GUI Script
User Manual
Beta version

Enable Your Design

ThaiEasyElec.com

On-line Electronics Shop for Embedded System

Revision History

Version	Date	Changes
1.00	23 Sep 2010	Original version, used with gui_engine_v3_00.h gui_sound_plugin_v2_00.h

1. Introduction

GUI script is a new concept for GUI development on non-OS application. It's designed for engineers who have skill in C programming and microcontroller. GUI script makes GUI development faster, upgradable and more flexible. Advantages of using GUI script are described in **BlueScreen SUN7 User Manual**. In this user manual user will have more detail commands, API and programming guides.

2. Related Files

Source files related with GUI script are:

gui_engine.h: The header of GUI engine, it processes GUI commands from script file.

gui_sound_plugin.h: In application with sound needed, this is a plug-in that works cooperatively with vs1011.c to manage MP3 files playing along the screens.

app_scr_func.c (and .h): This is the application specific source file created for users to writing their code. And users need to define quantity of objects in the header file.

GUI_ENGINE.lib: The library file.

The source file versions this user manual referenced to are issued in Revision History. Changes in source files can be seen in their header files.

3. Features

Screens:

- Wait time assignable for no activity screens
- Namable (assign name to screens for easy identification)
- MP3 files assignable to play at the beginning of screens
- Multiple auxiliary values assignable for user purpose
- Standard keypads selectable to screens
- Imaged backgrounds with language assignable
- Plain colored backgrounds
- Function called before screens being drawn
- Function called after screens being drawn
- Function every 100ms while the screen being displayed

Buttons:

- Imaged buttons with language assignable
- Plain colored buttons
- Image or color in normal, pressed and disabled states
- Transparency supported
- Movement (while being pressed) supported in downward and rightward direction only
- Namable (assign name to buttons for easy identification)
- On-button text with specifiable font, color and bold characteristic
- Key buttons supported
- Language mode change supported
- Auxiliary value assignable for key buttons or user purpose
- Going to screen ID when released
- Interlock buttons
- Function called when pressed
- Function called when released
- Function called when buttons are associated in all case

Images on screen:

- Language assignable
- Transparency supported

Textboxes:

- Text, background, and border color assignable

- Maximum characters assignable
- Password mode with assignable displayed character
- Specifiable font, color and bold characteristic for texts
- Text alignment: left, center and right
- Insertion allowed (with cursor enabled) or not (no cursor)

Tables:

- Row quantity, row height assignable
- Column width, caption color, caption text and content color assignable individually
- Specifiable font, color and bold characteristic for texts
- Border color assignable
- Function called when pressed
- Function called when tables are associated in all case

Labels:

- Language assignable
- Alignment: left, center and right referenced from origin position
- Specifiable font, color and bold characteristic for texts

Boxes (for making simple boxes or lines on screen):

- Color assignable

Fonts & Extension fonts:

- Accept binary font files generated from bmp2h_conv software

Language:

- Support up to 8 language modes

Paths:

- Support multi-path applications, if the hierarchy can be divided into paths and sharing some common screens, path can be set for easier implementation

4. Commands and Parameters

4.1 Command structure

GUI script is processed by console; the command used here is “GUI” (just like “LS” command that will show content in current directory). But with so many components to be set for GUI, a subcommand is needed. Each subcommand needs different parameters. Below is the command structure:

```
GUI subcommand parameter1 parameter2 … [parameterX]
```

Please note that:

1. Only “GUI” is not case-sensitive, since it is just a console command. The rest is case-sensitive.
2. With console version 2.04 or later, you can use *space* or *tab* to separate tokens.
3. With console version 2.04 or later, you can assign a number in 3 formats: straight number (e.g. 123, 4) character format (e.g. ‘a’, ‘A’) or hexagonal format (e.g. 0xA0, 0xFFFF).
4. Parameters written in [ ], are optional, if there is in a command, it’s always the last one. Unless they are set, default value will be used.
5. Every parameter that is text type (file paths, names, texts), may contains spaces. Console engine accepts all characters until the end of line found. This kind of parameters is always the last one in commands.

