**Modifications to BigTreeTech TFT35 V3.0 Source Code**

*C:\Users\tony\Documents\3D Printers\Marlin Firmware\BIGTREETECH-TouchScreenFirmware-master\Copy to SD Card root directory to update* - Unified Menu Material theme\config.ini

36,2: #//TG 11/1/20 changed from 5 to 6 for 250000 baud

151,2: #//TG 11/1/20 changed this

185,2: #//TG 11/1/20 changed BED was 150

202,2: #//TG 11/1/20 changed these

272,2: #//TG 11/1/20 modified these (6 maximum)

313,3: ##//TG 11/1/20 enabled this

318,2: #//TG 11/1/20 changed this was 0, SWX1 sensor is non-inverting

323,2: #//TG increased due to false triggers (was 100)

439,2: #//TG 11/1/20 enabled this

444,2: #//TG 11/1/20 enabled this

*C:\Users\tony\Documents\3D Printers\Marlin Firmware\BIGTREETECH-TouchScreenFirmware-master\TFT\src\User\API\menu.c*

5,1: //TG this code draws, handles, processes, and responds to menus

*C:\Users\tony\Documents\3D Printers\Marlin Firmware\BIGTREETECH-TouchScreenFirmware-master\TFT\src\User\API\parseACK.c*

550,33: // Parse and store ABL type //TG Is this broken??

*C:\Users\tony\Documents\3D Printers\Marlin Firmware\BIGTREETECH-TouchScreenFirmware-master\TFT\src\User\API\Settings.h*

30,40: #define ICON\_FLASH\_SIGN 20201031 //TG (YYYYMMDD) change if any icon(s) is added or removed

*C:\Users\tony\Documents\3D Printers\Marlin Firmware\BIGTREETECH-TouchScreenFirmware-master\TFT\src\User\Menu\FeatureSettings.c*

4,1: //TG Layout for the page

55,1: //TG here is where you would expand the list if needed, but also need to expand

111,5: //TG this list item has only a LABEL value, no ON/OFF TOGGLE (red dot)

361,29: #ifdef LED\_COLOR\_PIN //TG only on the E3 version

372,35: #ifdef LCD\_LED\_PWM\_CHANNEL //TG LCD brightness control

440,35: void menuFeatureSettings(void) //TG handles the Feature List Menu up,dn, and back buttons

449,51: key\_num = menuKeyGetValue(); //TG get a pressed key?

452,51: case KEY\_ICON\_5: //TG up arrow, load prev list

462,51: case KEY\_ICON\_6: //TG dn arrow, load next list

472,51: case KEY\_ICON\_7: //TG back arrow, load prev infoMenu

C:\Users\tony\Documents\3D Printers\Marlin Firmware\BIGTREETECH-TouchScreenFirmware-master\TFT\src\User\Menu\SettingsMenu.c

120,59: {ICON\_BACKGROUND, LABEL\_BACKGROUND}, //TG available button?

121,59: {ICON\_BACKGROUND, LABEL\_BACKGROUND}, //TG available button?

**Creating/Adding custom icon buttons to menus**

To create a new icon for a button to be used in menus, first choose a name for the icon **CUSTOM\_1**, then create a .bmp file for the button and name it **CUSTOM\_1.bmp**

Now, to be able to use it in menus do the following:

First, we must add a few new keywords into these 4 files so the rest of the code knows about our new icon:

*TFT\src\User\API\icon\_list.inc* add **X\_ICON (CUSTOM\_1)**

*TFT\src\User\API\Language\Language.inc* add **X\_WORD (CUSTOM\_1)**

*TFT\src\User\API\Language\language\_en.h* add **#define STRING\_CUSTOM\_1 "TG Menu"**

*TFT\src\User\API\Language\language\_keywords.h* add #**define LANG\_KEY\_CUSTOM\_1 "label\_custom\_1:"**

Any time a new keyword is added to the 4 files above, we have to change these signs in *TFT\src\User\API\Settings.h*:

#define **LANGUAGE\_FLASH\_SIGN** 20201107 //(YYYYMMDD) change if any keyword(s) in language pack is added or removed

#define **ICON\_FLASH\_SIGN** 20201107 //(YYYYMMDD) change if any icon(s) is added or removed

Now, to implement the new icon as a button in a menu, we find an empty slot in one of the existing menus and

modify the **const MENUITEMS** array for that parent menu. For example, we'll use the Settings Menu as the parent

menu for our new button because it has some empty positions in it (*each menu page can have 7 buttons max*).

So open the code file for the Settings Menu (TFT*\src\User\Menu\SettingsMenu.c* in this case).

We see there is a **const MENUITEMS** array named **settingsItems** and we also see this array has two unused

entries at indexes 5 and 6 (they're set to **ICON\_BACKGROUND,LABEL\_BACKGROUND** which means empty).

