UTFT_Buttons

Add-on Library for UTFT: Buttons

Manual

Introduction:

This library is an add-on to UTFT and will not work on its own.
This add-on library also requires the URTouch library.

This library adds simple but easy to use buttons to extend the use of the UTFT and URTouch libraries.

You can always find the latest version of the library at **http://www.RinkyDinkElectronics.com/**

For version information, please refer to **version.txt**.

IMPORTANT:

The library defaults to a maximum of 20 simultaneous buttons.

This number can be adjusted according to your needs by changing the number on the line:

```
#define MAX_BUTTONS 20
```

In the **UTFT_Buttons.h** file.

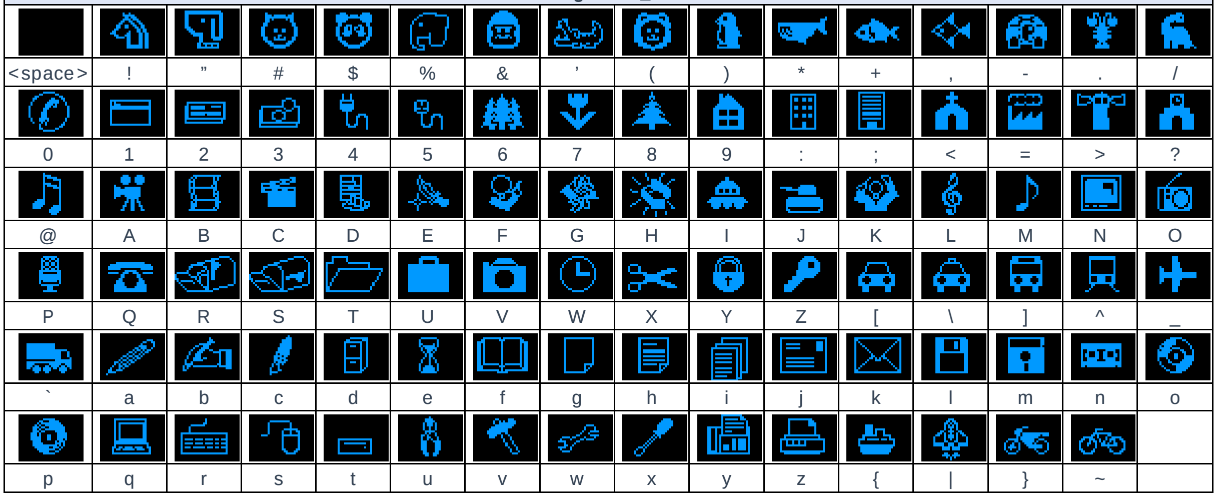
You should note that every possible button will reserve a small amount of RAM, 13-15 bytes depending on what development board you are using, whether it is used or not so you should not increase the number beyond what you actually need.

DEFINED LITERALS:

Status flags		
BUTTON_DISABLED:	0x0001	
BUTTON_SYMBOL:	0x0002	
BUTTON_SYMBOL_REP_3X:	0x0004	
BUTTON_BITMAP:	0x0008	(Should not be used manually)
BUTTON_NO_BORDER:	0x0010	(Only valid for bitmap buttons)
BUTTON_UNUSED:	0x8000	(Should not be used manually)

INCLUDED FONTS:

Dingbats1_XL



<space>	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

FUNCTIONS:

UTFT_Buttons(UTFT, URTouch);

The main class constructor.

Parameters:
```
UTFT   : A reference to an already created UTFT object
URTouch: A reference to an already created URTouch object
```

Usage:
```
UTFT_Buttons myButtons(&myGLCD, &myTouch); // Start an instance of the UTFT_Buttons class
```

Notes: Remember the ‘&’ in front of the object names

addButton(x, y, width, height, label[, flags]);

Add a new text or symbol button.

Parameters:
```
X     : x-coordinate for the upper left corner of the button
y     : y-coordinate for the upper left corner of the button
width : width of the button in pixels
height: height of the button in pixels
label : button text or character for symbol
flags : <optional>
        Can use any combination of BUTTON_DISABLED, BUTTON_SYMBOL and BUTTON_SYMBOL_REP_3X.
        Use | to combine. Default is <none>.
```

Returns: (INT) buttonID, -1 if no button could be added

Usage:
```
int but1 = myButtons.addButton( 10,  20, 300,  30, "Button 1"); // add a new button “Button 1”
```

Notes: Buttons will not be drawn on the screen until drawButton() or drawButtons() is called.

addButton(x, y, width, height, data[, flags]);

Add a new bitmap button.

