advanced digital chips inc

adStar-L

- SDK Reference Manual -

32bits EISC Microprocessor*adStar-L*

Ver 1.00
June 23, 2015

Advanced Digital Chips Inc.

History

2015-06-23	Created Preliminary Specification

adStar-L SDK Reference Manual

ⓒAdvanced Digital Chips Inc.
All right reserved.
No part of this document may be reproduced in any form without written permission from Advanced Digital Chips Inc.
Advanced Digital Chips Inc. reserves the right to change in its products or product specification to improve function or design at any time, without notice.

Office

22F, Bldg A, KeumkangPenterium IT Tower,
810, Gwanyang_dong, Dongan-Gu, Anyang-si, Gyeonggi-do, 431-060, Korea
Tel : +82-31-463-7500
Fax : +82-31-463-7588
URL : http://www.adc.co.kr

— Table of Contents —

1. SOFTWARE DEVELOPMENT ENVIRONMENT 12

2. USING ADSTAR-L SDK 18

2.1 adStar-L SDK organization and basic definitions. 18
2.2 adStar-L SDK library build 20
2.3 Demo Program Execution 21
2.3.1 Project build & download 22
2.3.2 Copying Files to Nand Flash (SD Card) 24
2.3.3 Demo Program build & download 24
2.4 New adStar-L Project Creation 25

3. BOOTLOADER 30

3.1Bootloader. (example / bootloader) 30
3.1.1 Use Bootloader 31
3.1.2 Bootloader Mode 32
A. Remote Communication Mode 32
B. Mass Storage Mode 34
C. Execute Mode Mode 35
D. User Define Mode (execute_fat) 37
3.2 Nand Boot Code 38
3.2.1 Use Nand Boot Code 38
3.3 Use without BootLoader 39

4. LIB_CONFIG.H 41

5. UART 43

5.1 uart_config 43
5.2 uart_putch 44
5.3 uart_putdata 44
5.4 uart_putstring 44
5.5 uart_getch 45
5.6 uart_getdata 45
5.7 uart_rx_flush 45
5.8 uart_tx_flush 46
5.9 set_debug_channel 46
5.10 get_debug_channel 46
5.11 debugprintf 47
5.12 debugstring 47
5.13 PRINTLINE 47
5.14 PRINTVAR(A) 48

5.15 UART Example 48

6. INTERRUPT **49**

6.1 init_interrupt 49
6.2 set_interrupt 49
6.3 enable_interrupt 49
6.4 Interrupt Example 51

7. TIMER **52**

7.1 set_timer 52
7.2 stop_timer 52
7.3 delayms 52
7.4 TIMER Example 53

8. GRAPHIC **54**

8.1 setscreen 54
8.2 createframe 54
8.3 setframebuffer 55
8.4 setdoubleframebuffer 56
8.5 setframebufferxy 56
8.6 set_draw_target 57
8.7 get_draw_target 57
8.8 getbackframe 57
8.9 getfrontframe 58
8.10 flip 58
8.11 getscreenwidth 58
8.12 getscreenheight 59
8.13 getscreenpitch 59
8.14 getscreenbpp 59
8.15 drawputpixel 60
8.16 draw_line 60
8.17 draw_rect 60
8.18 draw_rectfill 61
8.19 draw_roundrect 61
8.20 draw_roundrectfill 62
8.21 draw_circle 62
8.22 draw_circlefill 63
8.23 draw_ellipse 63
8.24 draw_ellipsefill 63
8.25 loadbmp 64
8.26 loadbmpp 64
8.27 loadjpg 65

8.28 loadjpgp *65*
8.29 loadtga *65*
8.30 loadtgap *66*
8.31 loadpng *66*
8.32 loadpngp *66*
8.33 loadsurf *67*
8.34 loadjpg_hw *67*
8.35 draw_surface *67*
8.36 draw_surface_rect *68*
8.37 draw_set_clip_winodw *68*
8.38 draw_surface_scale *69*
8.39 draw_surface_scalerect *69*
8.40 release_surface *70*
8.41 savebmp *71*
8.42 savejpg *71*
8.43 createsurface_from *72*
8.44 Single frame & double frame usage example *73*
8.45 Set frame buffer xy usage example *76*
8.46 Graphic Example *77*

9. MOVIE FILE PLAY **79**

9.1 loadmovie *79*
9.2 release_movie *79*
9.3 movie_play *80*
9.4 movie_drawnext *80*
9.5 movie_drawnextex *81*
9.6 movie_seek *81*
9.6 movie_current_time *82*
9.8 example *82*

10. SOUND **83**

10.1 sound_init() *83*
10.2 sound_loadwav *83*
10.3 sound_loadmp3 ..... *83*
10.4 sound_release *84*
10.5 sound_play *84*
10.6 sound_stop *85*
10.7 sound_vol *85*
10.8 sound_pause *85*
10.9 sound_resume *86*
10.10 sound_isplay *86*
10.11 sound_ispause *86*

10.12 Sound Example 87

11. FILE SYSTEM **89**

11.1 f_mount 89
11.2 f_chdrive 89
11.3 f_chdir 90
11.4 FILE System Example 90

12. FONT **91**

12.1 create_bmpfont 91
12.2 release_bmpfont 91
12.3 bmpfont_draw 91
12.4 bmpfont_draw_vleft 92
12.5 bmpfont_draw_vright 92
12.6 egl_font_set_color 93
12.7 bmpfont_makesurface 93
12.8 bmpfont_setkerning 93
12.9 bmpfont_setautokerning 94
12.10 create_bitfont 94
12.11 release_bitfont 94
12.12 bitfont_draw 95
12.13 bitfont_draw_vleft 95
12.14 bitfont_draw_vright 95
12.15 bitfont_makesurface 96
12.16 FONT Example 97

13. SPI **102**

13.1 spi_master_init 102
13.2 spi_set_freq 102
13.3 spi_master_xfer 102
13.4 spi_wait_empty_fifo 103
13.5 SPI Example 103

14. EGL LIBRARY **112**

WINDOW 115
▶*egl_create_window function* 116
▶*egl_window_show function* 117
▶*egl_window_isshow function* 118
▶*egl_window_invalidate function* 119
▶*egl_window_invalidate_rect function* 120
▶*egl_window_redraw_rect function* 121
▶*egl_window_set_bg function* 122

▶ *egl_window_get_bg function* *123*
▶ *egl_window_add_object function* *124*
▶ *egl_window_set_callback function* *125*
▶ *egl_window_get_active function* *126*
▶ *egl_user_touch_input function* *127*
▶ *egl_draw function* *128*
▶ *egl_visible_object function* *129*
▶ *egl_window_delete_object function* *130*
▶ *egl_release_window function* *131*
▶ *Window Example* *132*
BUTTON 134
▶ *egl_create_button function* *135*
▶ *egl_button_callback function* *136*
▶ *egl_release_button function* *137*
▶ *Button Example.* *138*
IMAGE BUTTON 139
▶ *egl_create_image_button function* *140*
▶ *egl_image_button_callback function* *141*
▶ *egl_release_image_button function* *141*
▶ *Image Button Example* *143*
TOGGLE IMAGE BUTTON 144
▶ *egl_create_toggle_image function* *145*
▶ *egl_toggle_image_callback function* *146*
▶ *egl_toggle_image_set_on function* *147*
▶ *egl_release_toggle_image function* *148*
▶ *Toggle Image Button Example.* *149*
LABEL 150
▶ *egl_create_label function* *151*
▶ *egl_label_callback function* *152*
▶ *egl_label_set_text function* *153*
▶ *egl_label_set_redraw_bg function* *154*
▶ *egl_label_set_color function* *155*
▶ *egl_release_label function* *156*
▶ *Label Example.* *157*
CHECK BUTTON / RADIO BUTTON 158
▶ *egl_create_checkbutton function* *159*
▶ *egl_checkbutton_callback function* *160*
▶ *egl_checkbutton_set_check function* *161*
▶ *egl_checkbutton_get_check function* *162*
▶ *egl_checkbutton_set_style function* *163*
▶ *egl_release_checkbutton function* *164*

▶ *Check / Radio button Example.* 165
PROGRESS BAR 167
▶ *egl_create_progressbar function* 168
▶ *egl_progressbar_set_barcolor function* 169
▶ *egl_progressbar_set_bgcolor function* 170
▶ *egl_progressbar_set_textcolor function* 171
▶ *egl_progressbar_set_text function* 172
▶ *egl_progressbar_set_pos function* 173
▶ *egl_progressbar_get_pos function* 174
▶ *egl_release_progressbar function* 175
▶ *Progress Bar Example.* 176
SCROLL BAR 177
▶ *egl_create_scrollbar function* 179
▶ *egl_scrollbar_callback function* 180
▶ *egl_scroll_set_position function* 181
▶ *egl_scroll_get_position function* 182
▶ *egl_scroll_set_totalcount function* 183
▶ *egl_scroll_get_totalcount function* 184
▶ *egl_scroll_set_viewcount function* 185
▶ *egl_scroll_get_viewcount function* 186
▶ *egl_scroll_set_upcount function* 187
▶ *egl_scroll_get_upcount function* 188
▶ *egl_scroll_set_bgcolor function* 189
▶ *egl_scroll_set_size function* 190
▶ *egl_release_scrollbar function* 191
▶ *Scroll Bar Example.* 192
SLIDER 193
▶ *egl_create_slider function* 195
▶ *egl_slider_callback function* 196
▶ *egl_slider_set_pos function* 197
▶ *egl_slider_get_pos function* 198
▶ *egl_slider_set_range function* 199
▶ *egl_slider_get_range function* 200
▶ *egl_slider_stepit function* 201
▶ *egl_slider_set_tick_frequency function* 202
▶ *egl_slider_set_tick_style function* 203
▶ *egl_slider_set_thumb_size function* 204
▶ *egl_slider_get_thumb_size function* 205
▶ *egl_slider_set_barcolor function* 206
▶ *egl_slider_set_tickcolor function* 207
▶ *egl_slider_set_transparent function* 208

▶ *egl_release_slider function* *209*
▶ *Slider Example.* *210*
LIST BOX 211
▶ *egl_create_listbox function* *213*
▶ *egl_listbox_callback function* *214*
▶ *egl_listbox_additem function* *214*
▶ *egl_listbox_delitem function* *216*
▶ *egl_listbox_delitem_text function* *218*
▶ *egl_listbox_delitem_num function* *219*
▶ *egl_listbox_alldelitem function* *220*
▶ *egl_listbox_get_all_itemlist function* *221*
▶ *egl_listbox_get_sel_item function* *222*
▶ *egl_listbox_get_multiple_sel_itemlist function* *223*
▶ *egl_listbox_set_bgcolor function* *224*
▶ *egl_listbox_set_selbgcolor function* *225*
▶ *egl_listbox_set_textcolor function* *226*
▶ *egl_listbox_set_seltextcolor function* *227*
▶ *egl_listbox_set_textalign function* *228*
▶ *egl_listbox_set_scrollwidth function* *229*
▶ *egl_listbox_change_item_text function* *230*
▶ *egl_listbox_change_item_num function* *231*
▶ *egl_release_listbox function* *232*
▶ *List box Example.* *233*
CIRCLE GAUGE 236
▶ *egl_create_circle_gauge function* *237*
▶ *egl_circle_gauge_set_value function* *239*
▶ *egl_circle_gauge_get_value function* *240*
▶ *egl_release_circle_gauge function* *241*
▶ *Circle Gauge Example.* *242*
BAR GAUGE 244
▶ *egl_create_bar_gauge function* *245*
▶ *egl_bar_gauge_set_value function* *247*
▶ *egl_bar_gauge_get_value function* *248*
▶ *egl_release_bar_gauge function* *249*
▶ *Bar Gauge Example* *250*
PICTURE 252
▶ *egl_create_picture function* *253*
▶ *egl_picture_callback function* *254*
▶ *egl_picture_set function* *255*
▶ *egl_release_picture function* *256*
▶ *Bar Gauge Example* *257*

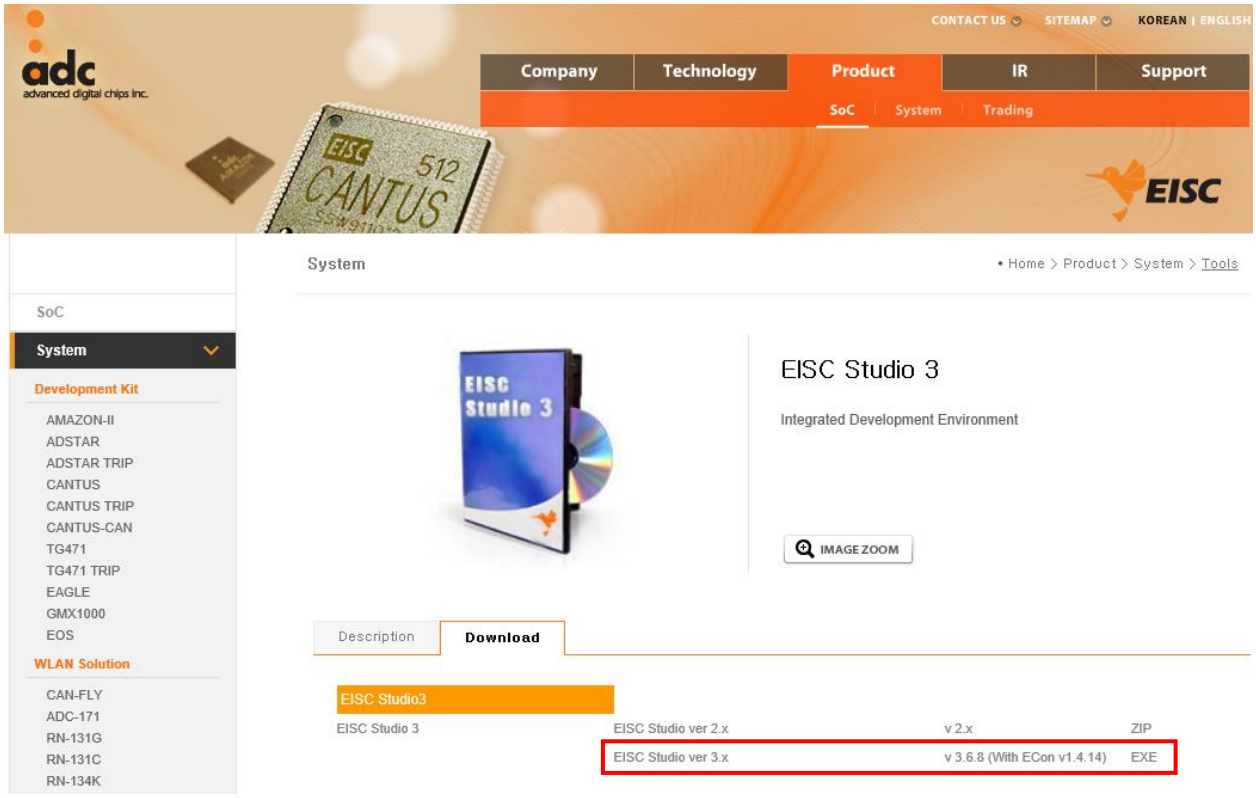
ANIMATION 259
▶ *egl_create_animation function* *260*
▶ *egl_release_animation function* *262*
▶ *Animation Example.* *263*
CUSTOM OBJECT 264
▶ *egl_create_custom_object function* *265*
▶ *Custom Example.* *267*
MESSAGEBOX 269
▶ *egl_show_messagebox function* *270*
▶ *MessageBox Example* *272*
EGL FONT 274
▶ *egl_get_font function* *275*
▶ *egl_set_font function* *276*
▶ *egl_font_set_bkmode function* *277*
▶ *egl_font_get_bk_color function* *278*
▶ *egl_font_set_bk_color function* *279*
▶ *egl_font_get_color function* *280*
▶ *egl_font_set_color function* *281*
▶ *create_bitfont function* *282*
▶ *release_bitfont function* *283*
▶ *create_bmpfont function* *284*
▶ *bmfont_release function* *285*
▶ *draw_text function* *286*
▶ *draw_text_pivot function* *287*
▶ *draw_text_len function* *288*
▶ *draw_text_in_box function* *289*
▶ *text_width function* *291*
EGL PRIMITIVES 292
▶ *draw_line function* *294*
▶ *draw_hline function* *295*
▶ *draw_vline function* *296*
▶ *draw_thickline function* *297*
▶ *draw_rect function* *298*
▶ *draw_rectfill function* *299*
▶ *draw_rectfill_gradient function* *300*
▶ *draw_rectfill_h_gradient function* *301*
▶ *draw_rectfill_v_gradient function* *302*
▶ *draw_roundrect function* *303*
▶ *draw_roundrectfill function* *304*
▶ *draw_arc function* *305*
▶ *draw_pie function* *306*

▶ *draw_piefill function* *307*
▶ *draw_ellipse function* *308*
▶ *draw_ellipsefill function* *309*
▶ *draw_circle function* *310*
▶ *draw_circlefill function* *311*
▶ *draw_bezier function* *312*
▶ *draw_polyline function* *314*
▶ *draw_polygon function* *315*
▶ *draw_polygonfill function* *316*
EGL ETC. 317
▶ *egl_init function* *318*
▶ *egl_show_object function* *319*
▶ *egl_object_set_redraw function* *320*
15. DEBUGGING **321**

1. SOFTWARE DEVELOPMENT ENVIRONMENT

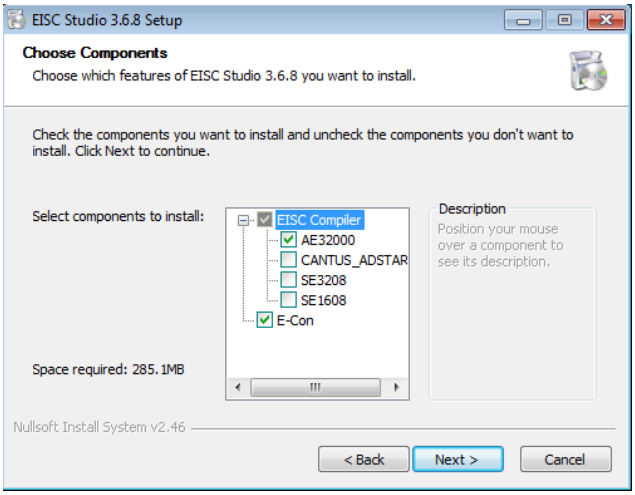
This chapter explains how to establish development environment of adStar-L based software. ADChips provides IDE called as EISC Studio3 that supports program code management, compilation, downloading and debugging. Therefore, users can set their development environment up easily by installing EISC Studio3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

1. Download EISC Studio ver 3.x – v3.6.8 from (http://www.adc.co.kr), Product → System → EISC Studio3 → Download.
   (Refer to an installation and usage guide on the page.)

