adc
advanced digital chips inc

AMAZONⅡ

- SDK Reference Manual-

Ver 1.08

September 22, 2016

Advanced Digital Chips Inc.

History

2013-07-24	Created
2014-09-18	Ver 1.02
	- Change ECON or ECon → E-Con
2014-10-20	Ver 1.03
	- Add 8.49 loadjpg_hw, 8.50 savebmp, 8.51 savejpg
	- Add 9 MOVIE FILE PLAY
2014-11-05	Ver 1.04
	- Add egl_visible_object / egl_window_delete_object function in EGL Library
	- Add release function for object. (window, button, checkbutton, etc …)
	- Modify 3. Using the SDK.
2014-12-11	Ver 1.05
	- Add 10.15 sound_playex, 10.16 sound_current_time
2015-02-09	Ver 1.06
	- Add description about nand booting at 2. Download.
2015-02-09	Ver 1.07
	- Add 8.52 surface_set_alpha.
2016-09-22	Ver 1.08
	- Add 8.41 draw_surface_rotate.

— Table of Contents —

1. SOFTWARE DEVELOPMENT ENVIRONMENT 9

2. DOWNLOAD 15

3. USING THE SDK 21

3.1 AMAZON Ⅱ SDK composition 21
3.2 AMAZON Ⅱ Board Switch 22
3.3 Serial Flash boot 22
3.4 NAND Flash boot 25

4. MEMORY SETTINGS 27

5. UART 29

5.1 uart_config 30
5.2 uart_putch 31
5.3 uart_putdata 31
5.4 uart_putstring 32
5.5 uart_getch 32
5.6 uart_getdata 33
5.7 uart_rx_flush 33
5.8 uart_tx_flush 34
5.9 set_debug_channel 34
5.10 get_debug_channel 35
5.11 debugprintf 35
5.12 debugstring 36
5.13 PRINTLINE 36
5.14 PRINTVAR(A) 36
5.15 UART Example 37

6. INTERRUPT 38

6.1 init_interrupt 39
6.2 set_interrupt 39
6.3 enable_interrupt 40
6.4 Interrupt Example 42

7. TIMER 43

7.1 set_timer 44
7.2 stop_timer 44
7.3 delayms 45
7.4 TIMER Example 46

8. GRAPHIC 47

8.1 ge_set_texture_mem 50
8.2 get_ge_target_buffer 50
8.3 setscreen 51
8.4 get_front_buffer 52
8.5 get_back_buffer 52
8.6 get_screen_width 53
8.7 get_screen_height 53
8.8 get_screen_pitch 53
8.9 get_screen_bpp 54
8.10 flip 54
8.11 flip_wait 55
8.12 async_flip 55
8.13 async_flip_wait 55
8.14 ge_wait_cmd_complete 56
8.15 ge_draw_rectfill 56
8.16 ge_draw_rectfillalpha 57
8.17 surface_set_alpha 57

8.18 draw_line *58*
8.19 draw_rect *59*
8.20 draw_rectfill *60*
8.21 draw_roundrect *61*
8.22 draw_roundrectfill *62*
8.23 draw_circle *63*
8.24 draw_circlefill *64*
8.25 draw_ellipse *65*
6.26 draw_ellipsefill *66*
8.27 loadbmp *67*
8.28 loadbmpp *67*
8.29 loadjpg *68*
8.30 loadjpgp *68*
8.31 loadtga *69*
8.32 loadtgap *69*
8.33 loadpng *70*
8.34 loadpngp *70*
8.35 loadsurf *71*
8.36 draw_surface *71*
8.37 draw_surface_rect *72*
8.38 draw_surface_scale *73*
8.39 draw_surface_scalerect *74*
8.40 draw_surface_rotate *75*
8.41 release_surface *76*
8.42 draw_set_clip_winodw *76*
8.43 osd_set_surface *77*
8.44 osd_set_surface_with_alpha *78*
8.45 osd_set_pal *79*
8.46 osd_set_off *79*
8.47 OSD_MOVE *80*
8.48 vga_mode_on *80*
8.49 vce_on_surface *81*
8.50 loadjpg_hw *82*
8.51 savebmp *82*
8.52 savejpg *83*
8.53 surface_set_alpha *84*
8.54 Graphic Example *85*

9. MOVIE FILE PLAY **87**

9.1 loadmovie *88*
9.2 release_movie *88*
9.3 movie_play *89*
9.4 movie_drawnext *90*
9.5 movie_drawnextex *91*
9.6 movie_seek *92*
9.7 movie_current_time *92*
9.8 example *93*

10. SOUND **94**

10.1 sound_init *95*
10.2 sound_loadwav *95*
10.3 sound_loadwavp *96*
10.4 sound_loadmp3 ..... *96*
10.5 sound_loadmp3p *97*
10.6 sound_release *97*
10.7 sound_play *98*
10.8 sound_stop *98*
10.9 sound_vol *99*
10.10 sound_vol_wav *99*
10.11 sound_pause *100*
10.12 sound_resume *100*
10.13 sound_isplay *101*

10.14 sound_ispause *101*
10.15 sound_playex *102*
10.16 sound_current_time *102*
10.17 Sound Example *103*

11. FILE SYSTEM **105**

11.1 f_mount *106*
11.2 f_chdrive *106*
11.3 f_chdir *107*
11.4 FILE System Example *108*

12. FONT **109**

12.1 create_bmpfont *110*
12.2 release_bmpfont *110*
12.3 bmpfont_draw *111*
12.4 bmpfont_draw_vleft *112*
12.5 bmpfont_draw_vright *113*
12.6 egl_font_set_color *113*
12.7 bmpfont_makesurface *114*
12.8 bmpfont_setkerning *114*
12.9 bmpfont_setautokerning *115*
12.10 create_bitfont *116*
12.11 release_bitfont *116*
12.12 bitfont_draw *117*
12.13 bitfont_draw_vleft *118*
12.14 bitfont_draw_vright *119*
12.15 bitfont_makesurface *119*

13. EGL LIBRARY **123**

WINDOW 125
▶*egl_create_window function* *126*
▶*egl_window_show function* *127*
▶*egl_window_isshow function* *128*
▶*egl_window_invalidate function* *129*
▶*egl_window_invalidate_rect function* *130*
▶*egl_window_redraw_rect function* *131*
▶*egl_window_set_bg function* *132*
▶*egl_window_get_bg function* *133*
▶*egl_window_add_object function* *134*
▶*egl_window_set_callback function* *135*
▶*egl_window_get_active function* *136*
▶*egl_user_touch_input function* *137*
▶*egl_draw function* *138*
▶*egl_visible_object function* *139*
▶*egl_window_delete_object function* *140*
▶*egl_release_window function* *141*
▶ *Window Example.* *142*
BUTTON 144
▶*egl_create_button function* *145*
▶*egl_button_callback function* *146*
▶*egl_release_button function* *147*
▶ *Button Example.* *148*
IMAGE BUTTON 149
▶*egl_create_image_button function* *150*
▶*egl_image_button_callback function* *151*
▶*egl_release_image_button function* *152*
▶ *Image Button Example.* *153*
TOGGLE IMAGE BUTTON 154

▶*egl_create_toggle_image function* *155*
▶*egl_toggle_image_callback function* *156*
▶*egl_toggle_image_set_on function* *157*
▶*egl_release_toggle_image function* *158*
▶ *Toggle Image Button Example.* *159*
LABEL 160
▶*egl_create_label function* *161*
▶*egl_label_callback function* *162*
▶*egl_label_set_text function* *163*
▶*egl_label_set_redraw_bg function* *164*
▶*egl_label_set_color function* *165*
▶*egl_release_label function* *166*
▶ *Label Example.* *167*
CHECK BUTTON / RADIO BUTTON 168
▶*egl_create_checkbutton function* *169*
▶*egl_checkbutton_callback function* *170*
▶*egl_checkbutton_set_check function* *171*
▶*egl_checkbutton_get_check function* *172*
▶*egl_checkbutton_set_style function* *173*
▶*egl_release_checkbutton function* *174*
▶ *Check / Radio button Example.* *175*
PROGRESS BAR 177
▶*egl_create_progressbar function* *178*
▶*egl_progressbar_set_barcolor function* *179*
▶*egl_progressbar_set_bgcolor function* *180*
▶*egl_progressbar_set_textcolor function* *181*
▶*egl_progressbar_set_text function* *182*
▶*egl_progressbar_set_pos function* *183*
▶*egl_progressbar_get_pos function* *184*
▶*egl_release_progressbar function* *185*
▶ *Progress Bar Example.* *186*
SCROLL BAR 187
▶*egl_create_scrollbar function* *189*
▶*egl_scrollbar_callback function* *190*
▶*egl_scroll_set_position function* *191*
▶*egl_scroll_get_position function* *192*
▶*egl_scroll_set_totalcount function* *193*
▶*egl_scroll_get_totalcount function* *194*
▶*egl_scroll_set_viewcount function* *195*
▶*egl_scroll_get_viewcount function* *196*
▶*egl_scroll_set_upcount function* *197*
▶*egl_scroll_get_upcount function* *198*
▶*egl_scroll_set_bgcolor function* *199*
▶*egl_scroll_set_size function* *200*
▶*egl_release_scrollbar function* *201*
▶ *Scroll Bar Example.* *202*
SLIDER 203
▶*egl_create_slider function* *205*
▶*egl_slider_callback function* *206*
▶*egl_slider_set_pos function* *207*
▶*egl_slider_get_pos function* *208*
▶*egl_slider_set_range function* *209*
▶*egl_slider_get_range function* *210*
▶*egl_slider_stepit function* *211*
▶*egl_slider_set_tick_frequency function* *212*

▶*egl_slider_set_tick_style function*....213
▶*egl_slider_set_thumb_size function*....214
▶*egl_slider_get_thumb_size function*....215
▶*egl_slider_set_barcolor function*....216
▶*egl_slider_set_tickcolor function*....217
▶*egl_slider_set_transparent function*....218
▶*egl_release_slider function*....219
▶ *Slider Example*....220
LIST BOX....221
▶*egl_create_listbox function*....223
▶*egl_listbox_callback function*....224
▶*egl_listbox_additem function*....225
▶*egl_listbox_delitem function*....226
▶*egl_listbox_delitem_text function*....227
▶*egl_listbox_delitem_num function*....228
▶*egl_listbox_alldelitem function*....229
▶*egl_listbox_ get_all_itemlist function*....230
▶*egl_listbox_ get_sel_item function*....231
▶*egl_listbox_ get_ multiple_sel_itemlist function*....232
▶*egl_listbox_ set_bgcolor function*....233
▶*egl_listbox_ set_selbgcolor function*....234
▶*egl_listbox_ set_textcolor function*....235
▶*egl_listbox_ set_seltextcolor function*....236
▶*egl_listbox_ set_textalign function*....237
▶*egl_listbox_ set_scrollwidth function*....238
▶*egl_listbox_change_item_text function*....239
▶*egl_listbox_change_item_num function*....240
▶*egl_release_listbox function*....241
▶ *List box Example.*....242
CIRCLE GAUGE....245
▶*egl_create_circle_gauge function*....246
▶*egl_circle_gauge_set_value function*....248
▶*egl_circle_gauge_get_value function*....249
▶*egl_release_circle_gauge function*....250
▶ *Circle Gauge Example*....251
BAR GAUGE....253
▶*egl_create_bar_gauge function*....254
▶*egl_bar_gauge_set_value function*....256
▶*egl_bar_gauge_get_value function*....257
▶*egl_release_bar_gauge function*....258
▶ *Bar Gauge Example*....259
PICTURE....261
▶*egl_create_picture function*....262
▶*egl_picture_callback function*....263
▶*egl_picture_set function*....264
▶*egl_release_picture function*....265
▶ *Pixture Example.*....266
ANIMATION....267
▶*egl_create_animation function*....268
▶*egl_release_animation function*....270
▶ *Animation Example*....271
CUSTOM OBJECT....272
▶*egl_create_custom_object function*....273
▶ *Custom Example.*....275
MESSAGEBOX....277
▶*egl_show_messagebox function*....278

▶ *MessageBox Example* *280*
EGL FONT 282
▶*egl_get_font function* *283*
▶*egl_set_font function* *284*
▶*egl_font_set_bkmode function* *285*
▶*egl_font_get_bk_color function* *286*
▶*egl_font_set_bk_color function* *287*
▶*egl_font_get_color function* *288*
▶*egl_font_set_color function* *289*
▶*create_bitfont function* *290*
▶*release_bitfont function* *291*
▶*create_bmpfont function* *292*
▶*bmfont_release function* *293*
▶*draw_text function* *294*
▶*draw_text_pivot function* *295*
▶*draw_text_len function* *296*
▶*draw_text_in_box function* *297*
▶*text_width function* *299*
EGL PRIMITIVES 300
▶*draw_line function* *302*
▶*draw_hline function* *303*
▶*draw_vline function* *304*
▶*draw_thickline function* *305*
▶*draw_rect function* *306*
▶*draw_rectfill function* *307*
▶*draw_rectfill_gradient function* *308*
▶*draw_rectfill_h_gradient function* *309*
▶*draw_rectfill_v_gradient function* *310*
▶*draw_roundrect function* *311*
▶*draw_roundrectfill function* *312*
▶*draw_arc function* *313*
▶*draw_pie function* *314*
▶*draw_piefill function* *315*
▶*draw_ellipse function* *316*
▶*draw_ellipsefill function* *317*
▶*draw_circle function* *318*
▶*draw_circlefill function* *319*
▶*draw_bezier function* *320*
▶*draw_polyline function* *322*
▶*draw_polygon function* *323*
▶*draw_polygonfill function* *324*
EGL ETC. 325
▶*egl_init function* *326*
▶*egl_show_object function* *327*
▶*egl_object_set_redraw function* *328*

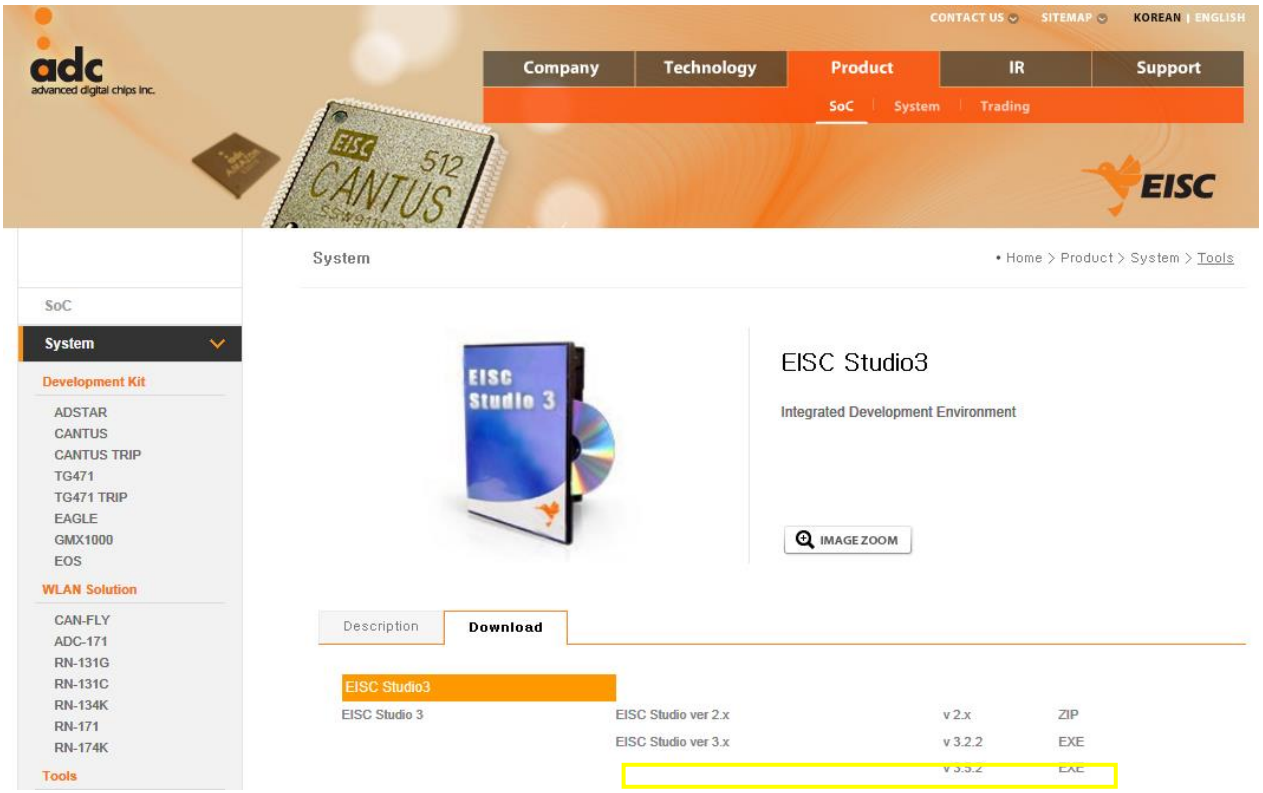
14. DEBUGGING 329

14.1 Preparing debugging *329*
14.2 Debugging *329*

1. Software Development Environment

This chapter explains how to establish development environment of AMAZONⅡ based software. ADChips provides IDE called as EISC Studio3 that supports program code management, compilation, downloading and debugging. Therefore, users can set their development environment up easily by installing EISC Studio3 installation file from an ADChips' product downloading page. EISC Studio3 is compatible with Windows XP or higher.

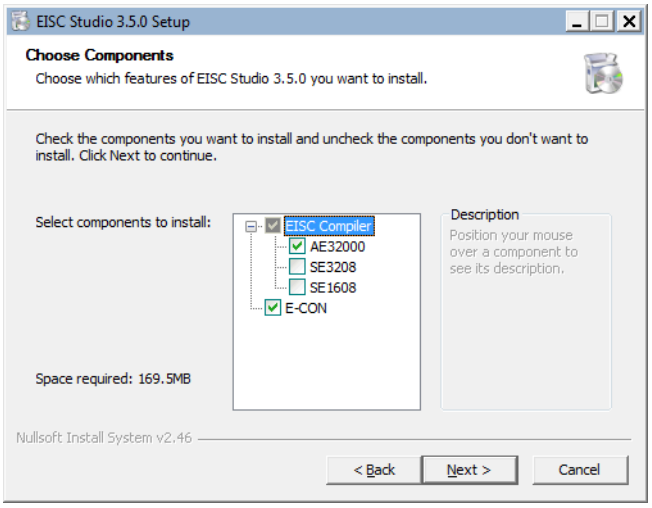
1. Download EISC Studio ver 3.x – v3.6.4 from (http://www.adc.co.kr), Product → System → EISC Studio3 → Download.
   (Refer to an installation and usage guide on the page.)



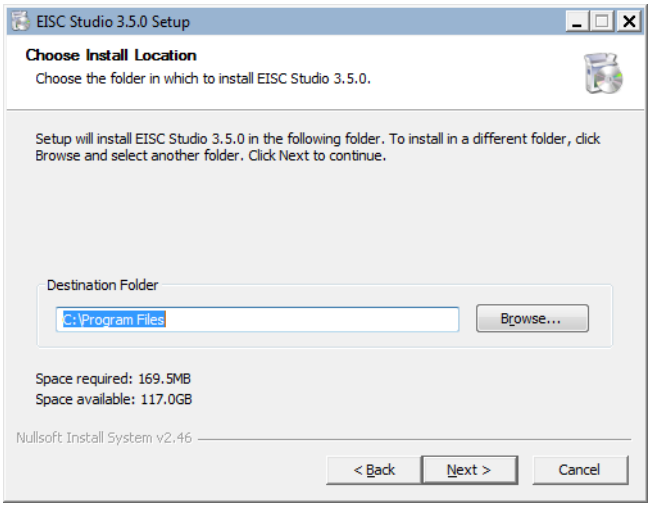
2. After executing downloaded file, ES3_setup_v3.6.4.exe then an installation window will be come up. To continue installation procedures, click 'Next'. Then, users should choose an appropriate E-Con driver and a compiler from a component selection window as below.
   Since, AMAZONⅡ contains AE32000 processor[1], user should choose AE32000 compiler. If a user is trying to use E-Con[2] device first time, then install the E-Con Driver by choosing E-Con. (If use window8, don't't check E-Con. Instead, refer to the window8_64bit_device

[1]ADChips developed 32bit Processor. (EISC Architecture)

_driver_en.pdf in pc-util folder)

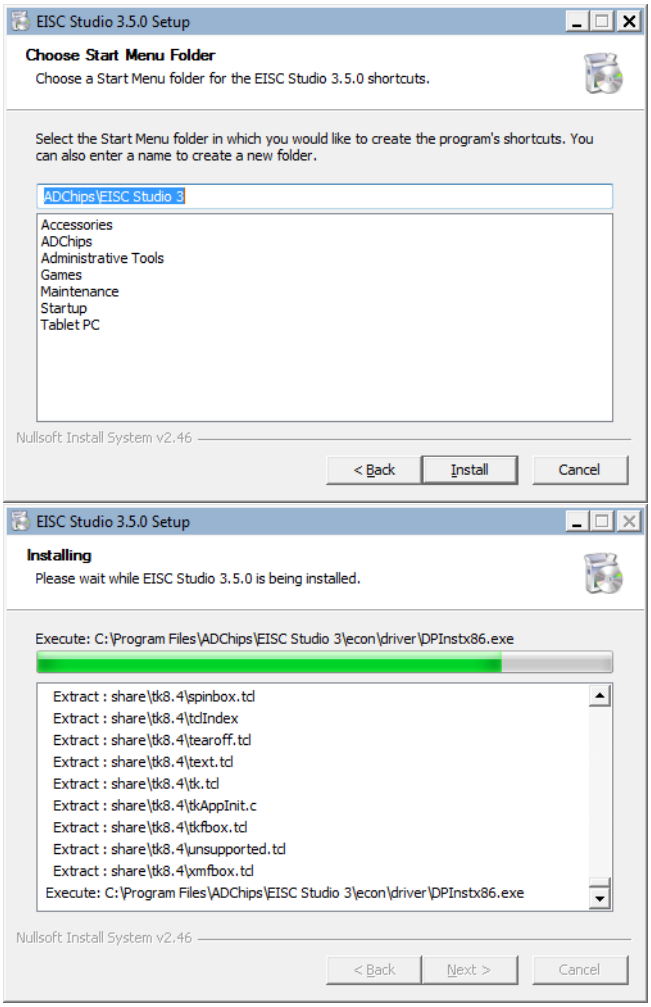


3. Click 'Next' after selecting the compiler and E-Con Driver. Then, user can decide a folder where they will be installed. A default folder name is "C:₩Program Files".
(Caution: when using 64bit OS, the default folder name is "C:₩Program Files(x86)". User must change the default folder name as c:₩Program Files.)

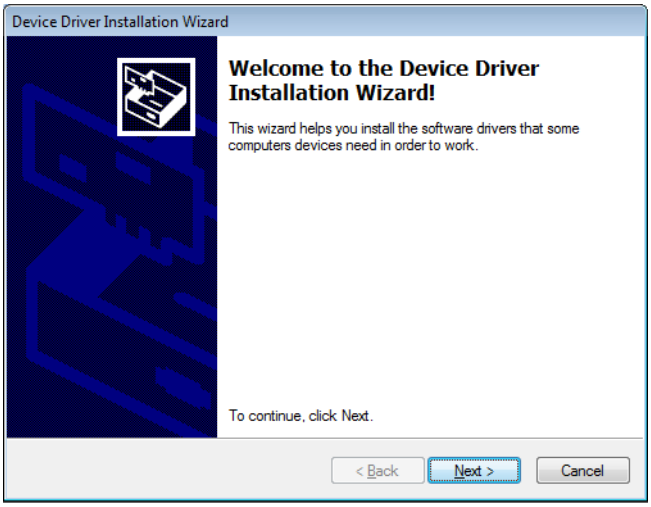


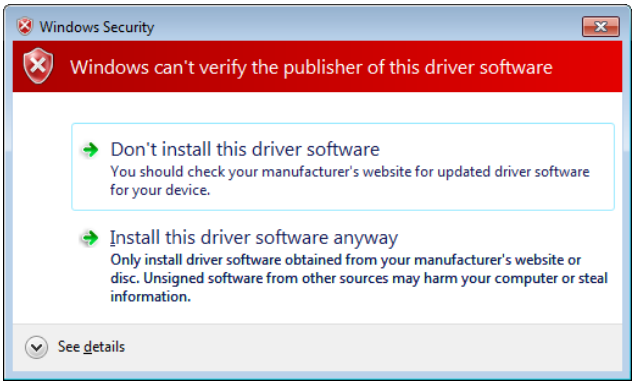
4. Below windows will be come up when click 'Next'. The chosen components are installed when click 'install'.

[2]Device for a program download and debugging by connecting with adStar.

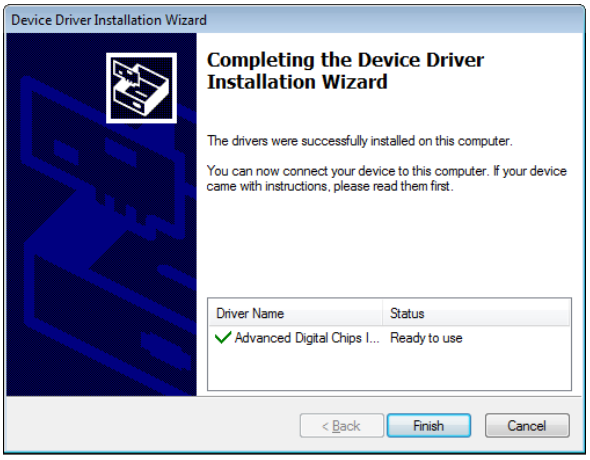


5. If E-Con is selected during the component selection procedure, following window for the E-Con driver installation is displayed. Clicking 'Next', start the driver installation. If Windows alarms secure warning, then choose "install this driver software".
   (Caution: E-Con should be connected with a computer unless the installation procedures could encounter problems..)

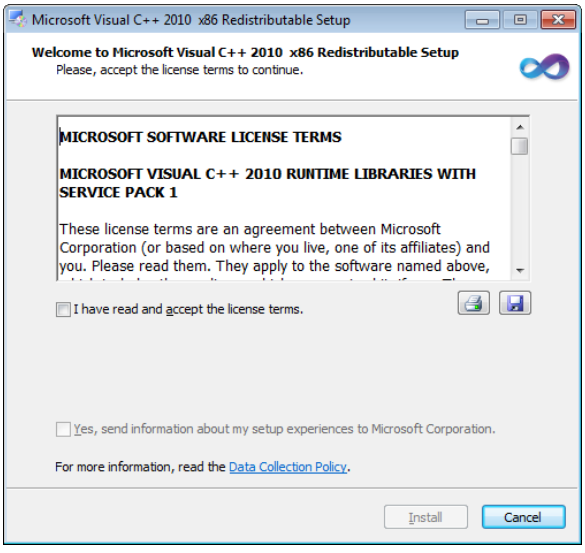




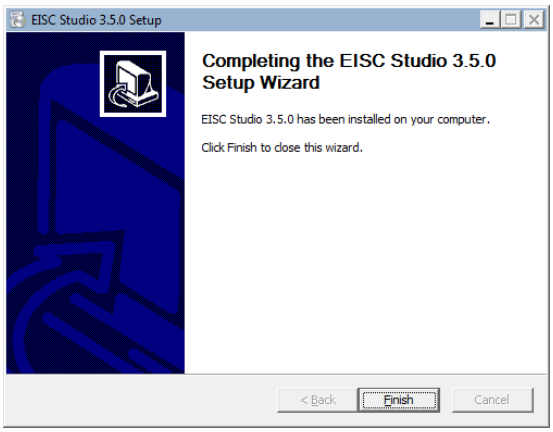
6. The end of the device driver installation loads below window, user can continue remained installation procedures of EISC Studio3 by clicking 'Next'.



7. If Microsoft Visual C++ 2010 Redistribute Package installation window come up, then check 'agree' and continue installation. It is necessary for executing EISC Studio3.

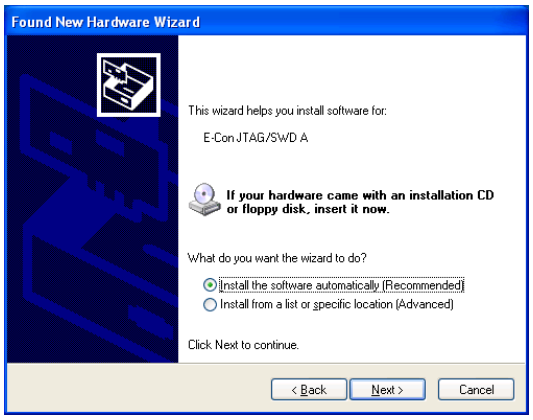


8. After finishing the installation, below window will be displayed. Clicking 'Next', installation will complete.

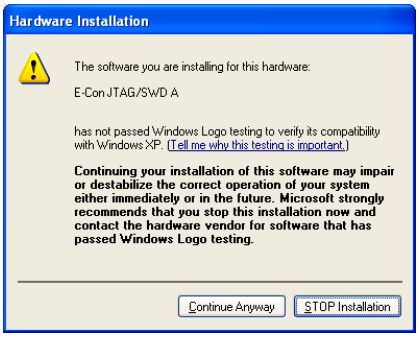


9. After installing EISC Studio 3, a download device; the E-Con driver should be installed. If the E-Con driver files are copied during the EISC Studio 3 installation, then install the E-Con driver by connecting E-Con with a computer. If not, copy the driver files by executing a DPInstx86.exe (32bit) or a DPInstx64 (64bit) in the econ₩driver folder and install the E-Con driver by connecting E-Con with a computer. If you use window 8, will install after refer Windows8_64bit_device_driver.pdf in pc-util folder.
   (DPInstx86.exe or DPInstx64.exe execution must be done with no connections between E-Con and a computer.)

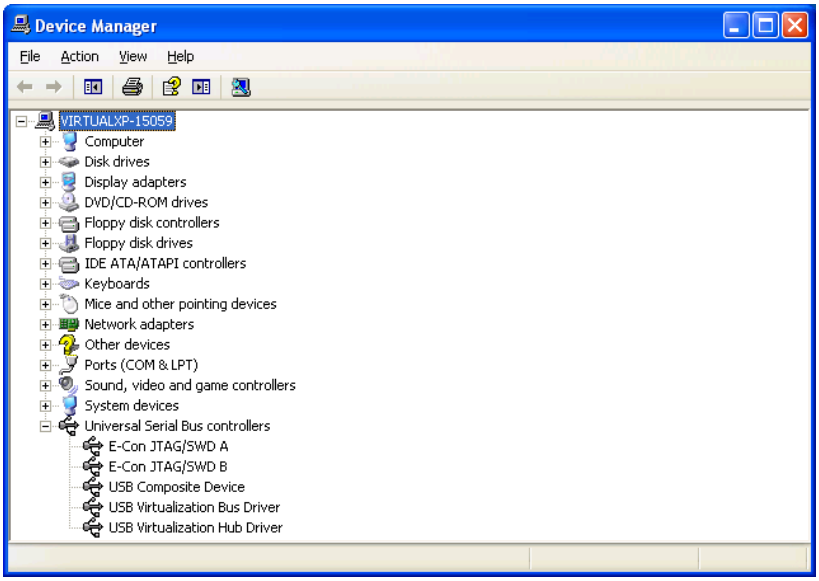
10. The E-Con driver files are copied and E-Con is connected with a computer, then the driver will be installed automatically in case of Windows 7. However, lower than Windows XP loads 'new hardware search wizard' window. Then, just check "install the software automatically" and click 'Next'.



During the installation, below warning messages could be ignored. Just keep clicking 'Next'.



11. E-Con consists of two channels, A and B. The B channel installation is same as what we explained above. User can check the driver installation result from the device manager like below. Refer to (www.adc.co.kr) Product→System→E-Con→Download→"E-Con Driver Install Guide" for more detailed information about the E-Con driver installation.
(If use window 8, refer to window8_64bit_device_driver_en.pdf in pc-util folder.)



2. Download

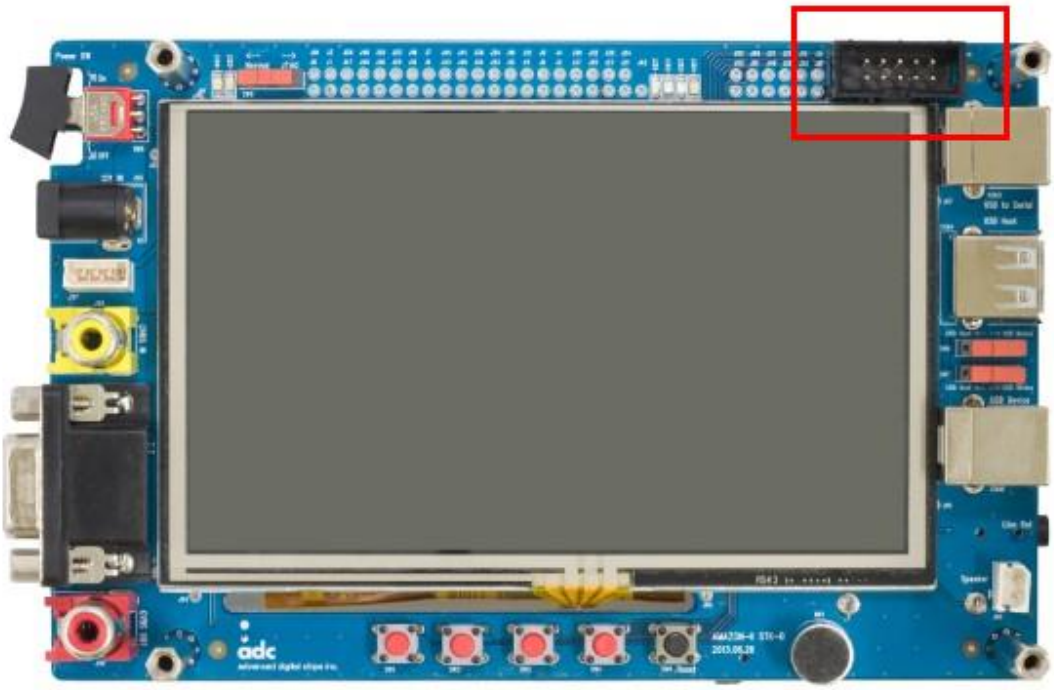
Download way is two. One way is that using E-Con. Other way is that using USB device.

USB device using method is described in Chapter "3. Using the SDK". In this chapter, I will describe using E-Con. In order to use E-Con download, need to install E-Con driver.

(Driver install method is described in Chapter "1.Software Development Enviroment")

Also, need EConMan program in EISC Studio 3 folder. (Path : c:₩ADChips₩EISC Studio 3₩econ)

Connect E-Con to red box on follow image.



Three area that can download using E-Con.