4.2 Parameter Descriptions

- Color: color value range depends on color depth used in the application. Color depth can be 8, 16 or 24 bpp. Below are example color values (also listed in lcd_ctrl.h). Note that default color is black (0), parameters don’t need to be set if default value used.

Common color table

Color	8 bpp	16 bpp	24 bpp
Red	0x03	0x001F	0x0000FF
Green	0x1C	0x07E0	0x00FF00
Blue	0xE0	0xF800	0xFF0000
Yellow	0x1F	0x07FF	0x00FFFF

Purple	0xE3	0xF81F	0xFF00FF
Black	0x00	0x0000	0x000000
White	0xFF	0xFFFF	0xFFFFFF

- Language: language variable in software is char-type. It's always one in 1,2,4,8,16,32,64,128 (1 << n, n=0-7). Anyway, the language value can be set from 1 to 255. This allows an object supports more than 1 language. In general case, use 1 2 8 … for single language objects and 255 for objects used for all language.

- Object ID: screen ID, button ID and so on, are indexed by short-type variable. A short-type variable consumes 2 bytes, so it can be 0-65535 (0xFFFF is 65535). Anyway, the value 65535 is used as some unset value and shouldn't be used by users.

5. GUI Commands

5.1 RBG: Register background images

The concept of GUI script is simple, images, background images and fonts must be first read and stored to SDRAM. An ID number must be assigned to each of them. And after that, this ID number will be used in the rest of script.

```
            GUI RBG ID Image_type Lang Path

e.g.        GUI RBG 0  2  255  img/bg1.bmp
```

ID: ID number of background image
Image_type: type of image;
0 for binary files (.bin)
1 for JPEG files
2 for BMP files
Lang: language for background image
Path: path of the file stated from current directory containing script file, for example, if the script file path is “sr/main.txt”, here the image path must be “sr/img/bg1.bmp”

5.2 RFONT: Register fonts

```
            GUI RFONT ID Width Height Gap Path

e.g.        GUI RFONT 0  25 24 1 fonts/arial_eng25x24.bin
```

ID: ID number of font
Width: width of character block in pixel
Height: height of character block in pixel, can only be 8, 16, 24… 128
Gap: Gap between characters for display
Path: see RBG command

5.3 RXFONT: Register extension fonts

```
            GUI RXFONT ID Width Height Adj Aux Path

e.g.        GUI RXFONT 0 25 24 1 0 fonts/arial_thai25x24.bin
```

ID: ID number of extension font
Width: width of character blocks in pixel
Height: height of character blocks in pixel; can only be 8, 16, 24… 128, anyway, Thai fonts are supported up to 64.
Adj: height adjustment for English font and extension font
Aux: auxiliary value, use dependently on extension language (for Thai, this is the height for shifting tone when upper vowel is presented)
Path: see RBG command

5.4 RIMG: Register images

```
GUI RIMG ID Image_type(0) Width Height Lang Path
GUI RIMG ID Image_type(2) Lang Path
GUI RIMG ID Image_type(3) Width Height Lang Color [Border]
```

```
e.g.    GUI RIMG 0 0 25  75 255 img/button0.bin
        GUI RIMG 1 2 255 img/button0.bmp
        GUI RIMG 2 3 25  75 255 0xFFFF
```

ID: ID number of image
Image_type: type of image;
0 for binary files (.bin)
1 for JPEG files, currently not supported
2 for BMP files
3 for plain color button
Width: width of image in pixel
Height: height of image in pixel
Lang: language for image
Color: color of button's body
Border: color of button's border, only applied when BT3_SMOOTH_LV = 0 (see obj_lib.c, default is 2)
Path: see RBG command

5.5 MEM: Show memory status

```
GUI MEM
```

Show memory status, this should be places after all file registering finished, where all memory used for images, fonts and extension fonts are summarized.

