)

Main:
'set the properties of the object, visibility, enabled etc...
    res=GUIObjPSet(21,&b00001100)
    res=GUIObjPSet(23,&b00001000)
    res=GUIObjPSet(25,&b00101100)
    res=GUIObjPSet(27,&b00101000)
    res=GUIObjPSet(29,&b00001100)
    res=GUIObjPSet(39,&b00001100)
    res=GUIObjPSet(40,&b00101100)
    res=GUIObjPSet(41,&b00001100)
    res=GUIObjPSet(42,&b00001000)
    res=GUIObjPSet(50,&b00001100)
    res=GUIObjPSet(51,&b00001100)
    res=GUIObjPSet(52,&b00001100)
    res=GUIObjPSet(61,&b00001100)
    res=GUIObjPSet(62,&b00001100)

    res=GUIObjPSet(71,&b00001100)
```

```
'nothing appears on the screen until you draw the object
'this allows multiple updates at once and you control when it happens
    GUIObjDraw 21
    GUIObjDraw 23
    GUIObjDraw 25
    GUIObjDraw 27
    GUIObjDraw 29
    GUIObjDraw 39
    GUIObjDraw 40
    GUIObjDraw 41
    GUIObjDraw 42
    GUIObjDraw 50
    GUIObjDraw 51
    GUIObjDraw 52
    GUIObjDraw 61
    GUIObjDraw 62

    GUIObjVSet 71,250
    GUIObjDraw 71

'a little example code
    q=1000
    If res <>-1 Then
        timer=0
'draw the progress bars for increasing value
        For n=1 To q step 25
                GUIObjVSet 61,n
                GUIObjDraw 61
                GUIObjVSet 62,q-n
                GUIObjDraw 62
        Next
'set the contents of a text box
        Ot(GUIObjFind(52))="R "+Str$(timer/n)+"mS"
        GUIObjDraw 52
    EndIf

'go round in a loop printing (to the console) the ID (numeric name) of
'any touched objects and how long it was touched for
    Do
        Timer=0
        q=ProcessTouch()
        If q<>-1 Then ? "Obj ";q,,"Dwell";Timer;"mS"
        'get the value from the slider and update a text box
        If q=71 Then
            For m=50 to 52
                'Ot(GUIObjFind(m))=Str$(GUIObjVGet(71)):GUIObjDraw m
            Next
        EndIf
        Pause 100
    Loop
    End



'------------------ The GUI pack
'Main touch processor. Regular calls here will get the numeric name of
'screen objects that are touched, return -1 if no object is touched
'animates and manages status etc
    Function ProcessTouch() As Integer
        ProcessTouch=-1
        GetTouch
        If Xt=-1 Then Exit Function
        Local Integer n,m
        For n=0 To P-1
            If Xt>=O(n,3) Then
                If Yt>=O(n,4) Then
                    If Xt<=O(n,5) Then
                        If Yt<=O(n,6) Then
                            If O(n,2) And 8 Then
                                If O(n,2) And 4 Then
                                    ProcessTouch=O(n,0)
                                    O(n,2)=O(n,2) Or &b00010000'touched
                                    Select Case O(n,1)'click animation
                                        Case 1
                                            O(n,9)=2
                                            GUIObjDraw O(n,0)
                                        Case 2
                                            For m=0 To P
                                                If O(n,0)<>O(m,0) Then
                                                    If O(m,1)=2 Then
                                                        If O(m,7)=O(n,7)
Then

                                                            O(m,2)=O(m,2)

And &b11011111

                                                            GUIObjDraw

O(m,0)