1. Serial Flash
2. NAND Flash.
3. Ram

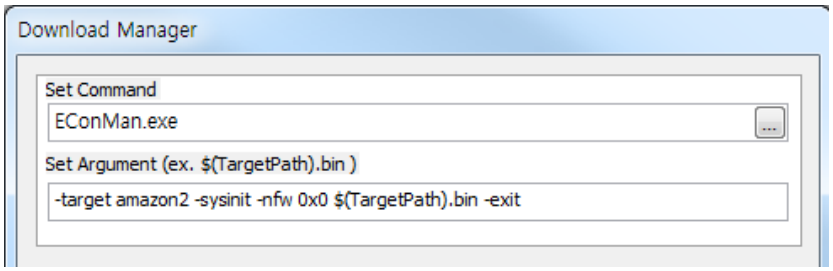
1) NAND Flash Download

In order to boot using NAND Flash, need to download Nand bootcode and bootloader using E-Con. Download method is following.

To use nand booting, need to set the config pin according to NAND Flash type.

AMAZONII STK board boot mode was fixed to fit the 2G08 NAND Flash.

1. Create binary file by build. (Run Build → Build Project)
2. Connect E-Con and board. Power on.
3. Run Build → Download option. Set 'Set Command' as the the E-Con download program. EConMan.exe that exists in EISC Studio 3 installed folder – econ folder. (Can use the default value 'EConMan.exe'.). 'Set Argument' as"-target amazon2 –sysinit –nfw 0x0 $(TargetPath).bin –exit' and click 'OK'
4. Run Build → Download to Target. binary file is downloaded in NAND Flash block 0.



Argument indicates,

-**target** is target name to download, amazon2

-**sysinit** is initialization command for download.

-**nfw** is command to download NAND Flash the build generated bin file, 0x0 is download address and $(TargetPath).bin is download file name. $(TargetPath).bin indicates a bin file of the currently opened project.

-**exit** is command to exit automatically after the download is finished.

(Refer to www.adc.co.kr → Product → System → E-Con → Download → E-Con.pdf file for detailed information of 'set arguments')

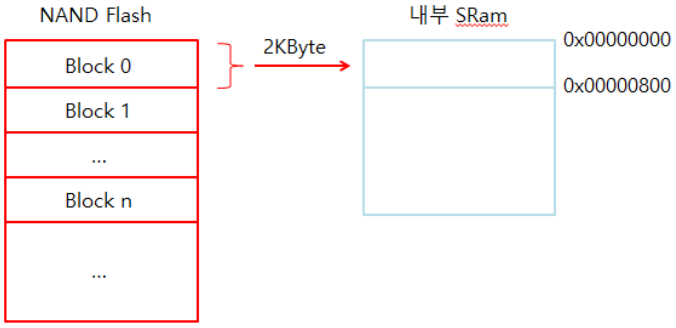
Now until, explained that download to NAND 0 block. NAND bootcode is downloaded to 0 block. But bootloader is downloaded to 1 block. Because NAND is written as block unit on EConMan.exe, should write 1 block start address in order to download bootloader.

1 block address, small page = 0x4000
large page = 0x20000

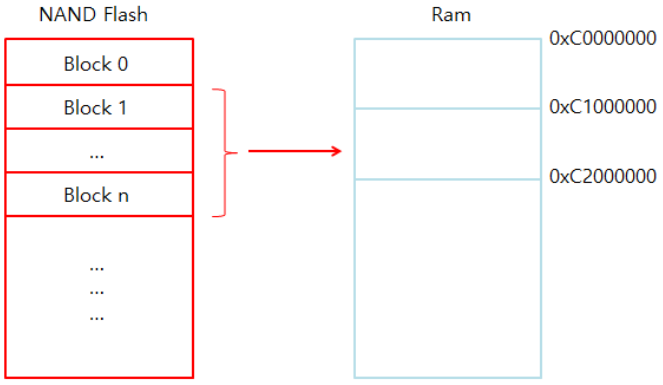
AMAZON2 STK board is embedded large page NAND Flash. Refer to 3.3 NAND Flash boot of detailed description about Bootloer download.

Boot sequence of NAND Boot mode is like follow.

1. After copy data on 2kbytes size from NAND Flash 0 block to SRam. Run program in SRam. So, NAND Boot code size has to be under 2Kbytes.



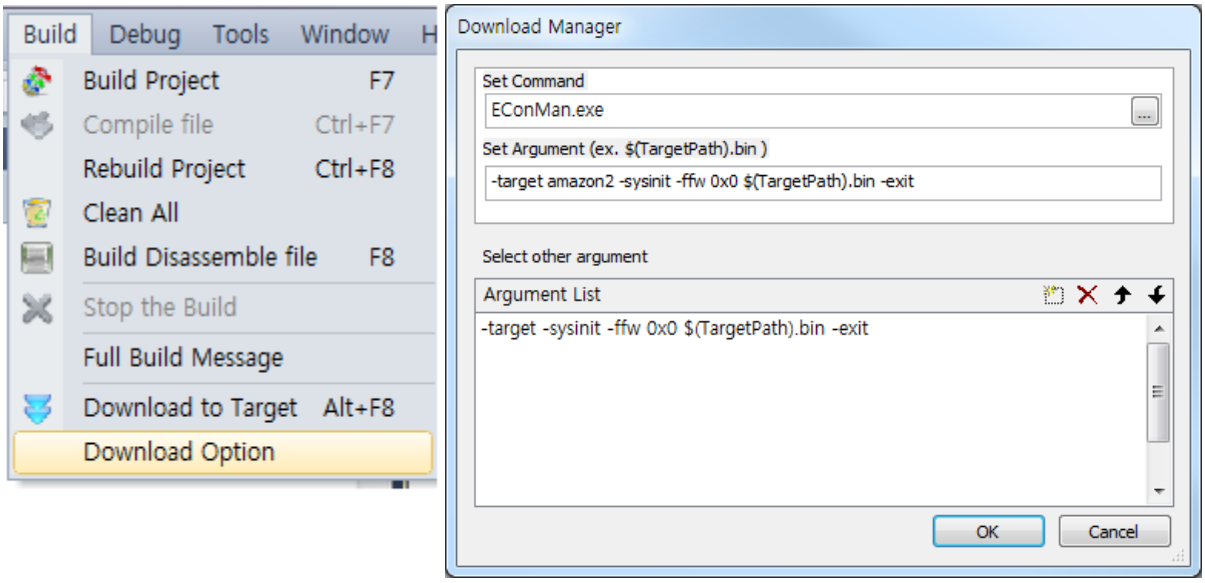
2. On NAND Boot code, copy bootloader data from NAND Flash 1block to 0xc1000000 of Ram area. Then Run bootloader in 0xc1000000.



2) Serial Flash Download

In order to run bootloader or application at serial flash, need to download serial flash.
Download method is following.
(Note that AMAZON2 STK board setting is fixed to the NAND Booting. So, if you use AMAZON2 STK board, refer next chapter 1) NAND Flash Download or 3.4 NAND Flash boot.)

1. Create binary file by build. (Run Build → Build Project)
2. Connect E-Con and board. And power on.
3. Run Build → Download option. Set 'Set Command' as the the E-Con download program. EConMan.exe that exists in EISC Studio 3 installed folder – econ folder. (Can use the default value 'EConMan.exe'.). 'Set Argument' as"-target amazon2 –sysinit –ffw 0x0 $(TargetPath).bin –exit' and click 'OK'
4. Run Build → Download to Target. binary file is downloaded in address 0.



Argument indicates,

-**target** is target name to download, amazon2

-**sysinit** is initialization command for download.

-**ffw** is command to download the build generated bin file, 0x0 is download address and $(TargetPath).bin is download file name. $(TargetPath).bin indicates a bin file of the currently opened project.

-**exit** is command to exit automatically after the download is finished.

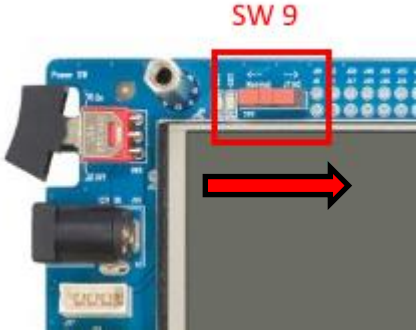
(Refer to www.adc.co.kr ➔ Product ➔ System ➔ E-Con ➔ Download ➔ E-Con.pdf file for detailed information of 'set arguments')

3) Ram Download

At this chapter, explain method that downloading ram. If use this method, can run program in ram without bootloader.

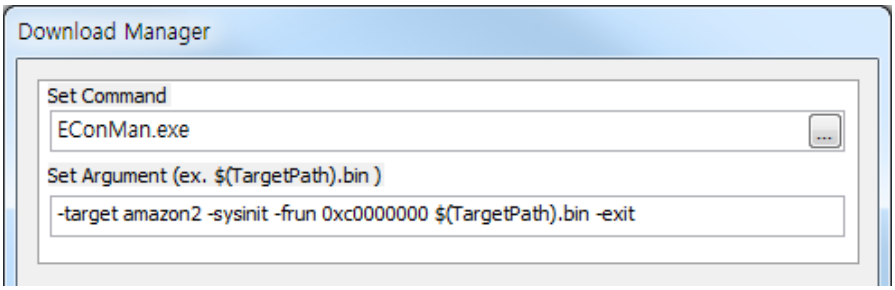
Power on in JTAG debug mode. (SW9 → JTAG) And continue following flow.

(need to use amazon2_ram.ld as Linker Script file)



1. Create binary file by build. (Run Build → Build Project)

2. Connect E-Con and board. Power on.
3. Run Build → Download option. Set 'Set Command' as the the E-Con download program. EConMan.exe that exists in EISC Studio 3 installed folder – econ folder. (Can use the default value 'EConMan.exe'.). 'Set Argument' as"-target amazon2 –sysinit –frun 0xC0000000 $(TargetPath).bin –exit' and click 'OK'
4. Run Build → Download to Target. binary file is downloaded in 0xC0000000 (Ram area) and run.



Argument indicates,

-**target** is target name to download, amazon2

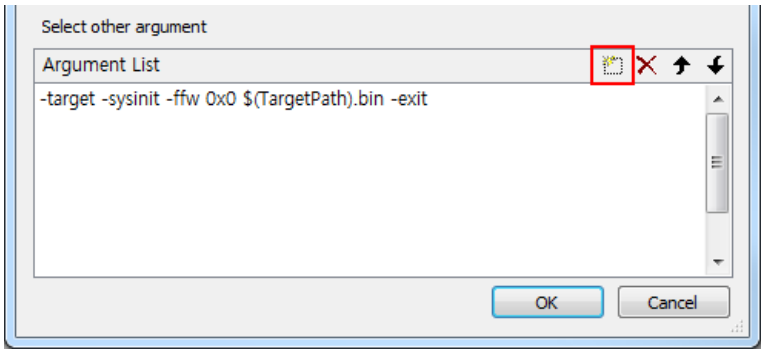
-**sysinit** is initialization command for download.

-**frun** is command to download Ram the build generated bin file and run, 0xC0000000 is download address and $(TargetPath).bin is download file name. $(TargetPath).bin indicates a bin file of the currently opened project.

-**exit** is command to exit automatically after the download is finished.

(Refer to www.adc.co.kr → Product → System → E-Con → Download → E-Con.pdf file for detailed information of 'set arguments')

4) Select other argument



You can change argument as easy using Select other argument. After saving argument, click argument in order to change argument. If click icon in red box, can add argument.

◈ Config pin and Boot mode.

Boot mode is determined by config pin 0 ~ 3.

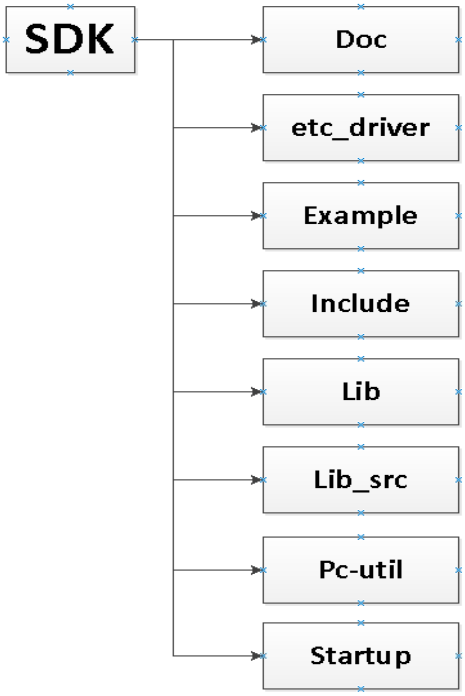
JTAG mode / Nomal mode are determined by config pin 0, and boot memory is determined by the config pin 1, 2, 3.

boot Mode	Config 1	Config 2	Config 3
NAND Small 3cycle	Low	Low	Low
NAND Small 4cycle	High	Low	Low
NAND Large 4cycle	Low	High	Low
NAND Large 5cycle	High	High	Low
NAND MLC 4bit	Low	Low	High
NAND MLC 24bit	High	Low	High
Serial Flash	Low	High	High

3. Using the SDK

3.1 AMAZONⅡ SDK composition

This shows how AMAZONⅡ SDK consists of.



Doc ➔ A folder where AMAZONⅡ related documents (including this) are located.

Example ➔A folder where AMAZONⅡ EVM Board example programs are located.
(Example -> flash_data contains images and sound files for the SDK examples.)

Include ➔AMAZONⅡ SDK's header files.

lib ➔AMAZONⅡ SDK's library files.

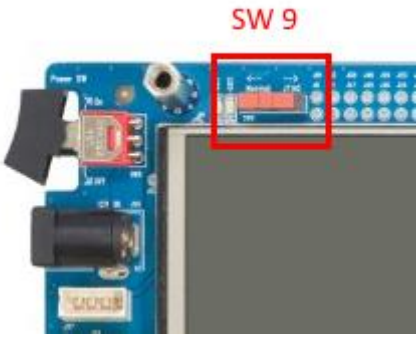
lib_src➔AMAZONⅡ SDK's library source files.
(Originally, libraries are just source files not a built library files. Thus, the libadstar.a library file must be generated in the 'lib' folder by building the adStar.epx project in the lib_src folder. To do this, just press F7 or 'build'-'build project' after opening adStar.epx.)

Pc-util➔A folder where utilities to use AMAZONⅡ are located. The adStar USB Driver exists.

Startup ➔all startup & link scrip code for the AMAZONⅡ development. Moreover, STK board basic configuration code exists.

3.2 AMAZONII Board Switch

SW9 is switch that set JTAG Debug mode at AMAZONII board.



JTAG Debug mode is boot as stop status on start address. So, you use for debugging and safety downloading.

AMAZONII board is fixed as NAND Large 5Cycle boot mode.

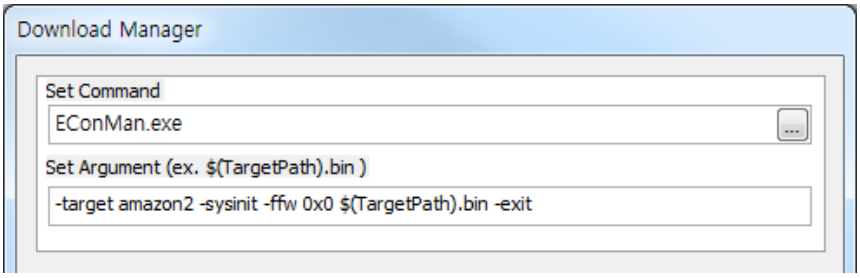
3.3 Serial Flash boot

You should set serial flash boot mode in order to use serial boot. If use serial flash boot, you can run application program using bootloader or without bootloader. It is depends on whatever Linker Script file use.

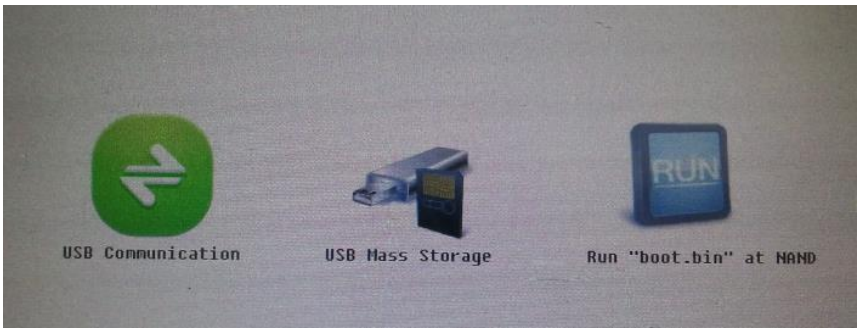
If you want to run application without bootloader, need to use amazon2.ld file. If want to run application in ram with using bootloader, need to use amazon2_ram.ld file.

3.3.1 Using Bootloader

By default, bootloader have been downloaded in EVM board. Because bootloader is in example folder, you can redownload. Open bootloader project in example folder. (double click .epx file) Run Build → Build Porject, Power on board after connect E-Con. Run Build → Download Option, and enter download command. Download bootloader in address 0 by run Download to Target in Build menu. (If fail download, retry in JTAG Debug mode.)



Boot on serial flash boot mode, after downloaded. Then, you can look folloing image in LCD.



Icons have each function.

USB Communication : Can Download application to dram by using USB device.

USB Mass Storage : Can use NAND Flash in EVM board like usb memory.

Run "boot.bin" at NAND : Can run boot.bin in NAND Flash.

3.3.1.1 USB Communication

After run USB Communication mode, connect USB cable in USB device connector. Then, find USB device. At first, you should install device driver. USB driver is in pc-util/usb_driver.

You can copy driver by executing a DPInstx86.exe(32bit) or DPInstx64(64bit) in the pc-util/usb_driver. And connect USB cable, driver will be installed automatically in case of Windows 7. Hower, lower than Windows XP loads 'new hardware search wizard' window. Then, just check "install the software automatically" and click 'NEXT'

When installation completed, you can check the driver installation result from the device manager. (device name : adchips USB device)

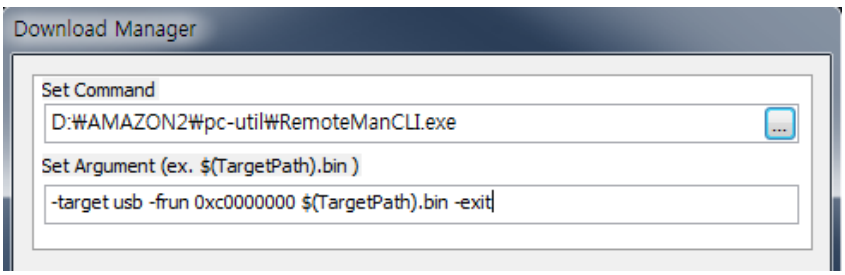
Ref : When use Windows 8, refer Window8_64bit_device_driver.pdf file in doc folger.

You can application by download on dram using USB device. Open Demo project in example folder. You should confirm Linker Script file amazon2_ram.ld

amazon2.ld : operate at serial flash.

amazon2_ram.ld : operate at dram.

Run Build → Build Project, Build → Download Option. Enter command and download binary by run Download to Target. Porgram is downloaded and run.

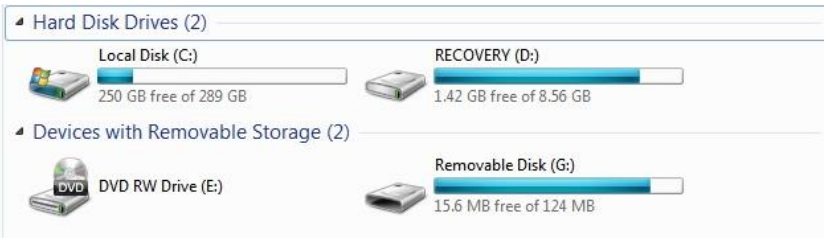


You should usd EConMan instead of RemoteManCLI program in pc-util folder.
Argument indicates,

-target is target anme to download, usb.
-frun is command to download ram the build generated bin file and run, 0xC0000000 is download address and $(TargetPath).bin is download file name. $(TargetPath).bin indicates a bin file of the currently opened project.
-exit is command to exit automatically after the download is finished.

3.3.1.2 USB Mass Storage

USB Mass Storage can use NAND Flash in EVM board like USB memory stick. When run USB Mass Storage, new storage device is appeared. This device is NAND Flash of the board, so copying files to the device indicates that the files could be used by the amazonⅡ program.
In order to operate the Demo Project should copy all image and sound files from ''example' – 'flash data'.

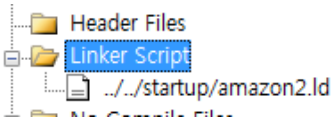


3.3.1.3 Run "boot.bin" at NAND

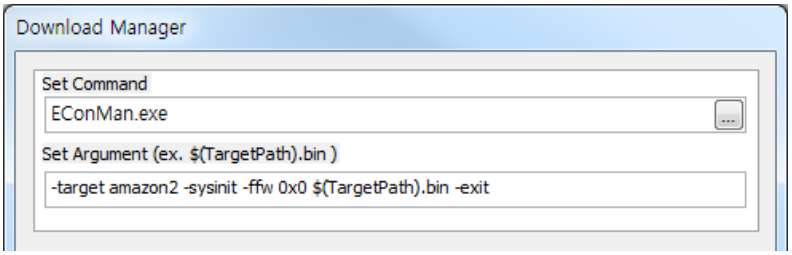
Run "boot.bin" at NAND is mode that run application in dram by copying boot.bin file in NAND Flash. After change binary file name boot.bin, copy to NAND Flash using USB Mass Storage mode. Run this mode, and application program is operated.

3.3.2 Use without Bootloader

You can run application program without bootloader. Open Demo project. After program open, change Linker Script file as amazon2.ld. By default, example is used amazon2_ram.ld. amazon2.ld file is in startup folder.



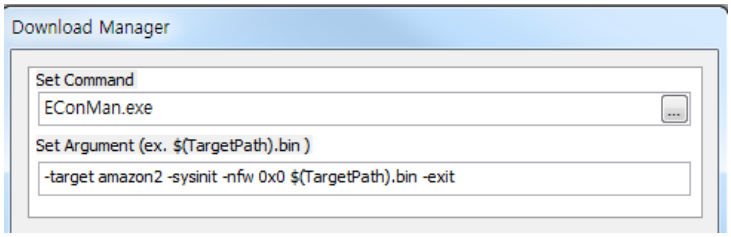
After change Linker Script file, run Build → Build Project. Run Build → Download Option, enter command like following. Run Build → Download to Target.



Reset and you can confirm operation program.
(If fail download, retry in JTAG Debug mode)

3.4 NAND Flash boot

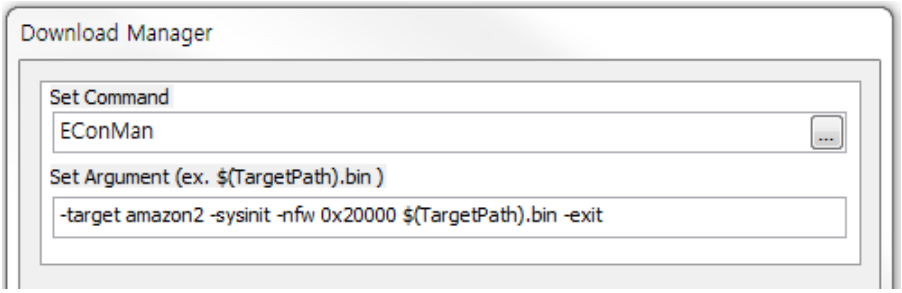
You need to download NAND boot code for NAND booting. NAND boot code is in example/NandBootCode.
Open the NandBootCode project and build. Enter like following, after run Download Option.
And run Download to Target, Nand Boot Code is downloaded in NAND Flash.



Argument indicates,
-target is target name to download, amazon2
-sysinit is initialization command for download.
-nfw is command to download NAND Flash the build generated bin file, 0x0 is download address and $(TargetPath).bin is download file name. $(TargetPath).bin indicates a bin file of the currently opened project.
-exit is command to exit automatically after the download is finished.

After download, power on board. Then Nand Boot Code is operated.
But Nand boot code operated copy binary data in NAND Flash block 1 to dram and run.
So, don't operate. I will wxplain method that download to NAND blodk 1.

Open Demo program project, run Build Project, run Download Option, enter like following.

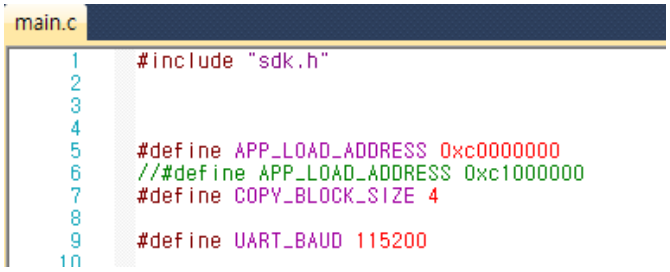


Unlike previous, download address is 0x20000 (block 1 address)
Because NAND should write as block unit, should write 1 block start address.

1 block address, small page = 0x4000
large page = 0x20000

Run Download to Target, program is downloaded. Booting on Nand Flash boot mode, you can confirm downloaded program is operating.

If you want use bootloader like Serial Flash boot mode on Nand Flash Boot mode, need to modify NandBootCode and Bootloader.



At NandBootCode, APP_LOAD_Address change 0xC0000000 to 0xC1000000.
At Bootloader, change Linker Script file. (amazon2-bootloader.ld → amazon2-bootloader_ram.ld)
And download like previously.

4. Memory settings

AMAZON2 EVM board has 128MByte DDR2 memory. 128MByte ram area is devided like following table in SDK.

0xC0000000 ~ 0xC1FFFFFF (32MByte)	text (Program code) data bss heap stack
0xC2100000 ~ 0xC2900000 (8MByte)	Frame buffer
0xC2900000 ~ 0xC7FFFFFF (87MByte)	Texture memory

0XC0000000 area is used as program memory like data, bss, heap, stack area.
Running program in ram area, program code stored in ram area.
(when use amazon2_ram.ld as Linker Script)
This area is adjusted by setting stack address in Linker Script file.

```
8   MEMORY
9   {
10      rom (rx)    : ORIGIN = 0x00000000,  LENGTH = 512K
11      /* vector table will stored at 0x18000000 by InitInterrupt()*/
12      spm(rwx)    : ORIGIN = 0x20000200,  LENGTH = (16*1024-0x200)
13      ram (!rx)   : ORIGIN = 0xC0000000,  LENGTH = 32M
14  }
```

```
79      ._stack 0xC1FFFFFC :
80      {
81        _stack = .;
82        /*PROVIDE (__stack = .);*/
83        *(._stack)
84      }
85  }
```

Above file is amazon2_ram.ld file of startup folder.
Set 0xC0000000 as ram start address and set 0xC1FFFFFC as stack pointer.
So, 32MByte ram area is used as program memory.
This area size is adjusted by setting stack pointer.

Frame buffer is area that image data is stored and displayed on screen.
Buffer cound is determined by setting single mode or double mode.
Buffer size is determined by resolution and RGB565/RGB888 mode.

One buffer is created on single buf mode.
Two buffers are created on double buf mode.

Frame buffer size is determined by following calculation.

frame buffer size = width x height x bpp

width / height < 512 → width / height = 512
width / height < 1024 → width / height = 1024
width / height < 2048 → width / height = 2048

Ex) 1024 x 600, RGB565 mode
frame buffer size = 1024 x 1024 x 2 = 2MByte

Frame buffer is created from address that set. (default 0xc2100000)

single buf / 1024 x 600 / RGB565 ➔ frame buffer count : 1 , frame buf size : 2MByte. total 2MByte
double buf / 1024 x 600 / RGB888 ➔ frame buffer count : 2 , frame buf size : 4MByte. total 8MByte

Textrure memory is area that loading image is stored. Graphic engine access this memory.
Start address and size are determined by ge_set_texture_mem( ).
Default setting value is 87MByte from 0xC2900000.
Memory in this area is allocated by the load function (like loadbmp( )) and is freed by the release_surface function.

5. UART

Function	Description
uart_config	Function to initialize UART.
uart_putch	Display the character.
uart_putdata	Display the character of specified length.
uart_putstring	Display the string.
uart_getch	Store the character at variable.
uart_getdata	Store the character of specified length at variable.
uart_rx_flush	Clear Receive FIFO.
uart_tx_flush	Clear Transmitter FIFO.
set_debug_channel	Set debugging message displaying UART channel.
get_debug_channel	Get debugging message displaying UART channel.
debugprintf	Display numbers, characters and variable.
debugstring	Display string.
PRINTLINE	Display a line number where this function is called.
PRINTVAR	Display a line number and a value where this function is called.

5.1 uart_config

```
BOOL  uart_config (
      int  ch,
      int  baud,
      UART_DATABITS  databits,
      UART_STOPBITS  stopbit,
      UART_PARITY  parity
);
```

Overview

Function to initialize UART.

Parameter

int ch	UART initial channel value.
int baud	UART baud rate value. SDK uses 115200 as default.
UART_DATABITS databits	UART transmission data bit setting value. SDK defines data bit as below. **typedefenum**{ DATABITS_5 = 5, DATABITS_6 = 6, DATABITS_7 = 7, DATABITS_8 = 8 }**UART_DATABITS**;
UART_STOPBITS stopbit	UART stop bit setting value. SDK defines stop bit as below. **Typedefenum**{ STOPBITS_1 = 1, STOPBITS_2 = 2 }**UART_STOPBITS**;
UART_PARITY parity	UART parity bit setting value. SDK defines parity bit as below. **Typedefenum**{ UART_PARNONE = 0, UART_PARODD, UART_PAREVEN }**UART_PARITY**;

Return Value

TRUE or FALSE

5.2 uart_putch

```
BOOL  uart_putch (
      int  n,
      char  ch
);
```

Overview

Display the 'ch' value through the UART 'n' channel.

Parameter

int n	Channel to display the 'ch' value.
char ch	Value to be displayed through UART.

Return Value

TRUE or FALSE

5.3 uart_putdata

```
BOOL  uart_putdata (
      int  n,
      U8*  buf,
      int  len
);
```

Overview

Display 'len' length characters stored in 'buf' through the UART n channel.

Parameter

int n	Channel to display the 'ch' value.
U8* buf	Buffer that stored characters to be displayed.
int len	Length of characters to be displayed.

Return Value

TRUE or FALSE

5.4 uart_putstring

```
BOOL  uart_putstring (
      int  n,
      U8*  buf
);
```

Overview

Display characters stored in 'buf' through the UART n channel.

Parameter

int n	Channel to display the 'n' value.
U8* buf	Buffer that stores characters to be displayed.

Return Value

TRUE or FALSE

5.5 uart_getch

```
BOOL  uart_getch (
      int  n,
      char*  ch
);
```

Overview

Store received 1 Byte value (1 character) through the UART n channel to 'ch'.

Parameter

int n	UART channel to be input the value.
char* ch	Variable to be input a value through UART.

Return Value

TRUE or FALSE

5.6 uart_getdata

```
int  uart_getdata (
     int  n,
     U8*  buf,
     int  bufmax
);
```

Overview

Read 'bufmax' length of characters through the UART n channel and store the characters to 'buf'.

Parameter

int n	UART channel to be input the value.
U8* buf	Buffer to be input the value through UART.
int bufmax	Number of values (characters) to read. (Byte unit)

Return Value

The total number of elements successfully read is returned as byte unit.

5.7 uart_rx_flush

```
void  uart_rx_flush (
     int  ch
);
```

Overview

Initialize rxfifo of the UART n channel.

Parameter

int ch	UART channel to be initialized.

Return Value

None.

5.8 uart_tx_flush

```
void  uart_tx_flush (
      int  ch
);
```

Overview

Initialize txfifo of the UART n channel.

Parameter

int ch	UART channel to be initialized.

Return Value

None.

5.9 set_debug_channel

```
void  set_debug_channel (
      int  ch
);
```

Overview

Set debugging message displaying UART channel that is used by debugging functions such as **debugprintf**, **debugstring**, **PRINTVAR and PRINTLINE** .

Parameter

int ch	Debugging purpose UART channel.

Return Value

None.

5.10 get_debug_channel

```
void  get_debug_channel ( );
```

Overview

Return the debugging purpose UART channel.

Parameter

None.

Return Value

None.

5.11 debugprintf

```
void  debugprintf (
        const char*  const format,
);
```

Overview

It has same role with C language's 'printf', displays numbers, characters and variable values through UART. The output UART channel is configured by **set_debug_channel()** function. The default debugging channel is 0.

Usage

debugprintf("result number : %d₩r₩n",result);

→Displays the variable value of 'result' in decimal number with "result number: "string through UART and breaks line. The usage is same as 'printf'. However, '₩r₩n' must be denoted when line breaking.

5.12 debugstring

```
void  debugstring (
        const char*  str
);
```

Overview

Function to display string. It is used only for string displaying. Output UART channel is the debugging channel from **set_debug_channel()** function. The default channel is 0.
Just for reference, it is possible to display only string with **debugprintf()** function.

Usage

debugstring("=== AMAZON2 Start ===₩r₩n");
→string in "" is displayed through UART.

5.13 PRINTLINE

Overview

It is a macro function, displays a line number where this function is called through UART. It uses the debugging channel. The default channel is 0.

Usage

PRINTLINE;
→ Displays a line number where this function is called.

5.14 PRINTVAR(A)

Overview

It is a macro function, displays a line number and a value of 'A' where this function is called through UART. It uses the debugging channel. The default channel is 0

Usage

int a = 10;
PRINTVAR(a);;
→ Displays a value of 'a'. Even a register value could be displayed.

5.15 UART Example

```
#include "sdk.h"

Int main()
{
      boardinit();
      uart_config(0, 115200, DATABITS_8, STOPBITS_1, UART_PARNONE );
                        // Sets the UART 0 channel as below.
                         // Baud rate = 115200
                         // Data bit = 8bit
                         // Stop bit = 1
                         // Parity = none
     debugstring("==========================================₩r₩n");
      debugprintf("AMAZON2 UART Example. System clock(%dMhz)₩r₩n",get_ahb_clock()/1000000);
      debugstring("==========================================₩r₩n");

      U8 ch;
      While(1)
      {
            if(uart_getch(0, &ch))  // Reads 1Byte data from UART 0 and stores 'ch'
            {
                  uart_putch(0, ch); // Displays UART 0 input data to UART 0
            }
      }
}
```

6. INTERRUPT

Function	Description
init_interrupt	The interrupt initialization function.
set_interrupt	Call back function registration function of the interrupt is called.
enable_interrupt	Function to enable the interrupt.

6.1 init_interrupt

```
void  init_interrupt ( );
```

Overview

Ther interrupt initialization function. It should be called to use the interrupt related function. startup_amazon2.S of the AMAZONⅡ SDK calls and initializes the interrupt.

6.2 set_interrupt

```
BOOL  set_interrupt (
      INTERRUPT_TYPE  intnum,
      void  (*fp)()
);
```

Overview

Call back function registration function of the interrupt is called.

Caution: the interrupt callback functions of UART, Sound mixer are already made and registered from SDK, no redundant operations are avoided.

Parameter

INTERRUPT_TYPE intnum	interrupt type (refer to next chapter about the interrupt type)
void (*fp)()	callback function of the interrupt.

Return Value

TRUE or FALSE

6.3 enable_interrupt

```
void  enable_interrupt (
      INTERRUPT_TYPE  intnum,
      BOOL  b
);
```

Overview

Function to enable the interrupt. Register the interrupt function with **set_interrupt()** and enable it with **enable_interrupt()**.