5.6 SCS: Screen settings

-SCS S: Set associated screen ID, with this command, scripts afterward will belong to this screen ID.

```
GUI SCS S ID
e.g.    GUI SCS S 0
```

ID: screen ID

-SCS n: Name the associated screen, for easy identifying in software; user can specify the screen with a name so the screen ID can be changed without code modification.

```
GUI SCS n Name
e.g.    GUI SCS n screen1
```

Name: a text to be used as the name of screen

-SCS P: Switch path to new path when the screen is shown up. Ignore this command for single path application.

```
GUI SCS P Path
e.g.    GUI SCS P 1
```

Path: new path to be switched to

-SCS C: Set char-type auxiliary values. Its range is 0-255. The quantity of value can be set in source code (default is 2), see appendix for more detail.

```
GUI SCS C Index Value
e.g.    GUI SCS C 0 5
```

Index: index of auxiliary values
Value: number between 0-255

-SCS L: Set long-type auxiliary values. Its range is 0x0-0xFFFFFFFF. The quantity of value can be set in source code (default is 1), see appendix for more detail.

```
GUI SCS L Index Value
```

```
e.g.        GUI SCS L 0 100000
```

Index: index of auxiliary values
Value: number between 0-0xFFFFFFFF

-SCS W: Set waiting time, in case that the associated screen may show up temporary and then switch to other screen (or switch to next screen).

```
            GUI SCS W Time [ID]

e.g.        GUI SCS W 50 10
```

Time: time in 100ms, with example above, the waiting time is 5 seconds
ID: target screen ID to be shown after the waiting completed, if ID is not entered, next adjacent screen will be shown

For multi-path applications:

```
            GUI SCS W Time ID#1 ID#2 …

e.g.        GUI SCS W 50 1 2 3
```

ID#n: target screen ID for path 0, path 1 and so on, single path assignment means that target screen IDs are the same for all paths

5.7 BG: Background image settings

-BG A: Add background images to associated screen, can be more than one (depend on number of language in the application).

```
            GUI BG A ID#1 [ID#2] [ID#3] … [ID#8]

e.g.        GUI BG A 0  1  2
```

ID: ID of registered background images, the quantity of ID can be up to MAX_LANGUAGE defined in the code

-BG C: Set background color in case of using plain color background, only one of BG A or BG C should be applied to a screen.

```
            GUI BG C Color
```

```
e.g.        GUI BG C 0xFFFF
```

Color: color of background, it's white in the example above (for 16 bpp)

5.8 IMG: Image box settings

-IMG O: Set origin position of the image box, default is (0, 0).

```
            GUI IMG O X Y

e.g.        GUI IMG O 100 200
```

X: horizontal position
Y: vertical position

-IMG I: Set image ID used for the image box, default is 0.

```
            GUI IMG I ID

e.g.        GUI IMG I 5
```

ID: ID of registered images

-IMG t: Set transparency, without this command, the image box will be displayed normally (showing white pixel).

```
            GUI IMG t
```

-IMG E: End current image box setting, end every image box setting with this command so the index increase for next one.

```
            GUI IMG E
```

5.9 BT: Button settings

-BT n: Name the button, for easy identifying in software; user can specify the button with a name.

```
            GUI BT n Name

e.g.        GUI BT n Next
```

Name: a text to be used as the name of button

-BT O: Set origin position of the button, default is (0, 0).

```
GUI BT O X Y

e.g.    GUI BT O 100 200
```

X: horizontal position
Y: vertical position

-BT N: Set image ID used for the button in normal state, default is 0.

```
GUI BT N ID

e.g.    GUI BT N 0
```

ID: ID of registered images

-BT P: Set image ID used for the button in pressed state, default is 0.

```
GUI BT P ID

e.g.    GUI BT P 1
```

ID: ID of registered images

-BT D: Set image ID used for the button in disabled state, default is 0.

```
GUI BT D ID

e.g.    GUI BT D 2
```

ID: ID of registered images

-BT M: Set the movement when the button is pressed, default is 0, 0.

```
GUI BT M X Y

e.g.    GUI BT M 2 2
```

X: rightward move (in pixel)
Y: downward move (in pixel)

-BT G: Set the screen to go to when the button is released. Or go to next screen.

```
GUI BT G [ID]
```

```
e.g.    GUI BT G 4
```

ID: screen ID to go to, if not entered, go to next screen

For multi-path applications:

```
GUI BT G ID#1 ID#2 …
```

```
e.g.    GUI BT G 1 2 3
```