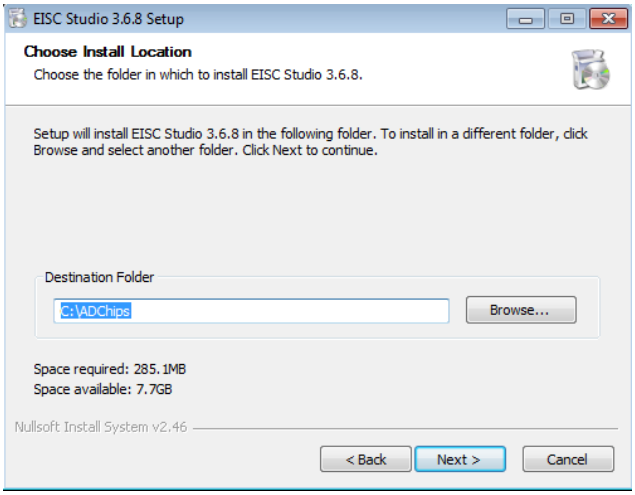


2. After executing downloaded file, ES3_setup_v3.6.8.exe then an installation window will be come up. To continue installation procedures, click 'Next'. Then, users should choose an appropriate E-CON driver and a compiler from a component selection window as below. Since, adStar-L contains AE32000 processor[1], user should choose AE32000. If a user is trying to use E-CON[2] device first time, then install the E-CON Driver by choosing E-CON.

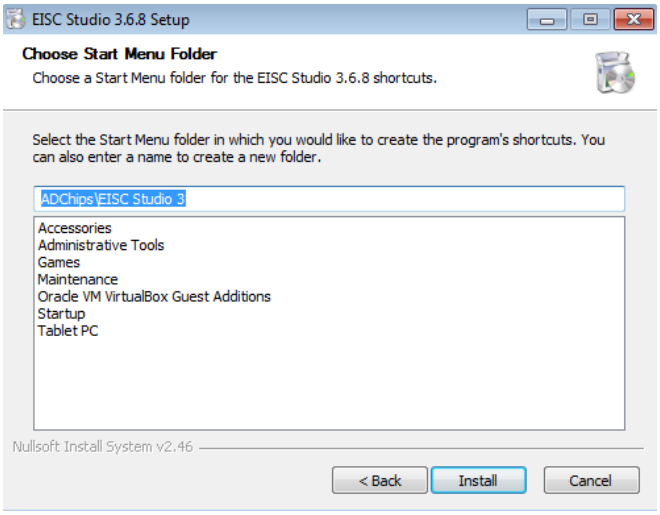
[1]ADChips developed 32bit Processor. (EISC Architecture)
[2]Device for a program download and debugging by connecting with adStar.

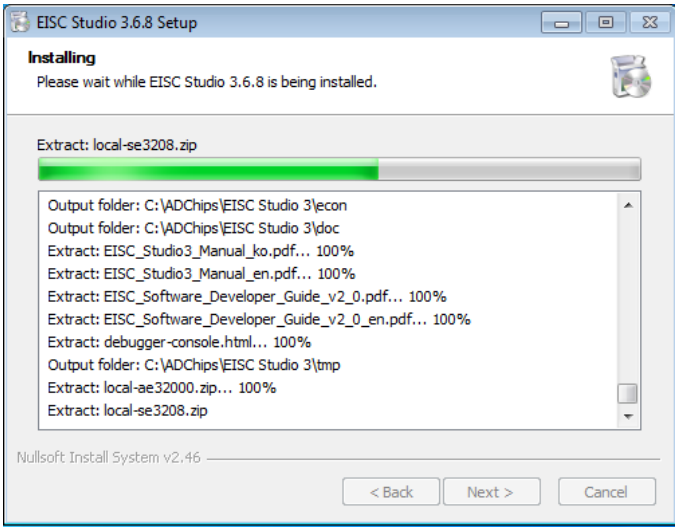


3. Click 'Next' after selecting the compiler and E-CON Driver. Then, user can decide a folder where they will be installed. A default folder name is "C:₩Program Files".
(Caution: when using 64bit OS, the default folder name is "C:₩Program Files(x86)". User must change the default folder name as c:₩Program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

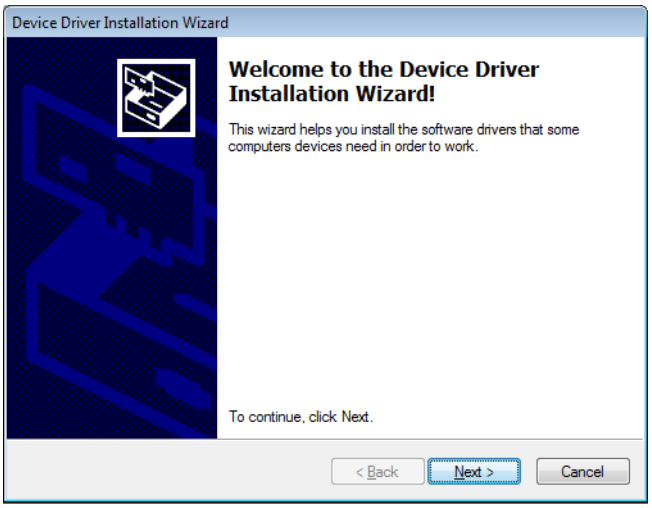


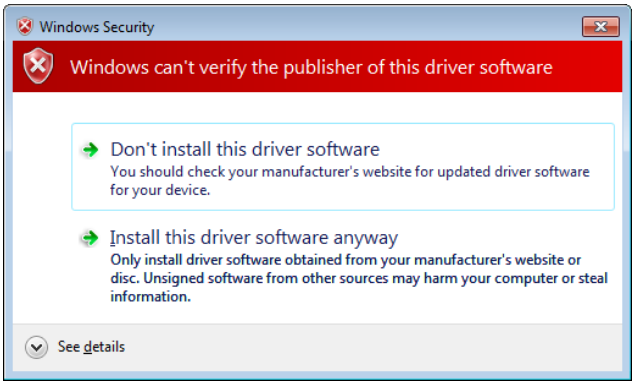
4. Below windows will be come up when click 'Next'. The chosen components are installed when click 'install'.



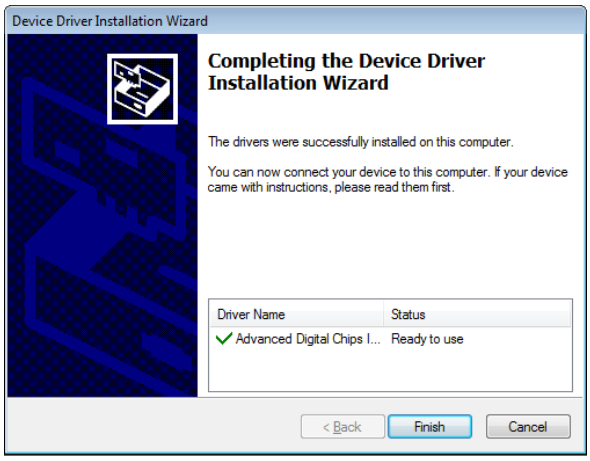


5. If E-CON is selected during the component selection procedure, following window for the E-CON driver installation is displayed. Clicking 'Next', start the driver installation. If Windows alarms secure warning, then choose "install this driver software".
(Caution: E-CON should be connected with a computer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

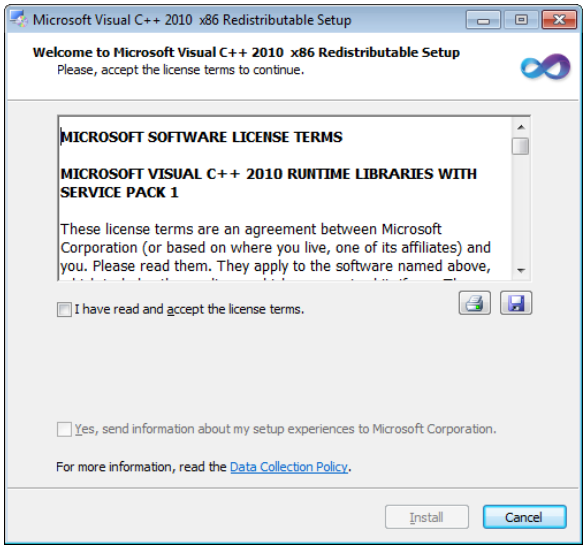




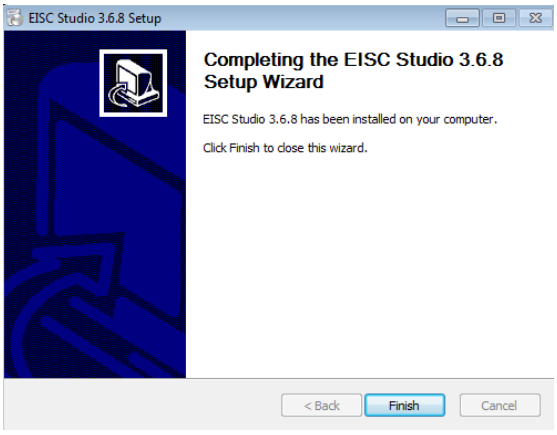
6. The end of the device driver installation loads below window, user can continue remained installation procedures of EISC Studio3 by clicking 'Next'.



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.

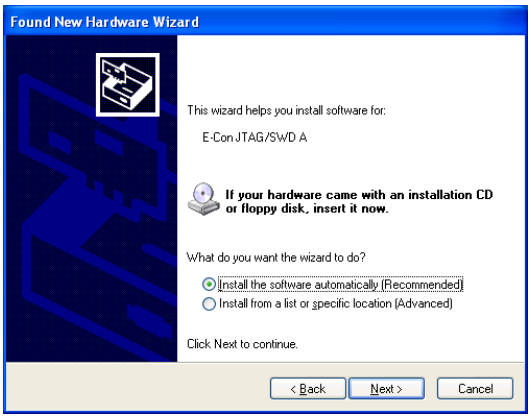


8. After finishing the installation, below window will be displayed. Clicking 'Next', installation will complete.

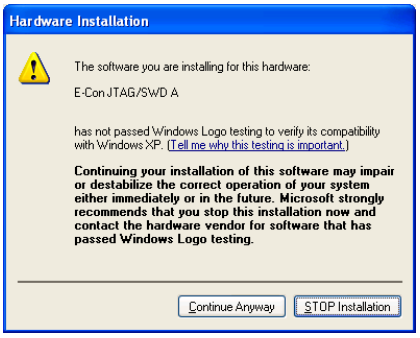


9. After installing EISC Studio 3, a download device; the E-CON driver should be installed. If the E-Con driver files are copied during the EISC Studio 3 installation, then install the E-Con driver by connecting E-Con with a computer. If not, copy the driver files by executing a DPInstx86.exe (32bit)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

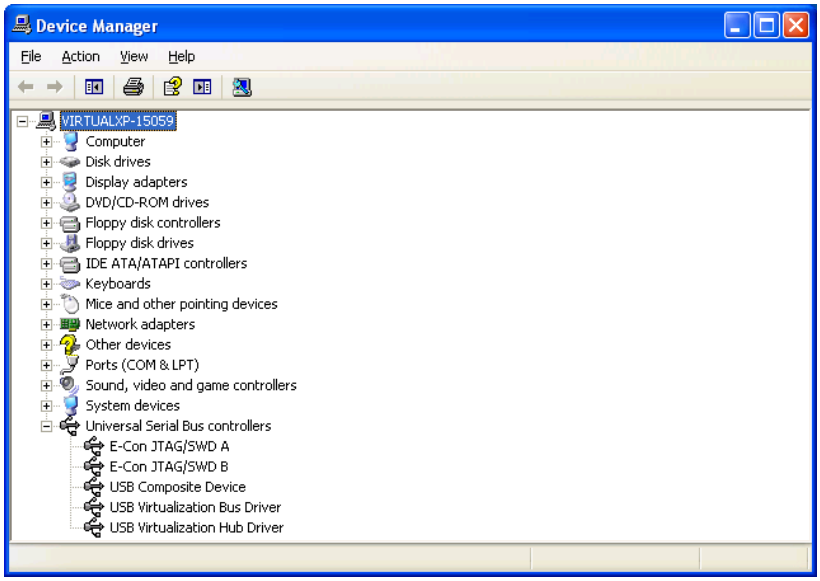
10. The E-Con driver files are copied and E-Con is connected with a computer, then the driver will be installed automatically in case of Windows 7. However, lower than Windows XP loads 'new hardware search wizard' window. Then, just check "install the software automatically" and click 'Next'.



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.

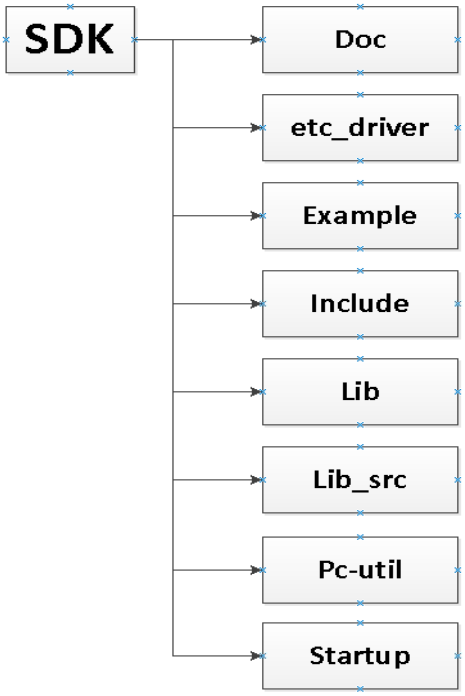

Device Manager
File Action View Help
VIRTUALXP-15059
Computer
Disk drives
Display adapters
DVD/CD-ROM drives
Floppy disk controllers
Floppy disk drives
IDE ATA/ATAPI controllers
Keyboards
Mice and other pointing devices
Network adapters
Other devices
Ports (COM & LPT)
Sound, video and game controllers
System devices
Universal Serial Bus controllers
E-Con JTAG/SWD A
E-Con JTAG/SWD B
USB Composite Device
USB Virtualization Bus Driver
USB Virtualization Hub Driver

2. Using adStar-L SDK

ADChips provides adStar-L SDK for better development environment. adStar-L SDK can be downloaded from (www.adc.co.kr) Product →SoC→ ADSTAR-L → Tools & Software.

2.1 adStar-L SDK organization and basic definitions.

This shows how adStar-L SDK consists of.



Doc → A folder where adStar-L related documents (including this) are located.

Example →A foler where adStar-L STK Board example programs are located.
(Example -> flash_data contains images and sound files for the SDK examples.)