Parameter

INTERRUPT_TYPE intnum	interrupt type (refer to next chapter about the interrupt type)
BOOL b	1 or TRUE => interrupt enable
	0 or FALSE => interrupt disable

Return Value

TRUE or FALSE

INTERRUPT_TYPE

INTERRUPT_TYPE
INTNUM_EIRQ0
INTNUM_TIMER0
INTNUM_SEIP
INTNUM_DMA0
INTNUM_FRAMESYNC
INTNUM_GPIO0
INTNUM_UART0
INTNUM_XDMA0
INTNUM_EIRQ1
INTNUM_TIMER1
INTNUM_PMU
INTNUM_DMA1
INTNUM_ICE
INTNUM_GPIO1
INTNUM_UART1
INTNUM_XDMA1
INTNUM_TIMER2
INTNUM_USBDEV
INTNUM_TWI
INTNUM_USBHOST
INTNUM_I2SRX1
INTNUM_UART2
INTNUM_SEIPRX
INTNUM_WATCHDOG
INTNUM_NAND
INTNUM_DMA2
INTNUM_SDHC
INTNUM_I2STX1

INTERRUPT_TYPE
INTNUM_SPI0
INTNUM_SOUNDMIXER
INTNUM_GPIO2
INTNUM_SPI1
INTNUM_TIMER3
INTNUM_UART3
INTNUM_GPIO3
INTNUM_MJPEG_FULL
INTNUM_DMA3
INTNUM_XDMA2
INTNUM_GPIO4
INTNUM_MJPEG_DEC_END
INTNUM_GPIO5
INTNUM_XDMA3
INTNUM_GPIO6
INTNUM_XDMA4
INTNUM_XDMA5
INTNUM_GPIO7
INTNUM_XDMA6
INTNUM_TIMER_CAP_OVER
INTNUM_GPIO8
INTNUM_XDMA7
INTNUM_GPIO9
INTNUM_GPIO10
INTNUM_GPIO11
INTNUM_GPIO12
INTNUM_GPIO13
INTNUM_GPIO14
INTNUM_GPIO15

6.4 Interrupt Example

```
#include "sdk.h"

void EIRQ0ISR( )
{
	debugprintf("EIRQ0 Interrupt₩r₩n");
}

int main()
{
	boardinit();
	uart_config(0, 115200, DATABITS_8, STOPBITS_1, UART_PARNONE );
	set_interrupt( INTNUM_EIRQ0, EIRQ0ISR );
	//Registers a callback function of the interrupt when External Interrupt 0 occurred.
	enable_interrupt( INTNUM_EIRQ0, TRUE );
	//Enables External Interrupt 0.

	while(1)
	return 0;
}
```

7. TIMER

Function	Description
set_timer	Function to configure and opeate the timer.
stop_timer	Function to stop timer.
delayms	Function to delay.

7.1 set_timer

```
BOOL  set_timer (
      int  nCh,
      U32  ms
);
```

Overview

Function to configure and operate the nCh channle timer. Determines the timer channel and period to be used and enables the timer operations at timer control register.
When this function is called after the timer interrupt is registered with **set_interrupt()** function, the timer interrupt occurred with the configured period.

Parameter

int nCh	Configured timer channel value.
U32 ms	timer interrupt period. (ms unit)

Return Value

TRUE or FALSE

7.2 stop_timer

```
BOOL  stop_timer (
      int  nCh
);
```

Overview

Stops the nCh channel timer. Disables the nCh channel timer operations.

Parameter

int nCh	Timer channel value to be disabled.

Return Value

TRUE or FALSE

7.3 delayms

```
BOOL  delayms (
      U32  ms
);
```

Overview

Takes millisecond delay. (ms unit)

Parameter

U32 ms	Time amount of delay, ms unit.

Return Value

TRUE or FALSE

7.4 TIMER Example

```
#include "sdk.h"

void TIMER0ISR( )
{
        debugprintf("==TIMER0ISR==₩r₩n");
}

int main()
{
        boardinit();
        uart_config(0, 115200, DATABITS_8, STOPBITS_1, UART_PARNONE );
        set_interrupt( INTNUM_TIMER0, TIMER0ISR );
                        //Register a callback function of the interrupt of Timer 0 Interrupt.
        set_timer( 0, 1000);
                        // Timer0 interrupt occurred every 1 second.
        delayms(5000);
                        // 5 seconds delay.
        stop_timer( 0 );
                        // Disables Timer 0 Interrupt.  No more timer.
        while(1)
        return 0;
}
```

8. GRAPHIC

Function	Description
ge_set_texture_mem	Set texture memory size and area.
get_ge_target_buffer	Get frame buffer that graphic engine accessed.
setscreen	Set LCD resolution and RGB mode.
get_front_buffer	Get front frame buffer.
get_back_buffer	Get back frame buffer.
get_screen_width	Get screen width value.
get_screen_height	Get screen height value.
get_screen_pitch	Get screen pitch value.
get_screen_bpp	Get screen bpp(bit per pixel) value.
flip	Function to switchover between frames.
flip_wait	Function to switchover between frames. Wait that switchover is completed.
async_flip	Function to switchover between frames. Perform async switchover.
async_flip_wait	Function to switchover between frames. Perform async switchover and Wait that switchover is completed.
ge_wait_cmd_complete	Wait that graphic engine operation is completed.
ge_draw_rectfill	Draw filled rectangle by graphic engine.
ge_draw_rectfillalpha	Draw filled rectangle with transparency effect by graphic engine.
surface_set_alpha	Set surface alpha value.
draw_line	Draw line.

Function	Description
draw_rect	Draw rectangle.
draw_rectfill	Draw filled rectangle.
draw_roundrect	Draw round rectangle.
draw_roundrectfill	Draw filled round rectangle.
draw_circle	Draw circle.
draw_circlefill	Draw filled circle.
draw_ellipse	Draw ellipse.
draw_ellipsefill	Draw filled ellipse.
loadbmp	Load BMP image file.
loadbmpp	Load BMP image from memory.
loadjpg	Load JPG image file.
loadjpgp	Load JPG image from memory.
loadpng	Load PNG image file.
loadpngp	Load PNG image from memory.
loadsurf	Load SURF image file.
draw_surface	Display the image that loaded.
draw_surface_rect	Display the specific region of an image.
draw_surface_scale	Display the image by scaling (scale up or scale down).
draw_surface_scalerect	Display the specific region of an image by scaling.
Draw_surface_rotate	Draw rotated image.

Function	Description
release_surface	Release the memory region that image load function created.
draw_set_clip_window	Set limit region that an image could be displayed.
osd_set_surface	Set OSD coordinate and region. And enable OSD.
osd_set_surface_with_alpha	Set OSD coordinate and region (with alpha effect).
osd_set_pal	Set palette of OSD.
osd_set_off	Disable OSD.
OSD_MOVE	Move OSD coordinate.
vga_mode_on	Set VGA (analog RGB) output mode. Must call this function in order to use VGA output mode.
vce_on_surface	Call function when use external video. External video data stored in specified SURFACE.
loadjpg_hw	Load jpg image file using H/W decoder
savebmp	Save the screen as bmp file.
savejpg	Save the screen as jpg file.
surface_set_alpha	Set alpha value of SURFACE.

8.1 ge_set_texture_mem

```
void  ge_set_texture_mem (
      U32  addr,
      U32  size
);
```

Overview

Set texture memory size and area. Texture memory is memory used by graphic engine.

Parameter

U32 addr	Texture memory start address.
U32 size	Texture memory size.

Return Value

None.

8.2 get_ge_target_buffer

```
SURFACE*  get_ge_target_buffer ( );
```

Overview

Get frame buffer that graphic engine accessed.

Parameter

None.

Return Value

SURFACE pointer of frame buffer that graphic engine accessed.

8.3 setscreen

```
void  setscreen (
      U32   framebufferAddr,
      SCREENRES   size,
      U32   scmode
);
```

Overview

Set LCD resolution and RGB mode.
It should be set first for LCD displaying.

Parameter

U32 framebufferAddr — Frame buffer start address.

SCREENRES size — LCD resolution value, the LCD Controller is configured according to the resolution value. SCREENRES are defined as below. (Each LCD has different characteristics. Thus, if LCD is not working correctly with the proper resolution then the values in **crt.c**'s **setscreen()** function should be modified by the LCD characteristics.)

```
Typedefenum {
                SCREEN_480x272 = 0,
                SCREEN_640x480,
                SCREEN_800x480,
                SCREEN_800x600,
                SCREEN_1024x768,
                SCREEN_TV_HD,
                SCREEN_TV_NTSC,
                SCREEN_TV_PAL,
               } SCREENRES;
```

U32 scmode — RGB mode setting value, defined as below.

SCREENMODE_RGB888
SCREENMODE_RGB565
SCREENMODE_SINGLEBUF
SCREENMODE_DOUBLEBUF

Select one of RGB888 and RGB565.
Select one of SINGLEBUF and DOUBLEBUF.

Return Value

None

8.4 get_front_buffer

```
SURFACE*  get_front_buffer ( );
```

Overview

Get SURFACE pointer of current screen frame. (front buffer)

Parameter

None.

Return Value

SURFACE pointer of current screen frame.

8.5 get_back_buffer

```
SURFACE*  get_back_buffer ( );
```

Overview

Get SURFACE pointer of hidden screen frame. (back buffer or rendering buffer)

Parameter

None.

Return Value

SURFACE pointer of back buffer.

8.6 get_screen_width

U32 get_screen_width ( );

Overview

Get current screen width size.

Parameter

None.

Return Value

Width size of screen.

8.7 get_screen_height

U32 get_screen_height ( );

Overview

Get current screen height size.

Parameter

None.

Return Value

Height size of screen.

8.8 get_screen_pitch

U32 get_screen_pitch ( );

Overview

Get pitch value of screen. Pitch value is changed by bpp and width.

width <= 512 → 512 * bpp / 8

width <= 1024 → 1024 * bpp / 8

width <= 2048 → 2048 * bpp / 8

Parameter

None.

Return Value

Pitch value of screen.

8.9 get_screen_bpp

```
U32  get_screen_bpp ( );
```

Overview

Get bpp(bit per pixel) value of screen.

Parameter

None.

Return Value

bpp value of screen.

8.10 flip

```
void  flip ( );
```

Overview

Switch between frame when use double frame buffer.
If you want to use double frame buffer, Set SCREENMODE_DOUBLEBUF in scmode argument in setscreen function.

Parameter

None.

Return Value

None.

8.11 flip_wait

```
void flip_wait ( );
```

Overview

Switch between frame when use double frame buffer.
Wait that switchover is completed.

Parameter

None.

Return Value

None.

8.12 async_flip

```
void async_flip ( );
```

Overview

Switch between frame when use double frame buffer.
Perform async switchover.

Parameter

None.

Return Value

None.

8.13 async_flip_wait

```
void async_flip_wait ( );
```

Overview

Switch between frame when use double frame buffer.
Perform async switchover and Wait that switchover is completed.

Parameter

None.

Return Value

None.

8.14 ge_wait_cmd_complete

```
void ge_wait_cmd_complete ( );
```

Overview

Wait that graphic engine operation is completed. Before operating that give influence to graphic engine, use this function.

Parameter

None.

Return Value

None.

8.15 ge_draw_rectfill

```
void ge_draw_rectfill (
    int x,
    int y,
    int w,
    int h,
    EGL_COLOR c
);
```

Overview

Draw filled rectangle by graphic engine.

Parameter

int x	x coordinate of rectangle.
int y	y coordinate of rectangle.
int w	Width of rectangle.
int h	Height of rectangle.
EGL_COLOR c	Color of rectangle. Use MAKE_COLORREF(r, g, b) macro function.

Return Value

None.

8.16 ge_draw_rectfillalpha

```
void  ge_draw_rectfillalpha (
    int  x,
    int  y,
    int  w,
    int  h,
    EGL_COLOR  acolor
 );
```

Overview

Draw filled rectangle with transparency effect by graphic engine.

Parameter

int x	x coordinate of rectangle.
int y	y coordinate of rectangle.
int w	Width of rectangle.
int h	Height of rectangle.
EGL_COLOR c	Color of rectangle. Use MAKE_COLORREF(r, g, b) macro function.

Return Value

None.

8.17 surface_set_alpha

```
BOOL  surface_set_alpha (
    SURFACE*  surf,
    U8  a
);
```

Overview

Set surface alpha value.

Parameter

SURFACE* surf	SURFACE that to set alpha.
U8 a	Aplha value.

Return Value

TRUE or FALSE

8.18 draw_line

```
void  draw_line (
    int  x1,
    int  y1,
    int  x2,
    int  y2,
    EGL_COLOR  color
);
```

Overview

draw a line from x1, y1 coordinate to x2, y2 coordinate.

Parameter

int x1	x coordinate of line start point.
int y1	y coordinate of line start point.
int x2	x coordinate of line end point.
int y2	y coordinate of line end point.
EGL_COLOR color	Color of line. Use MAKE_COLORREF(r, g, b) macro function.

Return Value

None.

8.19 draw_rect

```
void  draw_rect (
    int  x,
    int  y,
    int  w,
    int  h,
    EGL_COLOR  c
);
```

Overview

Draw rectangle.

Parameter

int x	x coordinate of rectangle.
int y	y coordinate of rectangle.
int w	Width of rectangle.
int h	Height of rectangle.
EGL_COLOR c	Color of rectangle. Use MAKE_COLORREF(r, g, b) macro function.

Return Value

None.

8.20 draw_rectfill

```
void  draw_rectfill (
    int  x,
    int  y,
    int  w,
    int  h,
    EGL_COLOR  c
);
```

Overview

Draw filled rectangle.

Parameter

int x	x coordinate of rectangle.
int y	y coordinate of rectangle.
int w	Width of rectangle.
int h	Height of rectangle.
EGL_COLOR c	Color of rectangle. Use MAKE_COLORREF(r, g, b) macro function.

Return Value

None

8.21 draw_roundrect

```
void  draw_roundrect (
    int  x0,
    int  y0,
    int  w,
    int  h,
    int  corner,
    EGL_COLOR  c
);
```

Overview

Draw round rectangle.

Parameter

int x0	x coordinate of round rectangle.
int y0	y coordinate of round rectangle.
int w	Width of round rectangle.
int h	Height of round rectangle.
int corner	Corner round strength. Bigger values indicates wider round.
EGL_COLOR c	Color of round rectangle. Use MAKE_COLORREF(r,g,b) macro function.

Return Value

None

8.22 draw_roundrectfill

```
void  draw_roundrectfill (
    int  x0,
    int  y0,
    int  w,
    int  h,
    int  corner,
    EGL_COLOR  c
);
```

Overview

Draw filled round rectangle.

Parameter

int x0	x coordinate of round rectangle.
int y0	y coordinate of round rectangle.
int w	Width of round rectangle.
int h	Height of round rectangle.
int corner	Corner round strength. Bigger values indicates wider round.
EGL_COLOR c	Color of round rectangle. Use MAKE_COLORREF(r,g,b) macro function.

Return Value

None

8.23 draw_circle

```
void  draw_circle (
    int  x,
    int  y,
    int  r,
    EGL_COLOR  color
);
```

Overview

Draw circle.

Parameter

int x	x coordinate of circle center.
int y	y coordinate of circle center.
int r	Radius of the circle.
EGL_COLOR color	Color of circle. Use MAKE_COLORREF(r,g,b) macro function.

Return Value

None.

8.24 draw_circlefill

```
void  draw_circlefill (
    int  x,
    int  y,
    int  r,
    EGL_COLOR  color
);
```

Overview

Draw filled circle.

Parameter

int x	x coordinate of circle center.
int y	y coordinate of circle center.
int r	Radius of circle.
EGL_COLOR color	Color of circle. Use MAKE_COLORREF(r,g,b) macro function.

Return Value

None

8.25 draw_ellipse

```
void  draw_ellipse (
    int  x,
    int  y,
    int  rx,
    int  ry,
    EGL_COLOR  color
);
```

Overview

Draw ellipse.

Parameter

int x	x coordinate of ellipse center.
int y	y coordinate of ellipse center.
int rx	Radius along the x axis of the ellipse.
int ry	Radius along the y axis of the ellipse.
EGL_COLOR color	Color of ellipse. Use MAKE_COLORREF(r,g,b) macro function.

Return Value

None.

6.26 draw_ellipsefill

```
void  draw_ellipsefill (
    int  x,
    int  y,
    int  rx,
    int  ry,
    EGL_COLOR  color
);
```

Overview

Draw filled ellipse.

Parameter

int x	x coordinate of ellipse center.
int y	y coordinate of ellipse center.
int rx	Radius along the x axis of the ellipse.
int ry	Radius along the y axis of the ellipse.
EGL_COLOR color	Color of ellipse. Use MAKE_COLORREF(r,g,b) macro function.

Return Value

None

8.27 loadbmp

```
SURFACE*  loadbmp (
    char*  fname
);
```

Overview

Function to load a bmp file. Loads a bmp file, creates a memory region, creates a SUFRACE structure with **draw_surface()** function and stores it.

Parameter

char* fname	BMP file name.

Return Value

A pointer to the image information and data stored memory region.

8.28 loadbmpp

```
SURFACE*  loadbmpp (
    U8*  startaddr
);
```

Overview

Function to load a bmp image from memory. It loads the image from memory not from a file like loadbmp function.

Parameter

U8* startaddr	Image stored memory address.

Return Value

A pointer to the image information and data stored memory region.

8.29 loadjpg

```
SURFACE*  loadjpg (
    char*  fname
);
```

Overview

Function to load a jpg file. Loads a jpg file, creates a memory region, creates a SUFRACE structure with **draw_surface()** function and stores it.
(**libjpeg.a** file should be added to the project for the jpg image output.)

Parameter

char* fname	JPG file name

Return Value

A pointer to the image information and data stored memory region.

8.30 loadjpgp

```
SURFACE*  loadjpgp (
    U8*  databuf,
    U32  len
);
```

Overview

Function to load a jpg image from memory. It loads the image from memory not from a file like Loadjpg function.

Parameter

U8* databuf	Image stored memory address.
U32 len	Length of the image data.

Return Value

A pointer to the image information and data stored memory region.

8.31 loadtga

```
SURFACE*  loadtga (
    char*  fname
);
```

Overview

Function to load a tga file. Loads a tga file, creates a memory region, creates a SUFRACE structure with **draw_surface()** function and stores it.

Parameter

char* fname	tga file name.

Return Value

A pointer to the image information and data stored memory region.

8.32 loadtgap

```
SURFACE*  loadtgap (
    U8*  startaddr
);
```

Overview

Function to load a tga image from memory. It loads the image from memory not from a file like Loadtga function.

Parameter

U8* startaddr	Image stored memory address.

Return Value

A pointer to the image information and data stored memory region.

8.33 loadpng

```
SURFACE*  loadpng (
    char*  filename
);
```

Overview

Function to load a png file. Loads a png file, creates a memory region, creates a SUFRACE structure with **draw_surface()** function and stores it.
(libz.a, libpng.a file should be added to the project for the png image output.)

Parameter

char* filename	PNG file name

Return Value

A pointer to the image information and data stored memory region.

8.34 loadpngp

```
SURFACE*  loadpngp (
    U8*  pngbuf,
    U32  datalen
);
```

Overview

Function to load a png image from memory. It loads the image from memory not from a file like Loadpng function.

Parameter

U8* pngbuf	Image stored memory address
U32 datalen	length of the image data

Return Value

A pointer to the image information and data stored memory region.

8.35 loadsurf

```
SURFACE*  loadsurf (
    char*  fname
);
```

Overview

Function to load a suf file. Loads a suf file, creates a memory region, creates a SUFRACE structure with **draw_surface()** function and stores it.
Suf file is that has data changed to match RGB mode (RGB888 or RGB565). Suf file has shortest load times than other images. You can make suf image by MakeSurface2.exe in SDK pc-util folder.

Parameter

char* fname	surf file name.

Return Value

A pointer to the image information and data stored memory region.

8.36 draw_surface

```
BOOL  draw_surface (
    SURFACE*  src_surf,
    int  dx,
    int  dy
);
```

Overview

Display the image that loaded.

Parameter

SURFACE* src_surf	SURFACE pointer of loaded image.
int dx	x coordinate of image.
int dy	y coordinate of image.

Return Value

TRUE or FALSE

8.37 draw_surface_rect

```
BOOL  draw_surface_rect (
    SURFACE*  src_surf,
    int  dx,
    int  dy,
    int  sx,
    int  sy,
    int  w,
    int  h
);
```

Overview

Function to output a specific region of an image. Specified region of the image (start point, size) is displayed at coordinate (dx,dy). It is used to output parts of the image.

Parameter

SURFACE* src_surf	SURFACE pointer of loaded image.
int dx	x coordinate of image.
int dy	y coordinate of image.
int sx	x coordinate of the image to be displayed.
int sy	y coordinate of the image to be displayed.
w	Horizontal length of the image to be displayed. It displays w length of the image from the coordinate sx.
h	Vertical length of the image to be displayed. It displays h length of the image from the coordinate sy.

Return Value

TRUE or FALSE

8.38 draw_surface_scale

```
BOOL  draw_surface_scale (
    SURFACE*  src_surf,
    int  dx,
    int  dy,
    int  dw,
    int  dh
);
```

Overview

Function to output an image with scaling. The image is up-scaled, if dw/dh values are bigger than the original image's width/height length. The image is down-scaled when vice-versa

Parameter

SURFACE* src_surf	SURFACE pointer of loaded image.
int dx	x coordinate of image.
int dy	y coordinate of image.
int dw	horizontal length of the displaying image. If this value is bigger than the original image's width then the horizontal length will be up-scaled or (smaller) down-scaled.
int dh	vertical length of the displaying image. If this value is bigger than the original image's height then the vertical length will be up-scaled or (smaller) down-scaled.

Return Value

TRUE or FALSE

8.39 draw_surface_scalerect

```
BOOL  draw_surface_scalerect (
    SURFACE*  src_surf,
    int  dx,
    int  dy,
    int  dw,
    int  dh,
    int  sx,
    int  sy,
    int  sw,
    int  sh
);
```

Overview

Function to merge the functions of **draw_surface_scale()** function and **draw_surface_rect()** function. Up/down scaled outputs the specific part of the image. Outputs a region of the image that is originated from (sx, sy) to sw, sh lengths to the coordinate (dx, dy) with dw, dh lengths. Up-scaled when sw, sh values are smaller than dw, dh, down-scaled vice-versa.

Parameter

SURFACE* src_surf	SURFACE pointer of loaded image.
int dx	x coordinate of image.
int dy	y coordinate of image.
int dw	Horizontal length of the displaying image. If this value is bigger than the original image's width then the horizontal length will be up-scaled or (smaller) down-scaled.
int dh	Vertical length of the displaying image. If this value is bigger than the original image's height then the vertical length will be up-scaled or (smaller) down-scaled.
int sx	Initial x coordinate of the part of the original image to be displayed.
int sy	Initial y coordinate of the part of the original image to be displayed.
int sw	Horizontal length of the output image. From sx to sw
int sh	Vertical length of the output image. From sy to sh

Return Value

TRUE or FALSE

8.40 draw_surface_rotate

```
BOOL  draw_surface_rotate (
    SURFACE*  src,
    int  cdx,
    int  cdy,
    int  ctx,
    int  cty,
    int  zoom,
    int  degrees
);
```

Overview

Draw rotated image.

Parameter

SURFACE* src_surf	SURFACE pointer of loaded image.
int cdx	x-coordinate of image.
int cdy	y-coordinate of image.
int ctx	x-coordinate of image rotate axis.
int cty	y-coordinate of image rotate axis.
int zoom	Image zoom in/out value. default 1. scale down : zoom value < 1 scale up : zoom value > 1
int degrees	Angle of rotation.

Return Value

TRUE or FALSE

8.41 release_surface

```
void  release_surfacet (
    SURFACE*  surf
);
```

Overview

Function to release the image load function created memory region. It is better to release unused image's memory region with this function because available memory is limited.

Parameter

SURFACE* surf	SURFACE pointer of loaded image.

Return Value

None

8.42 draw_set_clip_winodw

```
void  draw_set_clip_window (
    SURFACE*  dst,
    CLIP_RECT*  pRect
);
```

Overview

Function to limit the region that an image could be displayed. The image cannot be displayed outside of the region from coordinate (x,y) to w width and h height.

Parameter

SURFACE* dst	Target frame.
CLIP_RECT* pRect	Region to allow the image output. typedef struct { int x; int y; int endx; int endy; }CLIP_RECT;

Return Value

None

8.43 osd_set_surface

```
BOOL  osd_set_surface (
    SURFACE*  surf,
    int  x,
    int  y
);
```

Overview

Set OSD coordinate and region. And enable OSD.

Parameter

SURFACE* surf	SURFACE that want to use OSD.
int x	X coordinate of OSD.
int y	Y coordinate of OSD.

Return Value

TRUE or FALSE

8.44 osd_set_surface_with_alpha

```
BOOL  osd_set_surface_with_alpha (
    SURFACE*  surf,
    int  x,
    int  y,
    U8  alpha
);
```

Overview

Set OSD coordinate and region (with alpha effect).

Parameter

SURFACE* surf	SURFACE that want to use OSD.
int x	x coordinate of OSD.
int y	y coordinate of OSD.
U8 alpha	Alpha value of OSD.

Return Value

TRUE or FALSE

8.45 osd_set_pal

```
void  osd_set_pal (
    EGL_COLOR*  pal,
    int  num
);
```

Overview

Set palette of OSD.

Parameter

EGL_COLOR* pal	Palette color buffer.
int num	Number of Palette color.

Return Value

None.

8.46 osd_set_off

```
void  osd_set_off ( );
```

Overview

Disable OSD.

Parameter

None

Return Value

None

8.47 OSD_MOVE

```
OSD_MOVE ( x , y );
```

Overview

Move the OSD coordinate.

Parameter

x	x coordinate.
y	y coordinate.

Return Value

None

8.48 vga_mode_on

```
void vga_mode_on( void );
```

Overview

Set VGA (analog RGB) output mode. Must call this function in order to use VGA output mode.

Parameter

None

Return Value

None

8.49 vce_on_surface

```
BOOL  vce_on_surface (
   SURFACE*  dst,
   int  x,
   int  y,
   bool  halfmode
);
```

Overview

Call function when use external video. External video data stored in specified SURFACE struct. So, can display external video using draw_surface like display image. Because using SURFACE, need to create surface as create_surface function.

Parameter

SURFACE* dst	SURFACE to store the external video data.
int x	External video x-coordinate.
int y	External video y-coordinate.
bool halfmode	Determine halfmode.

Return Value

TRUE or FALSE

Example

```
SURFACE*  ext_surf = create_surface(720, 480, 16)
vce_on_surface(ext_surf,0,0,false);

draw_surface(ext_surf, 24, 24);
flip();
```

8.50 loadjpg_hw

```
SURFACE*  loadjpg_hw (
    char*  fname
);
```

Overview

Function to load a jpg file using H/W decoder. Loads a jpg file, creates a memory region, creates a SUFRACE structure with **draw_surface()** function and stores it.
Because H/W decoder is supported standard format, convert image file using makesurface2.exe in pc-util.

Parameter

char* fname	JPG file name

Return Value

A pointer to the image information and data stored memory region.

8.51 savebmp

```
BOOL  savebmp (
    char*  savefname,
    SURFACE* src
);
```

Overview

Save the screen as bmp image.

Parameter

char* savefname	BMP image file name
SURFACE* src	Screen image source

Return Value

TRUE or FALSE.

8.52 savejpg

```
BOOL  savejpg (
    char*  savefname,
    SURFACE* src
);
```

Overview

Save the screen as jpg image.

Parameter

char* savefname	jpg image file name
SURFACE* src	Screen image source

Return Value

TRUE or FALSE.

< savebmp/savejpg example >

```
SURFACE* current_frame = get_front_buffer();
res = savebmp("image1.bmp", current_frame); // Save the current screen as image1.bmp.
res = savejpg("image2.jpg", current_frame);  // Save the current screen as image2.bmp.
```

8.53 surface_set_alpha

```
BOOL  surface_set_alpha (
    SURFACE*  surf,
    U8  a
);
```

Overview

Set the alpha value of SURFACE. Value range is 0 to 255. The smaller the value becomes transparent.

Parameter

SURFACE* surf	SURFACE, to set an alpha value.
U8* a	Alpha value.

Return Value

TRUE or FALSE.

8.54 Graphic Example

<Single Frame>

```
#include "sdk.h"

int main()
{
        boardinit();
        uart_config(0,115200,DATABITS_8,STOPBITS_1,UART_PARNONE);

        display_clock_init();  // set LCD clock
        ge_set_texture_mem(0xc3000000, 0x50000000);
              // set texture memory. start address : 0xc3000000, size : 80MByte
       setscreen(0xc2000000,SCREEN_1024x600,SCREENMODE_RGB565|SCREENNMODE_SINGLEBUF);
              // 1024x600 LCD, frame buffer start address : 0xc2000000, RGB565, single buf mode.

        lcdon();  // turn on the lcd back light.

        draw_rectfill(0,0,get_screen_width(),get_screen_height(),MAKE_COLORREF(255,255,255));
              // draw 1024x600 white rectangle. ( screen cleared as white color )
        draw_line(0,0,100,100,MAKE_COLORREF(255,0,0));  // draw red line.
        draw_rect(100,100,100,100,MAKE_COLORREF(255,0,0));  // draw red rectangle.
        draw_circle(200,200,80,MAKE_COLORREF(0,0,255));  // draw blue circle.
        draw_ellipse(200,50,30,20,MAKE_COLORREF(0,255,0));  // draw green ellipse.
       draw_roundrect(100,270,100,100,10,MAKE_COLORREF(128,128,128));  // draw round rectangle.
       draw_roundrectfill(110,280,80,80,10,MAKE_COLORREF(255,0,0)); // draw round rectangle filled.

        while(1);
        return 0;
}
```

<Double Frame>

```
#include "sdk.h"

int main()
{
        boardinit();
        uart_config(0,115200,DATABITS_8,STOPBITS_1,UART_PARNONE);

        FATFS fs;
        f_mount(DRIVE_NAND, &fs);

        display_clock_init();  // set LCD clock
        ge_set_texture_mem(0xc3000000, 0x50000000);
             // set texture memory. start address : 0xc3000000, size : 80MByte
      setscreen(0xc2000000,SCREEN_1024x600,SCREENMODE_RGB565|SCREENNMODE_DOUBLEBUF);
             // 1024x600 LCD, frame buffer start address : 0xc2000000, RGB565, double buf mode.

        lcdon();  // turn on the lcd back light.

        SURFACE* bg = loadbmp("background.bmp");  // load bmp image.
        drawsurface(bg,0,0);  // draw image.
        flip();  // switch frame buffer.

        SURFACE* bg2 = loadbmp("background2.bmp");  // load bmp image.
        SURFACE* icon = loadjpg("icon.jpg");   // load jpg image.
        drawsurface(bg2,0,0);  // draw back ground image.
        drawsurface(icon,100,100);  // draw icon image.
        flip(); // switch frame buffer.

        while(1);
        return 0;
}
```

9. Movie file play

Movie file is generated by MovieGenerator™ in pc-util.

See the separate documentation for details about MovieGenerator™

Function	Description
loadmovie	Load movie file.
release_movie	Release memory that created by loadmovie function.
movie_play	Display first frame of video.
movie_drawnext	Display next frame on wanted position.
movie_drawnextex	Display next frame on wanted position. Can adjust size.
movie_seek	Change the current file position to specific time.
movie_current_time	Return time of current displayed frame.

9.1 loadmovie

```
MOVIE*  loadmovie(
    char*  fname,
    BOOL  bAll
);
```

Overview

Load movie file. All audio data is stored in ram memory. So, if there is audio data, should be ensured that sufficient memory.

11025Hz * 2 (16bit mono) * 30 sec = 661,500 bytes

You can release memory that used by release_movie function.

Parameter

char* fname	Movie file name
BOOL bAll	Whether to load all video data. If false, load video data on each every frame.

Return Value

Pointer to MOVIE

9.2 release_movie

```
void  release_movie(
    MOVIE*  mov
);
```

Overview

Release memory that used.

Parameter

MOVIE* mov	Pointer to MOVIE

Return Value

None

9.3 movie_play

```
BOOL  movie_play(
    MOVIE*  mov,
    int  x,
    int  y,
    BOOL  bAudio
);
```

Overview

Display first frame of video. If bAudio is true and there is audio data, start play audio.

Parameter

MOVIE* mov	Pointer to MOVIE.
int x	x coordinate.
int y	y coordinate
BOOL bAudio	Determine whether the output audio. If false, do not play audio.

Return Value

True or False.

9.4 movie_drawnext

```
MOVIE_DRAW_RESULT  movie_drawnext(
    MOVIE*  mov,
    int  x,
    int  y,
    BOOL  b_audiosync
);
```

Overview

Display next frame on wanted position. If b_audiosync is false, not synchronized video and audio are output in sequence.

Parameter

MOVIE* mov	Pointer to MOVIE.
int x	x coordinate.
int y	y coordinate
BOOL b_audiosync	Determine whether the sync about audio and video.

Return Value

MOVIE_DRAW_RESULT_FAIL – if mov is null, or mov is not running.
MOVIE_DRAW_RESULT_OK – if success.
MOVIE_DRAW_RESULT_EOF – if end of file.

9.5 movie_drawnextex

```
MOVIE_DRAW_RESULT  movie_drawnextex(
    MOVIE*  mov,
    int  x,
    int  y,
    int  w,
    int  h,
    BOOL  b_audiosync
);
```

Overview

Display next frame on wanted position and size. If b_audiosync is false, not synchronized video and audio are output in sequence.

Parameter

MOVIE* mov	Pointer to MOVIE.
int x	x coordinate.
int y	y coordinate
int w	width
int h	height
BOOL b_audiosync	Determine whether the sync about audio and video.

Return Value

MOVIE_DRAW_RESULT_FAIL – if mov is null, or mov is not running.
MOVIE_DRAW_RESULT_OK – if success.
MOVIE_DRAW_RESULT_EOF – if end of file.

9.6 movie_seek

```
U32  movie_seek(
    MOVIE*  mov,
    U32  ms,
);
```

Overview

Change the current file position to specific time.

Parameter

MOVIE* mov	Pointer to MOVIE.
U32 ms	Milliseconds.

Return Value

Number of video frame.

9.7 movie_current_time

```
U32  movie_current_time(
    MOVIE*  mov,
);
```

Overview

Return time of current displayed frame.

Parameter

MOVIE* mov	Pointer to MOVIE.

Return Value

Current miliseconds.

9.8 example

```
int main()
{
        …
        …
        MOVIE* mov = loadmovie(fname, FALSE);
        while(1)
        {
                if(movie_drawnext(mov, x, y, true) == MOVIE_DRAW_RESULT_EOF)
                        break;
                flip();
        }
        …
        …
}
```

10. SOUND

Function	Description
sound_init	Initialize a sound mixer.
sound_loadwav	Load WAV file.
sound_loadwavp	Load WAV data in memory.
sound_loadmp3	Load MP3 file.
sound_loadmp3p	Load MP3 data in memory.
sound_release	Release the memory that is created by load function.
sound_play	Play sound.
sound_stop	Stop sound.
sound_vol	Set volume of Sound mixer.
sound_vol_wav	Set volume of each Sound file.
sound_pause	Pause Sound.
sound_resume	Resume sound of pause status.
sound_isplay	Confirm that sound is playing.
sound_ispause	Confirm that sound is pause.
sound_playex	Play sound from a specific position.
sound_current_time	Return the current playing position.

