More Button Making – from Basic to Advanced

Some Button code has already been topics on the LFSLapper section of the LFS Forum. See –

Basic Button code
https://www.lfs.net/forum/thread/89097-Buttons---Learn-how-to-use-them---Basics-%28Long-post%29

Advanced Button Making - Escape Codes and symbol Glyphs
https://www.lfs.net/forum/thread/89157-Buttons---Learn-how-to-use-them---Advanced-%28Symbols-Glyphs%29

In making buttons, you will normally need to use the following code for colour of button backgrounds and for text colours:

Standard codes that can be used on button format:	**To create coloured text, use the following codes in front of the text you want to colour:**
0 - transparent button 16 - light button 32 - dark button 64 - align text to left 128 - align text to right ISB_LIGHT - light button (in place of 16) ISB_DARK – dark button (in place of 32) ISB_LEFT - align text to left on transparent button ISB_RIGHT - align text to right on transparent button ISB_LIGHT|ISB_LEFT – align text left on light grey button	^0 - black ^1 - red ^2 - green ^3 - yellow ^4 - blue ^5 - violet ^6 - cyan ^7 - white ^8 - no colour (light grey)

These are documented at the top of the LFSLapper.lpr code included in each version of LFSLapper.

However, some code is undocumented (as at Sept 2021, V7.080) and other code may be included within the code but may be difficult to understand.

And some of the known code can be used to make buttons that, even when seen, you might not realise are buttons.

Two screenshots of buttons taken 1 second apart – flashing buttons *off* on left, and reappearing on right

Code attached if you want to see all this working for yourself.

Note: *As you will see in the attached code, and in any examples I give here, a # symbol is used where I've wanted to show comments. This is used within the code to tell lapper to disregard anything* **on this line** *from this symbol on. In other code, you may have seen // as a way to show comments*

Some of the early examples I'm going to show are basic, so if you have already made changes to your lapper scripts, should be well known.

As a reminder, in LFS, a button is really a label until you add a sub (sub-routine). Adding a sub then makes the label clickable, so it's now a button that when clicked will then do specified thing(s). In lapper, it's always known, and coded as a button, regardless of how it is used.

The top 2 buttons in the image show the difference. You can try click the top button, but nothing will happen. It's just for showing a message. However, you can click the button underneath, and this will trigger a sub to be invoked.

As an example of the difference in code;

```
openPrivButton ("close_button",10,20,50,5,5,10,16,"[Close]"); #This is a button that will show text, but will be 'inert'

openPrivButton ("close_button1",10,25,50,5,5,10,16,"[Close]",DoSomething); #This is a button that will show text, and will follow instructions if you have a sub called 'DoSomething' – in this case, sub could be

Sub DoSomething ($KeyFlags,$id)
    ClosePrivButton ("close_button1"); # This will close any button(s) named 'close_button1'
EndSub
```

Normally when you make a button, you would have something like

```
openPrivButton ("test_button",10,40,50,5,5,10,0,"[This is a test]");
```

This would give you a button with the following parameters;

This is a ***basic*** private button that only displays a message. Only the driver will see this. As distinct from a global button that everyone will see at same time.

As a reminder;
1st part of code within brackets, and in quotes, is the button name (in this case "my_test_button")
Placement 10 (out of 200) going right from left of screen
Placement 40 (out of 200) down from top of screen. *As LFS already uses some screen real estate top and bottom to display times and options, best to be aware of where you are placing your buttons, and especially not use top end or bottom end of this 200, especially in the corners.*
Button width 50 (out of 200) *Remember, button placement + button width have to be maximum 200.*
Text height (in this case, 5)
If text on multiple lines, then line spacing (in this case, 5)
Number in seconds [in this case, 10. Use -1 when you don't want button to close, OR you use a Sub to close the button, eg ClosePrivButton ("test_button");]
Colour and placement of button (0 = transparent, 16 = light, etc)
Last part of button, within quotes, is the text that will appear on screen - in this case it will be [This is a test]