                                                        EndIf
                                                    EndIf
                                                EndIf
                                            Next
                                            O(n,2)=O(n,2) Or &h20
                                            GUIObjDraw O(n,0)
                                        Case 3
                                            If O(n,2) And &b00100000 Then
                                                O(n,2)=O(n,2) And &b11011111
                                            Else
                                                O(n,2)=O(n,2) Or &b00100000
                                            EndIf
                                            GUIObjDraw O(n,0)
                                        Case 4'slider
                                            Local w
                                            If CMM2 Then GUI Cursor Hide
                                            Do
                                                'Watchdog in here
                                                GetTouch
                                                If Xt=-1 Then Exit Do
                                                w=O(n,5)-O(n,3)
                                                Xt=Constrain(Xt-O(n,3),0,w)
                                                w=Xt/(w/(O(n,8)-O(n,7)))
                                                GUIObjVSet O(n,0),w
                                                If O(n,9) Then
Ot(GUIObjFind(O(n,9)))=Str$(w):GUIObjDraw O(n,9)
                                                EndIf
                                                GUIObjDraw O(n,0)
                                            Loop
                                            If CMM2 Then GUI Cursor Show
                                        Case 6
                                        Case 7
                                    End Select
                                    WaitNoTouch
                                    Select Case O(n,1)'release animation
                                        Case 1
                                            O(n,9)=1
                                            GUIObjDraw O(n,0)
                                        Case 6
                                        Case 7
                                    End Select
                                    Exit Function
                                EndIf
                            EndIf
                        EndIf
                    EndIf
                EndIf
            EndIf
        Next
    End Function

'waits for the stylus to be lifted off the panel
    Sub WaitNoTouch
        If CMM2 Then
            Do
            'Watchdog here if you want it
            Pause 10
            Loop Until Mouse(L,mp)=0
        Else
            Do
            'Watchdog here if you want it
            Pause 10
            Loop Until Touch(X)=-1
        EndIf
    End Sub

'Sub will draw any object in it's current state
```

```
    Sub GuiObjDraw(id As Integer)
        Local Integer n,m,x,y,z,w,h
        Local Float nn
        Local a$
        n=GUIObjFind(id)
        If n>-1 Then
            GUIObjCleanup n
            x=O(n,3):y=O(n,4)
            w=O(n,5)-x:h=O(n,6)-y
            Select Case O(n,1)
                Case 1'button
                    If O(n,2) And 8 Then'visible
                        Box x,y,w,h,,O(n,7),O(n,7)
                        GUIFrame n,0
                        If O(n,2) And 4 Then'Enabled
                            z=O(n,8)
                        Else'ghosted
                            z=Cgh
                        EndIf
                        Text w/2+x-FW()*Len(Ot(n))/2,h/2+y-
FH()/2,Ot(n),,,,z,O(n,7)
                    Else'erase
                        GUIObjCleanup n,1
                    EndIf
                Case 2'radio
                    If O(n,2) And 8 Then'visible
                        x=x+5:y=y+5
                        If O(n,2) And 4 Then'Enabled
                            If O(n,2) And 32 Then'set
                                Circle x,y,4,,,O(n,8),O(n,8)
                                Colour 0:For nn=2.34 To 5.5 Step 0.2:Pixel
x+6*Sin(nn),y+6*Cos(nn):Next
                                Colour &hffffff:For nn=5.5 To 8.64 Step
0.2:Pixel x+6*Sin(nn),y+6*Cos(nn):Next
                            Else
                                Circle x,y,4,,,CGy,CGy
                                Colour &hffffff:For nn=2.34 To 5.5 Step
0.2:Pixel x+6*Sin(nn),y+6*Cos(nn):Next
                                Colour 0:For nn=5.5 To 8.64 Step 0.2:Pixel
x+6*Sin(nn),y+6*Cos(nn):Next
                            EndIf
                            Colour 0
                        Else
                            Circle x,y,4,,,CGy,CGy
                            Colour Cgh:For nn=0 To 6.4 Step 0.2:Pixel
x+6*Sin(nn),y+6*Cos(nn):Next
                        EndIf
                        Text x+15,y+(h/2)-FH(),Ot(n)
                    Else'erase
                        GUIObjCleanup n,1
                    EndIf
                Case 3'checkbox
                    If O(n,2) And 8 Then'visible
                        Colour 0:Line x,y,x,y+11:Line x,y,x+11,y:Colour
&hffffff:Line x+11,y,x+11,y+11:Line x+11,y+11,x,y+11
                        If O(n,2) And 4 Then'Enabled
                            Colour 0
                        Else'ghosted
                            Colour Cgh
                        EndIf
                        If O(n,2) And 32 Then'checked
                            GUI Bitmap x+2,y+2,&h18386C6CC6020301
                        Else
                            GUI Bitmap x+2,y+2,0
                        EndIf
                        Text x+15,y+(h/2)-(FH()/2),Ot(n)
                    Else'erase
                        GUIObjCleanup n,1
                    EndIf
                Case 4'slider
                    If O(n,2) And 8 Then'visible
                        Box x-5,y,w+11,h+1,,CGY,CGy
                        y=y+(h/2)
                        Colour 0:Line x,y-2,x+w,y-2:Line x,y-2,x,y+2:Colour
&hffffff:Line x,y+2,x+w,y+2:Line x+w,y+2,x+w,y-2:
                        x=x+((w/(O(n,8)-O(n,7)))*O(n,10))
                        y=O(n,4)
                        'pointer
                        If O(n,2) And 4 Then'Enabled
                            Box x-5,y,11,h,,CGY,CGy
                            Colour 0:Line x+5,y+3,x+5,y+h:Line
x+5,y+h,x-5,y+h:Colour Cgh:Line x,y,x+5,y+3:Colour &hffffff:Line
x-5,y+h,x-5,y+3:Line x-5,y+3,x,y
                        Else
                            Box x-5,y,11,h,,CGY,CGy
                            Colour Cgh:Line x+5,y+3,x+5,y+h:Line
x+5,y+h,x-5,y+h:Line x,y,x+5,y+3:Line x-5,y+h,x-5,y+3:Line x-5,y+3,x,y
                        EndIf
                    Else'erase
                        GUIObjCleanup n,1
                    EndIf
                Case 5'textbox
                    If O(n,2) And 8 Then'visible
                        Box x,y,w+1,h+1,,O(n,7),O(n,7)
                        GUIFrame n,1
                        Select Case O(n,9)>>4
                            Case 0'L
                                Text x,y,Ot(n),,,,O(n,8),O(n,7)
                            Case 1'C
                                Text w/2+x-FW()*Len(Ot(n))/2,h/2+y-
FH()/2,Ot(n),,,,O(n,8),O(n,7)
                            Case 2'R
```