ID#n: target screen ID for path 0, path 1 and so on, single path assignment means that target screen IDs are the same for all paths

-BT d: Disable the button at startup (can be enabled later by software).

```
GUI BT d
```

-BT t: Set transparency, without this command, the button will be displayed normally (showing white pixel).

```
GUI BT t
```

-BT i: Set inverse mode, when pressed or released button will be inversed. With this mode set, movement setting is discarded.

```
GUI BT i
```

-BT T: Set on-button text.

```
GUI BT T Text
```

```
e.g.    GUI BT T Enter
```

Text: text to be displayed on the button

-BT F: Set font and/or extension font for on-button text, defaults are 0.

```
GUI BT F ID#1 [ID#2]
```

```
e.g.        GUI BT F 1 1
```

ID#1: font ID
ID#2: extension font ID

-BT B: Make on-button text bold.

```
            GUI BT B
```

-BT C: Set on-button text color, default is black.

```
            GUI BT C Color

e.g.        GUI BT C 0xF800
```

Color: color for on-button text, blue for the example above

-BT A: Set on-button text alignment, default is center (1).

```
            GUI BT A Alignment_type

e.g.        GUI BT A 2
```

Alignment type: 0 for left, 1 for center and 2 for right

-BT S: Save current button settings.

```
            GUI BT S

e.g.        GUI BT M 2 2
            GUI BT S
            GUI BT N 0
            GUI BT P 1
            GUI BT O 100 200
            GUI BT E
```

-BT L: Load settings to current button, example below shows the movement setting is copied from example above. The save and load commands help users in setting common parameters for buttons.

```
            GUI BT L

e.g.        GUI BT L
            GUI BT N 2
            GUI BT P 3
```

```
GUI BT O 300 200
GUI BT E
```

-BT V: Set value for the button. A 32-bit integer is used to store the value so the range is 0-0xFFFFFFFF.

```
GUI BT V Value
```

```
e.g.        GUI BT V 'A'
```

Value: value for the button

-BT X: Set special action for the button, default is 0.

```
GUI BT X Value
```

```
e.g.        GUI BT X 1
```

Value: special action for the button:
1 for key button, the value set by BT V will be used as a key character
2 for language change button, the value set by BT V will be used as new language

-BT E: End current button setting

```
GUI BT E
```

5.10 TXT: Textbox settings

-TXT O: Set origin position of the textbox, default is (0, 0).

```
GUI TXT O X Y
```

```
e.g.        GUI TXT O 100 200
```

X: horizontal position
Y: vertical position

-TXT S: Set textbox size.

```
GUI TXT S Width Height
```

```
e.g.        GUI TXT S 150 50
```

Width: width of the textbox
Height: height of the textbox

-TXT I: Set insertion of the textbox, disabled by default, with this command, the textbox can be pressed and the cursor is displayed. Then new characters can be inserted between the texts. Without this command (default), no curser shown and new characters will always be appended to the right side of the texts.

```
GUI TXT I
```

-TXT K: Set textbox active so it will receive characters from key buttons.

```
GUI TXT K
```

-TXT B: Make text on the textbox bold.

```
GUI TXT B
```

-TXT C: Set text and background colors, defaults are black.

```
GUI TXT C Text_color Back_color
```

```
e.g.    GUI TXT C 0  0xFFFF
```

Text_color: text color
Back_color: background color

-TXT F: Set font and/or extension font, defaults are 0.

```
GUI TXT F ID#1 [ID#2]
```

```
e.g.    GUI TXT F 1 1
```

ID#1: font ID
ID#2: extension font ID

-TXT A: Set on-text box text alignment, default is center (1).

```
GUI TXT A Alignment_type
```

```
e.g.    GUI TXT A 2
```

Alignment type: 0 for left, 1 for center and 2 for right

-TXT L: Set maximum characters in the textbox (length).

```
            GUI TXT L Length
e.g.        GUI TXT L 10
```

Length: maximum of characters

-TXT p: Enable password mode.

```
            GUI TXT p Password_character
e.g.        GUI TXT p 'X'
```