10.1 sound_init

```
BOOL  sound_init ( );
```

Overview

Initialize a sound mixer.

Parameter

None

Return Value

None

10.2 sound_loadwav

```
WAVE*  sound_loadwav (
    char*  filename
);
```

Overview

Load WAV file.

Parameter

char* filename	WAV file name.

Return Value

WAV file information and data stored WAVE structure.

10.3 sound_loadwavp

```
WAVE*  sound_loadwavp (
    U8*  buf,
    U32  len
);
```

Overview

Load WAV data in memory.

Parameter

U8* buf	Buffer that WAV data was stored.
U32 len	Length of WAV data.

Return Value

WAV file information and data stored WAVE structure.

10.4 sound_loadmp3

```
WAVE*  sound_loadmp3 (
    char*  filename,
);
```

Overview

Load MP3 file.

Parameter

char* filename	MP3 file name.

Return Value

MP3 file information and data stored WAVE structure.

10.5 sound_loadmp3p

```
WAVE*  sound_loadmp3p (
    U8*  buf,
    U32  mp3databufsize
);
```

Overview

Load MP3 data in memory.

Parameter

U8* buf	Buffer that MP3 data was stored.
U32 mp3databufsize	Length of MP3 data.

Return Value

MP3 file information and data stored WAVE structure.

10.6 sound_release

```
BOOL  sound_release (
    WAVE*  pWave
);
```

Overview

Release the memory that is create by load function
It is better to return disusing memory by calling **sound_release()** function about the disusing sound, because available memory space is limited.

Parameter

WAVE* pWave	WAVE structure that created by load function.

Return Value

TRUE or FALSE

10.7 sound_play

```
BOOL  sound_play (
    WAVE*  pWave
);
```

Overview

Play the sound.

Parameter

WAVE* pWave	WAVE structure that created by load function.

Return Value

TRUE or FALSE

10.8 sound_stop

```
BOOL  sound_stop (
    WAVE*  pWave
);
```

Overview

Stop the sound.

Parameter

WAVE* pWave	WAVE structure that created by load function.

Return Value

TRUE or FALSE

10.9 sound_vol

```
void  sound_vol (
      U8  vol
);
```

Overview

Set volume of Sound mixer.

Parameter

U8 vol	Volume value. Max value is 255.

Return Value

None

10.10 sound_vol_wav

```
void  sound_vol_wav (
      WAVE*  pWav,
      U8  vol
);
```

Overview

Set volume of each Sound file.

Parameter

WAVE* pWav	WAVE structure that created by load function.
U8 vol	Volume value. Max value is 255.

Return Value

None

10.11 sound_pause

```
BOOL  sound_pause (
    WAVE*  pWave
);
```

Overview

Pause Sound.

Parameter

WAVE* pWave	WAVE structure that created by load function.

Return Value

TRUE or FALSE

10.12 sound_resume

```
BOOL  sound_resume (
    WAVE*  pWav
);
```

Overview

Resume sound of pause status.

Parameter

WAVE* pWave	WAVE structure that created by load function.

Return Value

TRUE or FALSE

10.13 sound_isplay

```
BOOL  sound_isplay (
      WAVE*  pWave
);
```

Overview

Confirm that sound is playing.

Parameter

WAVE* pWave	WAVE structure that created by load function.

Return Value

TRUE(1 = play) or FALSE(0 = stop)

10.14 sound_ispause

```
BOOL  sound_ispause (
      WAVE*  pWave
);
```

Overview

Confirm that sound is pause.

Parameter

WAVE* pWave	WAVE structure that created by load function.

Return Value

TRUE(1 = pause) or FALSE(0 = none pause)

10.15 sound_playex

```
BOOL  sound_playex (
      WAVE*  pWave,
      U32  start_ms
);
```

Overview

Play sound from a specific position.

Parameter

WAVE* pWave	WAVE structure that created by load function.
U32 start_ms	Specific position (ms).

Return Value

TRUE or FALSE

10.16 sound_current_time

```
BOOL  sound_current_time (
      WAVE*  pWave
);
```

Overview

Return the current playing position.

Parameter

WAVE* pWave	WAVE structure that created by load function.

Return Value

Playing position (ms).

10.17 Sound Example

< WAV File Play >

```
#include "sdk.h"

Intmain()
{
    boardinit();
    uart_config(0,115200,DATABITS_8,STOPBITS_1,UART_PARNONE);

    FATFS fs;
    f_mount(DRIVE_NAND, &fs);

    sount_init();                  // sound mixer initialization for the sound output.
    WAVE* sound = sount_loadwav("test.wav");    // wav file loading

    sound_play(sound);                          // wav file playing

    delayms(1000);

    If(sound_isplay(sound))        // Examine whether the wav file is playing or not after one second
    {
        sound_stop(sound);                      // Stop if it is playing
    }

    sound_release(sound);                       // Remove no more using sound from memory.

      ...
      ...

    while(1);
    return 0;
}
```

< MP3 File Play >

```
#include "sdk.h"

Intmain()
{
    uart_config(0,115200,DATABITS_8,STOPBITS_1,UART_PARNONE);

    FATFS fs;
    f_mount(DRIVE_NAND, &fs);

    sound_init();                    // sound mixer initialization for the sound output.
    WAVE* sound = sount_loadmp3("test.mp3");    // mp3 file loading

    sound_play(sound);                          // mp3 file playing

    delayms(1000);

    sound_pause(sound);                // Pause the sound after one second

    delayms(1000);

    If(sound_ispause(sound))           // examine whether the sound is paused or not
    {
        sound_resume(sound);           // resume the sound if it is paused
    }

    …
    …

    while(1);
    return 0;
}
```

11. FILE System

This chapter explains how to configure Nand Flash and SD card as file storage devices.
Refer to fatfs manual for more detailed information about the file system
(http://elm-chan.org/fsw/ff/00index_e.html)

Function	Description
f_mount	Configure Nand Flash or SD card to use as file storage devices by mounting them.
f_chdrive	Change drive between the mounted drives.
f_chdir	Move directory on the mounted drive.

11.1 f_mount

```
FRESULT  f_mount (
    BYTE,
    FATFS*
);
```

Overview

Function to configure Nand Flash or SD card to use as file storage devices by mounting them.

Parameter

BYTE	Device numbers of Nand Flash or SD Card. These are defined as #define DRIVE_NAND 0 #define DRIVE_SDCARD 1
FATFS	device information holding structure.

Return Value

Result of the operations.

0 is success. Refer to fatfs manual about the others.

11.2 f_chdrive

```
FRESULT  f_chdrive (
    BYTE
);
```

Overview

Function to change drive between the mounted drives.

Parameter

BYTE	Device numbers of Nand Flash or SD Card. #define DRIVE_NAND 0 #define DRIVE_SDCARD 1

Return Value

Result of the operations.

0 is success. Refer to fatfs manual about the others.

11.3 f_chdir

```
FRESULT  f_chdir (
    const TCHAR*
);
```

Overview

Function to move directory on the mounted drive.

Parameter

TCHAR*	directory name to move

Return Value

Result of the operations.

0 is success. Refer to fatfs manual about the others.

11.4 FILE System Example

```
int main()
{
        init_interrupt();    // Interrupt 초기화
        uart_config(0, 115200, DATABITS_8, STOPBITS_1, UART_PARNONE );

       FATFS nand_fs, sdcard_fs;
        f_mount(DRIVE_NAND, &nand_fs);// Nand Flash mount
        f_mount(DRIVE_SDCARD, &sdcard_fs);// SD Card mount

       SURFACE* bmp = loadbmp("test.bmp"); // Load an image file from Nand Flash

        f_chdrive(DRIVE_SDCARD);              // change a drive to SD Card

       WAVE* wav = sound_loadwav("test.wav"); // Load a sound file from SD Card

        f_chdir("font");                      // Move to font directory of SD Card.
        //f_chdir("..");                      // Move to upper directory.
        //f_chdir("/..");                     // Move to root directory.
        ...
        ...

       while(1)
       return 0;
}
```

12. Font

Support two kinds of fonts. Image font and bit type font are them; this chapter explains the functions to use the fonts.

It is easy to various types of fonts with Image font, even if it need to create font images by using a program. Bit type font could be used with already provided fonts without any font creation jobs, however it is hard to change the font types.

Function	Description
create_bmpfont	Prepare to use a bmp font by loading a font file.
release_bmpfont	Release bmp font.
bmpfont_draw	Display strings on specific location using bmp font.
bmpfont_draw_vleft	Display string that rotate by left 90 degrees.
bmpfont_draw_vright	Display string that rotate by right 90 degrees.
egl_font_set_color	Change font color.
bmpfont_makesurface	Converse frequently used characters into an image.
bmpfont_setkerning	Set letter spacing of the bmp font.
bmpfont_setautokerning	Determine use the same letter spacing or not.
create_bitfont	Prepare to use a bit font
release_bitfont	Release bit font.
bitfont_draw	Display strings on specific location using bit font.
bitfont_draw_vleft	Display string that rotate by left 90 degrees.
bitfont_draw_vright	Display string that rotate by right 90 degrees.
bitfont_makesurface	Converse frequently used characters into an image.

<< bmp font >>

12.1 create_bmpfont

```
EGL_FONT*  create_bmpfont (
    const char  fname
);
```

Overview

Prepare to use a bmp font by loading a font file.

Parameter

fname	Font file name.

Return Value

BMP font pointer.

12.2 release_bmpfont

```
void  release_bmpfont (
    EGL_FONT*  pFont
);
```

Overview

Release bmp font.

Parameter

pFont	BMP font pointer.

Return Value

None

12.3 bmpfont_draw

```
int  bmpfont_draw (
     EGL_FONT*  pFont,
     int  x,
     int  y,
     const char* str
);
```

Overview

Display strings on specific location using bmp font.

Parameter

EGL_FONT* pFont	BMP font pointer.
int x	x coordinate.
int y	y coordinate.
const char* str	String.

Return Value

Width of displayed string.

12.4 bmpfont_draw_vleft

```
int  bmpfont_draw_vleft (
     EGL_FONT*  pFont,
     int  x,
     int  y,
     const char* str
);
```

Overview

Display string that rotate by left 90 degrees.

Parameter

EGL_FONT* pFont	BMP font pointer.
int x	x coordinate.
int y	y coordinate.
const char* str	String.

Return Value

Width of displayed string.

12.5 bmpfont_draw_vright

```
int  bmpfont_draw_vright (
     EGL_FONT*  pFont,
     int  x,
     int  y,
     const char* str
);
```

Overview

Display string that rotate by right 90 degrees.

Parameter

EGL_FONT* pFont	BMP font pointer.
int x	x coordinate.
int y	y coordinate.
const char* str	String.

Return Value

Width of displayed string.

12.6 egl_font_set_color

```
EGL_COLOR  egl_font_set_color (
     EGL_FONT*  pFont,
     EGL_COLOR  clr
);
```

Overview

Change font color.

Parameter

EGL_FONT* pFont	BMP font pointer.
EGL_COLOR clr	Font color. Use MAKE_COLORREF(r,g,b) macro function.

Return Value

Old color.

12.7 bmpfont_makesurface

```
SURFACE*  egl_font_set_color (
    EGL_FONT*  pFont,
    char*  text
);
```

Overview

Converse frequently used characters into an image. The characters could be displayed by draw_surface function when necessary.

Parameter

EGL_FONT* pFont	BMP font pointer.
EGL_COLOR text	String.

Return Value

SURFACE structure.

12.8 bmpfont_setkerning

```
BOOL  bmpfont_setkerning (
    EGL_FONT*  pFont,
    int  k
);
```

Overview

Set letter spacing of the bmp font.

Parameter

EGL_FONT* pFont	BMP font pointer.
int k	Letter spacing value.

Return Value

TRUE or FALSE

12.9 bmpfont_setautokerning

```
BOOL  bmpfont_setautokerning (
      EGL_FONT*  pFont,
      bool  b
);
```

Overview

Determine use the same letter spacing or not.

Parameter

EGL_FONT* pFont	BMP font pointer.
bool b	TRUE : Same letter spacing.
	FALSE : Different letter spacing.

Return Value

TRUE or FALSE

<< bit font >>

12.10 create_bitfont

```
EGL_FONT*  create_bitfont ( );
```

Overview

Prepare to use a bit font

Parameter

None

Return Value

Bit font pointer.

12.11 release_bitfont

```
void  release_bitfont ( );
```

Overview

Release bit font.

Parameter

pFont	Bit font pointer

Return Value

None

12.12 bitfont_draw

```
int  bitfont_draw (
     EGL_FONT*  pFont,
     int  x,
     int  y,
     const char* str
);
```

Overview

Display strings on specific location using bit font.

Parameter

EGL_FONT* pFont	Bit font pointer.
int x	x coordinate.
int y	y coordinate.
const char* str	String.

Return Value

Width of displayed string.

12.13 bitfont_draw_vleft

```
int  bitfont_draw_vleft (
     EGL_FONT*  pFont,
     int  x,
     int  y,
     const char* str
);
```

Overview

Display string that rotate by left 90 degrees.

Parameter

EGL_FONT* pFont	Bit font pointer.
int x	x coordinate.
int y	y coordinate.
const char* str	String.

Return Value

Width of displayed string.

12.14 bitfont_draw_vright

```
int  bitfont_draw_vright (
     EGL_FONT*  pFont,
     int  x,
     int  y,
     const char* str
);
```

Overview

Display string that rotate by right 90 degrees.

Parameter

EGL_FONT* pFont	Bit font pointer.
int x	x coordinate.
int y	y coordinate.
const char* str	String.

Return Value

Width of displayed string.

12.15 bitfont_makesurface

```
int  bitfont_makesurface (
     EGL_FONT*  pFont,
     char* str
);
```

Overview

Function to converse frequently used characters into an image. The characters could be displayed by **draw_surface()** function when necessary.

Parameter

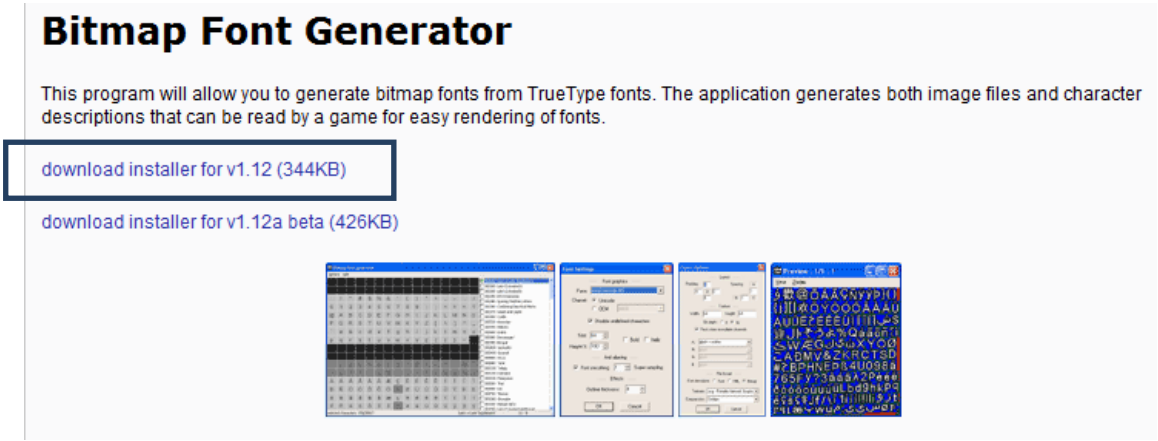
EGL_FONT* pFont	Bit font pointer.
const char* str	String.

Return Value

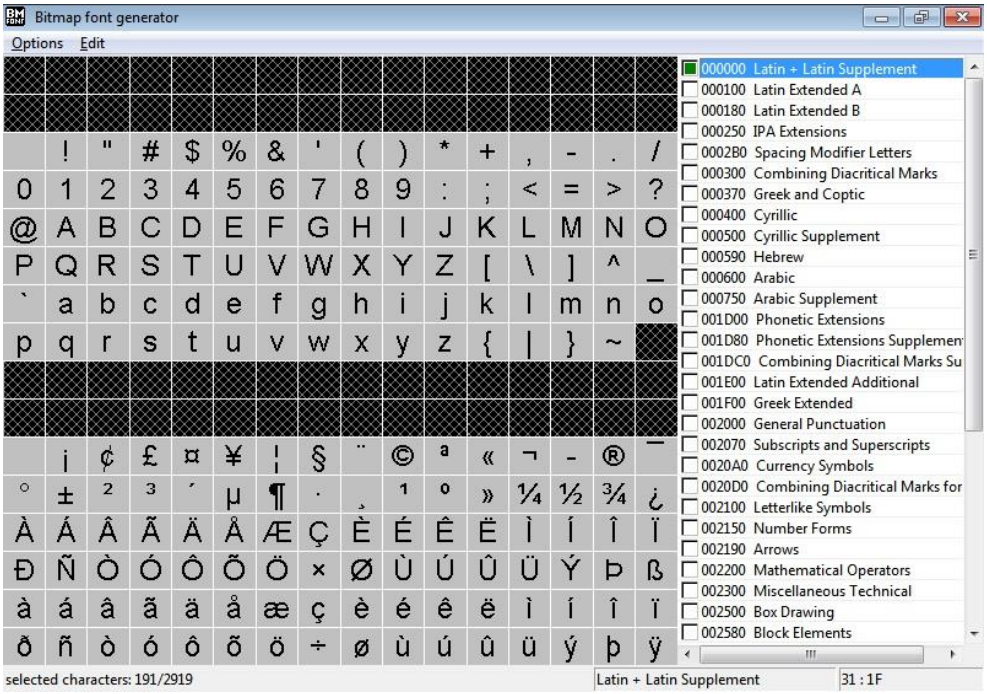
SURFACE structure

< how to create the image font >

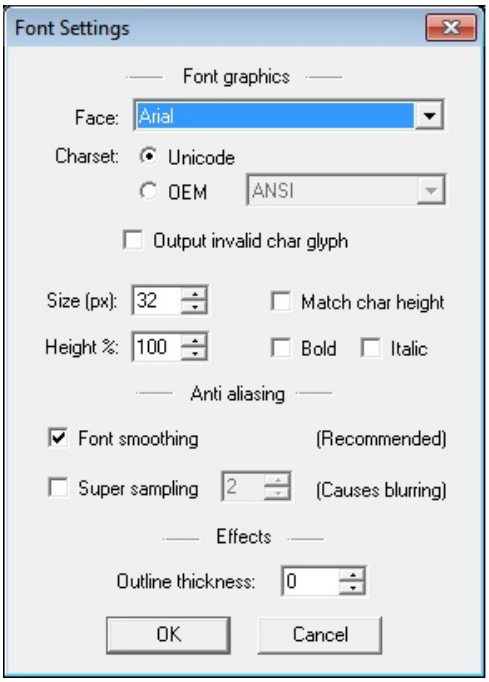
1. Download and install the Bitmap Font Generator v1.12 from http://www.angelcode.com/products/bmfont.



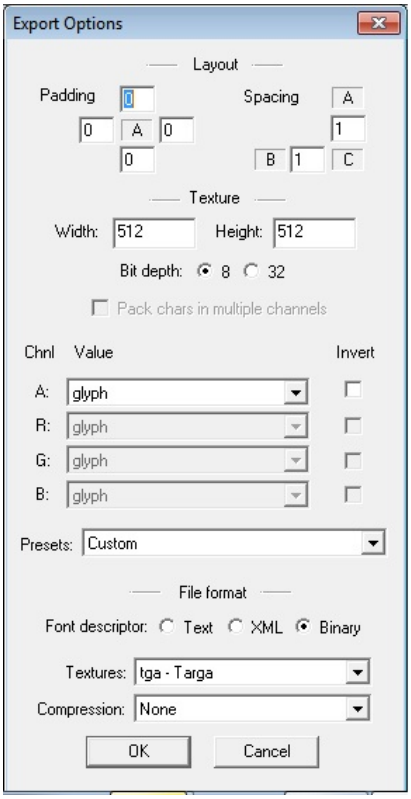
2. Next, execute the program shows up the below window.



3. Determine desired font and its size by choosing 'Options' → 'Font settings'.

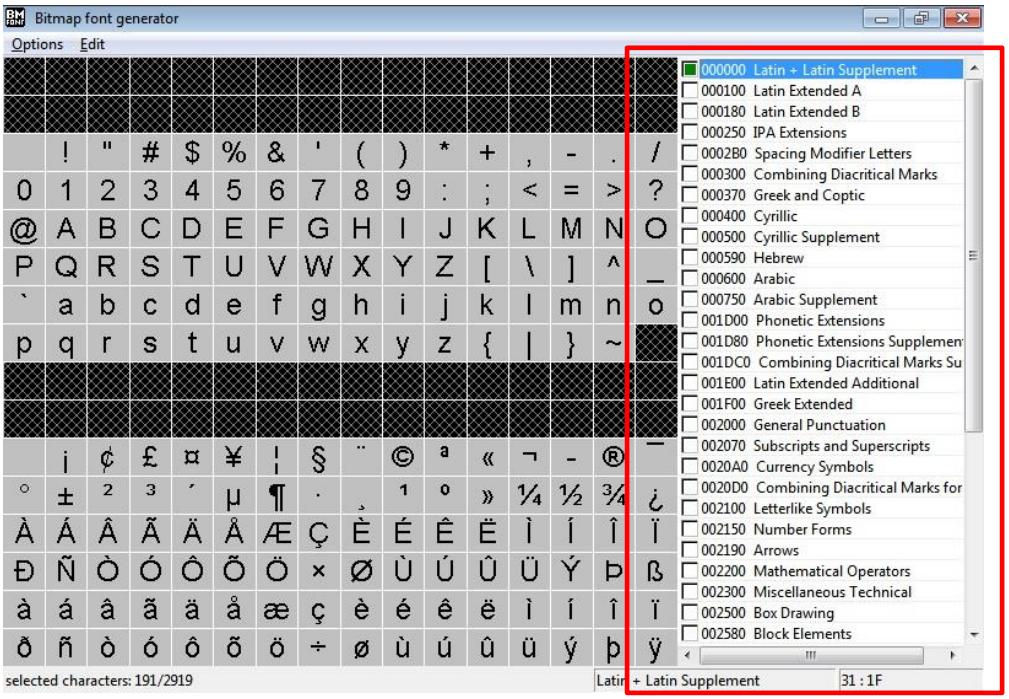


4. Set as the below window after executing 'Options' → 'Export options'.



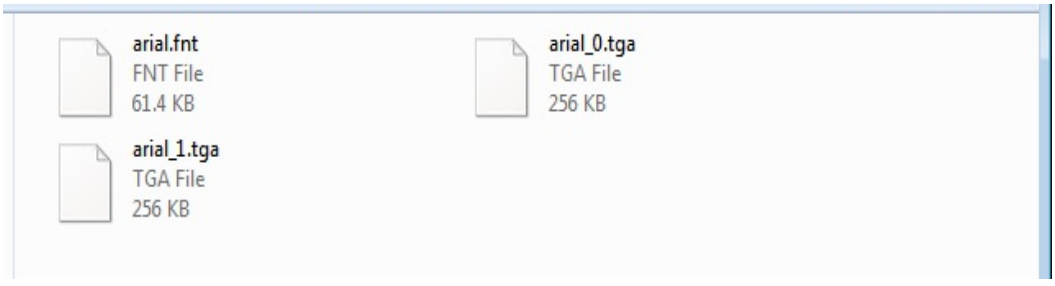
Width and Height are size of a single bitmap file's width and height. Bigger the size reduces total number of the bitmap files and wasted region could exist.

5. Check the desired font on the main window that is showed up at first execution. A font selected on the left side screen with a mouse won't be created as images.



6. Save the font with 'Option' → 'Save bitmap font as', FNAME.fnt and FNAME_xx.tga files are created. The font could be used by loading FNAME.fnt with create_bmpfont function.

7. fnt → stores location information of the font and auxiliary information for displaying
tga → actual font image



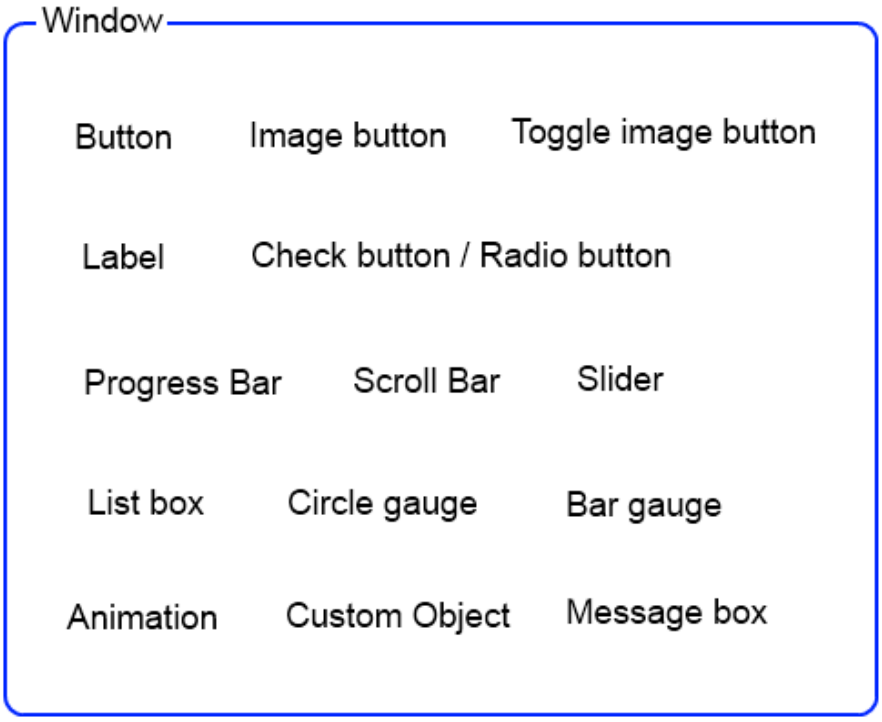
13. EGL Library

1) Itroduce

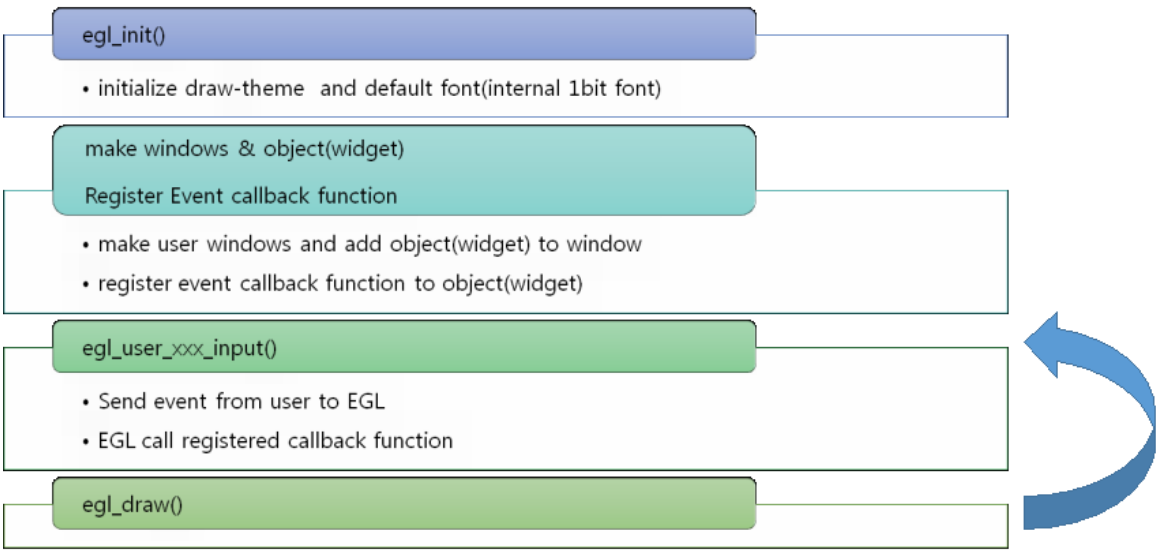
EGL(Embedded Graphic Library)is made for reduce time of graphic application development and use object by easy. EGL is offered function for use many object by easy. Event processing and object management is done in the EGL library. Use Callback function that created by developers for event action.

2) Composition

EGL is composed of follow objects.



3) Program base



< Source Code >

```
#include "adstar.h"

static void btn_callback(EGL_HANDLE  h, int  event) // button callback function.
{
    /* user call back function */
}

extern BOOL process_touch(BOOL*  touchdown, EGL_POINT*  pPoint);

int main()
{
...
  EGL_POINT  touch_pt;
  BOOL  touchdown = FALSE;
  EGL_HANDLE  hWin;
  EGL_HANDLE  btn;
  egl_init();
  hWin = egl_create_window("Main Window");          // create window
  btn = egl_create_button(100,100,200,50,"Button 1");  // create button obj
  egl_btn_callback(btn, btn_callback);
  egl_window_add_object(hWin, btn);                  // button obj add.
  egl_window_show(hWin);                             // window show

  while(1)
  {
       if( process_touch( &touchdown, &touch_pt ) )      // touch input.
       egl_user_touch_input( touchdown, &touch_pt ); // touch processing, msg processing.

     egl_draw();
   }
...
}
```

Window

Function	Description
egl_create_window	Create a window object.
egl_window_show	Decide window output or not.
egl_window_isshow	Return window output state.
egl_window_invalidate	Redraw object in window.
egl_window_invalidate_rect	Redraw object in specified area.
egl_window_redraw_rect	Redraw in specifien area.
egl_window_set_bg	Set the window back ground image.
egl_window_get_bg	Get the window back ground image.
egl_window_add_object	Add object on window.
egl_window_set_callback	Set callback function that will be called when window event occur.
egl_window_get_active	Return current window handle.
egl_user_touch_input	Send touch message to Msg_handler.
egl_draw	Draw object at current window.
egl_visible_object	Set visible status of object.
egl_window_delete_object	Delete object of window.
egl_release_window	Release window object.

▶egl_create_window function

```
EGL_HANDLE  egl_create_window(
      const char*  title
);
```

Overview

Create window. Window size is resolution same.

Parameter

const char* title	Window's title.

Return Value

Handle of created window.

Example

```
EGL_HANDLE  hWin;

hWin = egl_create_window("Main Window");
```

▶**egl_window_show** function

```
BOOL  egl_window_show(
    EGL_HANDLE  hWin,
    BOOL  bShow
);
```

Overview

Decide window output or not.

Parameter

EGL_HANDLE hWin	Handle of window.
BOOL bShow	Wether to output.

Return Value

TRUE or FALSE

Example

```
EGL_HANDLE  hWin;
hWin = egl_create_window("Main Window");
...
egl_window_show(hWin, TRUE);
```

▶egl_window_isshow function

```
BOOL  egl_window_isshow(
    EGL_HANDLE  hWin
);
```

Overview

Return window output state.

Parameter

EGL_HANDLE hWin	Handle of window.

Return Value

TRUE == window is shown.
FALSE == window is not shown.

Example

```
EGL_HANDLE  hWin;
hWin = egl_create_window("Main Window");
...
if(egl_window_isshow(hWin))
{
...
}
```

▶ **egl_window_invalidate** function

```
void  egl_window_invalidate( void );
```

Overview

Redraw object in window.

Parameter

None.

Return Value

None.

Example

```
EGL_HANDLE  hWin;
hWin = egl_create_window("Main Window");
...
egl_window_invalidate( );
```

▶egl_window_invalidate_rect function

```
void  egl_window_invalidate_rect(
    EGL_RECT*  pRect
);
```

Overview

Redraw object in specified area.

Parameter

EGL_RECT* pRect Specified area.

```
typedef  Struct  _tag_RECT{
    int  x;
    int  y;
    int  w;
    int  h;
}EGL_RECT
```

Return Value

None.

Example

```
EGL_HANDLE  hWin;
EGL_RECT pRect;
hWin = egl_create_window("Main Window");
...
pRect.x = 100;
pRect.y = 200;
pRect.w = 150;
pRect.h = 150;
egl_window_invalidate_rect(&pRect );
```

▶egl_window_redraw_rect function

```
Void  egl_window_redraw_rect(
      EGL_RECT*  pRect
);
```

Overview

Redraw in specifien area.

Parameter

EGL_RECT* pRect Specified area.

```
typedef  Struct  _tag_RECT{
    int  x;
    int  y;
    int  w;
    int  h;
}EGL_RECT
```

Return Value

None.

Example

```
EGL_HANDLE hWin;
EGL_RECT pRect;
hWin = egl_create_window("Main Window");
...
pRect.x = 100;
pRect.y = 200;
pRect.w = 150;
pRect.h = 150;
egl_window_redraw_rect(&pRect);
```

▶**egl_window_set_bg** function

```
BOOL  egl_window_set_bg(
    EGL_HANDLE  hWin;
    SURFACE*  Img;
);
```

Overview

Set the window back ground image.

Parameter

EGL_HANDLE hWin	Handle of window.
SURFACE* Img	Back ground image SURFACE.

Return Value

TRUE or FALSE

Example

```
EGL_HANDLE hWin;
SURFACE* surface = loadbmp("windowbg.bmp");
hWin = egl_create_window("Main Window");
egl_window_set_bg(hWin, surface);
```

▶**egl_window_get_bg** function

```
SURFACE*  egl_window_get_bg(
    EGL_HANDLE  hWin;
);
```

Overview

Get the window back ground image.

Parameter

EGL_HANDLE hWin Handle of window.

Return Value

Image of window.

Example

```
EGL_HANDLE hWin;
SURFACE* getimage;
hWin = egl_create_window("Main Window");
...
getimage = egl_window_get_bg(hWin);
```

▶**egl_window_add_object** function

```
BOOL  egl_window_add_object(
    EGL_HANDLE  hWin,
    EGL_HANDLE  hObj
);
```

Overview

Add object on window.

Parameter

EGL_HANDLE hWin	Handle of window.
EGL_HANDLE hObj	Handle of object to add.

Return Value

TRUE or FALSE

Example

```
EGL_HANDLE hWin;
EGL_HANDLE btn;
hWin = egl_create_window("Main Window");
btn = egl_create_button(100, 100, 100, 50, "Button");
egl_window_add_object(hWin, btn);
```

▶egl_window_set_callback function

```
void  egl_window_set_callback(
    EGL_HANDLE  hWin,
    EVENT_CALLBACK  cb
);
```

Overview

Set callback function that will be called when window event occur.

Parameter

EGL_HANDLE hWin	Handle of window.
EVENT_CALLBACK cb	Function that is called when event occur.

Return Value

None.

Example

```
Void window_callback(EGL_HANDLE h, int event)
{
…
}
int main()
{
   …
   EGL_HANDLE hWin;
   hWin = egl_create_window("Main Window");
   egl_window_callback(hWin, window_callback);
   …
}
```

▶egl_window_get_active function

```
EGL_HANDLE  egl_window_get_active( void );
```

Overview

Return current window handle.

Parameter

None.