So we can add our new button in like so, replacing with our new values as shown below:

const MENUITEMS settingsItems = {

// title

LABEL\_SETTINGS,

// icon label

{{ICON\_SCREEN\_SETTINGS, LABEL\_SCREEN\_SETTINGS},

{ICON\_MACHINE\_SETTINGS, LABEL\_MACHINE\_SETTINGS},

{ICON\_FEATURE\_SETTINGS, LABEL\_FEATURE\_SETTINGS},

{ICON\_SCREEN\_INFO, LABEL\_SCREEN\_INFO},

{ICON\_CONNECTION\_SETTINGS, LABEL\_CONNECTION\_SETTINGS},

{**ICON\_CUSTOM\_1, LABEL\_CUSTOM\_1**}, <-- we overwrite previous ICON\_BACKGROUND, LABEL\_BACKGROUND

{ICON\_BACKGROUND, LABEL\_BACKGROUND}, with the icon name and text label of our custom icon

{ICON\_BACK, LABEL\_BACK},}

};

Next, we need to add some code to handle what happens when our custom button is pressed. There's usually

a code block like the one shown shown below for the parent menu. It’s usually found not too far after the

**const MENUITEMS <parentmenu\_name>** block we modified above. Put your handling code in this code block.

while(infoMenu.menu[infoMenu.cur] == **menuSettings**)

{

key\_num = menuKeyGetValue();

switch(key\_num)

{

case KEY\_ICON\_0:

infoMenu.menu[++infoMenu.cur] = menuScreenSettings;

break;

case KEY\_ICON\_1:

mustStoreCmd("M503 S0\n");

infoMenu.menu[++infoMenu.cur] = menuMachineSettings;

break;

case KEY\_ICON\_2:

infoMenu.menu[++infoMenu.cur] = menuFeatureSettings;

break;

case KEY\_ICON\_3:

infoMenu.menu[++infoMenu.cur] = menuInfo;

break;

case KEY\_ICON\_4:

infoMenu.menu[++infoMenu.cur] = menuConnectionSettings;

break;

case KEY\_ICON\_5: <-- here we add the case for custom\_1 button

**infoMenu.menu[++infoMenu.cur] = menuTGmenu; at position 5 in the menu. You can call**

break; another menu or handle some task here.

Also notice there is no KEY\_ICON\_6,

case KEY\_ICON\_7: since that menu position is blank.

infoMenu.cur--;

break;

default:

break;

}

loopProcess();

}

Finally, we need to create a .c and .h module to perform some action (i.e. handle the **menuTGmenu** call above). Here is the C code file, **TGmenu.c**

#include "TGmenu.h"

#include "SendGcode.h"

#include "includes.h"

//extern char gcodeBuf[CMD\_MAX\_CHAR];

//char\* gcodeBufPtr = &gcodeBuf[0];

void menuTGmenu(void) <-- here is the new function to draw new menu

{

//TG examples of variable usage

/\*

static uint8\_t ublSlot;

static bool ublIsSaving = true;

static bool ublSlotSaved = false;

\*/

MENUITEMS TGmenuItems = { <-- this is the new MENUITEMS structure

// title

LABEL\_TGMENU,

// icon label

{{ICON\_Z\_0, LABEL\_Z\_0},

{ICON\_Z\_300, LABEL\_Z\_300},

{ICON\_M503, LABEL\_M503},

{ICON\_BACKGROUND, LABEL\_BACKGROUND},

{ICON\_BACKGROUND, LABEL\_BACKGROUND},

{ICON\_BACKGROUND, LABEL\_BACKGROUND},

{ICON\_BACKGROUND, LABEL\_BACKGROUND},

{ICON\_BACK, LABEL\_BACK}}

};

KEY\_VALUES key\_num = KEY\_IDLE;

The switch below is not used, just included to show how to modify an existing menu dynamically

//TG this looks like an example of how to modify an existing menu on the fly based on

// information or other variables elsewhere in code

/\* switch (infoMachineSettings.leveling)

{

case BL\_BBL:

autoLevelingItems.title.index = LABEL\_ABL\_SETTINGS\_BBL;

break;

case BL\_UBL:

autoLevelingItems.title.index = LABEL\_ABL\_SETTINGS\_UBL;

autoLevelingItems.items[1].icon = ICON\_EEPROM\_SAVE;

autoLevelingItems.items[1].label.index = LABEL\_SAVE;

autoLevelingItems.items[2].icon = ICON\_EEPROM\_RESTORE;

autoLevelingItems.items[2].label.index = LABEL\_LOAD;

break;

default:

break;

}

\*/

menuDrawPage(&TGmenuItems); <-- call to draw the menu

while (infoMenu.menu[infoMenu.cur] == menuTGmenu) <-- now wait in loop for an item to be pressed

{

key\_num = menuKeyGetValue(); <-- find out what menu item was touched

switch (key\_num) <-- do something for current touched menu item

{

case KEY\_ICON\_0:

storeCmd("G1 Z0\n");

break;

case KEY\_ICON\_1:

storeCmd("G1 Z300\n");

break;

case KEY\_ICON\_2:

/\*

gcodeBuf[0]=(char)'M';

gcodeBuf[1]=(char)'5';

gcodeBuf[2]=(char)'0';

gcodeBuf[3]=(char)'3';

gcodeBuf[4]=(char)'/n';

gcodeBuf[5]=0;

storeCmd(gcodeBuf);

\*/

storeCmd("M503\n");

//gcodeBuf[0] = 0;

infoMenu.menu[++infoMenu.cur] = menuTerminal;

break;

case KEY\_ICON\_7: <-- the “back” item sets current menu to previous

infoMenu.cur--;

break;

default:

break;

}

loopProcess();

}

}

And here is the .h file, **TGmenu.h**

#ifndef \_TGMENU\_H\_

#define \_TGMENU\_H\_

#ifdef \_\_cplusplus

extern "C" {

#endif

#include <stdbool.h>

void menuTGmenu(void);

#ifdef \_\_cplusplus

}

#endif

#endif

We are done!