Include →adStar-L SDK's header files.

lib →adStar-L SDK's library files.

lib_src→adStar-L SDK's library source files.

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

Startup →all startup & link scrip code for the adStar-L development. Moreover, STK board basic configuration code exists.

The adStar-L SDK defines variable types as below for the development convenience.
Refer to below table for referring the SDK Source.

Signed char	S8, s8
Signed short	S16, s16
Signed int	S32, s32
Unsigned char	U8, u8, __u8, BYTE, uchar
Unsigned short	U16, u16, __u16, WORD, ushort
Unsigned int	U32, u32, __u32
Unsigned long	DWROD, ulong
Unsigned long long	U64, u64, __u64
Volatile unsigned char	vU8
Volatile unsigned short	vU16
Volatile unsigned int	vU32
Volatile unsigned long long	vU64
1	TRUE, true
0	FALSE, false

Registers are defined with 'R_' prefix, so 'R_' prefixed variables could be considered as registers in SDK Source.

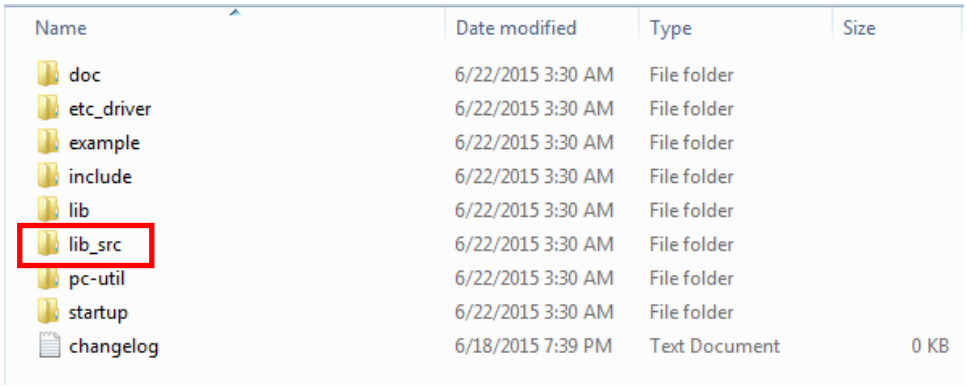
As an example, R_TM0CON is a channel 0 timer control register.

2.2 adStar-L SDK library build

After downloading the adStar-L SDK, the adStar-L SDK Library should be built.

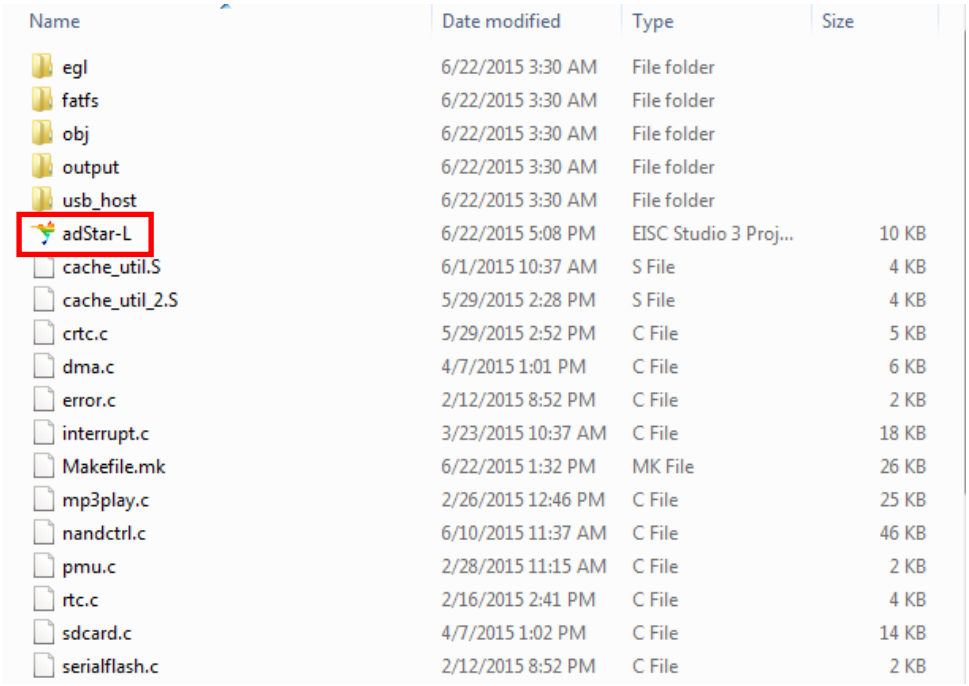
In order to build the library, open **'adStar-L.epx'** file in the **'lib_src'** folder and select 'build'-'build project'(or press F7 for 'project build').

If the process is finished then the **'libadStar-L.a'** library file will be generated in the **'lib'** folder. (It should be done first because all the adStar-L SDK examples need the libraries.)



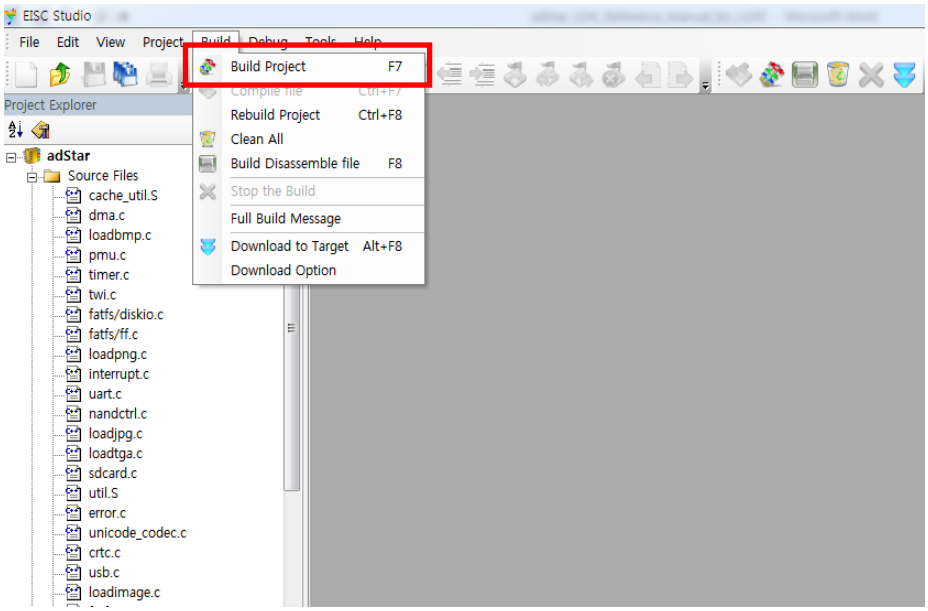
Name	Date modified	Type	Size
doc	6/22/2015 3:30 AM	File folder	
etc_driver	6/22/2015 3:30 AM	File folder	
example	6/22/2015 3:30 AM	File folder	
include	6/22/2015 3:30 AM	File folder	
lib	6/22/2015 3:30 AM	File folder	
lib_src	6/22/2015 3:30 AM	File folder	
pc-util	6/22/2015 3:30 AM	File folder	
startup	6/22/2015 3:30 AM	File folder	
changelog	6/18/2015 7:39 PM	Text Document	0 KB

[lib_src folder]

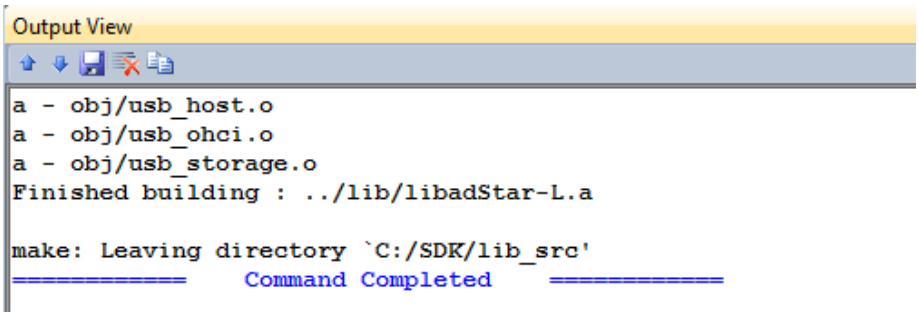


Name	Date modified	Type	Size
egl	6/22/2015 3:30 AM	File folder	
fatfs	6/22/2015 3:30 AM	File folder	
obj	6/22/2015 3:30 AM	File folder	
output	6/22/2015 3:30 AM	File folder	
usb_host	6/22/2015 3:30 AM	File folder	
adStar-L	6/22/2015 5:08 PM	EISC Studio 3 Proj...	10 KB
cache_util.S	6/1/2015 10:37 AM	S File	4 KB
cache_util_2.S	5/29/2015 2:28 PM	S File	4 KB
crtc.c	5/29/2015 2:52 PM	C File	5 KB
dma.c	4/7/2015 1:01 PM	C File	6 KB
error.c	2/12/2015 8:52 PM	C File	2 KB
interrupt.c	3/23/2015 10:37 AM	C File	18 KB
Makefile.mk	6/22/2015 1:32 PM	MK File	26 KB
mp3play.c	2/26/2015 12:46 PM	C File	25 KB
nandctrl.c	6/10/2015 11:37 AM	C File	46 KB
pmu.c	2/28/2015 11:15 AM	C File	2 KB
rtc.c	2/16/2015 2:41 PM	C File	4 KB
sdcard.c	4/7/2015 1:02 PM	C File	14 KB
serialflash.c	2/12/2015 8:52 PM	C File	2 KB

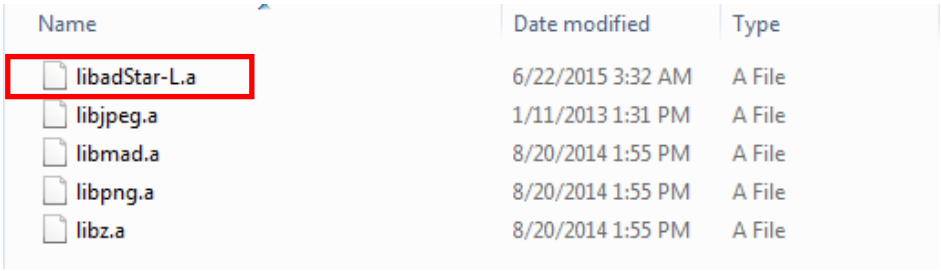
[adStar-L.epx file]



[Build Project]



[Build Project is finished well]



[libadStar-L.a file is generated]

If the adStar-L library is modified, then all the library build processes must be re-executed. In addition, the program that uses the adStar-L library must be built to adopt the modified library..

2.3 Demo Program Execution

User can easily check the adStar-L operations on the STK board with various examples in the SDK example folder.

This chapter explains how to use the **Demo** Project Source (of STK examples) such as build, download and execution.

At first, images and sound files Flash should be prepared to execute the demo example on Nand Flash. To copy the file to Nand Flash, execute the **usb_mass_storage** project in the example folder. User need to know how to build and download projects before executing projects.

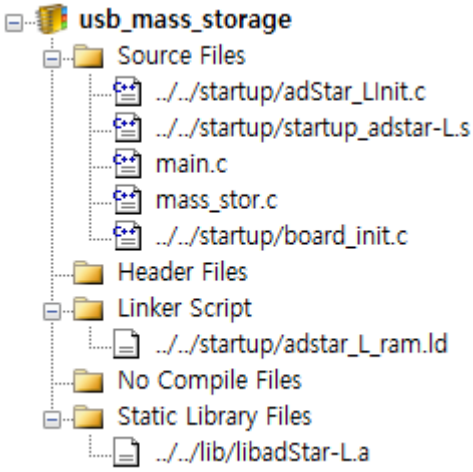
2.3.1 Project build & download

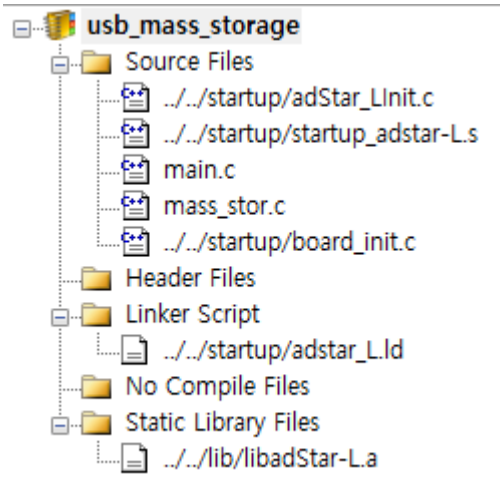
Open a project by double clicking **usb_mass_storage.epx** file in the **example - usb_mass_storage** folder.

After open the project, change Linker Script file in project Explorer window. Linker Script file is two. one of linker script is adStar-L.ld file and other one is adStar-L_ram.ld file.

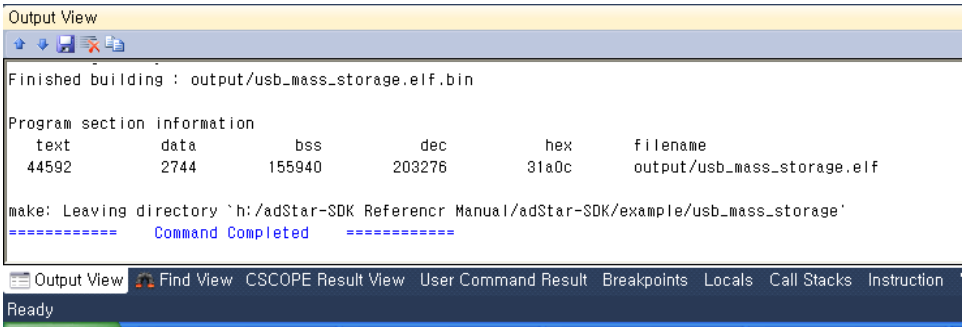
adStar-L_ram.ld file (use by default in SDK example) use when operate application in SDRam. So you need bootloader when using adStar-L_ram.ld file.

At this chapter, explain operate program without bootloader. So use adStar-L.ld file instead of adStar-L_ram.ld file. Click mouse right button at Linker Script, click Set Linker Script File and select adStar-L.ld file in startup folder.



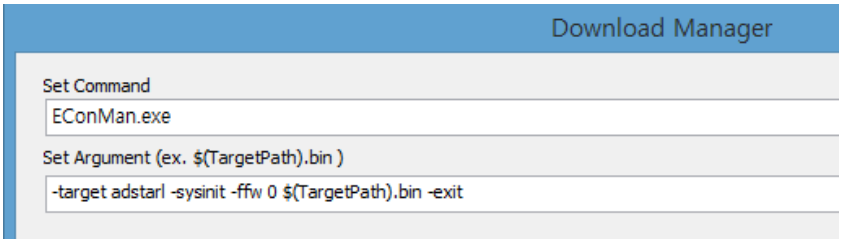


After change Linkger Script, choose **build → Build Project** or press F7 to build the project. Output View shows a build result as below when the build is finished.



When the build is finished well, user can see **usb_mass_storage.elf.bin** file in the output folder. Next, study how to download the built file, **usb_mass_storage.elf.bin** file, to the STK board. Download can be done with E-CON while opening a project to download on EISC Studio3. You follow the below steps.

1. Prepare the adStar-L board and the E-CON device for downloading.
   (EISC Studio3 and E-CON Driver should be installed. Refer to chapter 1, Software Development Environment for the installation.)
2. Turn on adStar-L board after connecting the board and the E-CON device.
3. Click 'Build' – 'Download Option'. Then the Download Manager window will come up as below. Set 'Set Command' as the E-CON download program, EConMan.exe that exists in EISC Studio3 installed folder – econ folder. (Can use the default value 'EconMan.exe'.) Set 'Set Argument' as "–target adStarl –sysinit –ffw 0x0 $(TargetPath).bin -exit" and click 'OK').



Argument indicates,

-target is target name to download, adStarl.

-sysinit is initialization command for download.

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

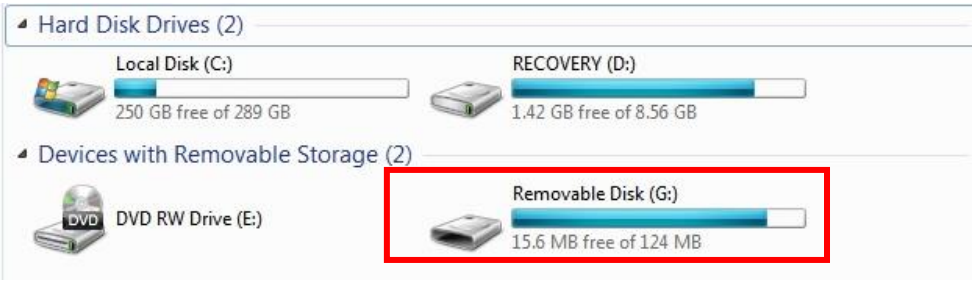
-exit is command to exit automatically after the download is finished.