```
                                Text O(n,5)-(FW()*Len(Ot(n))),h/2+y-
FH()/2,Ot(n),,,,O(n,8),O(n,7)
                        End Select
                    Else'erase
                        GUIObjCleanup n,1
                    EndIf
                Case 6'list static
                Case 7'list dropdown
                Case 8'progress bar
                    If O(n,2) And 8 Then'visible
                        GUIFrame n,1
                        If O(n,9) And &h10 Then'vert
                            z=(h/(O(n,8)-O(n,7)))*O(n,10)
                            Box x,y,w,h-z,,cGy,cGy
                            Box x,(y+h)-z,w,z,,O(n,9)>>8,O(n,9)>>8
                        Else'hoz
                            z=(w/(O(n,8)-O(n,7)))*O(n,10)
                            Box x+z,y,w-z,h,,cGy,cGy
                            Box x,y,z,h,,O(n,9)>>8,O(n,9)>>8
                        EndIf
                    Else'erase
                        GUIObjCleanup n,1
                    EndIf
                Case 9'frame
                    GUIFrame n,0
                    Colour 0,cGy
                    Text x+5,y-FH()/2,Ot(n)
            End Select
        EndIf
    End Sub

'set values of gadget by its numeric name
    Sub GUIObjVSet(id As Integer,cv As Integer)
        Local Integer n
        n=GUIObjFind(id)
        If n>-1 Then
            Select Case O(n,1)
                Case 4,8
                    O(n,10)=Constrain(cv,O(n,7),O(n,8))
            End Select
        EndIf
    End Sub

    Function GUIObjVGet(id As Integer)
        Local Integer n
        n=GUIObjFind(id)
        If n>-1 Then
            Select Case O(n,1)
                Case 2,3
                    GUIObjVGet=O(n,2) And 32
                Case 4,8
```

```
                    GUIObjVGet=O(n,10)
            End Select
        EndIf
    End Function

'draws any type of border around an object
    Sub GUIFrame(n As Integer,os As Integer)
        Local Integer c1,c2
        Select Case O(n,9) And &h0F
            Case 0'none
                c1=cGy:c2=cGy
            Case 1'button style raised
                c1=&hffffff:c2=0
            Case 2'button style depressed
                c1=0:c2=&hffffff
            Case 3'white
                c1=&hffffff:c2=&hffffff
            Case 4'black
                c1=0:c2=0
            Case 5'grey (disabled)
                c1=Cgh:c2=Cgh
        End Select
        Colour c1:Line O(n,3)-os,O(n,4)-os,O(n,3)-os,O(n,6)+os:Line O(n,3)-
os,O(n,4)-os,O(n,5)+os,O(n,4)-os:Colour c2:Line O(n,5)+os,O(n,4)-
os,O(n,5)+os,O(n,6)+os:Line O(n,3)-os,O(n,6)+os,O(n,5)+os,O(n,6)+os
    End Sub