Password_character: a character to be shown in password box

-TXT E: End current text box setting.

```
            GUI TXT E
```

5.11 LBL: Label settings

-LBL O: Set origin position of the label, default is (0, 0). Unlike all other objects, the alignment of label refers to its origin position. So the top-left corner of the label is not its origin position when the alignment set is center or right.

```
            GUI LBL O X Y
e.g.        GUI LBL O 100 200
```

X: horizontal position
Y: vertical position

-LBL B: Make text on the label bold.

```
            GUI LBL B
```

-LBL C: Set text color, default is black.

```
            GUI LBL C Text_color
e.g.        GUI LBL C 0
```

Text_color: text color

-LBL T: Set label text.

```
GUI LBL T Text
e.g.    GUI LBL T Name:
```

Text: label text

-LBL F: Set font and/or extension font, defaults are 0.

```
GUI LBL F ID#1 [ID#2]
e.g.    GUI LBL F 1 1
```

ID#1: font ID
ID#2: extension font ID

-LBL A: Set on-text box text alignment, default is center (1).

```
GUI LBL A Alignment_type
e.g.    GUI LBL A 2
```

Alignment type: 0 for left, 1 for center and 2 for right

-LBL L: Set language for label.

```
GUI LBL L Lang
e.g.    GUI LBL L 1
```

Lang: language for label

-LBL E: End current text box setting

```
GUI LBL E
```

5.12 BOX: Box settings

-BOX O: Set origin position of the box, default is (0, 0).

```
GUI BOX O X Y
```

```
e.g.    GUI BOX O 0 100
```

X: horizontal position
Y: vertical position

-BOX S: Set box size.

```
GUI BOX S Width Height
```

```
e.g.    GUI BOX S 800 2
```

Width: width of the box
Height: height of the box

-BOX C: Set box color, default is black.

```
GUI BOX C Color
```

```
e.g.    GUI BOX C 0
```

Color: box color

5.13 TAB: Table settings

-TAB O: Set origin position of the table, default is (0, 0).

```
GUI TAB O X Y
```

```
e.g.    GUI TAB O 50 200
```

X: horizontal position
Y: vertical position

-TAB B: Make texts on the table bold.

```
GUI TAB B
```

-TAB C: Set text and border colors, defaults are black.

```
GUI TAB C Text_color Border_color
```

```
e.g.        GUI TAB C 0xF800 0
```

Text_color: text color
Border_color: border color

-TAB c: Set column's parameters.

```
            GUI TAB c Column Width Cap_color Con_color [Text]

e.g.        GUI TAB c 0 100 0x07E0 0xFFFF No.
            GUI TAB c 1 200 0x07E0 0xFFFF Item Name
            GUI TAB c 2 400 0x07E0 0xFFFF Description
```

Column: column index
Width: column width
Cap_color: caption color (the top row)
Con_color: content color (every row but the top one)
Text: caption text

-TAB R: Set row's parameters.

```
            GUI TAB R Height nRow

e.g.        GUI TAB R 50 10
```

Height: row height
nRow: number of row

-TAB F: Set font and/or extension font, defaults are 0.

```
            GUI TAB F ID#1 [ID#2]

e.g.        GUI TAB F 1 1
```

ID#1: font ID
ID#2: extension font ID

-TAB A: Set on-text box text alignment, default is center (1).

```
            GUI TAB A Alignment_type

e.g.        GUI TAB A 2
```

Alignment type: 0 for left, 1 for center and 2 for right

-TAB E: End current table setting.

```
GUI TAB E
```

5.14 CFG: Configure global parameters

-CFG L: Configure default language, default is 1.

```
GUI CFG L Lang
e.g.    GUI CFG L 2
```

Lang: default (initial) language

5.15 SND: Play a MP3 file at the beginning of screen (gui_sound_plugin.c needed).

```
GUI SND ID
e.g.    GUI SND 0
```

ID: registered sound ID

5.16 END: End all settings, this command shows summarized objects created or registered.

```
GUI END
```

5.17 BEG: Set startup option. This subcommand is useful for development process; users can jump to specific screen for their test and debugging.