When it comes to (documented or not) button backgrounds, and text, you can have, or make;

- As standard, transparent, light grey, or dark grey buttons
- Combine codes to make new code to align text on coloured buttons, e.g. to make a light coloured button with text aligned to the left, combine the existing codes (16+64) to one new code 80
- For standard buttons, you can replace numbers for the colour and justification with ISB_ code, where ISB_LIGHT - light button (in place of 16), and ISB_LIGHT|ISB_LEFT (in place of 80) to align text left on light grey button
- Coloured buttons – Square, oblong or circular transparent buttons with oversized coloured text lines or symbols (not normal, but I've used this for making flags, and things, but sometimes difficult to overlay text on top)
- As standard, 9 colours of text. One is specified as 'no colour', which you might assume is clear, but it's light grey
- Justification of text. Unless specified Left or Right, it will be centred
- Text on more than single line. No need to try match buttons one above other, just use %nl% where you want a (new line) line break. Button will increase depth to incorporate all lines of text
- Blinking / flashing text and button. Both text and button flash together. If you don't want to see a flashing button, only flashing text, you can use coloured text on transparent button, overlaid onto coloured background button. Both text and button blink on/off together – per second, if no time specified.
- Language translation [e.g. (langEngine("%{message}%"));] You will need a language section, placed at end of your script
- Timer text - openPrivButton("countdown_button",95,137,10,8,1,**10**,16,"^1%cpt%"); # Countdown from **10** in seconds
- Combined colour text and button colour without specifying either button or text colour as you would in normal button code (some of the text colours only available with this combined code) but a combined number in colour button place, and no text colour specified in the button text.
- Combining text to make single string of text, cutting text string down in length and changing Upper to Lower case, and vice versa
- Using text in an array

Using standard code:

In the image (where TB = Test Button), and attached example code, you will see that some of the buttons are transparent, and some are either light grey, or dark grey.

TB1, TB4 and TB7 show uncoloured (light grey) text, centred on a transparent, a light grey and a dark grey button.

TB2, TB5 and TB8 show uncoloured (light grey) text, aligned left on a transparent, a light grey and a dark grey button.

TB3, TB6 and TB9 show uncoloured (light grey) text, aligned right on a transparent, a light grey and a dark grey button.

TB10 to TB18 show all the standard text colours centred on a light grey button.

TB22 shows standard dark grey button, but with various colours of text

openPrivButton ("test_button22",5,130,100,5,5,-1,32,"^8TB22 - ^1Assorted ^2colours ^3of ^4text ^5centred ^6on ^7dark ^8grey ^0button"); *# ^ marks in front of number tell lapper what colour to make text after this mark*

TB19, TB20 and TB21 show centred red text on transparent, light grey and dark grey buttons, but the buttons blink off and on every second.

openPrivButton("test_button 19",5,115,100,5,5,-1,**8**,"^1TB 19 - (8) This is red text centred on transparent button - blinks every second");

Whereas, you would normally use, 0, 16 or 32 to specify colour of button (transparent, light grey or dark grey) if you add 8 to this code, the button will blink. If you want to justify text left or right, then you have to add more code in.

Blinking buttons:

Transparent button, centred text ..... 0 + 8 = 8
Transparent button, text left..... 0 + 8 + 64 = 72
Transparent button, text right ..... 0 + 8 + 128 = 136
Light grey button, centred text ..... 16 + 8 = 24
Light grey button, text left ..... 16 + 8 + 64 = 88
Light grey button, text right ..... 16 + 8 + 128 = 152
Dark grey button, centred text ..... 32 + 8 = 40
Dark grey button, text left..... 32 + 8 + 64 = 104
Dark grey button, text right..... 32 + 8 +128 = 168

TB23 shows a button that has text on 3 lines

openPrivButton("test_button 23",5,140,20,5,5,-1,16,"^0TB 23 - This is**%nl%**^1text that**%nl%**^2is on 3 lines");

I've highlighted **%nl%** to show where the line breaks occur.