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

4. Click, 'Build' – 'Download to Target'. Start the download.

2.3.2 Copying Files to Nand Flash (SD Card)

After the downloading, turn the STK board power off and connect an usb cable to a computer and the STK board's device port. Turn the board power on, then 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 adStar-L programs.



In order to operate the Demo Project, should copy all image and sound files from 'example'-'flash data'.

To copy the files to a SD card instead of Nand Flash, use a definition in **mass_stor.c** of the **usb_mass_storage** project, '#define SDCARD_STORAGE'. In order to use the definition, remove comment flags on it, re-build and download the project to the board.

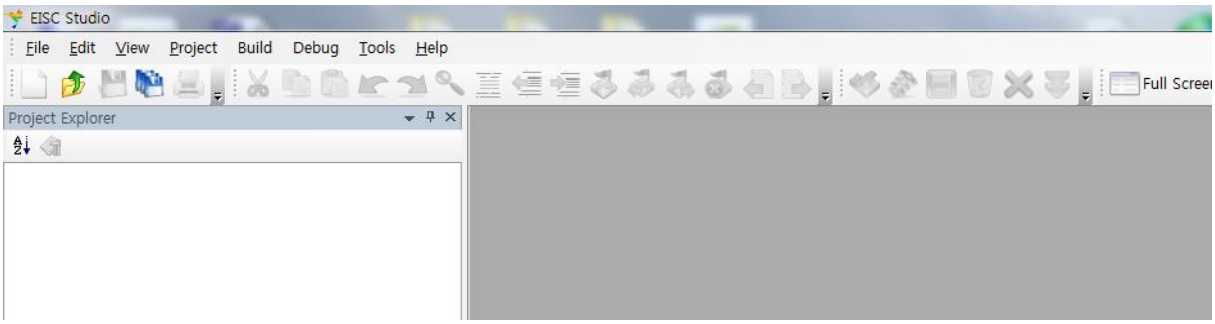
2.3.3 Demo Program build & download

After copying the image and sound files that will be used for the Demo project to Nand Flash, build the Demo project in the example folder. The built project could be downloaded via E-Con and can check the project operations.

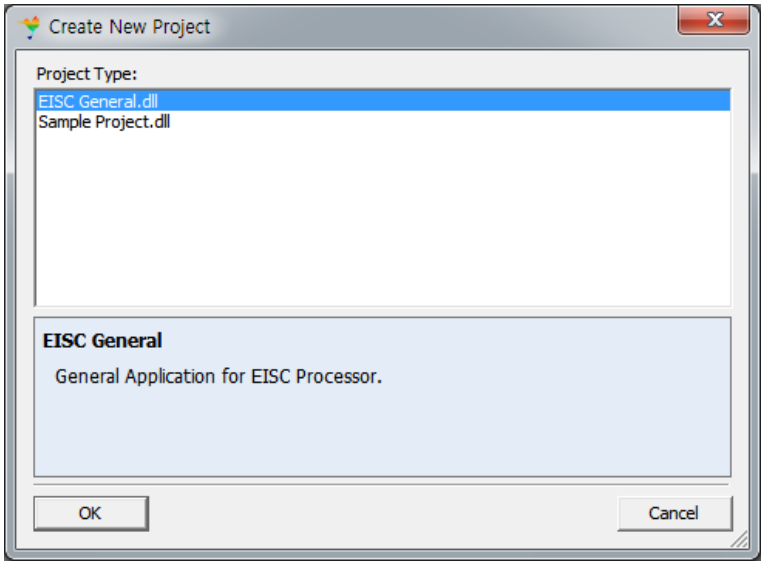
2.4 New adStar-L Project Creation

This chapter explains how to create a new project that displays "Hello adchips!" statement to UART.

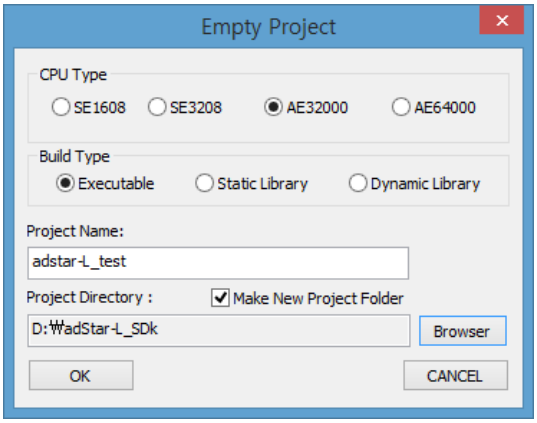
1. Execute EISC Studio3.



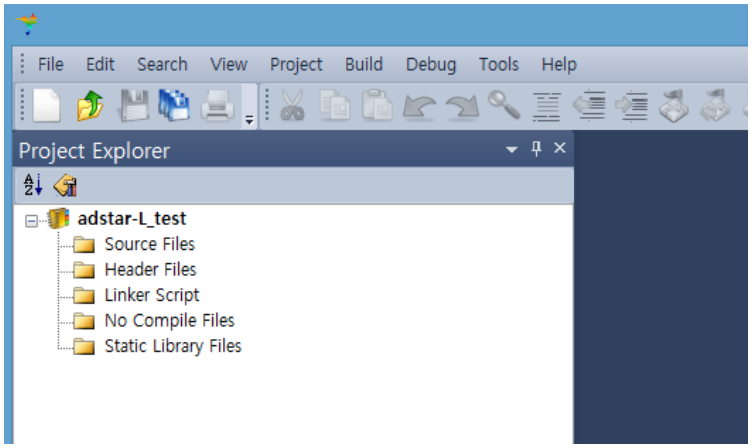
2. Click Menu, File → New → Project, then below window shows up. Then choose EISC General.dll as below and click 'OK'.



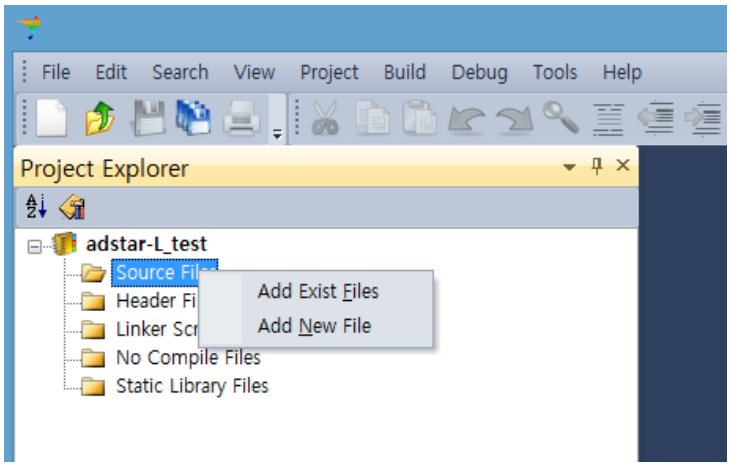
3. Then, a project configuration window shows up like below. User need to determine a CPU Type, a Build Type, a Project Name and a Project Directory. In order to use adStar-L, the CPU Type: AE32000, Build Type: Executable (because we will make an executable project). After that, we can confirm the Project Name and the Project directory. A check box (Make New Project Folder) besides the Project Directory determines that create a folder which has same name as the project or not. Checking on the box create a folder that has same name as the project in the Project Directory and create a Project file in it. Unless, create a Project file in the directory assigned as the Project Directory. After all the configurations are set, finish the new project creation process by clicking 'OK'.



4. Can check the project name and a project tree from a Project Explorer window after the new project creation. Need to add essential files for the adStar-L project.

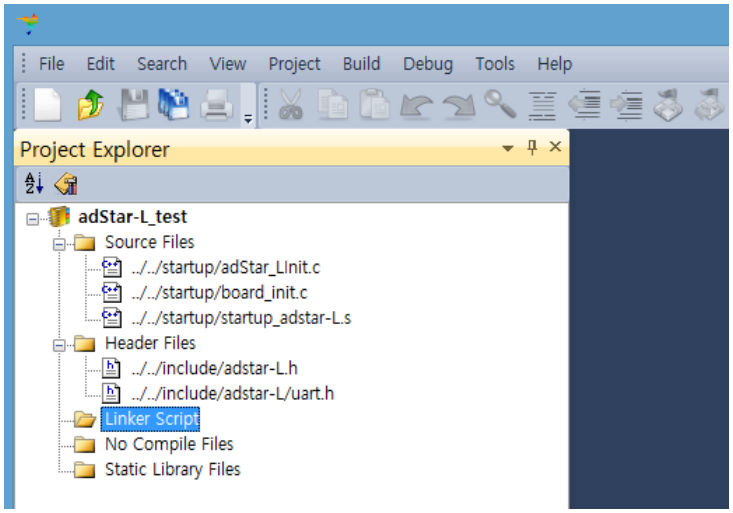


5. At first, click a right mouse button on 'Source Files' and choose 'Add Exist Files' to add the files. The additional files are **startup_adStar-Lt.s**, **adStar-Linit.c** and **stk_board.c** in SDK-'**startup**' folder. (add **stk_board.c** when using the stk board for adStar-L.)

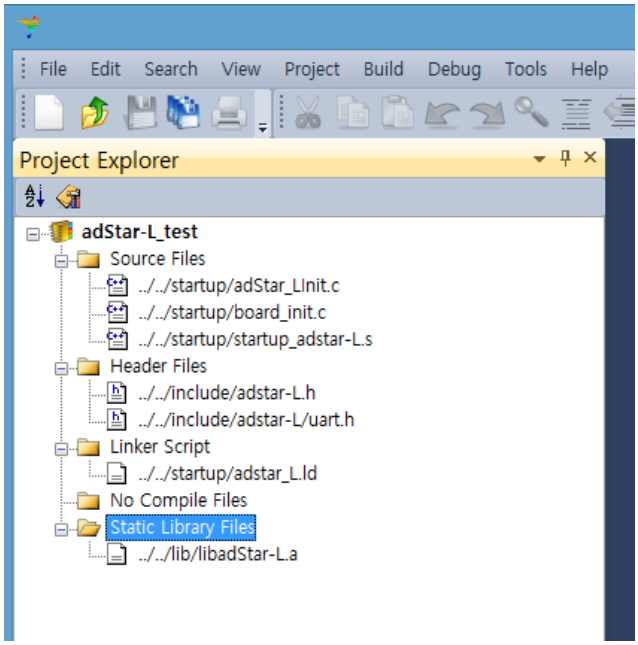


6. After that, same as the 'Source Files', click a right mouse buttion on 'Header Files' and add **adStar-L.h** file in the **include** folder. It is enough to add **adStar-L.h** file, but add other

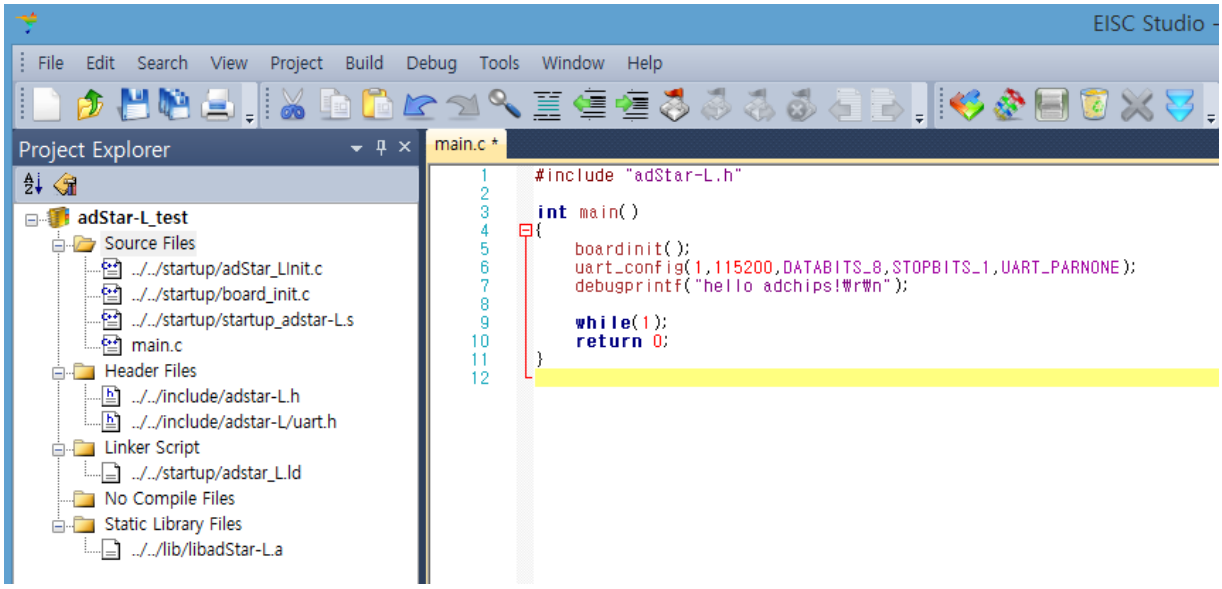
related header files in **include/adStar-L** folder for more easier function usage. We add **UART.h** file because we will make a UART using program.



7. Add **adStar-L.ld** file in the **startup** folder to the 'Linker Script' and add **libadStar-L.a** file in the **lib** folder to 'Static Library Files'. Also, add **libmad.a, libjpeg.a, libz.a** and **libpng.a** files for MP3 playing and JPG, PNG image file displaying (these are not necessary for currently making UART exmaple).
   (I'ts OK to add all library files in the lib folder because only needed library files are used.)

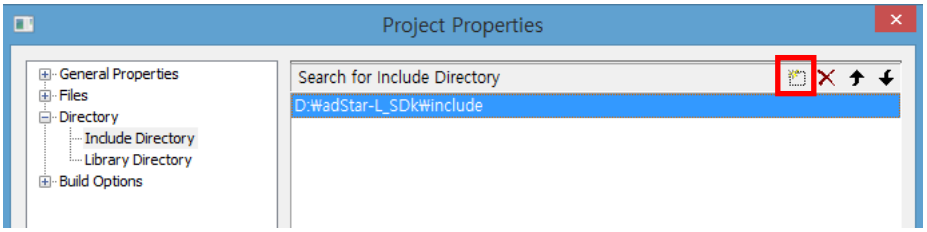


8. If the file addition processes are done as above, create main.c file for a main program by clicking a right mouse button on 'Source Files' and choose 'add new file'. We will explain how to make a code that displays "Hello adchips!" statement via UART.

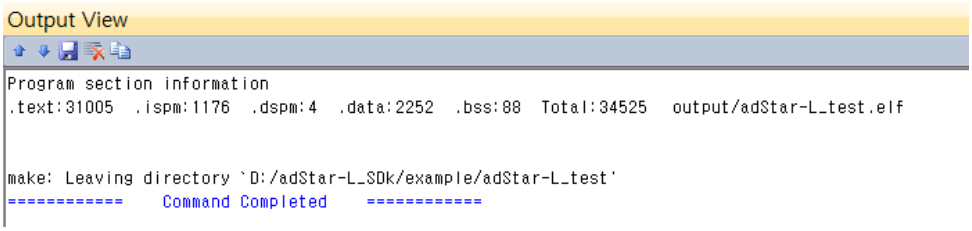


9. In main.c, first thing to do is that set a pin for an appropriate board by calling boardinit() function. (Since, boardinit() function is different with the using board, so set pin according to the board). Next, call uart_config() to initialize UART with denoted parameters.
   Finally, display "hello adchips!" by using debugprintf() function.

10. With all above processes, configure the project before the project build. Click right mouse button on the project name on the Project Explorer window and click 'Properties'. Add the include folder of SDK to 'Directory'-'Include Directory' as below. (Can add a folder by clicking the doted box on top-left)



11. After including the include folder, click 'Build'-'Build Project' or press F7 to start the project build. If the building is O.K. then the Output View window shows below results and '**Project name.elf.bin**' file is created in the output folder.



12. The created bin file could be downloaded to the STK board via E-Con. User can check that "hello adchips!" statement is displayed through UART channel 0.

3. Bootloader

adStar-L can Flash booting and Nand Booting.
So, bootloader is supported for Flash booting and Nand Booting.
At this chapter, explain the adStar-L bootloader.

3.1Bootloader. (example / bootloader)

Bootloader is booting program for Serial Flash.
Bootloader provide the following functions.

a. Remote Communication Mode.
b. Mass Storage Mode
c. Execute Mode
d. User Define Mode

Remote Communication Mode is that communication with adStar-L using RemoteManCLI program. Use this mode when download and run application program at DRAM using USB. Mass Storage Mode is that NAND flash is used as storage device like usb memory at computer. Execute Mode is that copy application program from 0x14000 in flash to SDRam and run that. User Define Mode is copying boot.bin file (application binary file) from NAND Flash to SDRam and run that.