-BEG S: Set startup screen, default is 0

```
GUI BEG S ID
e.g.    GUI BEG S 1
```

ID: screen ID

-BEG P: Set startup path, default is 0

```
GUI BEG P Path
```

```
e.g.    GUI BEG P 2
```

Path: path (route in the GUI flow)

6. SND Commands

This section is applicable when gui_sound_plugin.c is added to the project. The "SND" command is not the one with 5.15. "GUI SND" is a "GUI" command with a subcommand "SND". Now it's "SND" command and will be processed in the plug-in source code, not the GUI engine.

Unlike images or fonts, MP3 files are not stored on SDRAM. Only their paths are stored.

6.1 CFG: Configure global parameters

-CFG D: Configure sound directory, default is "/" (root);

```
SND CFG D Path

e.g.    SND CFG D sr/snd
```

Path: directory contains MP3 files.

6.2 RG0: Register MP3 files.

```
SND RG0 ID nSegment Volume Lang Path

e.g.    SND RG0 0  1 80 255 start.mp3
        SND RG0 1 10 80 255 numbers.mp3
```

ID: sound ID
nSegment: segments in sound file, this is 1 for general files playing all at once, but can be more then one for segmented files
For example; the numbers.mp3 sounds like "one two three … zero", each segment can be played individually.
Volume: sound volume, can be from 0-99.
Lang: language for MP3 file
Path: file path

6.3 RGC: Register MP3 files for multi-language application. Note that files can be assigned to an ID if their contents are the same but in multiple languages. Software will play the right file (with the language mode at a time) automatically.

```
            SND RGC Lang Path

e.g.        SND RG0 0 1 80   2 start_eng.mp3
            SND RGC          1 start_th.mp3
```

Lang: language for MP3 file
Path: file path

7. Command List

Command	Format	Section
Register background images	`GUI RBG ID Image_type Lang Path`	5.1
Register fonts	`GUI RFONT ID Width Height Gap Path`	5.2
Register extension fonts	`GUI RXFONT ID Width Height Adj Aux Path`	5.3
Register images	`GUI RIMG`	5.4
-Binary files	`GUI RIMG ID 0 Width Height Lang Path`	
-BMP files	`GUI RIMG ID 2 Lang Path`	
-Colored box	`GUI RIMG ID 3 Width Height Lang Color [Border]`	
Show memory status	`GUI MEM`	5.5
Screen settings	`GUI SCS`	5.6
- ID	`GUI SCS S ID`	
-Name	`GUI SCS n Name`	
-Switch path	`GUI SCS P Path`	
-Auxiliary value (char)	`GUI SCS C Index Value`	
-Auxiliary value (long)	`GUI SCS L Index Value`	
-Wait time (all paths)	`GUI SCS W Time [ID]`	
-Wait time (multi-path)	`GUI SCS W Time ID#1 ID#2 …`	
Background settings	`GUI BG`	5.7
-Add to screen	`GUI BG A ID#1 [ID#2] [ID#3] … [ID#8]`	
-Set background color	`GUI BG C Color`	
Image box settings	`GUI IMG`	5.8
-Origin	`GUI IMG O X Y`	
-Image ID	`GUI IMG I ID`	
-Transparency	`GUI IMG t`	
-End setting	`GUI IMG E`	
Button settings	`GUI BT`	5.9
-Name	`GUI BT n Name`	
-Origin	`GUI BT O X Y`	
-Normal state image	`GUI BT N ID`	
-Pressed state image	`GUI BT P ID`	
-Disabled state image	`GUI BT D ID`	
-Movement	`GUI BT M`	
-Screen to go (all paths)	`GUI BT G [ID]`	
-Screen to go (multi-path)	`GUI BT G ID#1 ID#2 …`	
-Disabled at startup	`GUI BT d`	
-Transparency	`GUI BT t`	
-Inverse	`GUI Bt i`	
-On-button text	`GUI BT T Text`	
-Fonts	`GUI BT F ID#1 [ID#2]`	
-Bold characteristic	`GUI BT B`	
-Text color	`GUI BT C Color`	
-Text alignment	`GUI BT A Alignment_type`	
-Save button settings	`GUI BT S`	