Return Value

Handle of current window.

Example

```
EGL_HANDLE hWin;
EGL_HANDLE active_win;
hWin = egl_create_window("Main Window");
...
active_win = egl_window_get_active( );
```

▶egl_user_touch_input function

```
void  egl_user_touch_input(
    BOOL  bPressed,
    EGL_POINT*  pt
);
```

Overview

Send touch message to Msg_handler.

Parameter

BOOL bPressed	State of touch.
EGL_POINT* pt	Coordinates of touch.

```
typedef struct _tagPOINT
{
    int x;
    int y;
} EGL_POINT;
```

Return Value

None.

Example

```
EGL_POINT touch_pt;
BOOL touchdown = FALSE;
...
if(process_touch(&touchdown, &touch_pt))
   egl_user_touch_input(touchdown, &touch_pt);
```

▶**egl_draw** function

```
void  egl_draw( void );
```

Overview

Draw object at current window.

Parameter

None.

Return Value

None.

Example

```
if(process_touch(&touchdown, &touch_pt))
    egl_user_touch_input(touchdown, &touch_pt);
egl_draw();
```

▶**egl_visible_object** function

```
void  egl_visible_object(
   EGL_HANDLE  h,
   Bool  b
 );
```

Overview

Set visible status of object.

Parameter

EGL_HANDLE h	Object handle.
Bool b	Visible status. TRUE : visible, FALSE : invisible

Return Value

None.

Example

```
egl_visible_object(button, FALSE);
...
egl_draw();
```

▶ **egl_window_delete_object** function

```
bool  egl_window_delete_object(
   EGL_HANDLE  hWin,
   EGL_HANDLE  hObj
 );
```

Overview

Delete object of window.

Parameter

EGL_HANDLE hWin	Window handle.
EGL_HANDLE hObj	Object handle.

Return Value

TRUE or FALSE

Example

```
egl_window_delete_object(hWin, button);
...
egl_draw();
```

▶egl_release_window function

```
bool  egl_release_window(
   EGL_HANDLE  hObj,
);
```

Overview

Release window object.

Parameter

EGL_HANDLE hObj Window handle.

Return Value

TRUE or FALSE

Example

```
If(hWin != NULL)
{
    egl_release_window(hWin);
    hWin = NULL;
}
```

▶ Window Example.

Example

```
#include "adstar.h"

static void btn1_callback(EGL_HANDLE h, int event)
{
     if( event == BTN_CLICKED )
     {
         debugprintf("btn1 clicked₩r₩n");
         egl_window_show( hWin2, TRUE );
       }
}

static void btn2_callback(EGL_HANDLE h, int event)
{
       if( event == BTN_CLICKED )
     {
         debugprintf("btn2 clicked₩r₩n");
         egl_window_show( hWin1, TRUE );
       }
}

extern BOOL process_touch(BOOL* touchdown, EGL_POINT* pPoint);

int main()
{
    ...
   EGL_POINT touch_pt;
   BOOL touchdown = FALSE;
   EGL_HANDLE hWin1;
   EGL_HANDLE hWin2;
   EGL_HANDLE btn1;
    EGL_HANDLE btn2;
    SURFACE* bg = loadbmp("bg.bmp");
    egl_init();
   hWin1 = egl_create_window("First Window");
   egl_windows_set_bg(hWin1, bg);
    hWin2 = egl_create_window("Second Window");
   btn1 = egl_create_button(100, 100, 200, 50, "Second Window");
```

```
    egl_btn_callback(btn1, btn1_callback);
   btn2 = egl_create_button(100, 200, 200, 50, "First Window");
    egl_btn_callback(btn2, btn2_callback);
   egl_window_add_object(hWin1, btn1);
   egl_window_add_object(hWin2, btn2);
   egl_window_show(hWin1);
    egl_draw();

   while(1)
   {
        if( process_touch(&touchdown, &touch_pt ) )
       egl_user_touch_input(touchdown, &touch_pt );

      egl_draw();
   }
}
```

Button

Function	Description
egl_create_button	Creates button object.
egl_button_callback	Set callback function that will be called when button event occur.
egl_release_button	Release button object

Define

BUTTON_EVENT

```
typedef enum
{
    BTN_CLICKED = 0,
    BTN_PRESSED,
    BTN_MAX,
}BUTTON_EVENT
```

▶**egl_create_button** function

```
EGL_HANDLE   egl_create_button(
      int   x,
      int   y,
      int   w,
      int   h,
      const char*   text
);
```

Overview

Creates button object.

Parameter

int x	X-coordinate of button.
int y	Y-coordinate of button.
int w	Width of button.
int h	Height of button.
const char* text	Text of button.

Return Value

Handle of button.

Example

```
EGL_HANDLE btn[2];

btn[0] = egl_create_button(100, 100, 100, 50, "Button1");

btn[1] = egl_create_button(300, 100, 100, 50, "Button2");
```

▶egl_button_callback function

```
BOOL  egl_button_callback(
      EGL_HANDLE  hObj,
      EVENT_CALLBACK  cb
);
```

Overview

Set callback function that will be called when button event occur.

Parameter

EGL_HANDLE hObj	Handle of button.
EVENT_CALLBACK cb	Function that is called when event occur.

Return Value

TRUE or FALSE

Example

```
void btn_callback(EGL_HANDLE h, int event)
{
If(event == BTN_CLICKED)
      {
            …
    }
}
int main()
{
   …
   EGL_HANDLE btn;
   btn = egl_create_button(100, 100, 100, 50, “Button1”);
   egl_button_callback(btn,btn_callback);
   …
}
```

▶**egl_release_button** function

```
BOOL  egl_release_button(
      EGL_HANDLE  hObj,
);
```

Overview

Release button object.

Parameter

EGL_HANDLE hObj	Handle of button.

Return Value

TRUE or FALSE

Example

```
if(button != NULL)
{
    egl_window_delete_object(hWin, button);
    egl_release_button(button);
    button = NULL;
}
```

▶ Button Example.

Example

```
void btn_callback(EGL_HANDLE h, int event)
{
      if(event == BTN_CLICKED)
      {
          debugprintf("buttton clicked.");
     }
}

extern BOOL process_touch(BOOL* touchdown, EGL_POINT* pPoint);

int main()
{
   …
    EGL_POINT touch_pt;
  BOOL touchdown = FALSE;
  EGL_HANDLE hWin;
  EGL_HANDLE btn;
  egl_init();
    btn = egl_create_button(100, 100, 100, 50, "Button Ex");
    egl_button_callback(btn,btn_callback);
  egl_window_add_object(hWin, btn);
  egl_window_show(hWin);
   egl_draw();

while(1)
{
      if( process_touch( &touchdown, &touch_pt ) )
     egl_user_touch_input( touchdown, &touch_pt );

    egl_draw();
 }
}
```

Image button

< img >

< pressed_img >

Function	Description
egl_create_image_button	Creates image button object.
egl_image_button_callback	Set callback function that will be called when image button event occur.
egl_release_image_button	Release image button.

▶**egl_create_image_button** function

```
EGL_HANDLE  egl_create_image_button(
    SURFACE*  img,
    SURFACE*  pressed_img,
    int  x,
    int  y,
    int  w,
    int  h
);
```

Overview

Creates image button object.

Parameter

SURFACE* img	Image of default button.
SURFACE* pressed_img	Image of pressed button.
int x	X-coordinate of image button.
int y	Y-coordinate of Image button.
int w	Width of image button.
int h	Height of image button.

Return Value

Handle of created image button.

Example

```
EGL_HADLE imagebtn;
SURFACE* img = loadbmp("btnimg.bmp");;
SURFACE* pressed_img = loadbmp("pressedimg.bmp");
imagebtn = egl_create_image_button(img, pressed_img, 100, 100, 150, 50);
```

▶egl_image_button_callback function

```
BOOL  egl_image_button_callback(
    EGL_HANDLE  hObj,
     EVENT_CALLBACK  cb
);
```

Overview

Set callback function that will be called when image button event occur.

Parameter

EGL_HADLE hObj	Handle of image button.
EVENT_CALLBACK cb	Function that is called when event occur.

Return Value

TRUE or FALSE

Example

```
void imgbtn_callback(EGL_HANDLE h, int event)
{
      …
}
int main()
{
   …
   EGL_HADLE imagebtn;
   SURFACE* img = loadbmp(“btnimg.bmp”);;
   SURFACE* pressed_img = loadbmp(“pressedimg.bmp”);
   imagebtn = egl_create_image_button(img, pressed_img, 100, 100, 150, 50);
   egl_image_button_callback(imgbtn, imgbtn_callback);
   …
}
```

▶egl_release_image_button function

```
BOOL   egl_release_image_button(
    EGL_HANDLE   hObj
);
```

Overview

Release image button.

Parameter

EGL_HADLE hObj	Handle of image button.

Return Value

TRUE or FALSE

Example

```
if(imgbtn != NULL)
{
    egl_window_delete_object(hWin, imgbtn);
    egl_release_image_button(imgbtn);
    imgbtn = NULL;
}
```

▶ Image Button Example.

Example

```
void imgbtn_callback(EGL_HANDLE h, int event)
{
      if(event == BTN_CLICKED)
      {
          debugprintf("image buttton clicked.");
      }
}

extern BOOL process_touch(BOOL* touchdown, EGL_POINT* pPoint);

int main()
{
   ...
    EGL_POINT touch_pt;
  BOOL  touchdown = FALSE;
  EGL_HANDLE hWin;
  EGL_HANDLE imgbtn;
    SURFACE* img = loadbmp("btnimg.bmp");;
    SURFACE* pressed_img = loadbmp("pressedimg.bmp");
    egl_init();
    imgbtn = egl_create_image_button(img, pressed_img, 100, 100, 150, 50);
    egl_button_callback(imgbtn, imgbtn_callback);
   egl_window_add_object(hWin, imgbtn);
   egl_window_show(hWin);
    egl_draw();

while(1)
{
      if( process_touch( &touchdown, &touch_pt ) )
      egl_user_touch_input( touchdown, &touch_pt );

    egl_draw();
  }
}
```

Toggle image button

< surf_on >

< surf_off >

Function	Description
egl_create_toggle_image	Creates toggle image button object.
egl_toggle_image_callback	Set callback function that will be called when toggle image button event occur.
egl_toggle_image_set_on	Set state of toggle image button.
egl_release_toggle_image	Release toggle image.

Define

TOGGLE_IMAGE_EVENT

```
typedef enum
{
    TOGGLE_IMAGE_ON = 0,
    TOGGLE_IMAGE_OFF,
} TOGGLE_IMAGE_EVENT;
```

▶egl_create_toggle_image function

```
EGL_HANDLE  egl_create_toggle_image(
    SURFACE*  surf_off,
    SURFACE*  surf_on,
    int  x,
    int  y,
    int  w,
    int  h
);
```

Overview

Creates toggle image button object.

Parameter

SURFACE* surf_off	Image of toggle imgae button when off state. (default state : 0ff)
SURFACE* surf_on	Image of toggle image button when on state.
int x	X-coordinate of toggle image button.
int y	Y-coordinate of toggle image button.
int w	Width of toggle image button.
int h	Height of toggle image button.

Return Value

Handle of created toggle image button.

Example

```
EGL_HADLE toggleimg;
SURFACE* surf_off = loadpng("off.png");;
SURFACE* surf_on = loadpng("on.png");
toggleimg = egl_create_toggle_image(surf_off, surf_on, 100, 100, 150, 50);
```

▶egl_toggle_image_callback function

```
BOOL  egl_toggle_image_callback(
    EGL_HANDLE  hObj,
    EVENT_CALLBACK  cb
);
```

Overview

Set callback function that will be called when toggle image button event occur.

Parameter

EGL_HANDLE hObj	Handle of toggle image button.
EVENT_CALLBACK cb	Function that is called when event occur.

Return Value

TRUE or FALSE

Example

```
void toggleimg_callback(EGL_HANDLE h, int event)
{
   if(event == TOGGLE_IMAGE_ON)
            …
    else
            …
}
int main()
{
   EGL_HADLE toggleimage;
   …
   toggleimage = egl_create_toggle_image(surf_off, surf_on, 100, 100, 150, 50);
   egl_toggle_image_callback(toggleimage, toggleimg_callback);
   …
}
```

▶egl_toggle_image_set_on function

```
BOOL  egl_toggle_image_set_on(
    EGL_HANDLE  hObj,
    BOOL  b
);
```

Overview

Set state of toggle image button.

Parameter

EGL_HANDLE hObj	Handle of toggle image button to set state.
BOOL b	Set value. TRUE => On FALSE => Off

Return Value

TRUE or FALSE

Example

```
EGL_HADLE toggleimg;
SURFACE* surf_off = loadpng("off.png");;
SURFACE* surf_on = loadpng("on.png");
toggleimg = egl_create_toggle_image(surf_off, surf_on, 100, 100, 150, 50);
egl_toggle_image_set_on(toggleimg, TRUE);
```

▶**egl_release_toggle_image** function

```
BOOL  egl_release_toggle_image(
    EGL_HANDLE  hObj
);
```

Overview

Release toggle image button.

Parameter

EGL_HANDLE hObj	Handle of toggle image button.

Return Value

TRUE or FALSE

Example

```
if( toggle != NULL)
{
    egl_window_delete_object(hWin, toggle);
    egl_release_toggle_image(toggle);
    toggle = NULL;
}
```

▶ Toggle Image Button Example.

Example

```
void toggleimg_callback(EGL_HANDLE h, int event)
{
    if(event == TOGGLE_IMAGE_ON)
            debugprintf(" toggle image on ₩r₩n");
    else
          debugprintf(" toggle image off ₩r₩n");
}

extern BOOL process_touch(BOOL* touchdown, EGL_POINT* pPoint);

int main()
{
    ...
    EGL_POINT touch_pt;
  BOOL  touchdown = FALSE;
  EGL_HANDLE  hWin;
   EGL_HANDLE  toggleimg;
    SURFACE* surf_off = loadpng("off.png");;
    SURFACE* surf_on = loadpng("on.png");
    egl_init();
    toggleimg = egl_create_toggle_image(surf_off, surf_on, 100, 100, 150, 50);
    egl_toggle_image_callback(toggleimg, toggleimg_callback);
    egl_toggle_image_set_on(toggleimg, TRUE);
   egl_window_add_object(hWin, toggleimg);
   egl_window_show(hWin);
    egl_draw();

while(1)
{
      if( process_touch( &touchdown, &touch_pt ) )
      egl_user_touch_input( touchdown, &touch_pt );

    egl_draw();
  }
}
```

Label

Function	Description
egl_create_label	Creates Label object.
egl_label_callback	Set callback function that will be called when label event occur.
egl_label_set_text	Set text of Label.
egl_label_set_redraw_bg	Set whether to draw back ground image of label.
egl_label_set_color	Set text color of label.
egl_release_label	Release label.

▶**egl_create_label** function

```
EGL_HANDLE   egl_create_label(
    int  x,
    int  y,
    int  w,
    int  h,
    const char* text
);
```

Overview

Creates Label object.

Parameter

int x	X-coordinate of label..
int y	Y-coordinate of label.
int w	Width of label.
int h	Height of label.
const char* text	Text of label.

Return Value

Handle of created label.

Example

```
EGL_HADLE label;
label = egl_create_label( 100, 100, 150, 50, "label test");
```

▶**egl_label_callback** function

```
BOOL  egl_label_calback(
      EGL_HANDLE  hObj,
      EVENT_CALLBACK  cb
);
```

Overview

Set callback function that will be called when label event occur.

Parameter

EGL_HANDLE hObj	Handle of label.
EVENT_CALLBACK cb	Function that is called when event occur.

Return Value

TRUE or FALSE

Example

```
void label_callback(EGL_HANDLE  h, int  event)
{
    If(event == LABEL_CLICKED)
    {
       ....
    }
}

int main()
{
    EGL_HANDLE  label;
    ....
    label = egl_create_label(100,100,150,50,"label");
    egl_label_callback(label, label_callback);
    ...
}
```

▶egl_label_set_text function

```
BOOL   egl_label_set_text(
    EGL_HANDLE   h,
      char*   text
);
```

Overview

Set text of Label.

Parameter

EGL_HANDLE h	Handle of label.
char* text	Text of label.

Return Value

TRUE or FALSE

Example

```
EGL_HADLE label;
label = egl_create_label( 100, 100, 150, 50, "label test" );
egl_label_set_text( label, "change text");
```

▶**egl_label_set_redraw_bg** function

```
void  egl_label_set_redraw_bg(
    EGL_HANDLE  h,
    BOOL  b
);
```

Overview

Set whether to draw back ground image of label. If the sole object is used, it should be set to True.

Parameter

EGL_HANDLE hObj	Handle of label.
BOOL b	Set value. TRUE or FALSE

Return Value

None.

Example

```
EGL_HADLE label;
label = egl_create_label( 100, 100, 150, 50, "label test" );
egl_label_set_redraw_bg(label, TRUE);
```

▶**egl_label_set_color** function

```
void  egl_label_set_color(
    EGL_HANDLE  h,
     EGL_COLOR  clr
);
```

Overview

Set text color of label.

Parameter

EGL_HANDLE h	Handle of label.
EGL_COLOR clr	Color value. Use MAKE_COLORREF(r, g, b) macro function.

Return Value

None.

Example

```
EGL_HADLE label;
label = egl_create_label( 100, 100, 150, 50, "label test" );
egl_label_set_color( label, MAKE_COLORREF( 0, 255, 0 ));
```

▶**egl_release_label** function

```
BOOL   egl_release_label(
    EGL_HANDLE   hObj
);
```

Overview

Release label.

Parameter

EGL_HANDLE hObj	Handle of label.

Return Value

TRUE or FALSE

Example

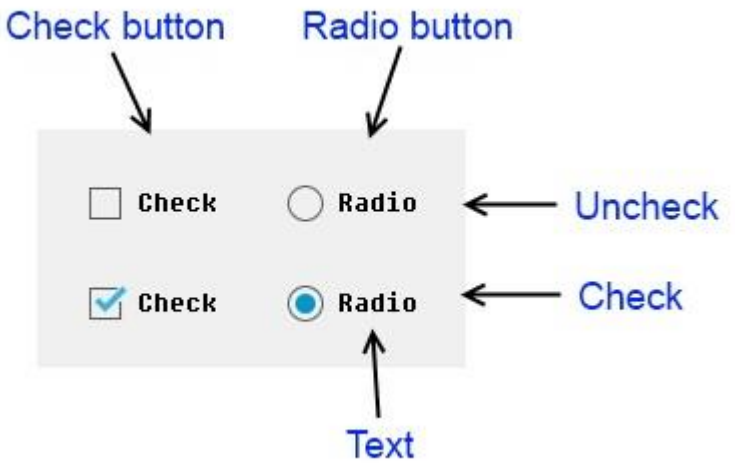
```
if(label != NULL)
{
    egl_window_delete_object(hWin, label);
    egl_release_label(label);
    label = NULL;
}
```

▶ Label Example.

Example

```
extern BOOL process_touch(BOOL* touchdown, EGL_POINT* pPoint);

int main()
{
    ...
    EGL_POINT touch_pt;
  BOOL touchdown = FALSE;
  EGL_HANDLE hWin;
   EGL_HANDLE label;
  egl_init();
    label = egl_create_label(50, 50, 100, 30, "label");
    egl_label_set_redraw_bg(label, TRUE);
    egl_label_set_color(label, MAKE_COLORREF(0, 0, 255);
    egl_window_add_object(hWin, label);
  egl_window_show(hWin);
   egl_draw();

while(1)
{
      if( process_touch( &touchdown, &touch_pt ) )
      egl_user_touch_input( touchdown, &touch_pt );

    egl_draw();
  }
}
```

Check button / Radio button



Function	Description
egl_create_checkbutton	Creates Check / Radio button.
egl_checkbutton_callback	Set callback function that will be called when Check / Radio button event occur.
egl_checkbutton_set_check	Set state of Check / Radio button. (check or uncheck)
egl_checkbutton_get_check	Get state of Check / Radio button.
egl_checkbutton_set_style	Set style of Check / Radio button.
egl_release_checkbutton	Release check / radio button.

Define	
CHECK_EVENT	typedef enum { CHECK_CHECKED = 0, CHECK_UNCHECKED,, } CHECK_EVENT;

▶egl_create_checkbutton function

```
EGL_HANDLE   egl_create_checkbuttonk(
      int   x,
      int   y,
      int   w,
      int   h,
      const char*   text,
      CHECK_STYLE   style
);
```

Overview

Creates Check / Radio button.

Parameter

int x	X-coordinate of Check / Radio button.
int y	Y-coordinate of Check / Radio button.
int w	Width of Check / Radio button.
int h	Height of Check / Radio button.
const char* text	Text of Check / Radio button.
CHECK_STLE style	Style of Check / Radio button. Check button ->CHECK_STYLE_CHECKBUTTON. Radio button ->CHECK_STYLE_RADIOBUTTON.

Return Value

Handle of created Check / Radio button.

Example

```
EGL_HANDLE check;
EGL_HANDLE radio;
check = egl_create_checkbutton(100,100, 100, 32, "Check", CHECK_STYLE_CHECKBUTTON);
radio = egl_create_checkbutton(250,  100, 100, 32, "Radio", CHECK_STYLE_RADIOBUTTON);
```

▶**egl_checkbutton_callback** function

```
BOOL  egl_button_callback(
      EGL_HANDLE  hObj,
      EVENT_CALLBACK  cb
);
```

Overview

Set callback function that will be called when Check / Radio button event occur.

Parameter

EGL_HANDLE hObj	Handle of Check / Radio button.
EVENT_CALLBACK cb	Function that is called when event occur.

Return Value

TRUE or FALSE

Example

```
Void check_callback(EGL_HANDLE h, int event)
{
if(event == CHECK_CHECKED)
…
     else if(event == CHECK_UNCHECKED)
…
}
int main()
{
   …
   EGL_HANDLE check;
   check = egl_create_checkbutton(100, 100, 100, 32, "Check", STYLE_CHECKBUTTON);
   egl_checkbutton_callback(check, check_callback);
   …
}
```

▶egl_checkbutton_set_check function

```
void egl_checkbutton_set_check(
    EGL_HANDLE  hObj,
    BOOL  b
);
```

Overview

Set state of Check / Radio button. (check or uncheck)

Parameter

EGL_HANDLE hObj	Handle of Check / Radio button.
BOOL b	Set value. TRUE = Check. FALSE = Uncheck.

Return Value

None.

Example

```
EGL_HANDLE check;

EGL_HANDLE radio;

check = egl_create_checkbutton(100, 100, 100, 32, "Check", STYLE_CHECKBUTTON);

radio = egl_create_checkbutton(200, 100, 100, 32, "Radio", STYLE_RADIOBUTTON);

egl_checkbutton_set_check(check, TRUE);

egl_checkbutton_set_check(radio, FASLE);
```

▶egl_checkbutton_get_check function

```
BOOL egl_checkbutton_get_check(
    EGL_HANDLE  hObj,
);
```

Overview

Get state of Check / Radio button.

Parameter

EGL_HANDLE hObj	Handle of Check / Radio button.

Return Value

State. TRUE = check.
FALSE = uncheck.

Example

```
EGL_HANDLE check;

EGL_HANDLE radio;

BOOL check_status = 0;

BOOL radio_status = 0;

check = egl_create_checkbutton(100, 100, 100, 32, "Check", STYLE_CHECKBUTTON);

radio = egl_create_checkbutton(200, 100, 100, 32, "Radio", STYLE_RADIOBUTTON);

check_status = egl_checkbutton_get_check(check);

radio_status = egl_checkbutton_get_check(radio);
```

▶egl_checkbutton_set_style function

```
void egl_checkbutton_set_style(
    EGL_HANDLE  hObj,
     CHECK_STYLE  style
);
```

Overview

Set style of Check / Radio button.

Parameter

EGL_HANDLE hObj	Handle of Check / Radio button.
CHECK_STYLE style	Style of Check / Radio button. Check button ->CHECK_STYLE_CHECKBUTTON. Radio button ->CHECK_STYLE_RADIOBUTTON.

Return Value

None.

Example

```
EGL_HANDLE check;

EGL_HANDLE radio;

check = egl_create_checkbutton(100,  100, 100, 32, "Check", CHECK_STYLE_CHECKBUTTON);

radio = egl_create_checkbutton(200, 100, 100, 32, "Radio", CHECK_STYLE_RADIOBUTTON);

egl_checkbutton_set_style(check, CHECK_STYLE_RADIOBUTTON);

egl_checkbutton_set_style(radio, CHECK_STYLE_CHECKBUTTON);
```

▶egl_release_checkbutton function

```
BOOL egl_release_checkbutton(
    EGL_HANDLE  hObj
);
```

Overview

Release check / radio button.

Parameter

EGL_HANDLE hObj	Handle of Check / Radio button.

Return Value

TRUE or FALSE

Example

```
if(check != NULL)
{
    egl_window_delete_object(hWin, check);
    egl_release_checkbutton(check);
    check = NULL;
}
```

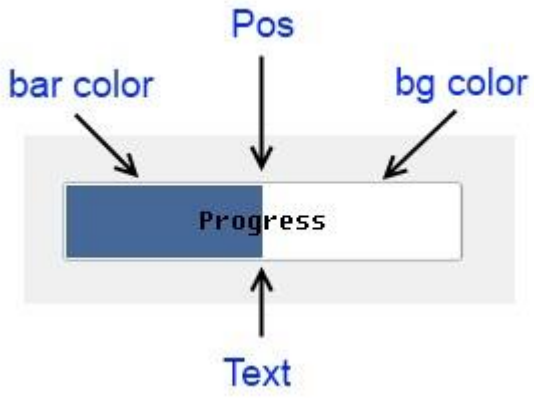
▶ Check / Radio button Example.

Example

```
 EGL_HANDLE check;
 EGL_HANDLE radio1;
 EGL_HANDLE radio2;

void check_callback(EGL_HANDLE h, int event)
{
    if(event == CHECK_CHECKED)
    {
        if(egl_checkbutton_get_check())
        {
            debugprintf("status : checked₩r₩n");
        }
        else
        {
            debugprintf("status : unchecked₩r₩n");
           }
     }
}

void radio1_callback(EGL_HANDLE h, int event)
{
    if(event == CHECK_CHECKED)
    {
        debugprintf("radio1 select₩r₩n");
        egl_checkbutton_set_check(radio2, FALSE);
     }
}

void radio2_callback(EGL_HANDLE h, int event)
{
    if(event == CHECK_CHECKED)
    {
        debugprintf("radio2 select₩r₩n");
        egl_checkbutton_set_check(radio1, FALSE);
     }
```

```
}

extern BOOL process_touch(BOOL* touchdown, EGL_POINT* pPoint);

int main()
{
   ...
   EGL_POINT touch_pt;
  BOOL touchdown = FALSE;
  EGL_HANDLE hWin;

   egl_init();
    check = egl_create_checkbutton(100, 100, 100, 32, "Check", CHECK_STYLE_CHECKBUTTON);
   egl_checkbutton_callback(check, check_callback);
    radio1 = egl_create_checkbutton(250, 100, 100, 32, "Radio1", CHECK_STYLE_RADIOBUTTON);
   egl_checkbutton_callback(radio1, radio1_callback);
    radio2 = egl_create_checkbutton(250, 160, 100, 32, "Radio2", CHECK_STYLE_RADIOBUTTON);
   egl_checkbutton_callback(radio2, radio2_callback);
    egl_window_add_object(hWin, check);
    egl_window_add_object(hWin, radio1);
    egl_window_add_object(hWin, radio2);
   egl_window_show(hWin);
    egl_draw();

while(1)
{
      if( process_touch( &touchdown, &touch_pt ) )
      egl_user_touch_input( touchdown, &touch_pt );

    egl_draw();
 }
}
```

Progress Bar



Function	Description
egl_create_progressbar	Creates progress bar.
egl_progressbar_set_barcolor	Set color of Progress bar.
egl_progressbar_set_bgcolor	Set back ground color of Progressbar.
egl_progressbar_set_textcolor	Set text color of Progressbar.
egl_progressbar_set_text	Set text of Progressbar.
egl_progressbar_set_pos	Set bar position of Progressbar.
egl_progressbar_get_pos	Get current bar position of Progress bar.
egl_release_progressbar	Release progressbar

▶**egl_create_progressbar** function

```
EGL_HANDLE  egl_create_progressbar(
    int  x,
    int  y,
    int  w,
    int  h,
    const  char* text,
   BOOL  style
   BOOL  bVertical
);
```

Overview

Creates progress bar.

Parameter

int x	X-coordinate of progress bar.
int y	Y-coordinate of progress bar.
int w	Width of progress bar.
int h	Height of progress bar.
const char* text	Text of progress bar.
BOOL style	Style of progress bar. Nomal = STYLE_PGBAR_NOMAL. Divide bar = STYLE_PGVAR_DIV.
BOOL bVertical	Direction of Progress bar. TRUE = Vertical. FALSE = Horizontal.

Return Value

Handle of created progress bar.

Example

```
EGL_HANDLE  pgbar;

pgbar = egl_create_progressbar(100, 100, 200, 40, "Progress", STYLE_PGBAR_NOMAL,FALSE);
```

▶**egl_progressbar_set_barcolor** function

```
void egl_progressbar_set_barcolor(
    EGL_HANDLE  hObj,
    EGL_COLOR  clr

);
```

Overview

Set color of progress bar.

Parameter

EGL_HANDLE hObj	Handle of progress bar.
EGL_COLOR clr	Color value. Use MAKE_COLORREF(r, g, b) macro function.

Return Value

None.

Example

```
EGL_HANDLE pgbar;

pgbar = egl_create_progressbar(100,100,200,40,"Progress", STYLE_PGBAR_NOMAL, FALSE);

egl_progressbar_set_barcolor(pgbar, MAKE_COLORREF(0, 0, 0xff));
```

▶egl_progressbar_set_bgcolor function

```
void egl_progressbar_set_bgcolor(
    EGL_HANDLE   hObj,
    EGL_COLOR   clr
);
```

Overview

Set back ground color of progress bar.

Parameter

EGL_HANDLE hObj	Handle of progress bar.
EGL_COLOR clr	Color value. Use MAKE_COLORREF(r, g, b) macro function.

Return Value

None.

Example

```
EGL_HANDLE pgbar;

pgbar = egl_create_progressbar(100,100,200,40,"Progress", STYLE_PGBAR_NOMAL, FALSE);

egl_progressbar_set_bgcolor(pgbar, MAKE_COLORREF(0, 0xff, 0));
```

▶egl_progressbar_set_textcolor function

```
void egl_progressbar_set_textcolor(
    EGL_HANDLE  hObj,
    EGL_COLOR  clr
);
```

Overview

Set text color of progress bar.

Parameter

EGL_HANDLE hObj	Handle of progress bar.
EGL_COLOR clr	Color value. Use MAKE_COLORREF(r, g, b) macro function.

Return Value

None.

Example

```
EGL_HANDLE pgbar;

pgbar = egl_create_progressbar(100,100,200,40,"Progress", STYLE_PGBAR_NOMAL, FALSE);

egl_progressbar_set_textcolor(pgbar, MAKE_COLORREF(0xff, 0, 0xff));
```

▶egl_progressbar_set_text function

```
void egl_progressbar_set_text(
    EGL_HANDLE   hObj,
    const char*  text
);
```

Overview

Set text of progress bar.

Parameter

EGL_HANDLE hObj	Handle of progress bar.
const char* text	Text of progress bar..

Return Value

None.

Example

```
    EGL_HANDLE pgbar;

    pgbar = egl_create_progressbar(100,100,200,40,"Progress", STYLE_PGBAR_NOMAL,
FALSE);

    egl_progressbar_set_text(pgbar, "text1");
```

▶**egl_progressbar_set_pos** function

```
void egl_progressbar_set_pos(
    EGL_HANDLE  hObj,
    int  value
);
```

Overview

Set bar position of progress bar.

Parameter

EGL_HANDLE hObj	Handle of progress bar.
int value	Bar position value. (0 ~ 100%)

Return Value

None.

Example

```
    EGL_HANDLE pgbar;

    pgbar    =    egl_create_progressbar(100,100,200,40,"Progress",    STYLE_PGBAR_NOMAL,
FALSE);

    egl_progressbar_set_pos(pgbar, 50);
```

▶**egl_progressbar_get_pos** function

```
int egl_progressbar_get_pos(
    EGL_HANDLE  hObj
);
```

Overview

Get current bar position of progress bar.

Parameter

EGL_HANDLE hObj	Handle of progress bar.

Return Value

Bar position value. (0 ~ 100%)

Example

```
    EGL_HANDLE pgbar;

    int pgbar_pos;

    pgbar = egl_create_progressbar(100,100,200,40,"Progress", STYLE_PGBAR_NOMAL,
FALSE);

    pgbar_pos = egl_progressbar_get_pos(pgbar);
```

▶**egl_release_progressbar** function

```
BOOL egl_release_progressbar(
    EGL_HANDLE  hObj
);
```

Overview

Release progressbar.

Parameter

EGL_HANDLE hObj	Handle of progress bar.

Return Value

TRUE or FALSE

Example

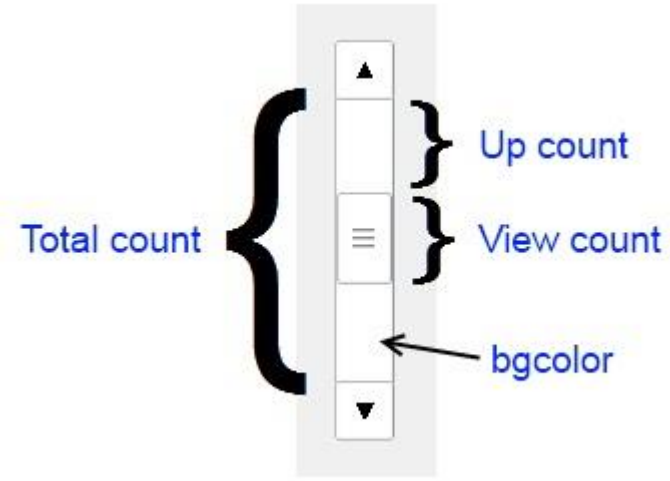
```
if(pgbar != NULL)
{
    egl_window_delete_object( hWin, pgbar );
    egl_release_progressbar(pgbar);
    pgbar = NULL;
}
```

▶ Progress Bar Example.