Note: if you don't specify text colour of new line of text, then text will be light grey.

TB24 (not shown on button, but is grey square button with red number to right of TB23 button with 3 lines) is a countdown button.

openPrivButton("test_button 24",40,140,10,8,4,10,16,"^1%cpt%"); *# Countdown from 10*

TB25 – "Greetings" button. The text on this button can be translated.

openPrivButton("test_button 25",55,140,50,5,5,-1,16,langEngine("%{Greeting}%")); *# See translation section at bottom (translation depends on player settings)*

You will need at least one Translation section otherwise you will get errors. This section is normally placed at bottom of any scripts.

In my code, because I use English language, and my LFS is set for English, everything that is preceded by "langEngine" has text in English. In some cases, I supply some translations for some of the text. Some of my 'English' text may be symbols. Where there is no foreign (to me) language translations, lapper will use the text in the English translation section.
For the "Greetings" text, I have

```
Lang "EN" # English
    Greeting = "^3Greetings";
    red_flag = "^1|";
    black_flag = "^0|";
    yellow_flag = "^3|";
EndLang

Lang "FR" # French
    Greeting = "^3Les salutations";
EndLang
```

Note: You have to specify the language and you need to put EndLang at end of each language section, same as you would put EndSub at the end of a sub routine.

Using Vars in place of numbers

In my first example button;

```
openPrivButton ("test_button",10,40,50,5,5,10,0,"[This is a test]");
```

you will note that the first 2 numbers relate to placement, as in

Placement 10 (out of 200) going right from left of screen
Placement 40 (out of 200) down from top of screen.

However, you can set various Vars for button placement, sizes and times. Vars start with dollar sign ($). This is especially useful if you have a lot of buttons in a table, so you can easily change placement, etc, of button by only changing one Var rather than every button.

For example;

```
$HFR = 110;          # HFR - How Far Right from Left edge of main content window (out of 200)
$HFD = 15;           # HFD - How Far Down from Top edge of main content window (out of 200)
$BRW = 38;           # BRW - Button Row Width (out of 200 max)
$BRHS = 5;           # BRHS - Button Row Height - short
$BRHL = 44;          # BRH - Button Row Height - long
$TBS = 5;            # TBS - This button spacing
$TBT = -1;           # TBT - This button Time (in seconds - where -1 means don't close)
```

This on a button, might look like;

```
openPrivButton ("2ndtest_buttons0",$HFR,$HFD,$BRW,$BRHS,$TBS,$TBT,0,"0 - combined text and button colour");
```

Where you have a group of buttons together, and you want a button to the side of, and/or underneath an existing button, then you can add (or take away) a number;

```
openPrivButton ("2ndtest_buttons1",$HFR,$HFD+5,$BRW,$BRHS,$TBS,$TBT,1,"1 - combined text and button colour");
```

Because I've added +5 to $HFD, then this button will be placed 5 below the first button, which is 5 high. Text is only single line, so the button spacing is ignored.

You can use your own description for the Vars, so you could have;

```
$1 = 10          or      $buttons_distance_from_left = 10

# 1 - Top Origin Left edge of main content window (1 right out of 200) from left edge where
# $buttons_distance_from_left = 10 -  buttons_distance_from_left_edge - Top Origin Left edge of main content window
```

Most times you'll want to use shorter var name, else your button code will be almost unmanageable.

Also, most buttons tend to be something like;

```
$bOrigL = 1        # $bOrigL - Top Origin Left edge of main content window
```

If you're making a script, you can set Global button Vars, so that every button that uses these vars, will all be connected, so a change to a value in a var will result in same change in buttons. Or you can have your vars just for use in a specific sub, etc.

Best practise if you are making your own button vars to have an original name for them, as you don't want to have lapper placing, resizing buttons based on vars you may have elsewhere in a script with same name.

Non standard code

All the buttons above the circular and Dutch flag, are not documented (and probably not many will know) and are made by combing the button and text colours together where you would put button colour code, and not putting text colour in front of the text that you want to display.