'removes an object from the screen by its internal reference
    Sub GUIObjCleanup(n,f)
        If f Or (O(n,2) And 64) Then
            Box O(n,3)-1,O(n,4)-1,O(n,5)-O(n,3)+1,O(n,6)-O(n,4)+1,,cGy,cGy
        EndIf
        O(n,2)=O(n,2) And &hBF'clear cleanup flag
    End Sub

'Sets the properties of an object
'Returns -1 if not found
    Function GUIObjPSet(id As Integer,b As Integer) As
Integer'id,enabled,visible,touched,Set,cleanup
        Local Integer n
        n=GUIObjFind(id)
        If n>-1 Then
            O(n,2)=b:Exit Function
        EndIf
        GUIObjPSet=-1'object not found
    End Function

'Returns the properties of an object by its numeric name
'Returns the properties or -1 if not found
    Function GUIObjPGet(id As Integer) As Integer
        Local Integer n
        n=GUIObjFind(id)
        If n>-1 Then
            GUIObjPGet=O(n,2):Exit Function
        EndIf
        GUIObjPGet=-1'object not found
    End Function

'Finds the internal reference for an object from its numeric name
'Returns the object number or -1 if not found
    Function GUIObjFind(id)
        Local Integer n
        For n=0 To P-1
            If O(n,0)=id Then GUIObjFind=n:Exit Function'object exists
        Next
        GUIObjFind=-1
    End Function

'force a value within boundaries
    Function Constrain(v As Integer,l As Integer,u As Integer) As Integer
        Constrain=Min(Max(v,l),u)
    End Function

'defines an object but doesn't draw anything. returns 0 if all OK.
'res=GUIObjDef(21,1,10,10,80,30,"Save",&hffggbb,,1)
    Function GUIObjDef(id As Integer,tp As Integer,x As Integer,y As
Integer,w As Integer,h As Integer,t$,g As Integer,c As Integer,d As Integer)
As Integer
        Local Integer n
        If P>Objs Then GUIObjDef=1:Exit Function'too many objects
        n=GUIObjFind(id)
        If n=-1 Then n=P:P=P+1'define or redefine. P always points to the
next free "slot"
        O(n,0)=id
        O(n,1)=tp
        O(n,2)=0
        O(n,3)=x
        O(n,4)=y
        O(n,7)=g
        O(n,8)=c
        O(n,9)=d
        O(n,10)=g
        Ot(n)=t$
        Select Case tp
            Case 1,5,8,9'button,textbox,progress bar,frame
                O(n,5)=x+w
                O(n,6)=y+h
                If O(n,7)+O(n,8)=0 Then O(n,7)=CGy
            Case 2,3'Radio,checkbox
                O(n,5)=x+15+FW()*Len(t$)
                O(n,6)=y+FH()
            Case 4'Slider
```

```
                O(n,4)=y-4
                O(n,5)=x+w
                O(n,6)=y+14
            Case 6'List Static
            Case 7'List DropDown
        End Select
    End Function

'compatibility functions
    Function FH() As Integer
        If CMM2 Then
            FH=MM.INFO(FONTHEIGHT)
        Else
            FH=MM.FONTHEIGHT
        EndIf
    End Function

    Function FW() As Integer
        If CMM2 Then
            FW=MM.INFO(FONTWIDTH)
        Else
            FW=MM.FONTWIDTH
        EndIf
    End Function

    Sub GetTouch
        If CMM2 Then
            GUI CURSOR MOUSE(x,0),MOUSE(y,0)'here or main loop
            Xt=-1
            If Mouse(R,mp) Then Save Image "GUI.bmp",0,0,320,240
            If Mouse(L,mp) Then
                Xt=mouse(X,mp):Yt=mouse(Y,mp)
            EndIf
        Else
            Xt=Touch(X):Yt=Touch(Y)
        EndIf
    End Sub
```