3.1.1 Use Bootloader

Bootloader is in SDK / example folder.

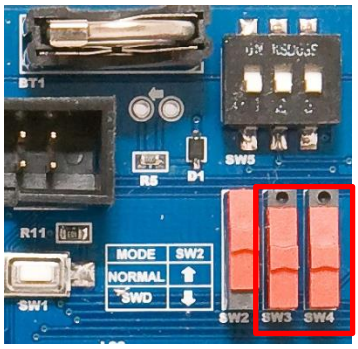
Bootloader project build and download. Then you can see that operate bootloader.

Bootloader has some mode. Following code is setting mode part.

```
*R_GPIDIR(2)=0x1<<7;//mode switch
*R_GPIDIR(3)=0x1<<0; //mode switch
int mode = ((*R_GPILEV(2)>>6)&(1<<1)) | ((*R_GPILEV(3))&1);

switch(mode)
{
case 0:
    bin_execute();
    break;
case 1:
    RSP_Run();
    break;
case 2:
    usb_clock_init();
    mass_storage_main();
    break;
case 3:
    bin_execute_fat();
    break;
}
```

In code, select mode by P2.6 (SW3) and P3.1 (SW4) input.

(SW2 is SWD mode switch.)



Like above picture, boot mode SW setting value is like following table.

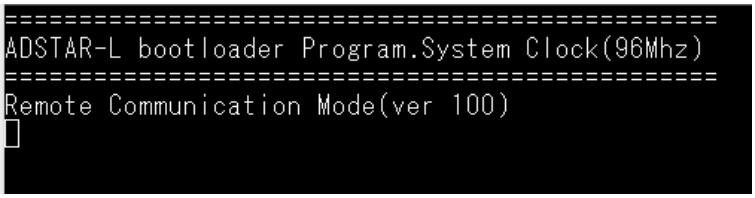
MODE	SW3	SW4
Execute Mode	down	down
USB Communication Mode	down	up
Mass Storage Mode	up	down
Execute Fat Mode	up	up

3.1.2 Bootloader Mode

A. Remote Communication Mode

Remote Communication Mode is that communication with adStar-L using RemoteManCLI program. Use this mode when download and run application program at DRAM using USB.

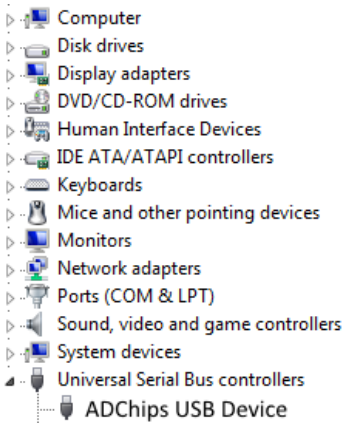
Booting Remote Communication Mode then display debug message like following.



[Remote Communication Mode]

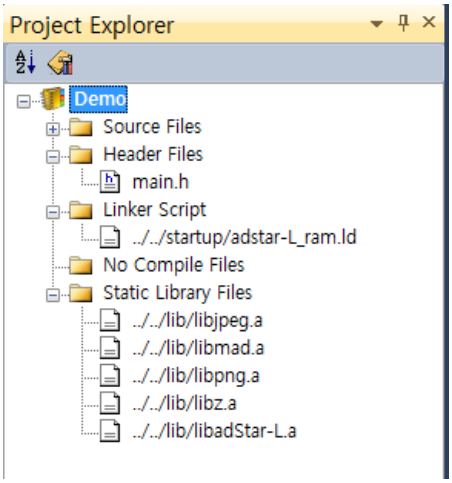
Because using adStar-L USB device at Remote Communication Mode, need to install adStar-L USB Driver. USB driver is in pc-util/usb_driver folder. Install driver way is to use ADChips USB Driver Install.exe in pc-util/usb_driver.

1. Run the ADChips USB Driver Install.exe in usb_driver folder.
2. After complete installation, connect USB cable and booting as Remote Communication Mode.
3. Then 'Installing device driver software' will be shown in tray icon. If you use windows 7, installed it automatically. but if you windws xp, displayed 'Found New Hardware Wizard' window. Then click 'Next' after selecting 'install the software automatically'.
4. After USB driver installation, you can see usb driver in device manager.



[Device Manager]

Now, you can run application program by download to SDRam in Remote Communication Mode. Before downloading, open the application project and check link script.



[Linker Script]

If Linker Script is adStar-L_ram.ld, then run build.

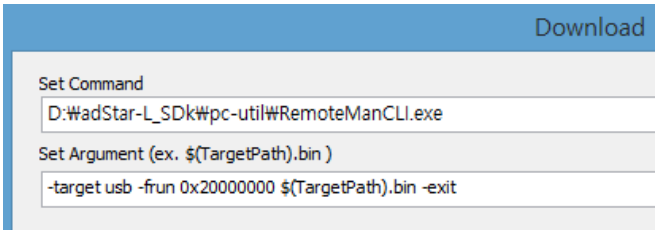
But if Linker Script is adStar-L.ld, after changing adStar-L_ram.ld, and run build.

In short,

Application that operated in flash is use adStar-L.ld.

Application that operated in SDRam is use adStar-L_ram.ld.

After checking the Linker Script, download to SDRam. Download way is like following.



1. Boot Remote Communication Mode.
2. Run build → download option.
3. Use RemoteMenCli as download program in 'pc-util'.
   option is like following.

   -target usb
   → target name.
   -frun 0x20000000 $(TargetPath).bin
   → command to binary file download and run.
   Download to SDram start address 0x20000000
   -exit
   → exit program.

After set download option, run build → download to target.

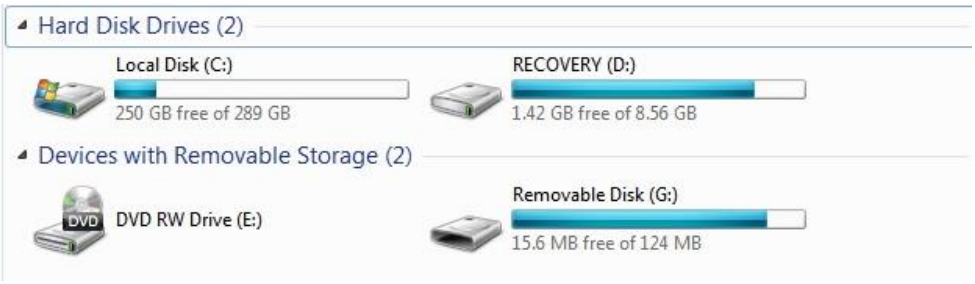
Then, download binary and application is operated.

B. Mass Storage Mode

Mass Storage Mode is that NAND flash is used as storage device like usb memory at computer. Boot Mass Storage Mode, then display message following. And new storage device appeared.

```
================================================
ADSTAR-L bootloader Program.System Clock(96Mhz)
================================================
NAND Manufacturer : Samsung, 128MiB 3,3V 8-bit
(../../lib_src/nandctrl.c:1282) make bad block inforamation done(bad-block(0))
USB Mass-Storage Mode Running(Interrupt Mode)
```

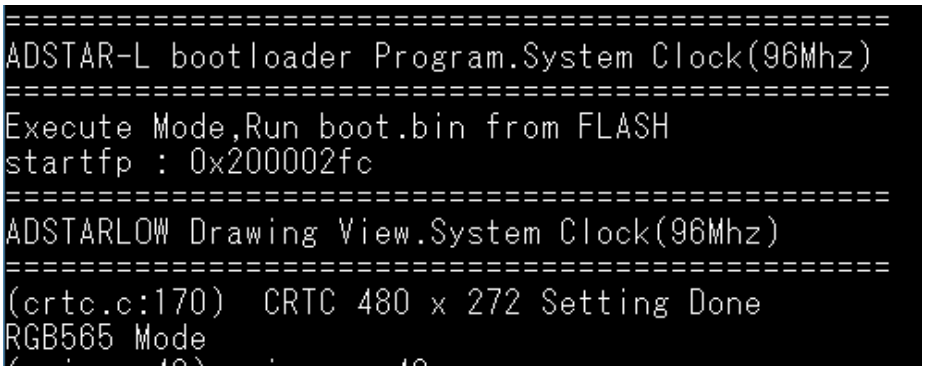
[Mass Storage Mode]



This device is Nand Flash of the board, so copying files to the device indicates that the files could be used by the adStar-L programs.

C. Execute Mode Mode

Execute mode is that mode as soon running application.
Boot execute mode then display message following.



[Execute Mode]

Execute mode is copying from flash data to SDRam. And run application.
You need to download to specified address. Specified address is 0x14000(sector number 20). And you need to use adStar-L_ram.ld as Linker Script.

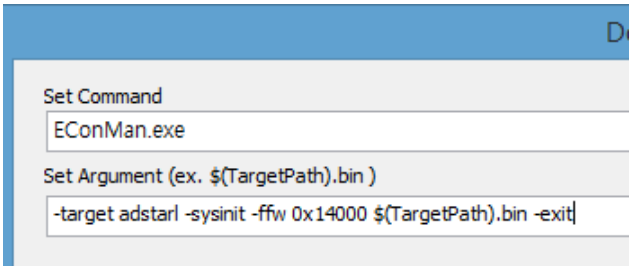
```
46  #define FLASH_APP_OFFSET  (1024*4*20)
47  static void bin_execute()
48  {
49      void (*entryfp)();
50      debugstring("Execute Mode,Run boot.bin from FLASH \r\n");
51      memcpy((void*)0x20000000,(void*)FLASH_APP_OFFSET,(512*1024)-FLASH_APP_OFFSET);
52      dcache_invalidate_way();
53      entryfp = (void(*)())*(U32*)0x20000000;
54      debugprintf("startfp : %#x\r\n",entryfp);
55      entryfp();
56  }
```

[bootloader main.c]

There is address defined at main.c line 46. If using bootloader bigger size than original, need to change this address.

There is two way to download 0x14000 in flash.

1) Use EconMan.

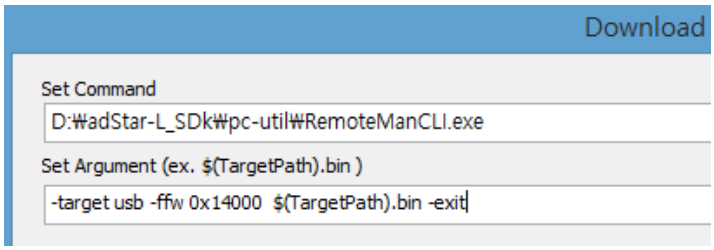


[download option]

Just change address from 0x0 to 0x14000.

2) Use RemoteMenCli.

After booting Remote Communication Mode, download by use following option.



[download option]

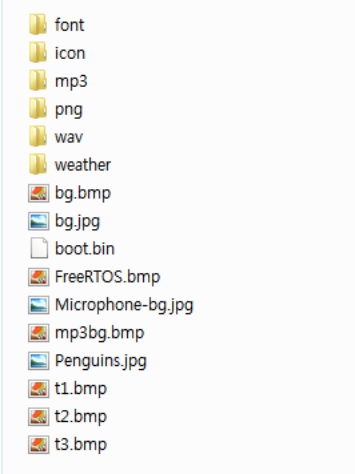
-flash_filewrite

→ flash download command.

After download, boot Execute Mode.

D. User Define Mode (execute_fat)

User Define Mode is copying boot.bin file (application binary file) from NAND Flash to SDRam and run that. So, change the file name of created file to boot.bin. And copy to NAND Flash by using Mass Storage Mode. After copying, run User Define Mode. Then you can see application is operated.



[boot.bin file of NAND Flash]

If you want using boot.bin in SD Card. you use '#define FAT_APP_TYPE DRIVE_SDCARD' instead of '#define FAT_APP_TYPE DRIVE_NAND' in main.c line 4.

3.2 Nand Boot Code

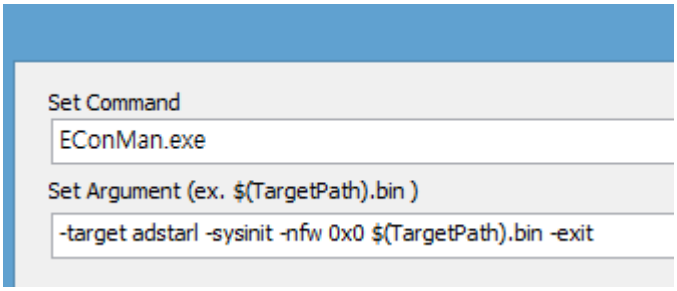
NAND Boot Code that used for NAND booting is used at adStar-L which does not embedded flash. If use NAND booting, need to set as NAND Flash Boot mode. Refer to datasheet for how set. NAND Boot mode is that copy 2Kbyte NAND Flash data in block 0 to SRam. Because NAND Boot mode is copied 2Kbyte NAND Flash data in block 0 to SRam, NAND Boot code size is limited as 2Kbyte. So, it is recommended to use as default status. NAND Boot Code operation is that copy 7 blocks data from block 1 to SDRam. And run application.

```
#define BOOTLOADER_STARTADDR 0x20000000
#define COPY_BLOCK 7//128K*7Mbyte
int main()
{
    void (*startfp)();
    boardinit();
    myuart_init();
    *R_NFMCFG  = 0x2222;
    nand_reset();
    mydebugstring("Nand Booting.₩r₩n");
    //U16 id = nand_id_read();
    copydata_from_nand((U8*)BOOTLOADER_STARTADDR,1,COPY_BLOCK);
    startfp = (void (*)())(*(volatile unsigned int*)BOOTLOADER_STARTADDR);
    startfp();
    while(1);
    return 0;
}
```

[main.c of nand boot code]

3.2.1 Use Nand Boot Code

Open NAND Boot code, build and download to NAND Flash block 0. Download option is like following.

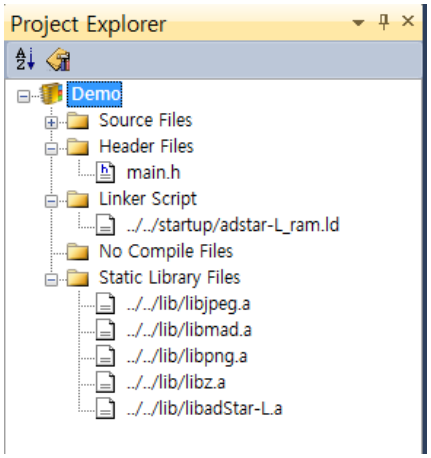


[Download Option]

-nfw

➔ NAND Flash download command.

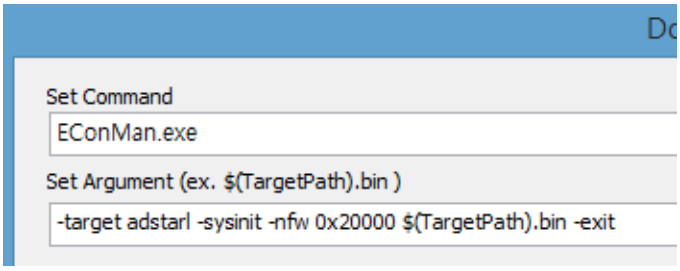
Next, explain the method about downloading application code which operated by copying to SDRam at NAND Boot code. Code that operated at SDRam is used adStar-L_ram.ld as Linker Script.



[Linker Script]

After checking Linker Script, build and download.
As before, you use '–nfw' command. Download address is different from the previous download. Download to NAND Flash block number 1.

You need to caution for different block number 1 address of each NAND FLASH.
(small page -> block 1 address : 0x4000, large page -> block 1 address : 0x20000)

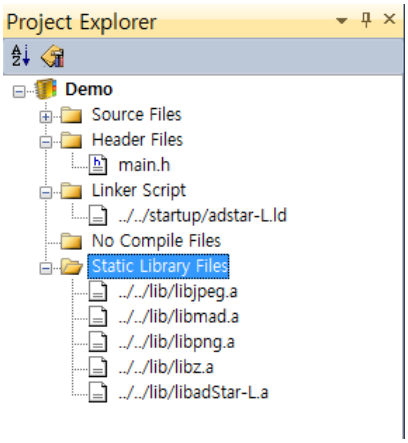


[Download Option]

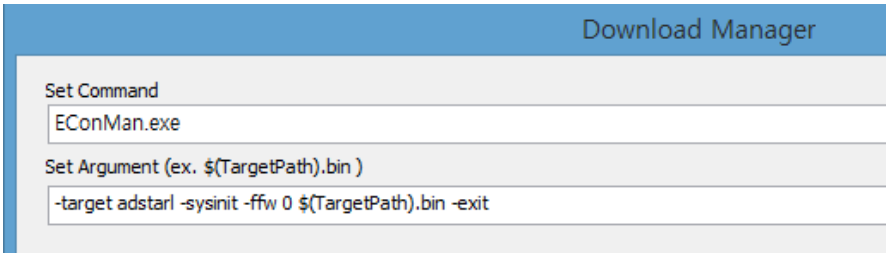
After download, turn on board. Then, display NAND Booting message. And application that downloaded to block 1 will operated.