-Load button settings	GUI BT L	
-Value	GUI BT V *Value*	
-Special action	GUI BT X *Value*	
-End setting	GUI BT E	
Textbox settings	GUI TXT	5.10
-Origin	GUI TXT O *X Y*	
-Size	GUI TXT S *Width Height*	
-Insertion	GUI TXT I	
-Key characters receiver	GUI TXT K	
-Bold characteristic	GUI TXT B	
-Colors	GUI TXT C *Text_color Back_color*	
-Fonts	GUI TXT F *ID#1 [ID#2]*	
-Text alignment	GUI TXT A *Alignment_type*	
-Text length (max. characters)	GUI TXT L *Length*	
-Password mode	GUI TXT p *Password_character*	
-End setting	GUI TXT E	
Label settings	GUI LBL	5.11
-Origin	GUI LBL O *X Y*	
-Bold characteristic	GUI LBL B	
-Color	GUI LBL C *Text_color*	
-Text	GUI LBL T *Text*	
-Fonts	GUI LBL F *ID#1 [ID#2]*	
-Text alignment	GUI LBL A *Alignment_type*	
-Language	GUI LBL L *Lang*	
-End setting	GUI LBL E	
Box settings	GUI BOX	5.12
-Origin	GUI BOX O *X Y*	
-Size	GUI BOX S *Width Height*	
-Color	GUI BOX C *Color*	
Table settings	GUI TAB	5.13
-Origin	GUI TAB O *X Y*	
-Bold characteristic	GUI TAB B	
-Color	GUI TAB C *Text_color Border_color*	
-Column's parameters	GUI TAB c *Column Width Cap_color Con_color [Text]*	
-Row's parameters	GUI TAB R *Height nRow*	
-Fonts	GUI TAB F *ID#1 [ID#2]*	
-Text alignment	GUI TAB A *Alignment_type*	
-End setting	GUI TAB E	
Configure global parameters	GUI CFG	5.14
-Default language	GUI CFG L *Lang*	
Add sound on screen startup	GUI SND *ID*	5.15
End GUI settings	GUI END	5.16
Set startup option	GUI BEG	5.17
-Startup screen	GUI BEG S *ID*	
-Startup path	GUI BEG P *Path*	
Configure global parameters (sound)	SND CFG	6.1
-Sound directory	SND CFG D *Path*	

Register MP3 file	`SND RG0 ID nSegment Volume Lang Path`	6.2
Register MP3 file (cont.)	`SND RGC Lang Path`	6.3

Remark: Parameters in [ ] are optional

8. Writing Script

8.1 Script Structure

After that you know commands. Now let’s see how to write your own script. A script file should have a head and body parts. The head part contains registering common objects to be used in the body part. The word common objects, whether they are used only one time or more, needed to be registered here. Then the body part describes what will be contains on screens and where they are.

8.1.1 Head Part

Objects to be registered here are font, extension font, background image, image and sound. Notice that these objects are memory-consuming. So put them here and then use them again and again. Ordering here is no needed since each object has an assigned ID. And each register function has its own ID domain. At the end of this part, memory consumed is summarized so show it with “GUI MEM” command.

```
##HEAD##
//register font(s)
GUI RFONT 0 …
GUI RFONT 1 …
…
//register extension font(s)
GUI RXFONT 0 …
GUI RXFONT 1 …
…
//register background image(s)
GUI RBG 0 …
GUI RBG 1 …
…
//register image(s)
GUI RIMG 0 …
GUI RIMG 1 …
…
SND CFG D …
SND RG0 0 …
SND RGC 0 …
SND RG0 1 …
SND RGC 1 …
…
//show memory used
GUI MEM
```

8.1.2 Body Part

The body part is segmented into screens. Begin each screen with “GUI SCS S” set some screen parameters if needed and then create buttons, tables or more as designed. End the script with “GUI END” so the engine shows how many objects created. Note that objects created here usually take many lines, that’s why we need “GUI x E” lines.

```
##BODY##
######screen 0######
GUI SCS S 0
GUI SCS C …
GUI SND …
GUI BG  A …

GUI BT  …
…
GUI BT  E

GUI BT  …
…
GUI BT  E

GUI LBL …
…
GUI LBL E

…

######screen 1######
GUI SCS S 1
GUI SND …
GUI BG  A …

GUI BT  …
…
GUI BT  E

GUI TAB …
…
GUI TAB E

…

######screen 2######
…

GUI END
```