Parameters:
```
X     : x-coordinate for the upper left corner of the button
y     : y-coordinate for the upper left corner of the button
width : width of the bitmap in pixels
height: height of the bitmap in pixels
data  : array containing the bitmap-data
flags : <optional>
        Can use any combination of BUTTON_DISABLED or BUTTON_NO_BORDER.
        Use | to combine. Default is <none>.
```

Returns: (INT) buttonID, -1 if no button could be added

Usage:
```
int but1 = myButtons.addButton( 10,  20, 300,  30, bitmap); // add a new bitmap button
```

Notes: Buttons will not be drawn on the screen until drawButton() or drawButtons() is called.
You can use the online-tool “ImageConverter 565” or “ImageConverter565.exe” supplied with UTFT to convert pictures into compatible arrays. The online-tool can be found on my website.

drawButtons();

Draw all currently added buttons on the screen.

Parameters: None

Usage:
```
myButtons.drawButtons(); // Draw all buttons
```

drawButton(buttonID);

Draw a single button on the screen.

Parameters:
```
buttonID: ID of the button to draw
```

Usage:
```
myButtons.drawButton(but1); // Draw button with buttonID but1
```

enableButton(buttonID[, redraw]);

Set button state to enabled/clickable.

Parameters:
```
buttonID: ID of the button to enable
redraw  : <optional>
          true : redraw button immediately
          false: do not redraw button yet (Default)
```

Usage:
```
myButtons.enableButton(but1, true); // Enable button with buttonID but1 and redraw it
```

disableButton(buttonID[, redraw]);

Set button state to disabled/unclickable.

Parameters:
```
buttonID: ID of the button to disable
redraw  : <optional>
          true : redraw button immediately
          false: do not redraw button yet (Default)
```

Usage:
```
myButtons.disableButton(but1); // Disable button with buttonID but1 but do not redraw it
```

buttonEnabled(buttonID);

Check the enabled/disabled status of a button.

Parameters:	`buttonID: ID of the button to disable`
Returns:	`(BOOLEAN)` **true** `if button is enabled, otherwise` **false**
Usage:	`boolean state = myButtons.buttonEnabled(but1); // Check if the button with ButtonID but1 is enabled`

relabelButton(buttonID, label[, redraw]);

Relabel a button.

Parameters:
```
buttonID: ID of the button to enable
label   : new button text or character for symbol
redraw  : <optional>
          true : redraw button immediately
          false: do not redraw button yet (Default)
```

Usage:
```
myButtons.relabelButton(but1, “New Label”); // Relabel button with buttonID but1 but do not redraw
```

deleteButton(buttonID);

Delete a button.

Parameters:	`buttonID: ID of the button to delete`
Usage:	`myButtons.deleteButton(but1); // Delete button with buttonID but1`
Notes:	Already drawn buttons will not be deleted from the screen, but they will no longer be detected by calling checkButtons()

deleteAllButtons();

Delete all current buttons.

Parameters:	None
Usage:	`myButtons.deleteAllButtons(); // Delete all buttons`
Notes:	Already drawn buttons will not be deleted from the screen, but they will no longer be detected by calling checkButtons()

checkButtons();

Check if any button is being pressed.

Parameters:	None
Returns:	(INT) buttonID of pressed button, -1 if no button is pressed
Usage:	`int pressed = myButtons.checkButtons(); // Check if any buttons are pressed`

setTextFont(fontname);

Select which font to use for button labels.

Parameters:	fontname: Name of the array containing the font you wish to use
Usage:	`myButtons.setTextFont(BigFont); // Select the font called BigFont`
Notes:	You must declare the font-array as an external or include it in your sketch.

setSymbolFont(fontname);

Select which font to use for button symbols.

Parameters:	fontname: Name of the array containing the font you wish to use
Usage:	`myButtons.setSymbolFont(Dingbats1_XL); // Select the font called Dingbats1_XL`
Notes:	You must declare the font-array as an external or include it in your sketch.

setButtonColors(text, inactive, border, highlight, background);

Set the colors used to draw the buttons.

Parameters:
```
text      : RGB565-encoded color to use for button text and symbols
inactive  : RGB565-encoded color to use for button text and symbols on disabled buttons
border    : RGB565-encoded color to use for button borders
highlight : RGB565-encoded color to use for button borders when selected
background: RGB565-encoded color to use for button background
```

Usage:
```
myButton.setButtonColors(VGA_WHITE, VGA_GRAY, VGA_WHITE, VGA_RED, VGA_BLUE); // Set default colors
```