3.3 Use without BootLoader

In this chapter, explain the method about operate adStar-L without bootloader. For operating adStar-L without bootloader, you need to change Linker Script file and download address. Use adStar-L.ld as Linker Script file. And use download address 0. Open project and change Linker Script. After build project, download to 0 in flash. When you reboot, you can see that application is operated without bootloader.



[Linker Script]



[Download Option]

4. lib_config.h

lib_config.h in SDK library source project file (adStar-L.epx) is config file. At this chapter, explain the lib_config.h file.

lib_config.h file begins as explain for nested interrupt setting.

```
#define SUPPORT_NESTED_INTNUM_EIRQ0 0
#define SUPPORT_NESTED_INTNUM_SYSTIMER 0
#define SUPPORT_NESTED_INTNUM_TIMER0 0
#define SUPPORT_NESTED_INTNUM_SOUNDMIXER 0

#define SUPPORT_NESTED_INTNUM_EIRQ1 0
#define SUPPORT_NESTED_INTNUM_FRAMESYNC 0
#define SUPPORT_NESTED_INTNUM_DMA0 1        //DMA is used by sound , 1 is recommended
#define SUPPORT_NESTED_INTNUM_GPIOA 0

#define SUPPORT_NESTED_INTNUM_UART0 0
#define SUPPORT_NESTED_INTNUM_DMA1 1
#define SUPPORT_NESTED_INTNUM_TIMER1 0
#define SUPPORT_NESTED_INTNUM_PMC 0

#define SUPPORT_NESTED_INTNUM_DMA2 1
#define SUPPORT_NESTED_INTNUM_ADC 0
#define SUPPORT_NESTED_INTNUM_USB 0
#define SUPPORT_NESTED_INTNUM_GPIOB 0

#define SUPPORT_NESTED_INTNUM_UART1 0
#define SUPPORT_NESTED_INTNUM_DMA3 1
#define SUPPORT_NESTED_INTNUM_SPI0 0
#define SUPPORT_NESTED_INTNUM_MJPEG0 0

#define SUPPORT_NESTED_INTNUM_GPIOC 0
#define SUPPORT_NESTED_INTNUM_SDHC 0
#define SUPPORT_NESTED_INTNUM_GPIOD 0
#define SUPPORT_NESTED_INTNUM_WATCHDOG 0

#define SUPPORT_NESTED_INTNUM_NAND 0
#define SUPPORT_NESTED_INTNUM_TWI 0
#define SUPPORT_NESTED_INTNUM_RTC 0
#define SUPPORT_NESTED_INTNUM_RTC_ALARM 0

#define SUPPORT_NESTED_INTNUM_SPI1 0
#define SUPPORT_NESTED_INTNUM_CAPOVERFLOW 0
#define SUPPORT_NESTED_INTNUM_MJPEG1 0
#define SUPPORT_NESTED_INTNUM_CLOCK_CTR_R 0
```

When set to 1, means you use nested interrupt on selected interrupt.
DMA interrupt has been set 1 by default. Because DMA interrupt use on mp3 decoding.
You can set up to max 14 nested interrupt.

```
/*****************************************************************************
    INTERNAL TIME OUT
*****************************************************************************/


#define TWI_RESP_TIME_OUT_COUNT (7200*100)  // About 100ms @ AHB 101MHz

/*****************************************************************************
    External NAND/SD-Card/USB Memory Supports
*****************************************************************************/

#define CONFIG_NAND 1
#define CONFIG_SDCARD 1
#define CONFIG_USBHOST 0
```

< Set the TWI timeout count and storage device to using >

```
/*****************************************************************************
    UART Config
*****************************************************************************/


#define UART_BUF_SIZE 512
#define CONFIG_UART_RX_INTERRUPT
//#define CONFIG_UART_TX_INTERRUPT

/*

*/
#d
```

< Set the uart interrupt and uart buf size >

```
/*****************************************************************************
    SYSTEM CLOCK
*****************************************************************************/
`




#define OSC_CLOCK 2000000

/*****************************************************************************
*   SOUND Mixer
*****************************************************************************/
/




*
#define WAVE_BUF_MAX (512*1024) //if you use MP3, > 5Kbyte
#define WAVE_RELOAD_BUF_MAX (1024*128)
#define DEFAULT_VOLUME 255//max 255


#define SND_OUTPUT_HZ   48000
```

< Set the crystal clock, sound buffer size and sound channel >

5. UART

5.1 uart_config

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

Function to initialize UART.

Parameter

ch UART initial channel value.

baud UART baud rate value.
SDK uses 115200 as default.

databits UART transmission data bit setting. SDK defines data bit as below.

```
typedefenum{
    DATABITS_5 = 5,
    DATABITS_6 = 6,
    DATABITS_7 = 7,
    DATABITS_8 = 8
}UART_DATABITS;
```

stopbit UART stop bit setting. SDK defines stop bit as below.

```
Typedefenum{
    STOPBITS_1 = 1,
    STOPBITS_2 = 2
}UART_STOPBITS;
```

parity UART parity bit setting. SDK defines parity bit as below.

```
Typedefenum{
    UART_PARNONE = 0,
    UART_PARODD,
    UART_PAREVEN
}UART_PARITY;
```

Return Value

TRUE(1) or FALSE(0)

5.2 uart_putch

BOOL uart_putch(**int** n, **char** ch)

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

Parameter

n	Channel to display the 'ch' value. adStar-L has 5 channels.
ch	Value to be displayed through UART.

Return Value

TRUE(1) or FALSE(0)

5.3 uart_putdata

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

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

Parameter

n	Channel to display the 'ch' value. adStar-L has 5 channels.
buf	Buffer that stores characters to be displayed.
len	Length of characters to be displayed.

Return Value

TRUE(1) or FALSE(0)

5.4 uart_putstring

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

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

Parameter

n	Channel to display the 'ch' value. adStar-L has 5 channels.
buf	Buffer that stores characters to be displayed.

Return Value

TRUE(1) or FALSE(0)

5.5 uart_getch

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

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

Parameter

n	UART channel to be input the value. adStar-L has 5 channels.
ch	Variable to be input a value through UART.

Return Value

TRUE(1) or FALSE(0)

5.6 uart_getdata

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

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

Parameter

n	UART channel to be input the value. adStar-L has 5 channels.
Buf	Buffer to be input the value through UART.
Bufmax	Number of values (characters) to read. (Byte unit)

Return Value

The total number of elements successfully read is returned as byte unit. If the value same as the 'Bufmax' value, this function is successfully finished, unless an error occurred.

5.7 uart_rx_flush

void uart_rx_flush(**int** ch)

Initialize rxfifo of the UART n channel.

Parameter

ch	UART channel to be initialized.

Return value

None.

5.8 uart_tx_flush

void uart_tx_flush(**int** ch)

Initialize txfifo of the UART n channel.

Parameter

ch UART channel to be initialized.

Return value

None.

5.9 set_debug_channel

Void set_debug_channel(**int** ch)

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

Parameter

ch Debugging purpose UART channel.

Return value

None.

5.10 get_debug_channel

int get_debug_channel( )

Return the debugging purpose UART channel.

Parameter

None

Return Value

Current debugging purpose UART channel value.

5.11 debugprintf

void debugprintf(**const char***const format, . . .)

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.

Note. Though set to use Tx interrupt, this function operates in a poling.

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)

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.

Note. Though set to use Tx interrupt, this function operates in a poling.

Usage

debugstring("=== adStar-L Start ===₩r₩n");

→string in "" is displayed through UART.

5.13 PRINTLINE

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)

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 "adStar-L.h"

int main()
{
    boardinit();
    uart_config(1, 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("ADSTAR-L UART Example. System clock(%dMhz)₩r₩n",get_ahb_clock()/1000000);
    debugstring("=============================================₩r₩n");
    U8 ch;
    While(1)
    {
        if(uart_getch(1, &ch))      // Reads 1Byte data from UART 0 and stores 'ch'.
        {
            uart_putch(1, ch);    // Displays UART 0 input data to UART 0.
        }
    }
}
```

6. INTERRUPT

6.1 init_interrupt

void init_interrupt(**void**);

The interrupt initialization function. It should be called to use the interrupt related function. **startup_adStar-L.s** of the adStar-L SDK calls and initializes the interrupt.

6.2 set_interrupt

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

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

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

Return value

TRUE or FALSE

6.3 enable_interrupt

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

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

Parameter

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

Return value

TRUE or FALSE

INTERRUPT_TYPE1

INTERRUPT_TYPE		INTERRUPT_TYPE	
INTNUM_EIRQ0	External interrupt 0	INTNUM_PMC	PMC interrupt
INTNUM_EIRQ1	External interrupt 1	INTNUM_ADC	ADC interrupt
INTNUM_SYSTIMER	System timer interrupt	INTNUM_USB	USB interrupt
INTNUM_TIMER0	Timer interrupt 0	INTNUM_SPIO	SPI 0 interrupt
INTNUM_TIMER1	Timer interrupt 1	INTNUM_SPI1	SPI 1 interrupt
INTNUM_SOUNDMIXER	Sound mixer interrupt	INTNUM_JPEG0	JPEG0 interrupt
INTNUM_FRAMESYNC	Frame sync interrupt	INTNUM_JPEG1	JPEG1 interrupt
INTNUM_DMA0	DMA interrupt 0	INTNUM_SDHC	SDHC interrupt
INTNUM_DMA1	DMA interrupt 1	INTNUM_NAND	NAND Flash interrupt
INTNUM_DMA2	DMA interrupt 2	INTNUM_WATCHDOG	Watch dog interrupt
INTNUM_DMA3	DMA interrupt 3	INTNUM_TWI	TWI interrupt
INTNUM_GPIOA	GPIO A interrupt	INTNUM_RTC	RTC interrupt
INTNUM_GPIOB	GPIO B interrupt	INTNUM_RTC_ALARM	RTC Alarm interrupt
INTNUM_GPIOC	GPIO C interrupt	INTNUM_CAPOVERFLOW	Capture overflow interrupt
INTNUM_GPIOD	GPIO D interrupt		
INTNUM_UART0	UART 0 interrupt	INTNUM_CLOCK_CTRL_R	Clock control interrupt
INTNUM_UART1	UART 1 interrupt		

6.4 Interrupt Example

```
#include “adStar-L.h“

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

int main()
{
	boardinit();
	uart_config(1, 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

7.1 set_timer

BOOL set_timer(**int** nCh, **U32** ms)

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

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

Return Value

TRUE(1) or FALSE (0)

7.2 stop_timer

BOOL stop_timer(**int** nCh)

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

Parameter

nCh	Timer channel value to be disabled.

Return Value

TRUE(1) or FALSE (0)

7.3 delayms

BOOL delayms(**U32** ms)

Takes ms delay. (ms unit)

Parameter

ms	Time amount of delay, ms unit.

Return Value

TRUE(1) or FALSE (0)

7.4 TIMER Example

```
#include "adStar-L.h"

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

int main()
{
        boardinit();
        uart_config(1, 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

8.1 setscreen

void setscreen(**SCREENRES** size, **U32** screenmode)

Set LCD resolution and RGB mode. It should be set first for LCD displaying.
Configure the LCD controller according to the set LCD resolution.

Parameter

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,
       } SCREENRES;
```

Screenmode — RGB mode setting value, defined as below.

SCREENMODE_RGB888
SCREENMODE_RGB565

Return value

None

8.2 createframe

SURFACE* createframe(**U32** w, **U32** h, **U32** bpp)

Function to create a frame (memory region) on a screen. Creates w width x h height region that images or shapes could be drawn. More than one frame is needed for the screen output.

Parameter

w	Sets width of the frame.
h	Sets height of the frame.
bpp	Sets bit per pixel value of the frame.

Return value

Returns the SURFACE structure that has the frame information. SURFACE is defined as below.

```
typedefstruct {
        U32 w;
        U32 h;
        U32 bpp;
        void*  pixels;
        U32 pixtype;
        PALETTE* pal;
        U8 ashiftbit;
        U8 rshiftbit;
        U8 gshiftbit;
        U8 bshiftbit;
        void* reserve;
} SURFACE;
```

8.3 setframebuffer

void setframebuffer(**SURFACE*** surf)

Function to choose a frame to be displayed on the screen. If the frame that is created by createframe() is chosen, then only the frame is shown on the screen. (Single frame)
For screen output, the frame must be chosen by calling one of these functions; setframebuffer(), setdoubleframebuffer().

Parameter

surf	createframe() created SURFACE structure.

Return value

None.

8.4 setdoubleframebuffer

void setdoubleframebuffer(**SURFACE*** surf, **SURFACE*** surf2)

Function to set two frames will be displayed on the screen. set two frames and show the frames alternately on the screen using **getbackframe()** and **flip()** functions (double frame). For convenience, the showing frame is called as a front frame and the other frame is called as a back frame.
(To use **setdoubleframebuffer()** function, two frames should be created.)
For screen output, the frame must be chosen by calling one of these functions; **setframebuffer()**, **setdoubleframebuffer()**.

Parameter

surf	**createframe()** created SURFACE structure First frame pointer for the screen output.
surf2	**createframe()** created SURFACE structure Second frame pointer for the screen output.

Return value

None.

8.5 setframebufferxy

void setframebufferxy(**SURFACE*** surf, **U32** x, **U32** y);

Function to control the on-screen frame's start location and shows the frame on the screen from the particular point of it. Could be used when the frame size is bigger than the screen size and max values of x/y could be difference value of the screen resolution within the frame size.
(Example: 800*480 resolution, frame size: 1024*512, then Max x = 224, Max y = 32)
Before using **setframebufferxy()**, the on-sceen frame must be chosen by calling one of these functions; **setframebuffer()**, **setdoubleframebuffer()**.

Parameter

surf	**createframe()** created SURFACE structure.
x	starting x coordinate to be displayed among all coordinate of the frame
y	starting y coordinate to be displayed among all coordinate of the frame.

Return value

None.

8.6 set_draw_target

Void set_draw_target(**SURFACE*** surf)

Function to choose frame to display images or to draw shapes. By denoting the frame created by **createframe()**, makes a drawer to draw things to the frame when the draw related functions are called.

Parameter

surf frame pointer. **createframe()** created SURFACE structure

Return value

None.

8.7 get_draw_target

SURFACE* get_draw_target( );

Bring the frame pointer that is a drawtarget currently.

Parameter

None.

Return value

frame pointer. **createframe()** created SURFACE structure.

8.8 getbackframe

SURFACE*getbackframe()

Function returns hidden screen frame's pointer, when using two frames with **setdoubleframebuffer()** function. Mostly, it is used to draw to the off-screen frame with **setdrawtarget()** function.

Parameter

None.

Return value

Returns **createframe()** created SURFACE structure. Has the frame information.

8.9 getfrontframe

SURFACE*getfrontframe()

Function returns the on-screen frame pointer when using two frames with **setdoubleframebuffer()** function.

Parameter

None.

Return value

Returns **createframe()** created SURFACE structure. Has the frame information.

8.10 flip

void flip()

Function to switchover between frames when using two frames with **setdoubleframebuffer()** function.

Parameter

None.

Return value

None.

8.11 getscreenwidth

U32 getscreenwidth( )

Function to return a current window's width size. The value is configured by **setscreen()** function is returned.

Parameter

None.

Return value

Returns window's width size.

8.12 getscreenheight

U32 getscreenheight( )

Function to return a current window's height size. The value is configured by **setscreen()** function is returned.

Parameter

None.

Return value

Returns window's height size.

8.13 getscreenpitch

U32 getscreenpitch()

Function to return the pitch value of the screen. The pitch value is configured by **setscreen()** function is returned. The pitch value is calculated by screen width × bpp÷ 8.

Parameter

None.

Return value

Returns screen's pitch value.

8.14 getscreenbpp

U32 getscreenbpp()

Function to return the bpp(bit per pixel) value of the screen. The bpp value is configured by **setscreen()** function is returned.

.

Parameter

None.

Return value

Returns screen's bbp value.

8.15 drawputpixel

void drawputpixel(**int** x, **int** y, **U8** r, **U8** g, **U8** b);

Function to draw a single pixel. Draws a pixel at specific location with specific color. It gives RGB values directly, so **drawsetrgb( )** function has no effect on it.