Example

```
EGL_HANDLE pgbar;
int value = 0;

void pgbar_callback(EGL_HANDLE h, int event)
{
        value++;
        if(value >100)
            value = 0;
      egl_progressbar_set_pos(pgbar, value);
}

extern BOOL process_touch(BOOL* touchdown, EGL_POINT* pPoint);

int main()
{
   ...
   EGL_POINT  touch_pt;
  BOOL  touchdown = FALSE;
  EGL_HANDLE  hWin;
   egl_init();
    pgbar = egl_create_progressbar(100, 100, 200, 40, "progress", STYLE_PGBAR_NOMAL,FALSE);
   egl_progress_callback(pgbar, pgbar_callback);
   egl_window_add_object(hWin, pgbar);
    egl_window_show(hWin);
   egl_draw();

while(1)
{
      if( process_touch( &touchdown, &touch_pt ) )
      egl_user_touch_input( touchdown, &touch_pt );

    egl_draw();
 }
}
```

Scroll Bar



Function	Description
egl_create_scrollbar	Create Scroll bar.
egl_scrollbar_callback	Set callback function that will be called when scroll bar event occur.
egl_scroll_set_position	Set thumb position of scroll bar.
egl_scroll_get_position	Get thumb position of Scroll bar.
egl_scroll_set_totalcount	Set total count of scroll bar.
egl_scroll_get_totalcount	Get total count of scroll bar.
egl_scroll_set_view_count	Set view count of Scroll bar.
egl_scroll_get_view_count	Get view count of scroll bar.
egl_scroll_set_upcount	Set up count of scroll bar.
egl_scroll_get_upcount	Get up count of Scroll bar.
egl_scroll_set_bgcolor	Set back ground color of scroll bar.
egl_scroll_set_size	Set size of scrollbar.

Function	Description
egl_release_scrollbar	Release scrollbar.
Define	
SCROLLBAR_EVENT	typedef enum { SCBAR_CLICKED = 0, } SCROLLBAR_EVENT;

▶**egl_create_scrollbar** function

```
EGL_HANDLE   egl_create_scrollbar(
    int  x,
    int  y,
    int  w,
    int  h,
    int  totalcount,
    int  viewcount,
    BOOL bVertical
);
```

Overview

Create scroll bar.

Parameter

int x	X-coordinate of scroll bar.
int y	Y-coordinate of scroll bar.
int w	Width of scroll bar.
int h	Height of scroll bar.
int totalcount	Total count of item.
int viewcount	Output count of item on 1 page.
BOOL bVertical	Direction of scroll bar. TRUE = Vertical. FALSE = Horizontal.

Return Value

Handle of created scrikk bar.

Example

```
EGL_HANDLE scroll;

scroll = egl_create_scrollbar(100, 100, 30, 200, 30, 10, TRUE);
```

▶egl_scrollbar_callback function

```
BOOL egl_scrollbar_callback(
    EGL_HANDLE  hObj,
    EVENT_CALLBACK  cb
);
```

Overview

Set callback function that will be called when scroll bar event occur.

Parameter

ELG_HANDLE hObj	Handle of scroll bar.
EVENT_CALLBACK cb	Function that is called when event occur.

Return Value

TRUE or FALSE

Example

```
void scroll_callback(EGL_HANDLE h, int event)
{

}
int main()
{
   ...
   EGL_HANDLE scroll;
   scroll = egl_create_scrollbar(100,100, 30, 200, 30, 10, TRUE);
   egl_scroll_callback(check, scroll_callback);
   ...
}
```

▶egl_scroll_set_position function

```
void  egl_scroll_set_position(
      EGL_HANDLE  hObj,
    int  totalcount,
    int  viewcount,
    int  upcount
);
```

Overview

Set thumb position of scroll bar.

Parameter

EGL_HANDLE hObj	Handle of scroll bar.
int totalcount	Total count of item.
int viewcount	Output count of item on 1 page.
int upcount	Count of item on previous pages.

Return Value

None.

Example

```
EGL_HANDLE scroll;

scroll = egl_create_scrollbar(100,100, 30, 200, 30, 10, TRUE);

egl_scroll_set_position(scroll, 25, 5, 10);
```

▶egl_scroll_get_position function

```
void  egl_scroll_get_position(
    EGL_HANDLE  hObj,
    int*  totalcount,
    int*  viewcount,
    int*  upcount
);
```

Overview

Get thumb position of scroll bar.

Parameter

EGL_HANDLE hObj	Handle of scroll bar.
int* totalcount	Pointer variable for storing total count.
int* viewcount	Pointer variable for storing view count.
int* upcount	Pointer variable for storing up count.

Return Value

None.

Example

```
EGL_HANDLE scroll;
int totalcount = 0;
int viewcount = 0;
int upcount = 0;
scroll = egl_create_scrollbar(100,100, 30, 200, 30, 10, TRUE);
egl_scroll_get_position(scroll, &totalcount, &viewcount, &upcount);
```

▶egl_scroll_set_totalcount function

```
void  egl_scroll_set_totalcount(
    EGL_HANDLE  hObj,
    int  totalcount
);
```

Overview

Set total count of scroll bar.

Parameter

EGL_HANDLE hObj	Handle of scroll bar.
int totalcount	Total count of item.

Return Value

None.

Example

```
EGL_HANDLE scroll;

scroll = egl_create_scrollbar(100,100, 30, 200, 30, 10, TRUE);

egl_scroll_set_totalcount(scroll, 45);
```

▶egl_scroll_get_totalcount function

```
int  egl_scroll_get_totalcount(
    EGL_HANDLE  hObj
);
```

Overview

Get total count of scroll bar..

Parameter

EGL_HANDLE hObj	Handle of scroll bar.

Return Value

Total count value.

Example

```
EGL_HANDLE scroll;
int totalcount = 0;
scroll = egl_create_scrollbar(100,100, 30, 200, 30, 10, TRUE);
totalcount = egl_scroll_get_totalcount(scroll);
```

▶egl_scroll_set_viewcount function

```
void  egl_scroll_set_viewcount(
    EGL_HANDLE  hObj,
    int  viewcount
);
```

Overview

Set view count of scroll bar.

Parameter

EGL_HANDLE hObj	Handle of scroll bar.
int viewcount	Output count of item on 1 page.

Return Value

None.

Example

```
EGL_HANDLE scroll;

scroll = egl_create_scrollbar(100,100, 30, 200, 30, 10, TRUE);

egl_scroll_set_viewcount(scroll, 5);
```

▶egl_scroll_get_viewcount function

```
int  egl_scroll_get_viewcount(
    EGL_HANDLE  hObj
);
```

Overview

Get view count of scroll bar.

Parameter

EGL_HANDLE hObj	Handle of scroll bar.

Return Value

View count value.

Example

```
EGL_HANDLE scroll;
int viewcount = 0;
scroll = egl_create_scrollbar(100,100, 30, 200, 30, 10, TRUE);
viewcount = egl_scroll_get_viewcount(scroll);
```

▶egl_scroll_set_upcount function

```
void  egl_scroll_set_upcount(
    EGL_HANDLE  hObj,
    int  upcount
);
```

Overview

Set upcount of scroll bar.

Parameter

EGL_HANDLE hObj	Handle of scroll bar.
int upcount	Count of item on previous pages.

Return Value

None.

Example

```
EGL_HANDLE scroll;

scroll = egl_create_scrollbar(100,100, 30, 200, 30, 10, TRUE);

egl_scroll_set_upcount(scroll, 10);
```

▶egl_scroll_get_upcount function

```
int  egl_scroll_get_upcount(
    EGL_HANDLE  hObj
);
```

Overview

Get up count of scroll bar.

Parameter

EGL_HANDLE hObj	Handle of scroll bar.

Return Value

Up count value.

Example

```
EGL_HANDLE scroll;
int upcount = 0;
scroll = egl_create_scrollbar(100,100, 30, 200, 30, 10, TRUE);
upcount = egl_scroll_get_upcount(scroll);
```

▶egl_scroll_set_bgcolor function

```
void  egl_scroll_set_bgcolor(
    EGL_HANDLE  hObj,
    unsigned char  r,
    unsigned char  g,
    unsigned char  b
)
```

Overview

Set back ground color of scroll bar.

Parameter

EGL_HANDLE hObj	Handle of scroll bar.
unsigned char r	Red color of RGB.
unsigned char g	Green color of RGB.
unsigned char b	Blue color of RGB.

Return Value

None.

Example

```
EGL_HANDLE scroll;

scroll = egl_create_scrollbar(100,100, 30, 200, 30, 10, TRUE);

egl_scroll_set_bgcolor(scroll, 0, 0, 0xff);
```

▶egl_scroll_set_size function

```
void  egl_scroll_set_size(
    EGL_HANDLE  hObj,
    int  w,
    int  h
)
```

Overview

Set size of scroll bar.

Parameter

EGL_HANDLE hObj	Handle of scroll bar.
int w	Width of scroll bar.
int h	Height of scroll bar.

Return Value

None.

Example

```
EGL_HANDLE scroll;
scroll = egl_create_scrollbar(100,100, 30, 200, 30, 10, TRUE);
...
egl_scroll_set_size(scroll, 40, 200);
```

▶egl_release_scrollbar function

```
BOOL   egl_release_scrollbar(
    EGL_HANDLE   hObj
)
```

Overview

Release scrollbar.

Parameter

EGL_HANDLE hObj	Handle of scroll bar.

Return Value

TRUE or FALSE

Example

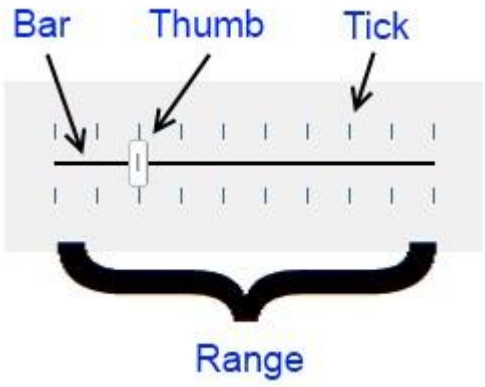
```
if(scroll != NULL)
{
    egl_window_delete_object( hWin, scroll );
    egl_release_scrollbar( scroll );
    scroll = NULL;
}
```

▶ Scroll Bar Example.

Example

```
EGL_HANDLE scroll;
void scroll_callback(EGL_HANDLE h, int event)
{
      int totalcount = 0;
      int viewcount = 0;
      int upcount = 0;

     if(event == SCBAR_CLICKED)
     {
          egl_scroll_get_position(scroll, &totalcount, &viewcount, &upcount);
          debugprintf("%d, %d, %d ₩r₩n", totalcount, viewcount, upcount);
       }
}
extern BOOL process_touch(BOOL* touchdown, EGL_POINT* pPoint);
int main()
{
   …
   EGL_POINT  touch_pt;
  BOOL  touchdown = FALSE;
  EGL_HANDLE  hWin;
   egl_init();
    scroll = egl_create_scrollbar(100, 100, 30, 200, 30, 10, TRUE);
    egl_scroll_set_totalcount(scroll, 40);
   egl_scroll_callback(scroll, scroll_callback);
   egl_window_add_object(hWin, scroll);
    egl_window_show(hWin);
   egl_draw();
while(1)
{
      if( process_touch( &touchdown, &touch_pt ) )
      egl_user_touch_input( touchdown, &touch_pt );

    egl_draw();
  }
}
```

Slider



Function	Description
egl_create_slider	Create slider.
egl_slider_callback	Set callback function that will be called when slider event occur.
egl_slider_set_pos	Set thumb position of slider.
egl_slider_get_pos	Get thumb position of slider.
egl_slider_set_range	Set range of slider.
egl_slider_get_range	Get range of slider.
egl_slider_stepit	Thumb is moved one step, increase or decrease.
egl_slider_set_tick_frequency	Set tick frequency of slider.
egl_slider_set_tick_style	Set tick style of slider.
egl_slider_set_thumb_size	Set thumb size of slider.
egl_slider_get_thumb_size	Get thumb size of slider.
egl_slider_set_barcolor	Set bar color of slider.
egl_slider_set_tickcolor	Set tick color of slider.

Function	Description
egl_slider_set_transparent	Set background transparent mode of slider.
egl_release_slider	Release slider
define	
SLIDER_EVENT	typedef enum { SLR_CLICKED = 0, } SLIDER_EVENT;

▶**egl_create_slider** function

```
EGL_HANDLE   egl_create_slider(
    int   x,
    int   y,
    int   w,
    int   h,
    int   range,
    TICKSTYLE   style,
    BOOL   bVertical
);
```

Overview

Create slider.

Parameter

int x	X-coordinate of slider.
int y	Y-coordinate of slider.
int w	Width of slider.
int h	Height of slider.
int range	Range of slider.
TICKSTYLE style	Tick style of slider. TICK_NONE : None. TICK_TOPLEFT : Output tick at top or left. TICK_BOTTOMRIGHT : Output tick at bottom or right. TICK_BOTH : Output tick at both.
BOOL bVertical	Direction of slider. TRUE = Vertical. FALSE = Horizontal.

Return Value

Handle of created slider.

Example

```
EGL_HANDLE slider;
slider = egl_create_slider(100, 100, 200, 40, 10, TICK_BOTH, FALSE);
```

▶egl_slider_callback function

```
BOOL  egl_slider_callback(
    EGL_HANDLE  hObj,
    EVENT_CALLBACK  cb
);
```

Overview

Set callback function that will be called when slider event occur.

Parameter

EGL_HANDLE hObj	Handle of slider.
EVENT_CALLBACK cb	Function that is called when event occur.

Return Value

TRUE or FALSE

Example

```
Void slider_callback(EGL_HANDLE h, int event)
{

}
int main()
{
   …
   EGL_HANDLE slider;
   slider = egl_create_slider(100,100, 200, 40, 10, TICK_BOTH, FALSE);
   egl_slider_callback(check, slider_callback);
   …
}
```

▶egl_slider_set_pos function

```
void  egl_slider_set_pos(
    EGL_HANDLE  hObj,
    int  nPos
);
```

Overview

Set thumb position of slider.

Parameter

EGL_HANDLE hObj	Handle of slider.
int nPos	Position value. (nPos is integer. 0 ~ range)

Return Value

None.

Example

```
EGL_HANDLE slider;
slider = egl_create_slider(100,100, 200, 40, 10, TICK_BOTH, FALSE);
egl_slider_set_pos(slider, 2);
```

▶egl_slider_get_pos function

```
int  egl_slider_get_pos(
    EGL_HANDLE  hObj
);
```

Overview

Get thumb position of slider.

Parameter

EGL_HANDLE hObj	Handle of slider.

Return Value

Position value.

Example

```
EGL_HANDLE slider;
int slider_pos = 0;
slider = egl_create_slider(100,100, 200, 40, 10, TICK_BOTH, FALSE);
slider_pos = egl_slider_get_pos(slider);
```

▶**egl_slider_set_range** function

```
void  egl_slider_set_range(
    EGL_HANDLE  hObj,
    int  nMinPos,
    int  nMaxPos
);
```

Overview

Set range of slider.

Parameter

EGL_HANDLE hObj	Handle of slider.
int nMinPos	Min position value. (range = Max position – Min position).
int nMaxPos	Max position value. (range = Max position – Min position).

Return Value

None.

Example

```
EGL_HANDLE slider;
slider = egl_create_slider(100,100, 200, 40, 10, TICK_BOTH, FALSE);
egl_slider_set_range(slider,  0, 15);
```

▶**egl_slider_get_range** function

```
void egl_slider_get_range(
    EGL_HANDLE  hObj,
     int*  lpMinPos,
     int*  lpMaxPos
);
```

Overview

Get range of slider.

Parameter

EGL_HANDLE hObj	Handle of slider.
int* lpMinPos	Pointer variable for storing min position value.
int* lpMaxPos	Pointer variable for storing max position value.

Return Value

None.

Example

```
EGL_HANDLE slider;
int max_pos = 0;
int min_pos = 0;
slider = egl_create_slider(100, 100, 200, 40, 10, TICK_BOTH, FALSE);
egl_slider_get_range(slider, &min_pos, &max_pos);
```

▶**egl_slider_stepit** function

```
void  egl_slider_stepit(
    EGL_HANDLE  hObj,
    BOOL  inc
);
```

Overview

Thumb is moved one step, increase or decrease.

Parameter

EGL_HANDLE hObj	Handle of slider.
BOOL inc	Set increase / decrease. TRUE = increase. FALSE = decrease.

Return Value

None.

Example

```
EGL_HANDLE slider;
slider = egl_create_slider(100, 100, 200, 40, 10, TICK_BOTH, FALSE);
egl_slider_stepit(slider, TRUE);
//egl_slider_stepit(slider, FALSE);
```

▶egl_slider_set_tick_frequency function

```
void  egl_slider_set_tick_frequency(
    EGL_HANDLE  hObj,
    int freq
);
```

Overview

Set tick frequency of slider.

Parameter

EGL_HANDLE hObj	Handle of slider.
int freq	Tick frequency.

Return Value

None.

Example

```
EGL_HANDLE slider;
slider = egl_create_slider(100, 100, 200, 40, 10, TICK_BOTH, FALSE);
egl_slider_set_tick_frequency(slider, 2);
```

▶egl_slider_set_tick_style function

```
void  egl_slider_set_tick_frequency(
    EGL_HANDLE  hObj,
    TICKSTYLE  style
);
```

Overview

Set tick style of slider.

Parameter

EGL_HADNE hObj	Handle of slider.
TICKSTYLE style	Tick style of slider. TICK_NONE : None. TICK_TOPLEFT : Output tick at top or left. TICK_BOTTOMRIGHT : Output tick at bottom or right. TICK_BOTH : Output tick at both.

Return Value

None.

Example

```
EGL_HANDLE slider;
slider = egl_create_slider(100, 100, 200, 40, 10, TICK_BOTH, FALSE);
egl_slider_set_tick_style(slider, TICK_BOTTOMRIGHT);
```

▶egl_slider_set_thumb_size function

```
void  egl_slider_set_thumb_size(
    EGL_HANDLE  hObj,
    int  width,
    int  height
);
```

Overview

Set thumb size of slider.

Parameter

EGL_HANDLE hObj	Handle of slider.
int width	Width of thumb.
int height	Height of thumb.

Return Value

None.

Example

```
EGL_HANDLE slider;
slider = egl_create_slider(100, 100, 200, 40, 10, TICK_BOTH, FALSE);
egl_slider_set_thumb_size(slider, 10, 30);
```

▶egl_slider_get_thumb_size function

```
void  egl_slider_get_thumb_size(
    EGL_HANDLE  hObj,
    int*  w,
    int*  h
)
```

Overview

Get thumb size of slider.

Parameter

EGL_HANDLE hObj	Handle of slider.
int* w	Pointer variable for storing width of thumb.
int* h	Pointer variable for storing height of thumb.

Return Value

None.

Example

```
EGL_HANDLE slider;
int w = 0;
int h = 0;
slider = egl_create_slider(100, 100, 200, 40, 10, TICK_BOTH, FALSE);
egl_slider_get_thumb_size(slider, &w, &h);
```

▶egl_slider_set_barcolor function

```
void  egl_slider_set_barcolor(
    EGL_HANDLE  hObj,
    EGL_COLOR  clr
);
```

Overview

Set bar color of slider.

Parameter

EGL_HANDLE hObj	Handle of slider.
EGL_COLOR clr	Color value. Use MAKE_COLORREF(r, g, b) macro function.

Return Value

None.

Example

```
EGL_HANDLE slider;
slider = egl_create_slider(100, 100, 200, 40, 10, TICK_BOTH, FALSE);
egl_slider_set_barcolor(slider, MAKE_COLORREF(0, 0xff, 0));
```

▶egl_slider_set_tickcolor function

```
void  egl_slider_set_tickcolor(
    EGL_HANDLE  hObj,
    EGL_COLOR  clr
);
```

Overview

Set tick color of slider.

Parameter

EGL_HANDLE hObj	Handle of slider.
EGL_COLOR clr	Color value. Use MAKE_COLORREF(r, g, b) macro function.

Return Value

None.

Example

```
EGL_HANDLE slider;
slider = egl_create_slider(100, 100, 200, 40, 10, TICK_BOTH, FALSE);
egl_slider_set_tickcolor(slider, MAKE_COLORREF(0xff, 0, 0));
```

▶**egl_slider_set_transparent** function

```
void  egl_slider_set_transparent(
    EGL_HANDLE  hObj,
    BOOL  bflag
);
```

Overview

Set background transparent mode of slider.

Parameter

EGL_HANDLE hObj	Handle of slider.
BOOL bflag	TRUE => transparent mode.
	FALSE => non transparent mode.

Return Value

None.

Example

```
EGL_HANDLE slider;
slider = egl_create_slider(100, 100, 200, 40, 10, TICK_BOTH, FALSE);
egl_slider_set_transparent(slider, TRUE);
```

▶egl_release_slider function

```
BOOL  egl_release_slider (
    EGL_HANDLE  hObj
);
```

Overview

Release slider.

Parameter

EGL_HANDLE hObj	Handle of slider.

Return Value

TRUE or FALSE.

Example

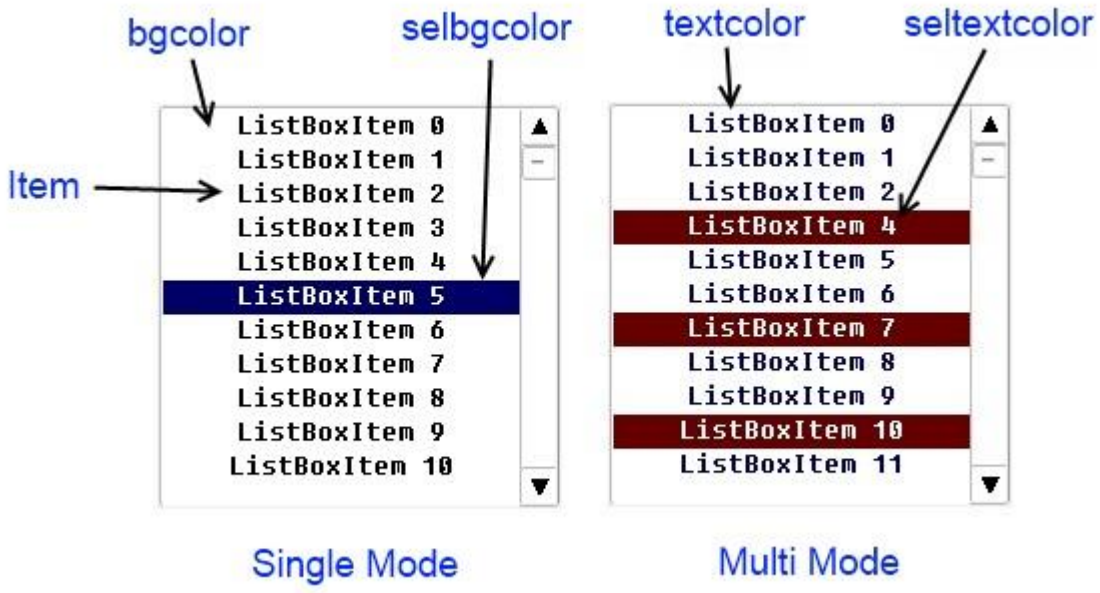
```
if(slider != NULL)
{
    egl_window_delete_object(hWin, slider);
    egl_release_slider(slider);
    slider = NULL;
}
```

▶ Slider Example.

Example

```
EGL_HANDLE slider;
void slider_callback(EGL_HANDLE h, int event)
{
      if(event == SLR_CLICKED)
      {
          debugprintf("position = %d₩r₩n",egl_slider_get_pos());
       }
}
extern BOOL process_touch(BOOL* touchdown, EGL_POINT* pPoint);
int main()
{
   …
   EGL_POINT touch_pt;
 BOOL touchdown = FALSE;
 EGL_HANDLE hWin;
  egl_init();
   slider = egl_create_slider(100, 100, 200, 40, 10, TICK_BOTH, FALSE);
   egl_sliderl_callback(slider, slider_callback);
  egl_window_add_object(hWin, slider);
   egl_window_show(hWin);
  egl_draw();
while(1)
{
      if( process_touch( &touchdown, &touch_pt ) )
      egl_user_touch_input( touchdown, &touch_pt );

   egl_draw();
 }
}
```

List Box



Function	Description
egl_create_listbox	Create list box.
egl_listbox_callback	Set callback function that will be called when list box event occur.
egl_listbox_additem	Add item of list box.
egl_listbox_delitem	Delete last item of list box.
egl_listbox_delitem_text	Delete item which match text of list box.
egl_listbox_delitem_num	Delete item N-th item of list box.
egl_listbox_alldelitem	Delete all item of list box.
egl_listbox_get_all_itemlist	Return all item list of list box.
egl_listbox_get_sel_item	Return selected item text of list box at single select mode.
egl_listbox_get_multiple_sel_itemlist	Return selected items text of list box at multi select mode.
egl_listbox_set_bgcolor	Set background color of list box.

Function	Description
egl_listbox_set_selbgcolor	Set background color of selected item.
egl_listbox_set_textcolor	Set font color of list box.
egl_listbox_set_seltextcolor	Set font color of selected item.
egl_listbox_set_textalign	Set align of list box item.
egl_listbox_set_scrollwidth	Set width of scrollbar.
egl_listbox_change_item_text	Change item which match text of list box.
egl_listbox_change_item_num	Change N-th item of list box.
egl_release_listbox	Release listbox

Define

LIST_EVENT

```
typedef enum
{
    LIST_CHANGED = 0,
} LIST_EVENT;
```

▶egl_create_listbox function

```
EGL_HANDLE  egl_create_listbox(
            int  x,
            int  y,
            int  w,
            int  h,
            bool bMultiple
);
```

Overview

Create list box.

Parameter

int x	X-coordinate of list box.
int y	Y-coordinate of list box.
int w	Width of list box.
int h	Height of list box.
Bool bMultiple	Set multi select mode of list box.

Return Value

Handle of created list box.

Example

```
EGL_HANDLE listbox[2];

listbox [0] = egl_create_button (100, 100, 200, 200, TRUE);

listbox [1] = egl_create_button (400, 100, 200, 200, FALSE);
```

▶egl_listbox_callback function

```
void  egl_listbox_evnet_callback (
      EGL_HANDLE  hObj,
      EVENT_CALLBACK  cb
);
```

Overview

Set callback function that will be called when list box event occur.

Parameter

EGL_HANDLE hObj	Handle of list box.
EVENT_CALLBACK cb	Function that is called when event occur.

Return Value

None.

Example

```
static void listbox_callback(EGL_HANDLE h, int event)
{
if(event == LIST_CHANGED)
    debugprintf("LIST_CHANGED");
}

void main(void)
{
   EGL_HANDLE listbox;

   egl_listbox_callback(listbox, listbox_callback);
}
```

▶egl_listbox_additem function

```
void  egl_listbox_additem (
      EGL_HANDLE  hObj,
      const char* text
);
```

Overview

Add item of list box.

Parameter

EGL_HANDLE hObj	Handle of list box.
const char* text	Text of add item.

Return Value

None.

Example

```
EGL_HANDLE listbox;

egl_listbox_additem(listbox, "ListboxItem 1");
```

▶egl_listbox_delitem function

```
void  egl_listbox_delitem (
      EGL_HANDLE  hObj
);
```

Overview

Delete last item of list box.

Parameter

EGL_HANDLE hObj	Handle of list box.

Return Value

None.

Example

```
EGL_HANDLE listbox;

egl_listbox_delitem(listbox);
```

▶egl_listbox_delitem_text function

```
BOOL  egl_listbox_delitem_text (
      EGL_HANDLE  hObj,
      const char* text
);
```

Overview

Delete item which match text of list box.

Parameter

EGL_HANDLE hObj	Handle of list box.
const char* text	Text of delete item.

Return Value

TRUE or FALSE

Example

```
EGL_HANDLE listbox;

if(egl_listbox_delitem_text(listbox, "ListboxItem 1") == TRUE)
   debugprintf("egl_listbox_delitem_text complete₩n");
```

▶egl_listbox_delitem_num function

```
BOOL  egl_listbox_delitem_num (
      EGL_HANDLE  hObj,
      Int  num
);
```

Overview

Delete N-th item of list box.

Parameter

EGL_HANDLE hObj	Handle of list box.
int num	Item number.

Return Value

TRUE or FALSE

Example

```
EGL_HANDLE listbox;

if(egl_listbox_delitem_text(listbox, 3) == TRUE)
   debugprintf("egl_listbox_delitem_text complete₩n");
```

▶egl_listbox_alldelitem function

```
void  egl_listbox_alldelitem (
      EGL_HANDLE  hObj,
);
```

Overview

Delete all item of list box.

Parameter

EGL_HANDLE hObj	Handle of list box.

Return Value

None.

Example

```
EGL_HANDLE listbox;

egl_listbox_alldelitem(listbox);
```

▶egl_listbox_ get_all_itemlist function

```
const char**  egl_listbox_get_all_itemlist (
      EGL_HANDLE  hObj,
      int* itemcnt
);
```

Overview

Return all item list of list box.

Parameter

EGL_HANDLE hObj	Handle of list box.
int* itemcnt	Pointer variable for storing item count.

Return Value

Text list pointer of all items.

※ Caution : You must release memory of list pointer that returned.

Example

```
EGL_HANDLE listbox;
int ItemCnt;
const char** ItemTextList;

ItemTextList = egl_listbox_all_itemlist(listbox, &ItemCnt);

free(ItemTextList);
```

▶egl_listbox_ get_sel_item function

```
const char*  egl_listbox_get_sel_item (
      EGL_HANDLE  hObj,
      int* index
);
```

Overview

Return selected item text of list box at single select mode.

Parameter

EGL_HANDLE hObj	Handle of list box.
Int* index	Pointer for storing selected item order.

Return Value

Text pointer of selected item.

Example

```
EGL_HANDLE listbox;
int selectItemIndex;
const char* selectItemText;

selectItemText = egl_listbox_get_sel_item(listbox, &selectItem);
```

▶egl_listbox_ get_ multiple_sel_itemlist function

```
const char**  egl_listbox_get_multiple_sel_itemlist (
      EGL_HANDLE  hObj,
      int* selcnt
);
```

Overview

Return selected items text of list box at multi select mode.

Parameter

EGL_HANDLE hObj	Handle of list box.
int* selcnt	Pointer variable for storing selected item count.

Return Value

Text list pointer of selected items.

※ Caution : You must release memory of list pointer that returned.

Example

```
EGL_HANDLE listbox;
int selectItemCnt;
const char** selectItemTextList;

selectItemTextList = egl_listbox_get_sel_itemlist (listbox, & selectItemCnt);

free(selectItemTextList);
```

▶egl_listbox_ set_bgcolor function

```
void egl_listbox_ set_bgcolor (
      EGL_HANDLE   hObj,
      EGL_COLOR   clr
);
```

Overview

Set background color of list box.

Parameter

EGL_HANDLE hObj	Handle of list box.
EGL_COLOR clr	Color value. Use MAKE_COLORREF(r, g, b) macro function.

Return Value

None.

Example

```
EGL_HANDLE listbox;

egl_listbox_set_bgcolor(listbox, MAKE_COLORREF(0x0, 0x0, 0xFF));  // Blue
```

▶egl_listbox_ set_selbgcolor function

```
void egl_listbox_ set_selbgcolor (
      EGL_HANDLE   hObj,
      EGL_COLOR   clr
);
```

Overview

Set background color of selected item.

Parameter

EGL_HANDLE hObj	Handle of list box.
EGL_COLOR clr	Color vaule. Use MAKE_COLORREF(r, g, b) macro function.

Return Value

None.

Example

```
EGL_HANDLE listbox;

egl_listbox_set_selbgcolor(listbox, MAKE_COLORREF(0xFF, 0x0, 0x0));  // Red
```

▶egl_listbox_ set_textcolor function

```
void egl_listbox_ set_textcolor (
      EGL_HANDLE   hObj,
      EGL_COLOR   clr
);
```

Overview

Set font color of list box.

Parameter

EGL_HANDLE hObj	Handle of list box.
EGL_COLOR clr	Color value. Use MAKE_COLORREF(r, g, b) macro function.

Return Value

None.

Example

```
EGL_HANDLE listbox;

egl_listbox_set_textcolor(listbox, MAKE_COLORREF(0x0, 0x0, 0x0));  // Black
```

▶egl_listbox_ set_seltextcolor function

```
void egl_listbox_ set_seltextcolor (
      EGL_HANDLE   hObj,
      EGL_COLOR   clr
);
```

Overview

Set font color of selected item.

Parameter

EGL_HANDLE hObj	Handle of list box.
EGL_COLOR clr	Color value. Use MAKE_COLORREF(r, g, b) macro function.

Return Value

None.

Example

```
EGL_HANDLE listbox;

egl_listbox_set_seltextcolor(listbox, MAKE_COLORREF(0xFF, 0xFF, 0xFF));  // White
```

▶egl_listbox_ set_textalign function

```
void egl_listbox_ set_textalign (
      EGL_HANDLE   hObj,
      int   align
);
```

Overview

Set align of list box item.

Parameter

EGL_HANDLE hObj	Handle of list box.
int align	Align value. EGL_ALIGN_LEFT / EGL_ALIGN_RIGHT / EGL_ALIGN_CENTER EGL_ALIGN_TOP / EGL_ALIGN_BOTTOM

Return Value

None.

Example

```
EGL_HANDLE listbox;

egl_listbox_set_textalign( listbox, EGL_ALIGN_LEFT);
```

▶egl_listbox_ set_scrollwidth function

```
void egl_listbox_ set_scrollwidth (
      EGL_HANDLE   hObj,
      int  width
);
```

Overview

Set width of scrollbar.

Parameter

EGL_HANDLE hObj	Handle of list box.
int width	Width of scrollbar.

Return Value

None.

Example

```
EGL_HANDLE listbox;

egl_listbox_set_scrollwidth(listbox, 40);
```

▶egl_listbox_change_item_text function

```
BOOL egl_listbox_change_item_text (
      EGL_HANDLE   hObj,
      const char*   text,
      const char*   changetext
);
```

Overview

Change item which match text of list box.

Parameter

EGL_HANDLE hObj	Handle of list box.
const char* text	Item text.
const char* changetext	Text value that is changed.

Return Value

TRUE or FALSE

Example

```
EGL_HANDLE listbox;

egl_listbox_change_item_text( listbox, "test1", "test2");
```

▶ **egl_listbox_change_item_num** function

```
BOOL egl_listbox_change_item_text (
      EGL_HANDLE   hObj,
      int   num,
      const char*   changetext
);
```

Overview

Change N-th item of list box.

Parameter

EGL_HANDLE hObj	Handle of list box.
int num	Item number.
const char* changetext	Text value that is changed.

Return Value

TRUE or FALSE

Example

```
EGL_HANDLE listbox;

egl_listbox_change_item_text( listbox, 3, "test2");
```

▶egl_release_listbox function

```
BOOL egl_release_listbox (
      EGL_HANDLE  hObj
);
```

Overview

Release list box.

Parameter

EGL_HANDLE hObj	Handle of list box.

Return Value

TRUE or FALSE

Example

```
if(listbox != NULL)
{
    egl_window_delete_object(hWin, listbox);
    egl_release_listbox(listbox);
    listbox = NULL;
}
```

▶ List box Example.