8.2 Script Making Guide Lines

Actually the screen ordering is not needed. You can even have screen 0, 1 and then 3. Also you can put a button, a table and then a button again if you like. However, making script should be done tidily so the one who takes care of C programming can understand it well. It's recommend that script file should be made as follow:

1. Place same type objects together for easier reading.

2. Name screens and special buttons (buttons that stimulate some events). Buttons with name can be indentified easily. Buttons with functional scripts (key buttons) may be left unnamed since programmer has nothing to do with them.

3. Utilize auxiliary values. Auxiliary value can be assigned to screens and buttons, utilizing them make the code more flexible as the value can be changed easier.

4. Use "save" and "load" for similar buttons. If a screen has buttons sharing some parameters, "save" and "load" reduce script lines and let you change parameters once for all.

5. Left parameters with default value unset (for example; black text, white background). Also reserve ID 0 for most common font. These save many of script lines.

10. Application Programming Interface

This section provides information on how to make application based on the GUI script. First of all, users should specify quantities for objects. So the GUI engine can provide enough memory for them. After that, creates their own functions and assign them to events.

10.1 Specify Numbers of Objects

Images, buttons and other objects need memory to store their parameters. Users must specify them first so there is enough memory reserved. At the final state, when the new firmware revision is released, users should specify quantities so more objects can be added (from script) in the future.

Specifying should be done in app_scr_func.h. Here default values are set. Note that background images use constant size of memory, whether they are assigned in script or not.

```
//a part from app_scr_func.h
#define MAX_SCR_OBJ         32
#define MAX_LANGUAGE        2
#define MAX_PATH            1
#define MAX_SCR             50          //all screens
…
```

Table below describes parameters specifiable; some of them have default value and don't need to be set.

Parameters	Description
MAX_SCR_OBJ	Reserved dynamic objects on a screen (see more detail about dynamic object later in this section)
MAX_LANGUAGE	Language modes, up to 8
MAX_PATH	Paths in application
MAX_SCR	Reserved screen slots
MAX_BACKGROUND	Reserved background image slots
MAX_IMAGE	Reserved image slots used for buttons and image boxes
MAX_BUTTON	Reserved button slots
MAX_IMAGE_BOX	Reserved image box slots
MAX_TEXTBOX	Reserved textbox slots
MAX_TABLE	Reserved table slots
MAX_TABLE_COL	Reserved columns in a table
MAX_TEXT_SIZE	Reserved text pool size (see more detail later in this section)

MAX_LABEL	Reserved label slots
MAX_BOX	Reserved box slots
MAX_FONT	Reserved font slots
MAX_EXT_FONT	Reserved extension font slots
AUX_INT_NUM	Number of integer-type auxiliary value in a screen slot, default is 1
AUX_CHAR_NUM	Number of character-type auxiliary value in a screen slot, default is 2
MAX_SOUND_FILE	Reserved sound file slots
MAX_SOUND	Reserved sound slots
MAX_SOUND_SEQ	Longest sound sequence on a screen

Remark: -All parameters must to be set to 1 or more
-Bold parameters have to be set
-Sound related parameters have to be set only when sound plug-in is used

On a screen, not every object is dynamic. Buttons, tables and textboxes have something to do when they are pressed, so they are dynamic. In the same time, labels, image boxes and boxes are static. Only dynamic objects are counted as objects on a screen and must not be over than MAX_SCR_OBJ.

Texts and names set with scripts are continuously concatenated in a text pool. Even a textbox having length set, will has memory reserved for its text here. All of these texts and names size together must not be over than MAX_TEXT_SIZE.

Besides of specifying numbers of objects, user can configure some settings in app_scr_func.h, table below shows some parameters that can be set for appropriate operation.

Parameters	Definition	Description
GUI_USE_SDRAM	Define it or remove it out	Have it defined make the GUI engine store object parameters in SDRAM (recommended)
SHOW_ACTIVITY	Define it as 0 or 1	Show activities on LCD while the script being read, left it undefined here to use default value 1 (show)