Parameter

x	x coordinate where draws the pixel
y	y coordinate where draws the pixel
r	r value of pixel RGB value. (0 ~ 255)
g	g value of pixel RGB value. (0 ~ 255)
b	b value of pixel RGB value. (0 ~ 255)

Return value

None.

8.16 draw_line

void draw_line(**U32** x1, **U32** y1, **U32** x2, **U32** y2, **EGL_COLOR** color)

Function to draw a line from x1, y1 coordinate to x2, y2 coordinate. The line color is configured by MAKE_COLORREF( ) macro function.

Parameter

x1	x coordinate where the line starts.
y1	y coordinate where the line starts.
x2	x coordinate where the line ends.
y2	y coordinate where the line ends.
color	Color value. Use MAKE_COLORREF(r, g, b) macro function.

Return Value

None.

8.17 draw_rect

void draw_rect(**S16** x, **S16** y, **U16** w, **U16** h, **EGL_COLOR** c)

Function to draw a rectangle (with width w / height h) that starts from coordinates x, y.

Parameter

x	x coordinate where the rectangle starts

y	y coordinate where the rectangle starts
w	rectangle's width.
h	rectangle's height.
c	Color value. Use MAKE_COLORREF(r, g, b) macro function.

Return value

None.

8.18 draw_rectfill

void draw_rectfill(**U32** x, **U32** y, **U32** w, **U32** h, **EGL_COLOR** c)

Function to draw a filled rectangle (with width w / height h) that starts from coordinates x, y.

Parameter

x	x coordinate where the rectangle starts
y	y coordinate where the rectangle starts
w	rectangle's width.
h	rectangle's height.
c	Color value. Use MAKE_COLORREF(r, g, b) macro function.

Return value

None.

8.19 draw_roundrect

void draw_roundrect(**S16** x0, **S16** y0, **U16** w, **U16** h, **U16** corner, **EGL_COLOR** c)

Function to draw a rounded corner rectangle (with width w / height h) that starts from coordinates x, y. The corner values adjust the round strength.

Parameter

x0	x coordinate where the rectangle starts
y0	y coordinate where the rectangle starts
w	rectangle's width.
h	rectangle's height.
corner	corner round strength. Bigger values indicates wider round.
c	Color value. Use MAKE_COLORREF(r, g, b) macro function.

Return value

None.

8.20 draw_roundrectfill

void draw_roundrectfill(**S16** x0, **S16** y0, **U16** w, **U16** h, **U16** corner, **EGL_COLOR** c)

Function to draw a rounded corner rectangle (with width w / height h) that starts from coordinates x, y. The rectangle is filled. The corner values adjust the round strength.

Parameter

x0	x coordinate where the rectangle starts
y0	y coordinate where the rectangle starts
w	rectangle's width.
h	rectangle's height.
corner	corner round strength. Bigger values indicates wider round.
c	Color value. Use MAKE_COLORREF(r, g, b) macro function.

Return value

None.

8.21 draw_circle

void draw_circle(**U32** x, **U32** y, **U32** rad, **EGL_COLOR** color)

Function to draw a circle that has coordinate (x, y) as a center with a radius (rad).

Parameter

x	x coordinate of the center
y	y coordinate of the cetner
rad	radius of the circle
color	Color value. Use MAKE_COLORREF(r, g, b) macro function.

Return value

None.

8.22 draw_circlefill

void draw_circlefill(**U32** x0, **U32** y0, **U32** rad, **EGL_COLOR** c)

Function to draw a filled circle that has coordinate (x, y) as a center with a radius (rad).

Parameter

x	x coordinate of the center
y	y coordinate of the cetner
rad	radius of the circle
c	Color value. Use MAKE_COLORREF(r, g, b) macro function.

Return value

None.

8.23 draw_ellipse

void draw_ellipse (**S16** x0, **S16** y0, **U16** Xradius, **U16** Yradius, **EGL_COLOR** c);

Function to draw an ellipse that has coordinate (x0, y0) as a center with a radius along the x axis (Xradius) and a radius along the y axis (Yradius).

Parameter

x0	x coordinate of the center
y0	y coordinate of the cetner
Xradius	radius along the x axis of the ellipse
Yradius	radius along the y axis of the ellipse
c	Color value. Use MAKE_COLORREF(r, g, b) macro function.

Return value

None.

8.24 draw_ellipsefill

void draw_ellipsefill (**S16** x0, **S16** y0, **U16** Xradius, **U16** Yradius, **EGL_COLOR** c);

Function to draw a filled ellipse that has coordinate (x0, y0) as a center with a radius along the x axis (Xradius) and a radius along the y axis (Yradius).

Parameter

x0	x coordinate of the center
y0	y coordinate of the cetner
Xradius	radius along the x axis of the ellipse
Yradius	radius along the y axis of the ellipse
c	Color value. Use MAKE_COLORREF(r, g, b) macro function.

Return value

None.

8.25 loadbmp

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

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

.

Parameter

fname	bmp file name

Return value

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

8.26 loadbmpp

SURFACE* loadbmpp(**U8*** startaddr)

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

Parameter

startaddr	Image stored memory address .

Return value

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

8.27 loadjpg

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

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

Parameter

fname	jpg file name

Return value

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

8.28 loadjpgp

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

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

Parameter

databuf	Image stored memory address.
len	length of the image data

Return value

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

8.29 loadtga

SURFACE* loadtga(**char*** fname)

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

Parameter

fname	tga file name

Return value

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

8.30 loadtgap

SURFACE* loadtga(**U8*** startaddr)

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

Parameter

Startaddr Image stored memory address.

Return value

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

8.31 loadpng

SURFACE* loadpng(**char*** filename)

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

Parameter

filename png file name

Return value

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

8.32 loadpngp

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

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

Parameter

Pngbuf Image stored memory address
Datalen length of the image data

Return value

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

8.33 loadsurf

SURFACE* loadsurf(**char*** fname)

Function to load a suf file. Loads a suf file, creates a memory region, creates a SUFRACE structure with **drawsurface()** function and stores it.
suf file is created by MakeSurface2.exe in pc-util folder. MakeSurface2.exe is converting program (BMP, JPG, PNG ➔ suf). suf file loading time is shortest.

Parameter

fname	File name

Return value

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

8.34 loadjpg_hw

SURFACE* loadjpg_hw(**char*** fname)

Function to load a jpg file using H/W decoder. Loads a jpg file, creates a memory region, creates a SURFACE 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

fname	JPG file name

Return value

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

8.35 draw_surface

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

Function to output an image.

Parameter

src_surf	A pointer to the memory region storing the returned image data from load function.
dx	x coordinate where the image locates.

dy	y coordinate where the image locates.

Return value

TRUE or FALSE

8.36 draw_surface_rect

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

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

src_surf	A pointer to the memory region storing the returned image data from load function.
dx	x coordinate where the image locates.
dy	y coordinate where the image locates.
sx	x coordinate of the image to be displayed.
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

8.37 draw_set_clip_winodw

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

Function to limit the region that an image could be displayed. The image cannot be displayed outside of the region from pRect.

Parameter

dst	Destination frame.
pRect	Display area.

```
typedef_struct
{
    int  x;
```

```
    int  y;
    int  endx;
  int  endy;
} CLIP_RECT;
```

Return value

None.

8.38 draw_surface_scale

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

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

src_surf	A pointer to the memory region storing the returned image data from load function.
dx	x coordinate where the image is displayed.
dy	y coordinate where the image is displayed.
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.
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 drawsurfacescalerect(**SURFACE*** src_surf, **int** dx, **int** dy, **int** dw, **int** dh,
int sx, **int** sy, **int** sw, **int** sh)

Function to merge the functions of **drawsurfacescale()** function and **drawsurfacerect()** 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

src_surf	A pointer to the memory region storing the returned image data from load function.
dx	x coordinate where the image is displayed.
dy	y coordinate where the image is displayed.
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.
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.
sx	initial x coordinate of the part of the original image to be displayed.
sy	initial y coordinate of the part of the original image to be displayed.
sw	horizontal length of the output image. From sx to sw
sh	vertical length of the output image. From sy to sh

Return value

TRUE or FALSE

8.40 release_surface

void release_surface(**SURFACE*** surf);

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

surf	A memory region pointer storing returned image data from load function.

Return value

None.

8.41 savebmp

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

Save the screen as bmp image.

Parameter

savefname	BMP image file name
src	Screen image source

Return value

TRUE or FALSE

8.42 savejpg

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

Save the screen as jpg image.

Parameter

savefname	JPG image file name
src	Screen image source

Return value

TRUE or FALSE

<savebmp/savejpg example>

```
SURFACE* current_frame = getfrontframe();
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.jpg
```

8.43 createsurface_from

SURFACE* createsurface_from(**SURFACE*** src, **U32** option);

Make copy of the surface. Can make rotated copy. Option is like following.

```
#define PIVOT_RIGHT (1)      // 90 degree right rotation
#define PIVOT_LEFT (1<<1)    // 90 degree left rotation.
#define PIVOT_180 (1<<2)     // 180 degree.
#define PIVOT_VFLIP (1<<4)   // up down invert.
#define PIVOT_HFLIP (1<<5)   // right left invert
```

Parameter

src	Original surface.
option	copy option.

Return value

copied surface..

Example

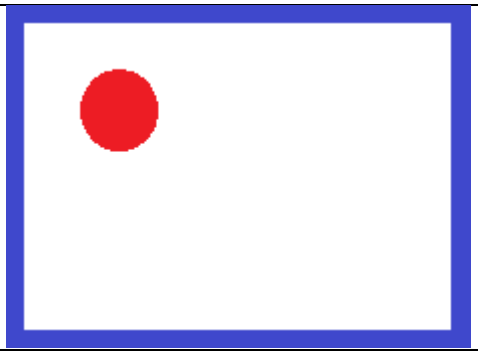
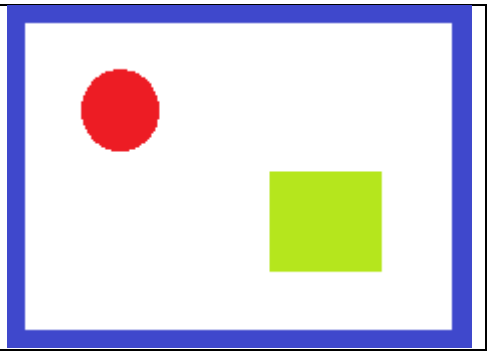
```
SURFACE* surface = loadbmp("test.bmp");
SURFACE* surface_right =createsurface_from(surface, PIVOT_RIGHT)
SURFACE* surface_left =createsurface_from(surface, PIVOT_LEFT)
SURFACE* surface_180 =createsurface_from(surface, PIVOT_180)
SURFACE* surface_vflip =createsurface_from(surface, PIVOT_VFLIP)
SURFACE* surface_hflip =createsurface_from(surface, PIVOT_HFLIP)

drawsurface(surface,0,0);
drawsurface(surface_right,200,0);
drawsurface(surface_left,0,200);
```

8.44 Single frame & double frame usage example

< SINGLE FRAME >

Single frame creates only one on-screen frame.

It draws images and shapes on currently showing screen.

Showing frame		

Like above figure, a single frame is created and images and shapes are drawn to the same frame.

How to configure and use the single frame.

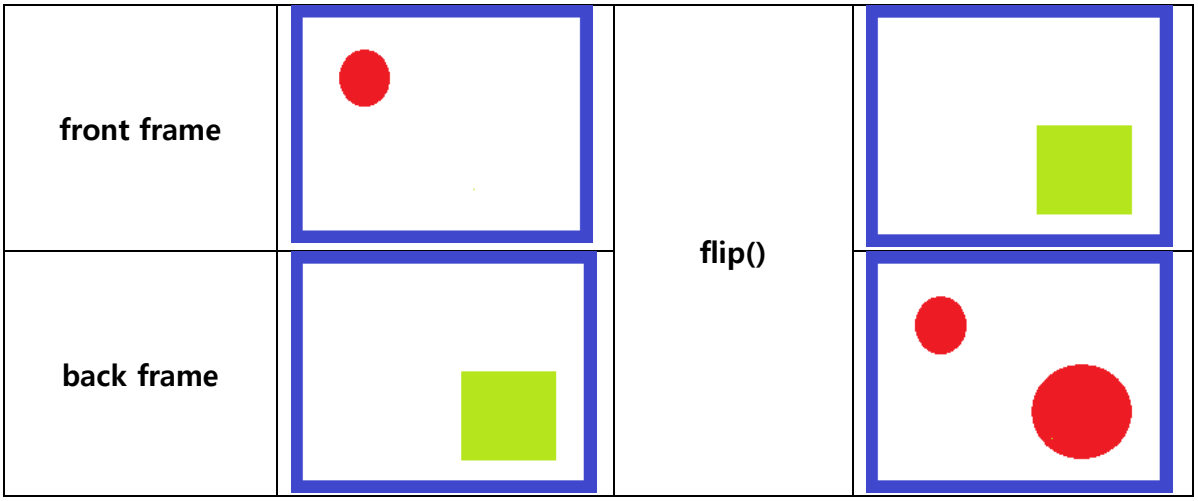
```
setscreen(SCREEN_800x480, SCREENMODE_RGB565 )
SURFACE* frame = createframe(800, 480, 16);
setframebuffer(frame);
setdrawtarget(frame);

drawcirclefill(50,50,10,MAKE_COLORREF(255,0,0));
drawrect(100,100,100,100,MAKE_COLORREF(0,255,0));
```

Creates a frame, configures the frame with **setframebuffer()** function and **setdrawtarget()** function ,and draws shapes or images.

<DOUBLE FRAME >

Double frame creates two frames. One frame is showing and the other is not on the screen. In convenience, the showing frame is called as the front frame and the other is called as the back frame.

Double frame has advantages that system can draw images and shapes on the back frame without any influences on the front frame.



As depicted in the above figure, even if the shapes are drawn on the back frame, no changes occurred on the screen (no effects on the front frame). When calling flip() function after drawing shapes on the back frame, switching between the frames occurred. As a result, the back frame is shown on the screen (becomes the front frame) and the front frame is gone back to the screen (becomes the back frame).

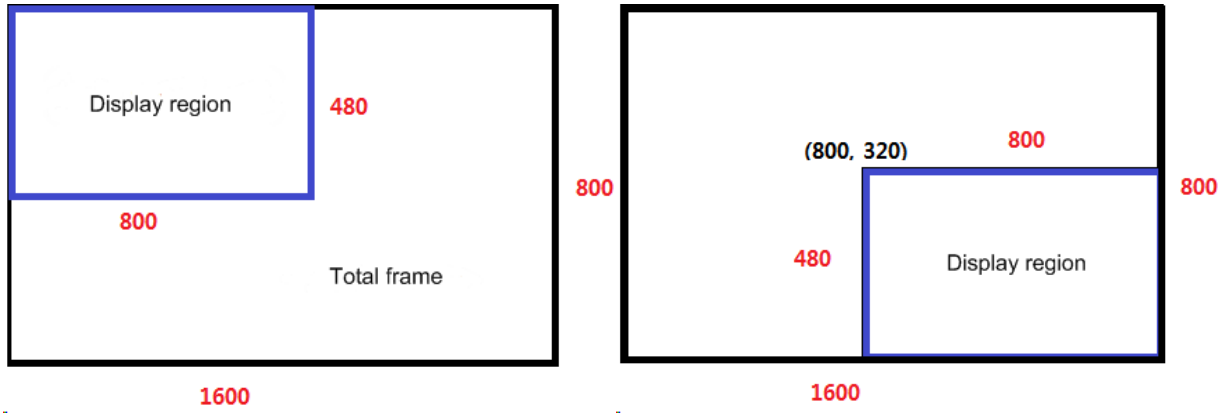
How to use the double frame.

```
setscreen(SCREEN_800x480, SCREENMODE_RGB565 )
SURFACE* frame1 = createframe(800, 480, 16);
SURFACE* frame2 = createframe(800, 480, 16);

setdoubleframebuffer(frame1, frame2);

set_draw_target(getbackframe());
draw_circlefill(50,50,10,MAKE_COLORREF(255,0,0));
flip();

set_draw_target(getbackframe());
draw_rectfill(600,300,100,100,MAKE_COLORREF(0,255,0));
flip();
```

```
set_draw_target(getbackframe());
draw_circlefill(600,300,20,MAKE_COLORREF(255,0,0));
```

Creates two frames, makes the back frame to a draw target with **setdrawtarget()** function and draws shapes or images. After that, brings the back frame on the screen by making the back frame to the front frame with **flip()** function. The front frame becomes the back frame and shapes and images are drawn on the frame with **setdrawtarget()** function that makes the frame as a draw target.