Buttons 1, 9, 17, 25, 33 and 45, are all Yellow text on different coloured buttons, some of which blink. But in none of these buttons have I put the colour number for Yellow in the button text field. The number 9 is only to show me number I used to get that particular button style.

Button 9;

```
openPrivButton ("2ndtest_buttons9",$HFR+40,$HFD+5,$BRW,$BRHS,$TBS,$TBT,9,"9 - combined text and button colour");
```

The 9 in the code equates to blinking clear button with yellow text, whereas 17 would be yellow text on light grey button.

When I first tried the numbers (I have 1 to 43), I tried making few numbers higher than 43, but these didn't work. Maybe if you go higher still you'll get different results. Especially as I was expecting 4 more buttons (to match 36 to 39, but blinking).

Buttons 0 to 7 are coloured text on transparent buttons. However, for this group, I placed a much larger light grey button to use as a backdrop.

Backdrop is;

```
openPrivButton ("2ndtest_buttonback",$HFR-1,$HFD-2,$BRW+2,$BRHL,$TBS,$TBT,16,"");
```

Where $BRHL = 44; that is, height of button, and 16 at end is colour of background, and there is no text to display.

Note: Most people use one colour of back (either light grey, or dark grey). However, there's nothing to stop you to put slightly smaller back on top so that you have a light or dark border around your smaller inserted back. Also, if you want slightly different colour of grey, you can try put a light button on top of light; light on top of dark; dark on top of dark; dark on top of light.

Flags

As you can see on the screen shot, there are 3 flags.

These are made from transparent buttons that have coloured symbols on them (in place of text).

For instance, part of the Dutch flag code is;

```
openPrivButton( "test_dutch_flagpart1",145,90,40,90,8,25,64,"^1-%nl%^7-%nl%^4-");
```

This code has text of horizontal lines on 3 lines, each with it's own colour.

What makes this work is that the size of the text on the button is 90, which is huge considering most button text is usually 4, 5 or 6 high.

In the Belgian flag, I've done something similar, in that I used a | symbol to give me a vertical line.

Because my original flag script had a huge number of flags in it, I use the symbol in my language section, so it could be reused wherever it was needed.

If you want to see more flags, I made a video that you can find on YouTube –
https://www.youtube.com/watch?v=UCm36TlU6uY

Everything on screen, excluding the car, track and LFS timer in top right corner, are buttons; Yisc[NL]'s pitboard, flags, messages, speedtrap, etc.

Other Shapes

You will see that there is a circular button … 'Yoda says'.

This is one that I use for my circular drift scorer. When I first made this, I actually used it with countdown numbers.

Like the flags, it's just symbols that have been blown up to a huge size.

If you want to see more symbols used to make road traffic signs, you can watch video I made –
https://www.youtube.com/watch?v=WjsdywOmz7E

There are a few other things that can be used in making buttons, one of which is in the use of arrays. Sorry, but this is beyond my expertise, although Yisc[NL] made something for a Tops table I was coding. Can be found at;
https://www.lfs.net/forum/post/1967377#post1967377

It's also possible to make a button that adds text together into a single string together using a full stop in the code, as in;

(extract from Sub PstInfo($userName) within the LFSLapper.lpr script)

```
openPrivButton( "PSNickName",75,59,50,5,5,-1,ISB_DARK, $currPly["NickName"] . " ^7( " . $currPly["PSCountry"] . " )" );
```

This will place your name on a button, open a bracket, then place the name of your country, close bracket, and put that next to your name.

Difference between a message and a button is that a message appears in top left of screen (like using the T command within LFS to send messages), and a button can have more variables (such as time it's visible) and can be placed in the location of your choice.

Taken all together, this whole guide may seem quite daunting, but as I've attached all the code for my examples, and because I've tried to break it down into sections, should make things slightly easier to manage. At least, that's my hope.

Anyway, hope the guide helps, and maybe even gives you some ideas on what you can accomplish, without knowing any very complicated code.