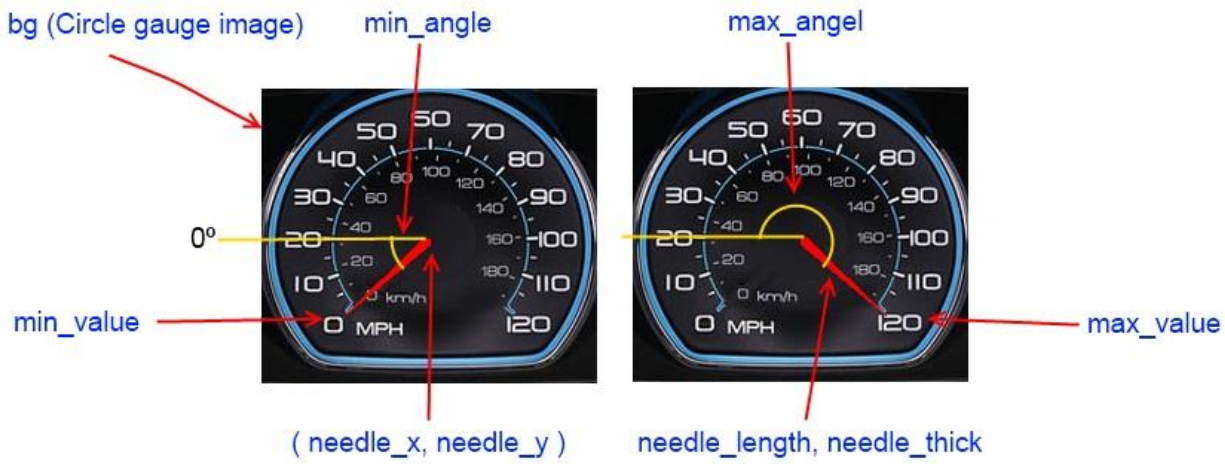
Example

```
#include "adStar.h"
extern BOOL process_touch(BOOL* touchdown,EGL_POINT* pPoint);

static void listbox_cb1(EGL_HANDLE h, int event)
{
	int itemcnt, index;
	const char **itemlist;
	const char *selitem;
	int i;

	if(event == LIST_CHANGED) {
		debugprintf("LIST_CHANGED₩r₩n");
		// Get all item list
		itemlist = egl_listbox_get_all_itemlist(h, &itemcnt);
		debugprintf("All Item %d₩r₩n", itemcnt);
		for(i = 0; i < itemcnt; i++)
			debugprintf("Item : [%s]₩r₩n", itemlist[i]);
		// Get selected item list
		selitem = egl_listbox_get_sel_item(h, &index);
		debugprintf("Selected Item [%d] : [%s]₩r₩n", index, selitem);
	}
	free(itemlist);
}

static void listbox_cb2(EGL_HANDLE h, int event)
{
	int selcnt;
	const char **selitemlist;
	int i;

	if(event == LIST_CHANGED) {
		debugprintf("LIST_CHANGED₩r₩n");
		selitemlist = egl_listbox_get_multiple_sel_itemlist(h, &selcnt);
		debugprintf("Selected Items %d ₩r₩n", selcnt);
		for(i = 0; i < selcnt; i++)
			debugprintf("Selected Item : [%s]₩r₩n", selitemlist[i]);
	}
```

```
	free(selitemlist);
}
void user_main()
{
	float addpos=1.0f;
	EGL_POINT touch_pt;
	BOOL touchdown=FALSE;
	EGL_HANDLE hWin;
	EGL_HANDLE listbox[2];

	egl_init();

	hWin = egl_create_window("Main Window");

	listbox[0] = egl_create_listbox(100, 100, 200, 200, FALSE);
	listbox[1] = egl_create_listbox(400, 100, 200, 200, TRUE);

	egl_listbox_callback(listbox[0], listbox_cb1);
	egl_listbox_callback(listbox[1], listbox_cb2);

	{
		int i;
		char str_text[16];
		for(i=0;i<100;i++)
		{
			sprintf(str_text,"ListBoxItem %d",i);
			egl_listbox_additem(listbox[0],str_text);
			egl_listbox_additem(listbox[1],str_text);
		}
	}
	// Delete last item
	egl_listbox_delitem(listbox[0]);
	// Delete "ListBoxItem 3" item
	egl_listbox_delitem_text(listbox[1], "ListBoxItem 3");

	// Set background color
	egl_listbox_set_bgcolor(listbox[1], MAKE_COLORREF(0xff, 0xff, 0xff));
	// Set selected background color
	egl_listbox_set_selbgcolor(listbox[1], MAKE_COLORREF(0x66, 0x0, 0x0));
	// Set text color
	egl_listbox_set_textcolor(listbox[1], MAKE_COLORREF(0, 0, 0x66));
```

```
        // Set selected text color
        egl_listbox_set_seltextcolor(listbox[1], MAKE_COLORREF(0xff, 0xff, 0xff));
        egl_window_add_object(hWin, listbox[0]);
        egl_window_add_object(hWin, listbox[1]);

       egl_window_show(hWin, TRUE);
        egl_draw();

        while(1)
        {
                BOOL bEvent = TRUE;

                if(process_touch(&touchdown, &touch_pt))
                        egl_user_touch_input(touchdown, &touch_pt);
                else
                        bEvent = FALSE;
                egl_draw();
        }
}
```

Circle Gauge



Function	Description
egl_create_circle_gauge	Create Circle gauge.
egl_circle_gauge_set_value	Set Circle gauge value.
egl_circle_gauge_get_value	Get Circle gauge value.
egl_release_circle_gauge	Release circle gauge

▶egl_create_circle_gauge function

```
EGL_HANDLE  egl_create_circle_gauge(
    SURFACE*  bg,
      int  x,
      int  y,
    EGL_CIRCLE_GAUGE_INFO*  pInfo
);
```

Overview

Create Circle gauge.

Parameter

SURFACE* bg	Circle gauge image.
int x	Circle gauge x-coordinate.
int y	Circle gauge y-coordinate.
EGL_CIRCLE_GAUGE_INFO* pInfo	Circle gauge infomation.

```
typedef struct  _tagCIRCLE_GAUGE_INFO
{
    int  needle_x;          // gauge needle x-coordinate..
    int  needle_y;          // gauge needle y-coordinate.
    int  needle_length;     // gauge needle length.
    int  needle_thick;      // gauge needle thickness.
    int  min_value;         // Minimum value of gauge.
    int  max_value;         // Maximum value of gauge.
    int  min_angle;         // Minimum angle of gauge.
    int  max_angle;         // Maximum angle of gauge.
}EGL_CIRCLE_GAUGE_INFO;
```

Return Value

Handle of created circle gauge.

Example

```
SURFACE* gauge_bg;
EGL_HANDLE c_gauge;
EGL_CIRCLE_GAUGE_INFO CGInfo;
CGInfo.needle_x = 113;
CGInfo.needle_y = 101;
CGInfo.min_angle = -42;
CGInfo.max_angle = 222;
CGInfo.min_valie = 0;
CGInfo.max_value = 120;
CGInfo.needle_length = 80;
CGInfo.needle_thick = 3;
gauge_bg = loadbmp("gauge.bmp");
c_gauge = egl_create_circle_gauge(gauge_bg, 286, 37, &CGInfo);
```

▶**egl_circle_gauge_set_value** function

```
BOOL  egl_circle_gauge_set_value(
    EGL_HANDLE  h,
    int  value
);
```

Overview

Set Circle gauge value.

Parameter

EGL_HANDLE h	Handle of circle gauge.
int vaule	Gauge value.

Return Value

TRUE or FALSE

Example

```
…
c_gauge = egl_create_circle_gauge(gauge_bg, 286,37,&CGInfo);
egl_circle_gauge_set_value(c_gauge, 80);
…
```

▶egl_circle_gauge_get_value function

```
int  egl_circle_gauge_get_value(
    EGL_HANDLE  h
);
```

Overview

Get Circle gauge value.

Parameter

EGL_HANDLE h	Handle of circle gauge.

Return Value

Gauge value.

Example

```
...
c_gauge = egl_create_circle_gauge(gauge_bg, 286,37,&CGInfo);
int gauge_value = egl_circle_gauge_get_value(c_gauge);
...
```

▶egl_release_circle_gauge function

```
BOOL  egl_release_circle_gauge(
    EGL_HANDLE  hObj
);
```

Overview

Release circle gauge.

Parameter

EGL_HANDLE hObj	Handle of circle gauge.

Return Value

TRUE or FALSE

Example

```
if(circle != NULL)
{
    egl_window_delete_object(hWin, circle);
    egl_release_circle(circle);
    circle = NULL;
}
```

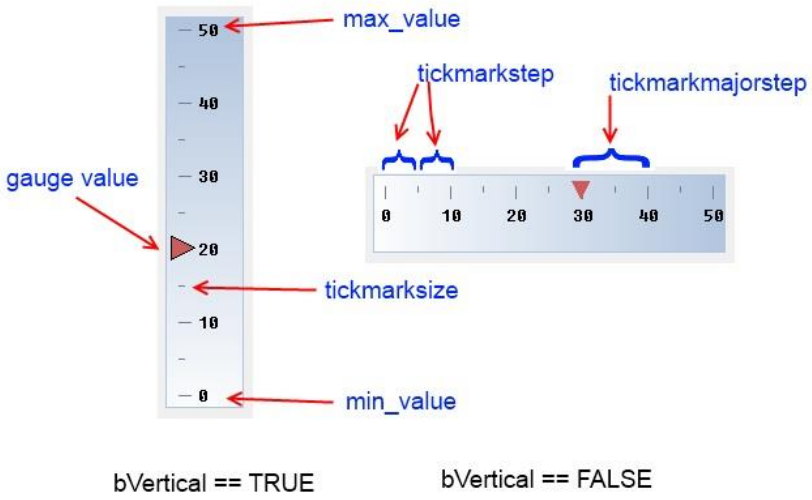
▶ Circle Gauge Example.

Example

```
extern BOOL process_touch(BOOL* touchdown, EGL_POINT* pPoint);
int main()
{
   …
   EGL_POINT touch_pt;
 BOOL touchdown = FALSE;
 EGL_HANDLE hWin;
   SURFACE* gauge_bg;
   EGL_HANDLE c_gauge;
   EGL_CIRCLE_GAUGE_INFO CGInfo;
  egl_init();
   CGInfo.needle_x = 113;
   CGInfo.needle_y = 101;
   CGInfo.min_angle = -42;
   CGInfo.max_angle = 222;
   CGInfo.min_valie = 0;
   CGInfo.max_value = 120;
   CGInfo.needle_length = 80;
   CGInfo.needle_thick = 3;
   gauge_bg = loadbmp("gauge.bmp");
   c_gauge = egl_create_circle_gauge(gauge_bg, 286, 37, &CGInfo);
   egl_window_add_object(hWin, c_gauge);
   egl_window_show(hWin);
  egl_draw();
  int gauge_value = 0;
  int gauge_flag = 0;
while(1)
{
      if( process_touch( &touchdown, &touch_pt ) )
     egl_user_touch_input( touchdown, &touch_pt );

   if(gauge_flag)
   {
      gauge_value++;
       if(gauge_value > 120)
```

```
            {
              gauge_value = 120;
              gauage_flag = 0;
            }
          }
          else
          {
             gauge_value--;
             if(gauge_value < 0)
             {
                gauge_value = 0;
                gauge_flag = 1;
              }
           }
           egl_circle_gauge_set_valut(c_gauge, gauge_value);
           egl_draw();
      }
}
```

Bar Gauge



bVertical == TRUE bVertical == FALSE

Function	Description
egl_create_bar_gauge	Create bar gauge.
egl_bar_gauge_set_value	Set bar gauge value.
egl_bar_gauge_get_value	Get bar gaugevalue.
egl_release_bar_gauge	Release bar gauge

▶**egl_create_bar_gauge** function

```
EGL_HANDLE  egl_create_bar_gauge(
    int  x,
    int  y,
    int  w,
    int  h,
    EGL_BAR_GAUGE_INFO*  pInfo
);
```

Overview

Create bar gauge.

Parameter

int x	Bar gauge x-coordinate.
int y	Bar gauge y-coordinate..
int w	Bar gauge width.
int h	Bar gauge height.
EGL_BAR_GAUGE_INFO* pInfo	Bar gauge information.

```
typedef struct  _tagBAR_GAUGE_INFO
{
int min_value;            // Minimum value of gauge.
int max_value;            // Maximum value of gauge.
int tickmarksize;         // tick size. major tick = tick size * 2
int tickmarkstep;         // tick step size.
int tichmarkmajorstep;    // major tick step size.
BOOL bVertical;           // vertical mode select.
SURFACE* bg;              // back ground image.
} EGL_BAR_GAUGE_INFO;
```

Return Value

Handle of created bar gauge.

Example

```
EGL_HANDLE b_gauge;
EGL_BAR_GAUGE_INFO BGInfo;
BGInfo.min_value = 0;
BGInfo.max_value = 50;
BGInfo.bVertical = TRUE;
BGInfo.tickmarksize = 5;
BGInfo.tickmarkstep = 5;
BGInfo.tickmarkmajorstep = 10;
BGInfo.bg = NULL;
b_gauge  = egl_create_bar_gauge(10, 40, 60, 300, &BGInfo);
```

▶egl_bar_gauge_set_value function

```
BOOL  egl_bar_gauge_set_value(
    EGL_HANDLE  h,
    int  value
);
```

Overview

Set bar gauge value.

Parameter

EGL_HANDLE h	Handle of bar gauge.
int value	Gauge value.

Return Value

TRUE or FALSE

Example

```
...
b_gauge = egl_create_bar_gauge(10, 40, 60, 300, &BGInfo);
egl_bar_gauge_set_value(b_gauge, 20);
...
```

▶egl_bar_gauge_get_value function

```
int  egl_bar_gauge_get_value(
    EGL_HANDLE  h
);
```

Overview

Get bar gauge value.

Parameter

EGL_HANDLE h	Handle of bar gauge.

Return Value

Gauge value.

Example

```
...
b_gauge = egl_create_bar_gauge(10, 40, 60, 300, &BGInfo);
int gauge = egl_bar_gauge_get_value(b_gauge);
...
```

▶**egl_release_bar_gauge** function

```
BOOL   egl_release_bar_gauge(
    EGL_HANDLE   hObj
);
```

Overview

Release bar gauge.

Parameter

EGL_HANDLE hObj	Handle of bar gauge.

Return Value

TRUE or FALSE

Example

```
if(bar_gauge != NULL)
{
    egl_window_delete_object( hWin, bar_gauge);
    egl_release_bar_gauge(bar_gauge);
    bar_gauge_NULL;
}
```

▶ Bar Gauge Example.

Example

```
extern BOOL process_touch(BOOL* touchdown, EGL_POINT* pPoint);
int main()
{
   ...
   EGL_POINT touch_pt;
 BOOL touchdown = FALSE;
 EGL_HANDLE hWin;
   EGL_HANDLE b_gauge;
   EGL_BAR_GAUGE_INFO BGInfo;
   egl_init();
   BGInfo.min_value = 0;
   BGInfo.max_value = 50;
   BGInfo.bVertical = TRUE;
   BGInfo.tickmarksize = 5;
   BGInfo.tickmarkstep = 5;
   BGInfo.tickmarkmajorstep = 10;
   BGInfo.bg = NULL;
   b_gauge  = egl_create_bar_gauge(10,40,60,300,&BGInfo);
   egl_window_add_object(hWin, b_gauge);
   egl_window_show(hWin);
  egl_draw();
  int gauge_value = 0;
  int gauge_flag = 0;
while(1)
{
      if( process_touch( &touchdown, &touch_pt ) )
     egl_user_touch_input( touchdown, &touch_pt );

   if(gauage_flag)
   {
      gauge_value++;
       if(gauge_value > 50)
       {
         gauge_value = 50;
         gauge_flag = 0;
```

```
            }
        }
        else
        {
            gauge_value--;
            if(gauge_value < 0)
            {
                gauge_value = 0;
                gauge_flag = 1;
             }
         }
         egl_bar_gauge_set_valut(b_gauge, gauge_value);
         egl_draw();
    }
}
```

Picture

Function	Description
egl_create_picture	Create Picture object
egl_picture_callback	Set callback function that will be called when picture event occur.
egl_picture_set	Set the picture image.
egl_release_picture	Release picture.

▶egl_create_picture function

```
EGL_HANDLE  egl_create_picture(
    SURFACE*  surf,
    int  x,
    int  y,
    int  w,
    int  h
);
```

Overview

Create Picture object.

Parameter

SURFACE* surf	Image of Picture object.
int x	Picture object x-coordinate.
int y	Picture object y-coordinate..
int w	Picture object width.
int h	Picture object height.

Return Value

Handle of created Picture object.

Example

```
SURFACE* surf = loadbmp("image.bmp");
EGL_HANDLE picture;
picture = elg_create_picture(surf,10,40,128,128);
```

▶egl_picture_callback function

```
BOOL  egl_picture_callback(
    EGL_HANDLE  hObj,
    EVENT_CALLBACK  cb
);
```

Overview

Set callback function that will be called when picture event occur.

Parameter

EGL_HANDLE hObj	Handle of Picture.
EVENT_CALLBACK cb	Function that is called when event occur.

Return Value

TRUE or FALSE

Example

```
static void picture_callback(EGL_HANDLE h, int event)
{
    if(event == PICTURE_CLICKED)
        debugprintf("Picture clicked₩r₩n");
}

void main(void)
{
   EGL_HANDLE picture;
   ...
   egl_picture_callback(picture, picture_callback);
   ...
}
```

▶**egl_picture_set** function

```
SURFACE*  egl_picture_set(
    EGL_HANDLE  hObj,
    SURFACE*  surf
);
```

Overview

Set the picture image.

Parameter

EGL_HANDLE h	Handle of Picture.
SURFACE* surf	Image

Return Value

Old image.

Example

```
...
SURFACE* new_image = loadbmp("new_image.bmp");
SURFACE* old_image;
old_image = egl_picture_set(picture, new_image);
...
```

▶egl_release_picture function

```
BOOL egl_release_picture(
    EGL_HANDLE  hObj
);
```

Overview

Release picture.

Parameter

EGL_HANDLE hObj Handle of Picture.

Return Value

TRUE or FALSE

Example

```
if(picture != NULL)
{
    egl_window_delete_object(hWin, picture);
    egl_release_picture(picture);
    picture = NULL;
}
```

▶ Pixture Example.

Example

```
extern BOOL process_touch(BOOL* touchdown, EGL_POINT* pPoint);
int main()
{
      …
      EGL_POINT touch_pt;
      BOOL touchdown = FALSE;
     EGL_HANDLE hWin;
      EGL_HANDLE picture;
      egl_init();
      hWin = egl_create_window("window1");

      SURFACE*  surf = loadbmp("test.bmp");
      picture  = egl_create_picture(surf, 10, 40, 128, 128);
      egl_window_add_object(hWin, b_gauge);
      egl_window_show(hWin);
     egl_draw();

    while(1)
    {
        if( process_touch( &touchdown, &touch_pt ) )
              egl_user_touch_input( touchdown, &touch_pt );

        egl_draw();
    }
}
```

Animation

Function	Description
egl_create_animation	Create animation object.
egl_release_animation	Release animation

▶egl_create_animation function

```
EGL_HANDLE  egl_create_animation(
    int  x,
    int  y,
    int  w,
    int  h,
    SURFACE**  surflist,
    int  surfcnt,
    int  delaycnt
);
```

Overview

Create animation object. Animation object is drawn consecutive images one by one.

Parameter

int x	Animation object x-coordinate.
int y	Animation object y-coordinate.
int w	Width of animation object.
int h	Height of animation object.
SURFACE** surflist	Animation image list.
int surfcnt	Animation image count.
int delaycnt	Animation image change time.

Return Value

Handle of created animation object.

Example

```
EGL_HANDLE btn_ani;
SURFACE* surf_ani[10];
char fname[12];
int i;
for(i=0;i<10;i++)
{
   sprint(fname, "Frame%d.bmp",i);
   surf_ani[i] = loadbmp(fname);
}
btn_ani = egl_create_animation(350, 150, 128, 128, surf_ani, 10, 0);
```

▶egl_release_animation function

```
BOOL   egl_release_animation(
       EGL_HANDLE   hObj
);
```

Overview

Release animation.

Parameter

EGL_HANDLE hObj	Handle of animation.

Return Value

TRUE or FALSE

Example

```
if(animation != NULL)
{
    egl_window_delete_object(hWin, animation);
    egl_release_animation(animation);
    animation = NULL;
}
```

▶ Animation Example.

Example

```
extern BOOL process_touch(BOOL* touchdown, EGL_POINT* pPoint);
int main()
{
    ...
    EGL_POINT touch_pt;
  BOOL touchdown = FALSE;
  EGL_HANDLE hWin;
    EGL_HANDLE btn_ani;
    SURFACE* surf_ani[10];
    char fname[12];
    egl_init();
    int i;
    for(i=0;i<10;i++)
    {
       sprint(fname, "Frame%d.bmp",i);
       surf_ani[i] = loadbmp(fname);
    }
    btn_ani = egl_create_animation(350, 150, 128, 128, surf_ani, 10, 0);
    egl_window_add_object(hWin, btn_ani);
    egl_window_show(hWin);
   egl_draw();
  while(1)
{
       if( process_touch( &touchdown, &touch_pt ) )
      egl_user_touch_input( touchdown, &touch_pt );

       egl_draw();
  }
}
```

Custom Object

Function	Description
egl_create_custom_object	Create custom object.

Define

EGL_MSG_ID

```
typedef enum enumMSG
{
    EGL_MSG_DRAW = 0,
    EGL_MSG_DELETE,
    EGL_MSG_FOCUS,
    EGL_MSG_UNFOCUS,
    EGL_MSG_KEY_UP,
    EGL_MSG_KEY_DOWN,
    EGL_MSG_TOUCHED,
    EGL_MSG_UNTOUCHED,
    EGL_MSG_MOVE,
    EGL_MSG_TIMETICK
} EGL_MSG_ID;
```

EGL_MSG

```
typedef struct _tagMessage
{
    EGL_MSGID msgID;
    EGL_HANDLE hObj;
    EGL_HANDLE hWin;
    Union
    {
        EGL_POINT point;
        U32 key;
    } param;
} EGL_MSG;
```

▶egl_create_custom_object function

```
EGL_HANDLE   egl_create_custom_object(
    int  x,
    int  y,
    int  w,
    int  h,
    void* (*msg_handler)(EGL_MSG*  pMsg)
);
```

Overview

Create custom object.

Parameter

int x	Custom object x-coordinate.
int y	Custom object y-coordinate.
int w	Width of custom object.
int h	Height of custom object.
void* (*msg_handler)(EGL_MSG* pMsg)	Msg processing function of custom object.

Return Value

Handle of created custom object.

Example

```
Static void* custom_obj_msghandler(EGL_MSG* pMsg)
{
     Switch(pMsg->msgID)
    {
        Case EGL_MSG_DRAW:
```

```
            /* draw */
            Break;
        }
        Return NULL;
}
 …
 Int main()
 {
 …
 EGL_HANDLE custom_obj;
 custom_obj = egl_create_custom(350, 240, 100, 30, custom_obj_msghandler);
 …
 }
```

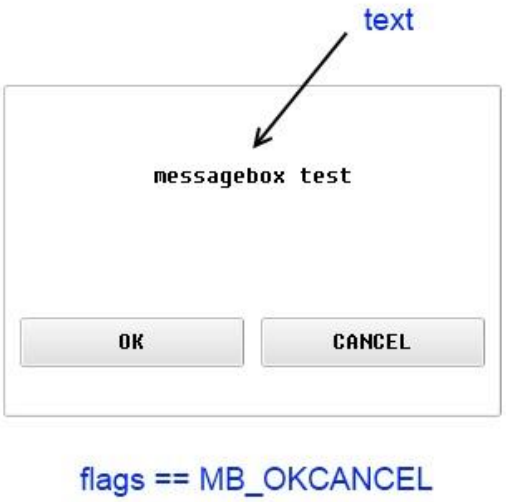
▶ Custom Example.

Example

```
static char speed_str[16];
static EGL_HANDLE hSpeed;

static void speed_draw(EGL_OBJECT_PTR pObj)
{
    draw_text_in_box(pObj->pFont, &pObj->rect, speed_str, EGL_ALIGN_CENTER);
}

static void* speed_msghandler(EGL_MSG* pMsg)
{
    switch(pMsg->msgID)
    {
        case EGL_MSG_DRAW:
            speed_draw(EGL_HANDLE_TO_OBJECT(pMsg->hObj));
            break;
     }
}

extern BOOL process_touch(BOOL* touchdown, EGL_POINT* pPoint);

int main()
{
   ...
   EGL_POINT touch_pt;
 BOOL  touchdown = FALSE;
 EGL_HANDLE hWin;
   EGL_HANDLE hSpeed;
   EGL_FONT* myfont;
   egl_init();
   myfont = egl_create_default_font();
   hSpeed = egl_create_custom_object(350, 240, 100, 30, speed_msghandler);
   egl_window_add_object(hWin, hSpeed);
   egl_set_font(hSpeed, myfont);
   strcpy(speed_str, "    0 Mile/h");
   egl_window_show(hWin);
  egl_draw();
```

```
    while(1)
   {
        if( process_touch( &touchdown, &touch_pt ) )
        egl_user_touch_input( touchdown, &touch_pt );

        egl_draw();
    }
}
```

Messagebox



Function	Description
egl_show_messagebox	Output message box.

▶egl_show_messagebox function

```
int  egl_show_messagebox(
    const char* text,
    int  flags,
    BOOL  (*user_input_function)(EGL_MSG* pMsg)
);
```

Overview

Output message box.

Parameter

const char* text	Text of messagebox.
int flags	Messagebox type. MB_OK MB_OKCANCEL MB_YESNO MB_YESNOCANCEL
BOOL (*user_input_function)(EGL_MSG* pMsg)	User input function. Function which check pressed button on message box.

Return Value

Return pressed button type on message box.

```
enum
 {
   IDOK = 0,
   IDCANCEL,
   IDABORT,
   IDRETRY,
   IDIGNORE,
   IDYES,
   IDNO,
   IDCONTINUE,
 }
```

Example

```
BOOL my_touchinput(EGL_MSG* pMsg)
{
    BOOL touchdown;
    EGL_POINT touch_pt;
    if(process_touch(&touchdown, &touch_pt))
    {
        if(touchdown)
           pMsg -> msgID = EGL_MSG_TOUCHED;
        else
           pMsg -> msgID = EGL_MSG_UNTOUCHED;
        pMsg -> param.point.x = touch_pt.x;
        pMsg -> param.point.y = touch_pt_y;
        return TRUE;
    }
    return FALSE;
}

int main()
{
...
...
int type = egl_show_messagebox("messagebox test", MB_OKCANCEL, my_touchinput);
...
 }
```

▶ MessageBox Example.

Example

```
void btn_callback(EGL_HANDLE h, int event)
{
      if(event == BTN_CLICKED)
      {
          debugprintf("buttton clicked.");
    }
}

BOOL my_touchinput(EGL_MSG* pMsg)
{
    BOOL  touchdown;
    EGL_POINT  touch_pt;
    if(process_touch(&touchdown, &touch_pt))
    {
        if(touchdown)
           pMsg -> msgID = EGL_MSG_TOUCHED;
        else
           pMsg -> msgID = EGL_MSG_UNTOUCHED;
        pMsg -> param.point.x = touch_pt.x;
        pMsg -> param.point.y = touch_pt_y;
        return TRUE;
    }
    return FALSE;
}

extern BOOL process_touch(BOOL* touchdown, EGL_POINT* pPoint);

int main()
{
   …
   EGL_POINT touch_pt;
 BOOL touchdown = FALSE;
 EGL_HANDLE hWin;
 EGL_HANDLE btn;
 egl_init();
   btn = egl_create_button(100, 100, 100, 50, "Button Ex");
   egl_button_callback(btn, btn_callback);
```

```
    egl_window_add_object(hWin, btn);
    egl_window_show(hWin);
     egl_draw();

    int type = egl_show_messagebox("messagebox test", MB_OKCANCEL, my_touchinput);

  while(1)
  {
       if( process_touch( &touchdown, &touch_pt ) )
      egl_user_touch_input( touchdown, &touch_pt );

     egl_draw();
   }
}
```

EGL Font

Function	Description
egl_get_font	Return font information of object.
egl_set_font	Set font of object.
egl_font_set_bkmode	Set whether use background color of font.
egl_font_get_bk_color	Get background color of font.
egl_font_set_bk_color	Set background color of font.
egl_font_get_color	Get color of font.
egl_font_set_color	Set color of font.
create_bitfont	Create bit type font for use.
release_bitfont	Release bit type font.
create_bmpfont	Create image font fot use.
bmfont_release	Release image font.
draw_text	Draw text.
draw_text_pivot	Draw text. Left / Right rotation by 90 degrees.
draw_text_len	Draw text. (Set the length)
draw_text_in_box	Draw text in box.
text_width	Get text width.

▶**egl_get_font** function

```
EGL_FONT*  egl_get_font(
    EGL_HANDLE  h
);
```

Overview

Return font information of object.

Parameter

EGL_HANDLE h	Handle of object that use font.

Return Value

Font information of object. (EGL_FONT Struct)

Example

```
EGL_HANDLE object;
…
…
EGL_FONT* cur_font = egl_get_font(object);
```

▶**egl_set_font** function

```
EGL_FONT*  egl_set_font(
    EGL_HANDLE  h,
    EGL_FONT*  font
);
```

Overview

Set font of object.

Parameter

EGL_HANDLE h	Handle of object.
EGL_FONT* font	Font pointer.

Return Value

Previously set font information.

Example

```
EGL_HANDLE object;
EGL_FONT* font;
...
font = create_bitfont();
egl_set_font(object , font);
```

▶egl_font_set_bkmode function

```
void  egl_font_set_bkmode(
    EGL_FONT*  font,
    int  mode
);
```

Overview

Set whether use background color of font.

Parameter

EGL_FONT* font	Font pointer.
int mode	Whether use background color. TRUE or FALSE.

Return Value

None.

Example

```
 EGL_FONT* font;
 ...
 font = create_bitfont();
 egl_set_font(object , font);
egl_font_set_bkmode(font, TRUE);
```

▶**egl_font_get_bk_color** function

```
EGL_COLOR  egl_font_get_bk_color(
    EGL_FONT*  font
);
```

Overview

Get background color of font.

Parameter

EGL_FONT* font	Font pointer.

Return Value

Current font background color.

Example

```
EGL_FONT* font;
EGL_COLOR font_bk_color;
...
font = create_bitfont();
font_bk_color = egl_font_get_bk_color(font);
```

▶**egl_font_set_bk_color** function

```
EGL_COLOR  egl_font_set_bk_color(
    EGL_FONT*  font,
    EGL_COLOR  clr
);
```

Overview

Set background color of font.

Parameter

EGL_FONT* font	Font pointer.
EGL_COLOR clr	Background color. Use (MAKE_COLORREF(r, g, b) macro function.)

Return Value

Previously set font back ground color.

Example

```
EGL_FONT* font;
EGL_COLOR font_bk_color;
...
font = create_bitfont();
font_bk_color = egl_font_set_bk_color(font, MAKE_COLORREF(0, 255, 0));
```

▶egl_font_get_color function

```
EGL_COLOR  egl_font_get_color(
    EGL_FONT*  font
);
```

Overview

Get color of font.

Parameter

EGL_FONT* font	Font pointer.

Return Value

Current font color.

Example

```
EGL_FONT* font;
EGL_COLOR font_color;
...
font = create_bitfont();
font_color = egl_font_get_color(font);
```

▶**egl_font_set_color** function

```
EGL_COLOR  egl_font_set_color(
    EGL_FONT*  font,
    EGL_COLOR  clr
);
```

Overview

Set color of font.

Parameter

EGL_FONT* font	Font pointer.
EGL_COLOR clr	Font color. (MAKE_COLORREF(r, g, b) macro function.)

Return Value

Previously set font color.

Example

```
EGL_FONT* font;
EGL_COLOR font_color;
...
font = create_bitfont();
font_color = egl_font_set_color(font, MAKE_COLORREF(0, 0, 255));
```

▶**create_bitfont** function

```
EGL_FONT*   create_bitfont( void );
```

Overview

Create bit type font for use.

Parameter

None

Return Value

Pointer of created bit font struct.

Example

```
EGL_FONT* bit_font;
EGL_HANDLE object;
…
bit_font = create_bitfont();
egl_set_font(object, bit_font );
…
draw_text(bit_font, 100, 100, "font test");
```

▶**release_bitfont** function

```
void  release_bitfont(
      EGL_FONT*  pFont
 );
```

Overview

Release bit type font.

Parameter

EGL_FONT* pFont Bit font pointer.

Return Value

None.

Example

```
EGL_FONT* bit_font;
…
bit_font = create_bitfont();
…
release_bitfont(bit_font);
```

▶**create_bmpfont** function

```
EGL_FONT*  create_bmpfont(
      const char *fname
);
```

Overview

Create image font fot use.

Parameter

const char *fname	Image font file name.

Return Value

Pointer of created image font struct.

Example

```
EGL_FONT* bm_font;
EGL_HANDLE object;
...
bm_font = create_bmpfont("font/batang_24.fnt");
egl_set_font(object, bm_font);
...
draw_text(bm_font, 100, 100, "image font test");
```

▶**bmfont_release** function

```
void  bmfont_release(
      EGL_FONT*  pFont
);
```

Overview

Release image font.

Parameter

EGL_FONT* pFont	Image font pointer.

Return Value

None.

Example

```
EGL_FONT* bm_font;
…
bm_font = create_bmpfont("font/batang_24.fnt");
…
bmfont_release(bm_font);
```

▶**draw_text** function

```
int  draw_text(
     EGL_FONT*  pFont,
     int  x,
     int  y,
     const char*  text
);
```

Overview

Draw text.

Parameter

EGL_FONT* pFont	Font that use in characters output.
int x	X-coordinate of text.
int y	Y-coordinate of text.
const char* text	To the output string.

Return Value

Number of text.

Example

```
EGL_FONT* bit_font;
…
bit_font = create_bitfont();
…
draw_text(bit_font, 100, 100, "font test");
```

▶**draw_text_pivot** function

```
int  draw_text_pivot(
      EGL_FONT*  pFont,
      int  x,
      int  y,
      const char*  text,
      int  pivot
);
```

Overview

Draw text. Left / Right rotation by 90 degrees.

Parameter

EGL_FONT* pFont	Font that use in characters output.
int x	X-coordinate of text.
int y	Y-coordinate of text.
const char* text	To the output string.
int pivot	PIVOT_RIGHT (Right rotation by 90 degrees)
	PIVOT_LEFT (LEFT rotation by 90 degrees)

Return Value

Number of text.

Example

```
EGL_FONT* bit_font;
…
bit_font = create_bitfont();
…
draw_text(bit_font, 100, 100, “font test”,PIVOT_RIGHT);
```

▶**draw_text_len** function

```
void  draw_text_len(
      EGL_FONT*  pFont,
      int  x,
      int  y,
      const char*  text,
      int  len
);
```

Overview

Draw text. (Set the length)

Parameter

EGL_FONT* pFont	Font that use in characters output.
int x	X-coordinate of text.
int y	Y-coordinate of text.
const char* text	To the output string.
int len	Text length.

Return Value

없음.

Example

```
EGL_FONT* bit_font;
…
bit_font = create_bitfont();
…
draw_text(bit_font, 100, 100, "font test",4);
```

▶**draw_text_in_box** function

```
void  draw_text_in_box(
      EGL_FONT*  pFont,
      EGL_RECT*  pRect,
      const char*  text,
      int align
);
```

Overview

Draw text in box.

Parameter

EGL_FONT* pFont	Font that use in characters output.

