* Programming by: Robert Townsley (MegaHurts)
6/2013
* Concept "***Button-as-Label***" method of displaying changing data.
* Program description:
First off, ***Hello*** to all **V-TFT** users that are reading this. I hope to show you something you may not be aware of that can be done in **V-TFT** that may be of great help to you with displaying Alpha-Numeric data on a **mmB TFT** display.

I have discovered that we (*Users*), can use the **Button** components (both square and rounded types), to display changing text and numbers that is easier, requires less "***User Code***" programming, automatically erases old displayed data when new data is "drawn" and gives you, the user, ***Left/Center/Right*** justification of the data displayed also, and options to have data shown in a outline box, color filled box or any combination of a **Button** objects properties settings!!!!!!

WOW!! you say. **HOW?** you ask. It is all explained in this demo **V-TFT** project. I have made multiple examples showing the "***Standard Label***" object method **Mikroe** has implemented in **V-TFT** and my method using **Button** objects doing the same thing for you to see and compare the code usage and a working application you can load to a **mmB** for **PIC32** to see the differences of how each method actually works when used.

I realized that a **Button** component has to draw everything within it's boarders when it is drawn or redrawn and that it is perfect for displaying text and numbers that changes rapidly or not, but when you want the display to be perfect, without any residual pixels left over from the previous data that was being displayed.

The **Button** component has the advantage of having the "*Caption*" property and 3 alignment settings which the **Label** component does not have - *Left*,*Right* and *Center* justification!

With a **Buttons** properties, you can create new and never done displays and spice it up however you like. You can have ***gradient*** colors for a background now also!! (*see the demo running or load the project into **V-TFT** to see example of this*.)

Using a solid color, ensures old data gets erased **WHILE** new data gets drawn. ***The only property that does not work for using a Button as a label is the "Transparent" property.*** With it set to **TRUE**, old data will *not* get erased completely.
Thankfully, there is a "Max Length" property that determines the number of characters the *Caption* property can be loaded with just like the **Label** component has, so learning how to use it (a *Button-as-Label*), this way is very easy to change to with the programming. The hard part may be deciding when and where to go with this method. Normal **Labels** still have a usage as they exist currently. Use them for any alpha-numeric data that is *static* and will not change during your applications running like before. The *Button-as-Label* does have a drawback though, in that they take up and effect a little more screen area than a **Label** does. Remember, they will draw everything within its boarders, so other objects will have to be placed so as to not be affected when the **Button** is drawn or redrawn.

S C R E E N 1 D E M O O P E R A T I O N

This project has two (2) screens that demonstrate both methods to display changing screen data.
Screen1 demo displays rapidly changing numbers using a counter that uses a "*word*"
type variable, so the count is from zero (**0**), to **65535**, then starts over at **0**.
There are only two (2) controls for this demo; the **Start/Stop** Button and a custom
Button control I call a "Cycle-value control button". This control is click based
and will cycle change the value shown in the little box (a *Button-as-label* display also),
from Ten (**10**), (initial value and fastest speed setting), to One (**1**), (Slowest speed setting),
and back to Ten (**10**), if pressed again and repeats the cycle changes for every press.

At the top of the screen, there are two indicators that are touch controls also.
The red highlighted one indicates which screen you are currently on, the other
one that looks ghosted or grayed out is a touch control to switch to the other
demo screen. This is true of both demo screens. Press the grayed-out box to
change to that demo screen. There is also a pointer to show which is to be pressed.

S C R E E N 2 D E M O O P E R A T I O N

This demo screen (Screen2), is designed to show you changing text messages
using the **V-TFT Label** method in the top boxed area and the same text displayed
using a "**Button**" Objects "Caption" property as a label in the bottom boxed area.
In the center of the screen, you will see six (**6**), buttons. Pressing any of these
will change the text message displayed in the top and bottom box areas.
Press each all you want to see what happens. Be sure to check the code section
for this screen to see the big difference in how it was programmed!

You will find detailed program code with comments in the **V-TFT** project file:
"*Object_Labels_Example_events_code.mbas*"
which is included with this demo project.

You can examine ,without fear of damaging, this demo in **V-TFT** to see all of the
screens objects properties and placement for seeing exactly how I made the demo.
(I made sure to program this so it is **V-TFT** compliant and friendly.)

You may also make changes to this demo in **V-TFT** and/or *compiler* as needed.
Other development Language Users can change the project's compiler setting to
see the **V-TFT** generated code for your compiler.

**(Do any changes with a copy of this project in case)

Changing the *compiler* will only change the code **V-TFT** generates, not the code
I added to all "*User Code*" areas, so you **C** and **PASCAL** users will have to convert
manually all of the Code I wrote, sorry. At least **BASIC** is easy to read and
similar to **PASCAL**. Conversion should not be too hard for those needing to do so.

No conversion is required in order to run this demo on a **mmB** for **PIC32** device.
Just use the ***Hex*** file I compiled from this **mBASIC** code to load into your **mmB**.

You can load the project into **mBASIC** for **PIC32** to fully examine all project code.

You can copy/paste anything to your own project if needed also.

I encourage you to explore and examine and try and test anything here.

I made this for you users to see and use and gain from my trials and experimenting.

Enjoy and I hope you find my idea useful also, ***Robert***. (***MegaHurts***) **B^)**