EGL_RECT* pRect	Output area.
	`typedef struct _tag_RECT{`
	`int x;`
	`int y;`
	`int w;`
	`int h;`
	`} EGL_RECT;`

const char* text	To the output string..

int align	Alignment within the area.
	`EGL_ALIGN_LEFT,`
	`EGL_ALIGN_RIGHT,`
	`EGL_ALIGN_TOP,`
	`EGL_ALIGN_BOTTOM,`
	`EGL_ALIGN_CENTER,`
	`EGL_ALIGN_MULTILINE`

Return Value

None.

Example

```
EGL_FONT* bit_font;
EGL_RECT rect;
…
bit_font = create_bitfont();
rect.x = 100;
rect.y = 100;
rect.w = 200;
rect.h = 40;
draw_text_in_box(bit_font, &rect, "font test", EGL_ALIGN_CENTER);
```

▶**text_width** function

```
int  text_width(
     EGL_FONT*  font,
     const char*  str
);
```

Overview

Get text width. (Unit pixel.)

Parameter

EGL_FONT* font	Font that use in characters output.
const char* str	String.

Return Value

String width.

Example

```
EGL_FONT* bit_font;
int font_width;
…
bit_font = create_bitfont();
…
font_width = text_width(bit_font, "font test");
```

EGL Primitives

Function	Description
draw_line	Draw line.
draw_hline	Draw horizontal line.
draw_vline	Draw vertical line.
draw_thickline	Draw thick line.
draw_rect	Draw a rectangle.
draw_rectfill	Draw a filled rectangle.
draw_rectfill_gradient	Draw a rectangle with a gradient effect.
draw_rectfill_h_gradient	Draw a rectangle with a horizontal gradient effect.
draw_rectfill_v_gradient	Draw a rectangle with a vertical gradient effect.
draw_roundrect	Draw a round rectangle.
draw_roundrectfill	Draw a filled round rectangle.
draw_arc	Draw an arc.
draw_pie	Draw a pie.
draw_piefill	Draw a filled pie.
draw_ellipse	Draw an ellipse.
draw_ellipsefill	Draw a filled ellipse.
draw_circle	Draw a circle.
draw_circlefill	Draw a filled circle.
Function	**Description**

draw_bezier	Draw Bezier curve.
draw_polyline	Draw poly line.
draw_polygon	Draw a polygon.
draw_polygonfill	Draw a filled polygon.

▶**draw_line** function

```
void  draw_line(
    int  x,
    int  y,
    int  x2,
    int  y2,
    EGL_COLOR  c
);
```

Overview

Draw line from the start point to the end point.

Parameter

int x	X-coordinate of start point.
int y	Y-coordinate of start point.
int x2	X-coordinate of end point.
int y2	Y-coordinate of end point.
EGL_COLOR c	Line color. (Use MAKE_COLORREF(r, g, b) macro function.)

Return Value

None.

Example

```
draw_line(100, 100, 200, 200, MAKE_COLORREF(0, 0, 0);
draw_line(100, 100, 100, 200, MAKE_COLORREF(255, 0, 0);
```

▶**draw_hline** function

```
void draw_hline(
    int  x,
    int  y,
    int  x2,
    EGL_COLOR  c
);
```

Overview

Draw horizontal line from the (x, y) to the (x2, y).

Parameter

int x	X-coordinate of start point.
int y	Y-coordinate of start point and end point.
int x2	X-coordinate of end point.
EGL_COLOR c	Line color. (Use MAKE_COLORREF(r, g, b) macro function.)

Return Value

None.

Example

```
draw_hline(100, 100, 200, MAKE_COLORREF(0, 0, 0));
draw_hline(100, 200, 200, MAKE_COLORREF(0, 0, 255));
```

▶**draw_vline** function

```
void draw_vline(
    int  x,
    int  y,
    int  y2,
    EGL_COLOR  c
);
```

Overview

Draw certical line from the (x, y) to the (x, y2).

Parameter

int x	X-coordinate of start point and end point.
int y	Y-coordinate of start point.
int y2	Y-coordinate of end point.
EGL_COLOR c	Line color. (Use MAKE_COLORREF(r, g, b) macro function.)

Return Value

None.

Example

```
draw_vline(100, 100, 200, MAKE_COLORREF(0, 0, 0));
draw_vline(200, 100, 200, MAKE_COLORREF(0, 255, 0));
```

▶**draw_thickline** function

```
void draw_thickline(
    int  x1,
    int  y1,
    int  x2,
    int  y2,
    U8  width,
    EGL_COLOR  color
);
```

Overview

Draw thick line from the start point to the end point.

Parameter

int x1	X-coordinate of start point.
int y1	Y-coordinate of start point.
int x2	X-coordinate of end point.
int y2	Y-coordinate of end point.
U8 width	Thickness of line.
EGL_COLOR color	Thick line color. (Use MAKE_COLORREF(r, g, b) macro function.)

Return Value

None.

Example

```
draw_thickline(100, 130, 200, 130, 2, MAKE_COLORREF(0, 0, 0));
draw_thickline(100, 200, 100, 300, 3, MAKE_COLORREF(255, 0, 0));
```

▶**draw_rect** function

```
void draw_rect(
    int  x,
    int  y,
    int  w,
    int  h,
    EGL_COLOR  c
);
```

Overview

Draw a rectangle with given start point, width and height.

Parameter

int x	X-coordinate of start point.
int y	Y-coordinate of start point.
int w	Width of rectangle.
int h	Height of rectangle.
EGL_COLOR c	Rectangle color. (Use MAKE_COLORREF(r, g, b) macro function.)

Return Value

None.

Example

```
draw_rect(100, 100, 200, 200, MAKE_COLORREF(0, 0, 0));
draw_rect(150, 150, 100, 100, MAKE_COLORREF(0, 0, 255));
```

▶**draw_rectfill** function

```
void draw_rectfill(
    int  x,
    int  y,
    int  w,
    int  h,
    EGL_COLOR  c
);
```

Overview

Draw a filled rectangle with given start point, width and height.

Parameter

int x	X-coordinate of start point.
int y	Y-coordinate of start point.
int w	Width of rectangle.
int h	Height of rectangle.
EGL_COLOR c	Rectangle color. (Use MAKE_COLORREF(r, g, b) macro function.)

Return Value

None.

Example

```
draw_rectfill(100, 100, 50, 50, MAKE_COLORREF(0, 255, 0));
draw_rectfill(200, 200, 50, 50, MAKE_COLORREF(0, 0, 255));
```

▶**draw_rectfill_gradient** function

```
void draw_rectfill_gradient(
    int  x,
    int  y,
    int  w,
    int  h,
    EGL_COLOR startcolor,
    EGL_COLOR endcolor,
    BOOL  bVertical
);
```

Overview

Draw a rectangle with a gradient effect.

Parameter

int x	X-coordinate of start point.
int y	Y-coordinate of start point.
int w	Width of rectangle.
int h	Height of rectangle.
EGL_COLOR startcolor	Gradient start color. (Use MAKE_COLORREF(r, g, b) macro function.)
EGL_COLOR endcolor	Gradient end color. (Use MAKE_COLORREF(r, g, b) macro function.)
BOOL bVertical	Gradient direction. TRUE == vertical, FALSE == horizontal.

Return Value

None.

Example

```
draw_rectfill_gradient(100,100,200,50,MAKE_COLORREF(0,255,0),MAKE_COLORREF(255,255,255),FALSE);
draw_rectfill_gradient(100,200,50,200,MAKE_COLORREF(255,255,255),MAKE_COLORREF(0,0,255),TRUE);
```

▶**draw_rectfill_h_gradient** function

```
void draw_rectfill_h_gradient(
    int  x,
    int  y,
    int  w,
    int  h,
    EGL_COLOR  startcolor,
    EGL_COLOR  endcolor
);
```

Overview

Draw a rectangle with a horizontal gradient effect.

Parameter

int x	X-coordinate of start point.
int y	Y-coordinate of start point.
int w	Width of rectangle.
int h	Height of rectangle.
EGL_COLOR startcolor	Gradient start color. (Use MAKE_COLORREF(r, g, b) macro function.)
EGL_COLOR endcolor	Gradient end color. (Use MAKE_COLORREF(r, g, b) macro function.)

Return Value

None.

Example

```
draw_rectfill_h_gradient(50,50,200,50,MAKE_COLORREF(255,255,255),MAKE_COLORREF(255,0,0));
```

▶**draw_rectfill_v_gradient** function

```
void draw_rectfill_v_gradient(
    int  x,
    int  y,
    int  w,
    int  h,
    EGL_COLOR  startcolor,
    EGL_COLOR  endcolor
);
```

Overview

Draw a rectangle with a vertical gradient effect.

Parameter

int x	X-coordinate of start point.
int y	Y-coordinate of start point.
int w	Width of rectangle.
int h	Height of rectangle.
EGL_COLOR startcolor	Gradient start color. (Use MAKE_COLORREF(r, g, b) macro function.)
EGL_COLOR endcolor	Gradient end color. (Use MAKE_COLORREF(r, g, b) macro function.)

Return Value

None.

Example

```
draw_rectfill_v_gradient(50,50,200,50,MAKE_COLORREF(255,255,255),MAKE_COLORREF(255,0,0));
```

▶**draw_roundrect** function

```
void draw_roundrect(
    int  x0,
    int  y0,
    int  w,
    int  h,
    int  corner,
    EGL_COLOR  c
);
```

Overview

Draw a round rectangle with given start point, width and height.

Parameter

int x0	X-coordinate of start point.
int y0	Y-coordinate of start point.
int w	Width of round rectangle.
int h	Height of round rectangle.
int corner	Corner roundness value.
EGL_COLOR c	Round rectangle color. (Use MAKE_COLORREF(r, g, b) macro function.)

Return Value

None.

Example

```
draw_roundrect(100, 100, 100, 100, 10, MAKE_COLORREF(0, 0, 0);
draw_roundrect(400, 200, 200, 200, 20, MAKE_COLORREF(0, 255, 0);
```

▶**draw_roundrectfill** function

```
void draw_roundrectfill(
    int  x0,
    int  y0,
    int  w,
    int  h
    int  corner,
    EGL_COLOR  c
);
```

Overview

Draw a filled round rectangle with given start point, width and height.

Parameter

int x0	X-coordinate of start point.
int y0	Y-coordinate of start point.
int w	Width of round rectangle.
int h	Height of round rectangle.
int corner	Corner roundness value.
EGL_COLOR c	Round rectangle color. (Use MAKE_COLORREF(r, g, b) macro function.)

Return Value

None.

Example

```
draw_roundrectfill(100, 100, 100, 100, 10, MAKE_COLORREF(255, 0, 0));
draw_roundrectfill(300, 200, 200, 200, 15, MAKE_COLOEREF(0, 255, 0));
```

▶**draw_arc** function

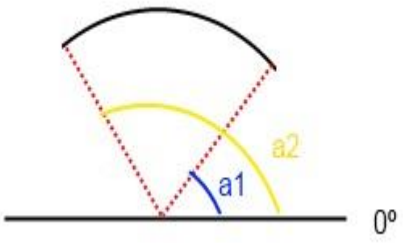
```
void draw_arc(
    int  x,
    int  y,
    int  rx,
    int  ry,
    int  a1,
    int  a2,
    EGL_COLOR  c
);
```

Overview

Draw an arc with given center coordinate, horizontal radius and vertical radius.
Draw from the start angle (a1) to the end angle (a2).

Parameter

int x	X-coordinate of center.
int y	Y-coordinate of center.
int rx	Horizontal radius.
int ry	Vertical radius.
int a1	Start angle.
int a2	End angle.
EGL_COLOR c	Arc color. (Use MAKE_COLORREF(r, g, b) macro function.)



Return Value

None.

Example

```
draw_arc(100, 100, 30, 30, 0, 90, MAKE_COLORREF(255, 0, 0));
draw_arc(200, 200, 40, 30, 90, 180, MAKE_COLORREF(0, 0, 255));
```

▶**draw_pie** function

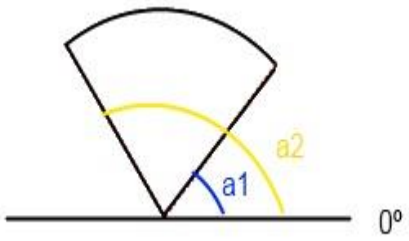
```
draw_pie(
    int  x,
    int  y,
    int  rx,
    int  ry,
    int  a1,
    int  a2,
    EGL_COLOR  c
);
```

Overview

Draw a pie with given center coordinate, horizontal radius and vertical radius.
Draw from the start angle (a1) to the end angle (a2).

Parameter

int x	X-coordinate of center.
int y	Y-coordinate of center.
int rx	Horizontal radius.
int ry	Vertical radius.
int a1	Start angle.
int a2	End angle.
EGL_COLOR c	Pie color. (Use MAKE_COLORREF(r, g, b) macro function.)



Return Value

None.

Example

```
draw_pie(100, 100, 50, 50, 45, 90, MAKE_COLORREF(0, 0, 255));
draw_pie(200, 100, 60, 60, 90, 180, MAKE_COLORREF(255, 0, 0));
```

▶**draw_piefill** function

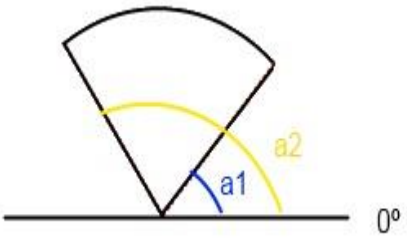
```
void  draw_piefill(
    int  x,
    int  y,
    int  rx,
    int  ry,
    int  a1,
    int  a2,
    EGL_COLOR  c
);
```

Overview

Draw a filled pie with given center coordinate, horizontal radius and vertical radius.
Draw from the start angle (a1) to the end angle (a2).

Parameter

int x	X-coordinate of center.
int y	Y-coordinate of center.
int rx	Horizontal radius.
int ry	Vertical radius.
int a1	Start angle.
int a2	End angle.
EGL_COLOR c	Pie color. (Use MAKE_COLORREF(r, g, b) macro function.)



Return Value

None.

Example

```
draw_pie(100, 100, 50, 50, 45, 90, MAKE_COLORREF(0, 0, 255));
draw_pie(200, 100, 60, 60, 90, 180, MAKE_COLORREF(255, 0, 0));
```

▶**draw_ellipse** function

```
void draw_ellipse(
    int  x,
    int  y,
    int  rx,
    int  ry,
    EGL_COLOR  c
);
```

Overview

Draw an ellipse with given center coordinate, horizontal radius and vertical radius.

Parameter

int x	X-coordinate of center.
int y	Y-coordinate of center.
int rx,	Horizontal radius.
int ry	Vertical radius.
EGL_COLOR c	Ellipse color. (Use MAKE_COLORREF(r, g, b) macro function.)

Return Value

None.

Example

```
draw_ellipse(100, 100, 50, 100, MAKE_COLORREF(0, 0, 255));
draw_ellipse(300, 200, 100, 50, MAKE_COLORREF(255, 0, 0));
```

▶**draw_ellipsefill** function

```
void draw_ellipsefill(
    int  x,
    int  y,
    int  rx,
    int  ry,
    EGL_COLOR  c
);
```

Overview

Draw a filled ellipse with given center coordinatem, horizontal radius and vertical radius.

Parameter

int x	X-coornate of center.
int y	Y-coornate of center.
int rx,	Horizontal radius.
int ry	Vertical radius.
EGL_COLOR c	Ellipse color. (Use MAKE_COLORREF(r, g, b) macro function.)

Return Value

None.

Example

```
draw_ellipsefill(100, 100, 50, 100, MAKE_COLORREF(0, 0, 255));
draw_ellipsefill(300, 200, 100, 50, MAKE_COLORREF(255, 0, 255));
```

▶**draw_circle** function

```
void draw_circle(
    int  x,
    int  y,
    int  r,
    EGL_COLOR  c
);
```

Overview

Draw a circle with given center coordinate and radius.

Parameter

int x	X-coordinate of center.
int y	Y-coordinate of center.
int r	Radiuse.
EGL_COLOR c	Circle color. (Use MAKE_COLORREF(r, g, b) macro function.)

Return Value

None.

Example

```
draw_circle(100, 100, 50, MAKE_COLORREF(255, 0, 255));
draw_circle(300, 200, 100, MAKE_COLORREF(0, 0, 255));
```

▶**draw_circlefill** function

```
void draw_circlefill(
    int  x,
    int  y,
    int  r,
    EGL_COLOR  c
);
```

Overview

Draw a filled circle with given center coordinate and radius.

Parameter

int x	X-coordinate of center.
int y	Y-coordinate of center.
int r	Radius.
EGL_COLOR c	Circle color. (Use MAKE_COLORREF(r, g, b) macro function.)

Return Value

None.

Example

```
draw_circle(100, 100, 70, MAKE_COLORREF(0, 0, 255));
draw_circle(300, 200, 50, MAKE_COLORREF(0, 255, 0));
```

▶**draw_bezier** function

```
void draw_bezier(
    EGL_POINT*  pts,
    int  n,
    int  s,
    EGL_COLOR  c
);
```

Overview

Draw Bezier curve.

Parameter

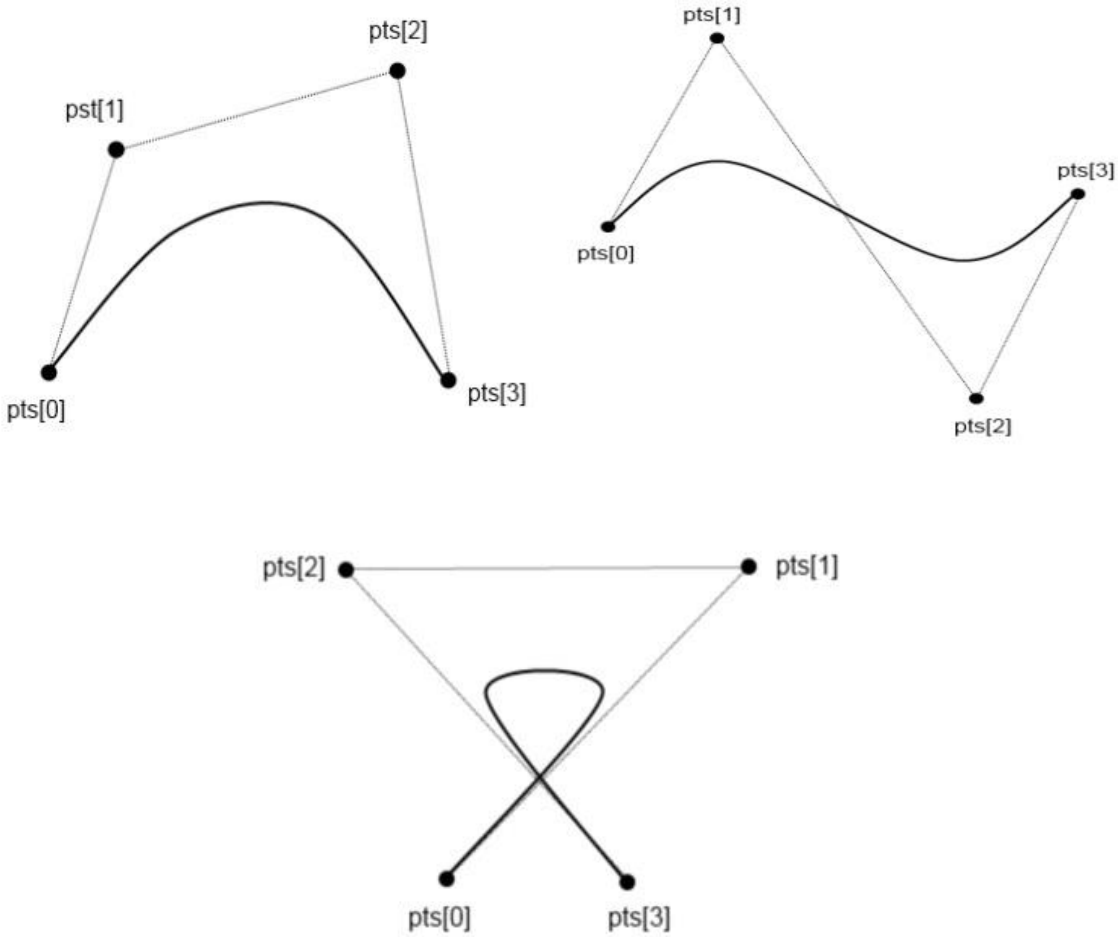
EGL_POINT* pts	Pointer to the coordinate array of Bezier curve.
int n	Number of coordinate.
int s	Decide Number of pointer between the coordinate and the coordinate.
EGL_COLOR c	Bezier curve color. (Use MAKE_COLORREF(r, g, b) macro function.)

Return Value

None.

Example

```
EGL_POINT* pts[4];
pts[0].x = 100; pts[0].y = 200;
pts[1].x = 200; pts[1].y = 50;
pts[2].x = 300; pts[2].y = 50;
pts[3].x = 400; pts[3].y = 200;

draw_bezier(pts, 4, 20, MAKE_COLORREF(255, 0, 255));
```



< Bezier curve image >

▶**draw_polyline** function

```
void draw_polyline(
    EGL_POINT*  p,
    int  n,
    EGL_COLOR  c
);
```

Overview

Draw poly line.

Parameter

EGL_POINT* p	Pointer to the coordinate array of poly line.
int n	Number of coordinate.
EGL_COLOR c	Poly line color. (Use MAKE_COLORREF(r, g, b) macro function.)

Return Value

None.

Example

```
EGL_POINT p[3];
p[0].x = 100; p[0].y = 100;
p[1].x = 130; p[1].y = 100;
p[2].x = 150; p[2].y = 150;
draw_polyline(p, 3, MAKE_COLORREF(0, 0, 0));
```

▶**draw_polygon** function

```
void draw_polygon(
    EGL_POINT* ptable,
    int  cnt,
    EGL_COLOR  c
);
```

Overview

Draw a polygon.

Parameter

EGL_POINT* p	Pointer to the coordinate array of polygon.
int n	Number of coordinate.
EGL_COLOR c	Polygon color. (Use MAKE_COLORREF(r, g, b) macro function.)

Return Value

None.

Example

```
EGL_POINT ptable[5];
ptable[0].x = 100; ptable[0].y = 100;
ptable[1].x = 150; ptable[1].y = 50;
ptable[2].x = 200; ptable[2].y = 100;
ptable[3].x = 135; ptable[3].y = 150;
ptable[4].x = 115; ptable[4].y = 150;
draw_polygon(ptable, 5, MAKE_COLORREF(0, 0, 255));
```

▶**draw_polygonfill** function

```
void draw_polygonfill(
    EGL_POINT*  ptable,
    int  cnt,
    EGL_COLOR  c
);
```

Overview

Draw a filled polygon.

Parameter

EGL_POINT* p	Pointer to the coordinate array of filled polygon.
int n	Number of coordinate.
EGL_COLOR c	Polygon color. (Use MAKE_COLORREF(r, g, b) macro function.)

Return Value

None.

Example

```
EGL_POINT ptable[3];
ptable[0].x = 100; ptable[0].y = 100;
ptable[1].x = 150; ptable[1].y = 150;
ptable[2].x = 100; ptable[2].y = 150;
draw_polygon(ptable, 3, MAKE_COLORREF(0, 255, 0));
```

EGL etc.

Function	Description
egl_init	EGL init function.
egl_show_object	Decide whether draw object.
egl_object_set_redraw	Decide whether redraw object.

Define

EGL_MSG_ID

```
typedef  enum  enumMSG
{
    EGL_MSG_DRAW = 0,
    EGL_MSG_DELETE,
    EGL_MSG_FOCUS,
    EGL_MSG_UNFOCUS,
    EGL_MSG_KEY_UP,
    EGL_MSG_KEY_DOWN,
    EGL_MSG_TOUCHED,
    EGL_MSG_UNTOUCHED,
    EGL_MSG_MOVE,
    EGL_MSG_TIMETICK
 } EGL_MSG_ID;
```

EGL_MSG

```
typedef  struct  _tagMessage
{
    EGL_MSGID  msgID;
    EGL_HANDLE  hObj;
    EGL_HANDLE  hWin;
    Union
    {
        EGL_POINT  point;
        U32  key;
    } param;
} EGL_MSG;
```

▶ **egl_init** function

```
BOOL  egl_init( void );
```

Overview

EGL init function. You need to call this function.

Parameter

None.

Return Value

TRUE or FALSE

▶**egl_show_object** function

```
void  egl_show_object(
      EGL_HANDLE  h,
      BOOL  bShow
);
```

Overview

Decide whether to draw object.

If bShow is FALSE, does not draw the corresponding object in egl_draw function.

Parameter

EGL_HANDLE h	Handle of object.
BOOL bShow	Whether to Draw. TRUE or FALSE.

Return Value

None.

▶**egl_object_set_redraw** function

```
void  egl_object_set_redraw(
      EGL_HANDLE  handle
);
```

Overview

Decide whether redraw object.

Parameter

EGL_HANDLE handle	Handle of object.

Return Value

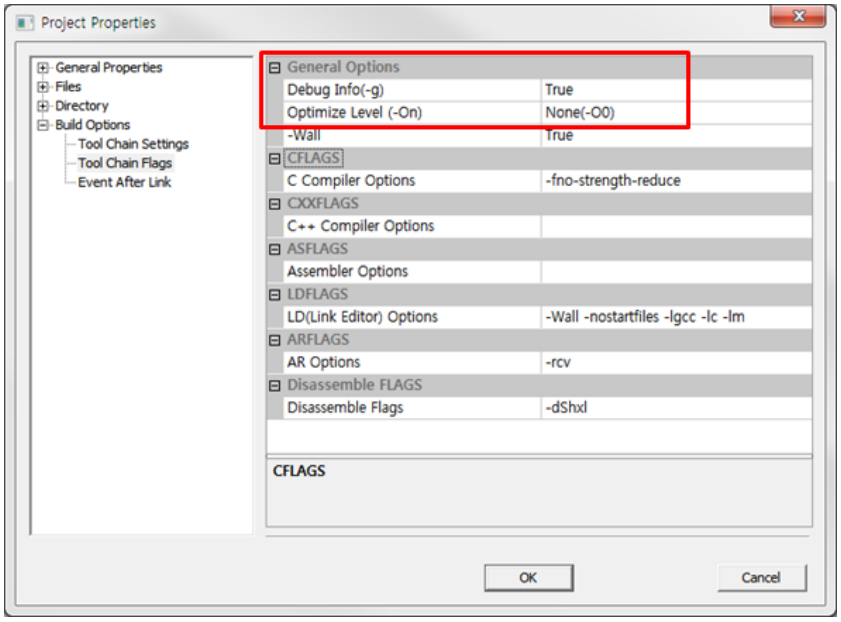
None.

14. Debugging

You need E-Con(JTAG device) and EConMan program in order to do debugging. In order to use E-Con, need to install device driver. If you don't install, refer to "1.Software Development Enviroment"

14.1 Preparing debugging

Open the project for debugging. And run Project → Properties.



If window is displayed like above image,

Set to TRUE on Debug Info(-g), Set to None(-O0) on Optimize Level(-On).

And run Build → Rebuild Project.

14.2 Debugging

There are two way for debugging according to status.

1. Case that is operate program in address 0 without bootloader
2. Case that is operate program in ram using bootloader

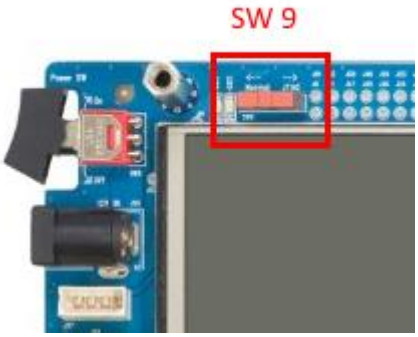
Can distinguish two case by Linker Script file.

If Linker Script file is amazon2.ld, program is that operated in address 0.

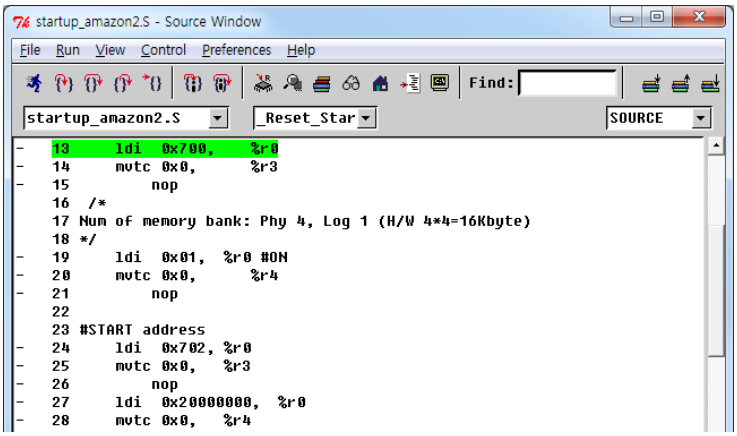
If Linker Script file is amazon2_ram.ld, program is that operated in ram.

▶ Case that is operate program in address 0 (in Serial Flash)

1) Download binary file that created by build on address 0
2) Set debug mode (JTAG mode) by set the SW9. Connect E-Con and power on.
   (Debug mode is status that SW9 was set right side)

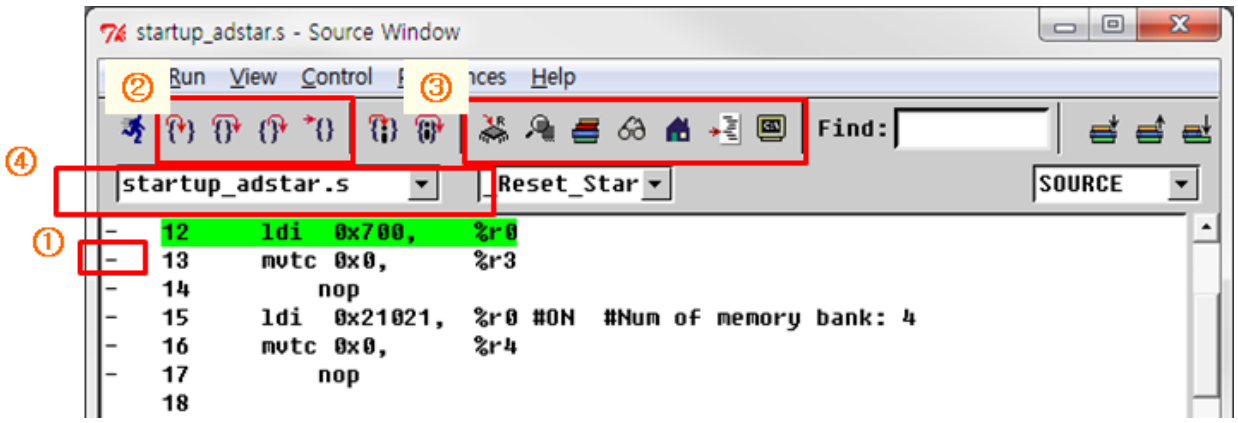


3) Run Debug → Start Debugger (F5).
   Then, following window is displayed, and can start debugging.



   If green bar is output like above image, will had connected well.
   If green bar is not output, retry.

4) Debugger window function can be described like following.



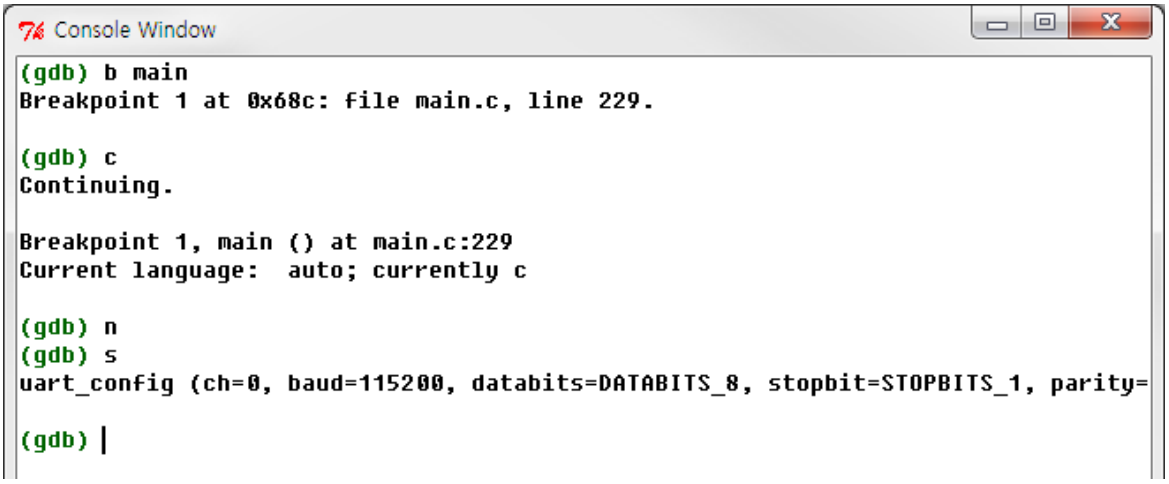
① : Break point set/unset

② : 'step', 'next', 'finish', 'continue' icon

③ : 'register', 'memory', 'stack', 'watch expressions', 'local variable' icon
'c:' is icon that console window displayed.
you can run debugging with GDB command in console window.

④ : Move source code.

5) Debugging in console window.



b : Set break point.

c : continue. Run until next break point.

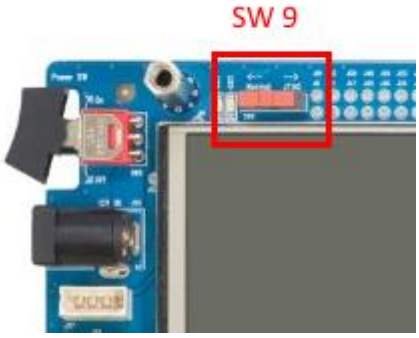
n : next. Run one command.

s : step. Run one command. (run go inside function in the case of function)

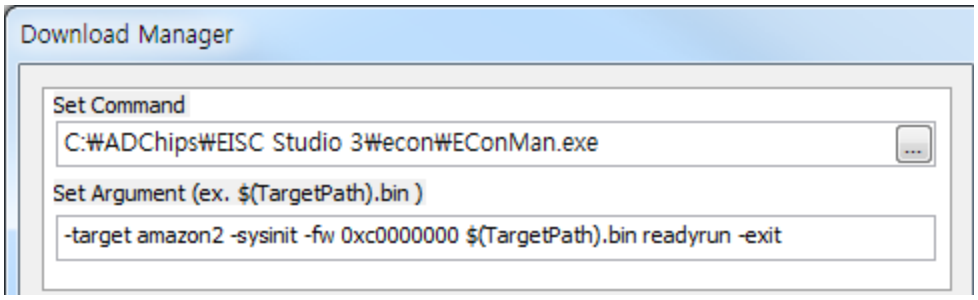
q : exit.

▶ Case that us ioerate program in Ram

1) Power on as debug mode in status that bootloader downloaded.
(Debug mode is status that SW9 was set right side)



2) Run Build → Download Option.
Then, following window is displayed, and write argument like following.



-target amazon2 –sytsinit –fw 0xc0000000 $(TargetPath).bin readyrun –exit

3) Run Build → Download to Target, Debug→Start Debugger (F5).
4) Then, following window is displayed, and can start debugging. If confirm green bar then start debugging like previously.