8.45 Set frame buffer xy usage example



Using setframebufferxy function, when a bigger frame than the screen size, only particular parts could be displayed on the screen as depicted in the above figure.

```
setscreen(SCREEN_800x480, SCREENMODE_RGB565);
SURFACE *frame = createframe(1600,800,16);
setframebuffer(frame);
set_draw_target(frame);
.....
.....
setframebufferxy(frame,800,320);    // Like the right side figure of the above, the parts of
                                       the frame is shown (from (800, 320) of the frame).
```

Configures the frame size is bigger than the screen resolution and chooses the frame that has bigger resolution than the screen resolution with **setframebuffer( )** function. After that, configures the origin of the parts to be shown then the frame is shown on the screen from the origin

.

8.46 Graphic Example

< SINGLE FRAME >

```
#include "adStar-L.h"
int main()
{
	boardinit();
	uart_config(1,115200,DATABITS_8,STOPBITS_1,UART_PARNONE);
	debugstring("================================================₩r₩n");
	debugprintf("ADSTAR-L Graphic.System Clock(%dMhz)₩r₩n",get_ahb_clock()/1000000);
	debugstring("================================================₩r₩n");

	 crtc_clock_init();   // Configures CRT Controller.

	 setscreen(SCREEN_800x480,SCREENMODE_RGB565);// Configures resolution and draw mode.
	 SURFACE *frame = createframe(800,480,16);// Generates a frame.
	 setframebuffer(frame);   // Selects a frame to display on a screen.
	 setfrawtarget(frame);// Selets a frame to drow image or shapes.

	 lcdon();
	 draw_rectfill(0,0,getscreenwidth(),getscreenheight(),MAKE_COLORREF(255,255,255));
	                  // After clears screen, drows white rectangle from 0,0 to 800,480.

	 draw_line(30,30,100,100,MAKE_COLORREF(255,0,0));
	                                  // Drows line from 30,30 to 100,100.
	 draw_rect(10,10,100,100,MAKE_COLORREF(255,0,0));
	                                  // Drows rectangle from 10,10 with hight 100 and width 100.
	 draw_circle(100,100,10,MAKE_COLORREF(255,0,0));
	                                 // Drows a circle that its origin is 100,100 and a radius is 10.
	 draw_circlefill(200,200,20,MAKE_COLORREF(255,0,0));
	                                // Drows a filled circle that its origin is 200,200 and a radius is 20.
	 draw_roundrect(50,50,100,100,5,MAKE_COLORREF(255,0,0));
	                  // Drows a rounded rectangle from 50,50 with hight is 100 and width is 100.
	 draw_ellipse (300, 100, 20, 40,MAKE_COLORREF(255,0,0));
	  // Drows an ellipse that its origin is 300,100 and radius of x-axis and y-axis are 20 and 40,
	     respectively.
	  while(1);
	  return 0;
}
```

< DOUBLE FRAME >

```
#include "adStar-L.h"

int main()
{
    boardinit();
    uart_config(1,115200,DATABITS_8,STOPBITS_1,UART_PARNONE);
    debugstring("================================================₩r₩n");
    debugprintf("ADSTAR-L Graphic.System Clock(%dMhz)₩r₩n",GetAHBclock()/1000000);
    debugstring("================================================₩r₩n");

    FATFS fs;
    f_mount(DRIVE_NAND, &fs);

      crtc_clock_init();   //Set CRT Controller.
      setscreen(SCREEN_800x480,SCREENMODE_RGB565);//Set resolution and the draw mode
      SURFACE *frame1 = createframe(800,480,16);          //Create frame1.
      SURFACE *frame2 = createframe(800,480,16);          //Create frame2.
                                          // Creates 2 frames in order to use double frame.
      setdoubleframebuffer(frame1,frame2);              //Configure as double frame
      set_draw_target(getbackframe());                    // Set the draw target as the back frame.

      lcdon();
      SURFACE* image1 = loadbmp("test1.bmp");   // load test1.bmp file from Nand Flash.
      draw_surface(image1,0,0);                      // Display the bmp image on (0, 0). (back frame)
      flip();                                 // switch over frames, show the back frame on the screen.

      set_draw_target(getbackframe());// Set the draw target as the back frame

      SURFACE* image2 = loadjpg("test2.jpg");// load test2.jpg file from Nand Flash.
     draw_surface_rect(image2,0,0,100,100,200,200);// Display parts of the image from (100, 100)
                                          //with vertical/horizontal lengths 200 (both) on (0, 0).
      flip();                    // switch over frames, show the back frame on the screen (jpg image).
      release_surface(image1);          // remove not using image1 from memory.
      while(1);
       return 0;
}
```

9. Movie file play

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

See the separate documentation for details a abput MovieGenerator™

9.1 loadmovie

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

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

fname	Movie file name.
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);

Release memory that used.

Parameter

mov	Pointer to MOVIE

Return value

None.

9.3 movie_play

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

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

Parameter

mov	Pointer to MOVIE
x	x coordinate
y	y coordinate
bAudio	Determine whether the ouput 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);

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

Parameter

mov	Pointer to MOVIE.
x	x coordinate.
y	y coordinate.
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);

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

Parameter

mov	Pointer to MOVIE.
x	x coordinate.
y	y coordinate.
w	width
h	height
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);

Change the current file position to specific time.

Parameter

mov	Pointer to MOVIE
ms	Milliseconds

Return value

Number of video frame.

9.6 movie_current_time

U32 movie_current_time(**MOVIE*** mov)

Return time of current displayed frame.

Parameter

mov Pointer to MOVIE.

Return value

Curent 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

This chapter explains the adStar-L's sound output.

adStar-L supports 8bit/16bit, signed/unsigned, mono/stereo WAV and MP3.

10.1 sound_init()

BOOL sound_init( )

Function to initialize a soundmixer for the sound output.

Parameter

None.

Return value

None.

10.2 sound_loadwav

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

Function to load a WAV file. Creates WAVE structure by loading the WAV file and stores the structure to the memory.

Parameter

filename WAV file

Return value

WAV file information and data stored WAVE structure.

10.3 sound_loadmp3

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

Function to load a MP3 file. Creates MP3 structure by loading the MP3 file and stores the structure to the memory.

Parameter

filename MP3 file

Return value

MP3 file information and data stored MP3 structure.

10.4 sound_release

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

Function to release the memory that is created by **load_soundwav()** or **load_soundmp3()** functions. It is better to return disusing memory by calling **sound_release()** function about the disusing sound, because available memory space is limited.

Parameter

pWave **sound_loadwav( )** or **sound_loadmp3( )** created WAVE structure.

Return value

TRUE(1) or FALSE(0)

10.5 sound_play

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

Function to play sound that is loaded by **load_soundwav()** or **load_soundmp3().**

Parameter

pWave **sound_loadwav( )** or **sound_loadmp3( )** created WAVE structure.

Return value

TRUE(1) or FALSE(0)

➔adStar-L can choose 1 channel of sound output among 4 channels. Channel 0&1 are outputs via an I2S and channel 2&3 are outputs via a digital modulator. SDK configures channel 2 as a default output. In order to change this, substitute digit "2" (it indicates channel 2) of "#define SND_OUTPUT_CHANNEL 2" in libconfig.h file (include folder) to other digit.

10.6 sound_stop

BOOL sound_stop(**WAVE*** pWave)

Function to stop playing sound.

Parameter

pWave WAVE structure of playing sound.

Return value

TRUE(1) or FALSE(0)

10.7 sound_vol

void sound_vol(**U8** vol);

Function to control sound volume.

Parameter

vol volume ranges are 0 - 255.

Return value

None

10.8 sound_pause

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

Function to pause playing sound.

Parameter

pWave WAVE structure of playing sound.

Return value

TRUE(1) or FALSE(0)

10.9 sound_resume

BOOL sound_resume(**WAVE*** pWave);

Function to resume paused sound.

Parameter

pWave WAVE structure of paused sound.

Return value

TRUE(1) or FALSE(0)

10.10 sound_isplay

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

Function to confirm whether the pWave sound is playing.

Parameter

pWave WAVE structure of the sound that is examined that it is playing or not

Return value

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

10.11 sound_ispause

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

Function to confirm whether the pWave sound is paused.

Parameter

pWave WAVE structure of the sound that is examined that it is paused or not.

Return value

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

10.12 Sound Example

< WAV File Play >

```
#include "adStar-L.h"

int main()
{
	boardinit();
	uart_config(1,115200,DATABITS_8,STOPBITS_1,UART_PARNONE);
	debugstring("================================================₩r₩n");
	debugprintf("ADSTAR-L Wave Play.System Clock(%dMhz)₩r₩n",get_ahb_clock()/1000000);
	debugstring("================================================₩r₩n");

	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 “adStar-L.h”

int main()
{
	boardinit();
	uart_config(1,115200,DATABITS_8,STOPBITS_1,UART_PARNONE);
	debugstring("==================================================₩r₩n");
	debugprintf("ADSTAR-L MP3Play.System Clock(%dMhz)₩r₩n",get_ahb_clock()/1000000);
	debugstring("==================================================₩r₩n");

	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

adStar-L SDK can use Nand Flash and SD card as file storage devices by adopting the file system. This chapter explains how to configure Nand Flash and SD card as file storage devices. Refer to **adStar-L SDK/doc** folder – fatfs manual for more detailed information about the file system (SDK/doc/fatfs/doc/00index_e.html).

11.1 f_mount

FRESULT f_mount(BYTE, FATFS*);

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

Result value

Result of the operations.

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

11.2 f_chdrive

FRESULT f_chdrive(BYTE);

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*);

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()
{
        board_init();
        uart_config(1, 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 에

        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

adStar-L SDK supports 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.

<< bmp font >>

12.1 create_bmpfont

EGL_FONT* create_bmpfont(**const char** *fontFile)

Function to prepare to use an bmp font by loading a font file.

Parameter

fontFile font file name

Result value

bmp font pointer.

12.2 release_bmpfont

void release_bmpfont(**EGL_FONT** *pFont)

Function to release no more using bmp font.

Parameter

pFont bmp font pointer.

Result value

None.

12.3 bmpfont_draw

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

Draw text with bmp font.

Parameter

pFont	bmp font pointer.
x	x coordinate.
y	y coordinate.
str	string to be displayed.

Result value

width of displayed string.

12.4 bmpfont_draw_vleft

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

Draw string that rotated 90 degrees to the left with bmp font.

Parameter

pFont	bmp font pointer.
x	x coordinate.
y	y coordinate.
str	string to be displayed.

Result value

width of displayed string.

12.5 bmpfont_draw_vright

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

Draw string that rotated 90 degrees to the right with bmp font.

Parameter

pFont	bmp font pointer.
x	x coordinate.
y	y coordinate.
str	string to be displayed.

Result value

width of displayed string.

12.6 egl_font_set_color

EGL_COLOR egl_font_set_color(**EGL_FONT** *Font, **EGL_COLOR** clr)

Change font color.

Parameter

pFont bmp font pointer.

clr Font color. Use MAKE_COLORREF(r,g,b) macro function.

Result value

Old font color.

12.7 bmpfont_makesurface

SURFACE* bmpfont_makesurface(**EGL_FONT** *pFont, **char** *text)

Function to converse frequently used characters into an image. The characters could be displayed by draw_surface function when necessary. If no more using font, have to release by release_surface() function.

Parameter

pFont bmp font pointer.

text strings to be converted into an image.

Result value

Pointer to SURFACE structure.

12.8 bmpfont_setkerning

bool bmpfont_setkerning(**EGL_FONT** *pFont, **int** k)

Function to set character spacing of the bmp font.

Parameter

pFont bmp font pointer.

k character gap

Return value

TRUE(1) or FALSE(0)

12.9 bmpfont_setautokerning

bool bmpfont_setautokerning(**EGL_FONT** *pFont, **bool** b)

Set whether character spacing same.

Parameter

pFont bmp font pointer.

b TRUE or FALSE

Return value

TRUE(1) or FALSE(0)

<< bit font >>

12.10 create_bitfont

EGL_FONT* create_bitfont( )

Function to initialize to use the bit type font.

Return value

bit font pointer.

12.11 release_bitfont

void* release_bitfont(**EGL_FONT*** pFont)

Function to release no more using bit font.

Parameter

pFont bit font pointer.

Return value

bit font pointer.

12.12 bitfont_draw

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

Function to display characters using the bit type font.

Parameter

pFont	bit font pointer.
x	x coordinate of the characters output
y	y coordinate of the characters output
str	characters to be displayed

Result value

width of displayed string.

12.13 bitfont_draw_vleft

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

Draw string that rotated 90 degrees to the left with bit font.

Parameter

pFont	bit font pointer.
x	x coordinate of the characters output.
y	y coordinate of the characters output.
str	string to be displayed.

Result value

width of displayed string.

12.14 bitfont_draw_vright

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

Draw string that rotated 90 degrees to the right with bit font.

Parameter

pFont	bit font pointer.
x	x coordinate of the characters output.
y	y coordinate of the characters output.

str string to be displayed.

Result value

width of displayed string.

12.15 bitfont_makesurface

SURFACE* bf_makesurface(**EGL_FONT** *pFont, **char** *str)

Function to converse frequently used characters into an image. The characters could be displayed by **drawsurface()** function when necessary. If no more using font, have to release by release_surface() function.

Parameter

pFont bit font pointer.

str characters to be converted into the image.

Return value

pointer to SURFACE structure

12.16 FONT Example

< bm font example >

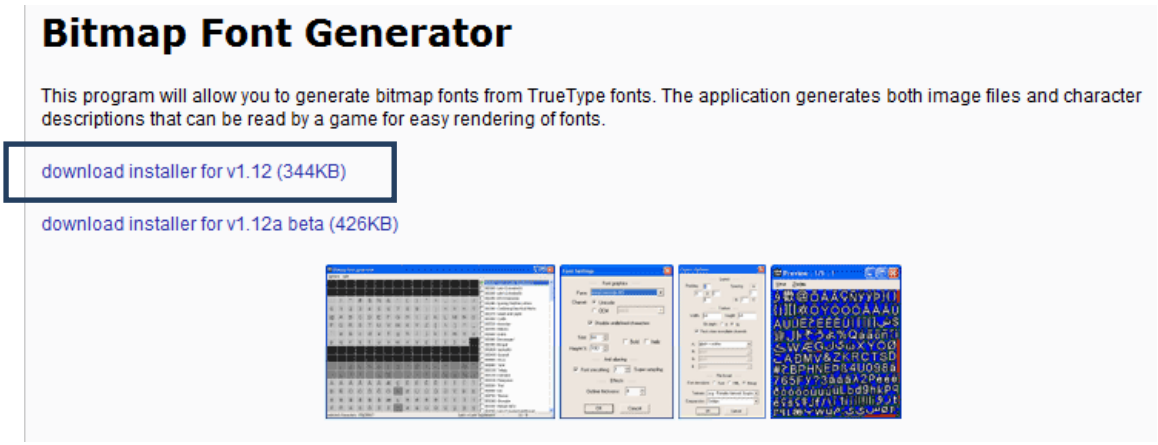
```
#include "adStar-L.h"

int main()
{
        boardinit();
        uart_config(1,115200,DATABITS_8,STOPBITS_1,UART_PARNONE);
        debugstring("==========================================₩r₩n");
        debugprintf("ADSTAR-L FONT Example. System Clock(%dMhz)₩r₩n",get_ahb_clock()/1000000);
        debugstring("==========================================₩r₩n");
        FATFS fs;
        f_mount(DRIVE_NAND,&fs);
        f_chdrive(DRIVE_NAND);
        crtc_clock_init();
        SURFACE*  frame = createframe(800,480,16);
        setscreen(SCREEN_800x480,SCREENMODE_RGB565);
        lcdon();
        setframebuffer(frame);
        setdrawtarget(frame);
        draw_rectfill(0,0,getscreenwidth(),getscreenheight(),MAKE_COLORREF(255,255,255));
        f_chdir("font");        //move to the font folder where font file exist.
        EGL_FONT*  pFont = create_bmpfont("nanum_8bit_16px_han.fnt");
                                // set to load and use nanum_8bit_16px_han.fnt file.
        egl_font_set_color(pFont, MAKE_COLORREF(0, 0, 255));
                                // change the color of  font as blue.
        bmpfont_draw(pFont, 100,100,"Test nanum font");
                                // display the characters onto coordinate (100,100)
        SURFACE*  fontsurf = bmpfont_makesurface(pFont, "bmfont_makesurface surface");
                                //convert the characters to an image. It could be used as an image using
                                //drawsurface function.
        drawsurface(fontsurf,100,300);
                                // display the converted image onto coordinate (100,300).
        while(1);
        return 0;
}
```

< bit font example >

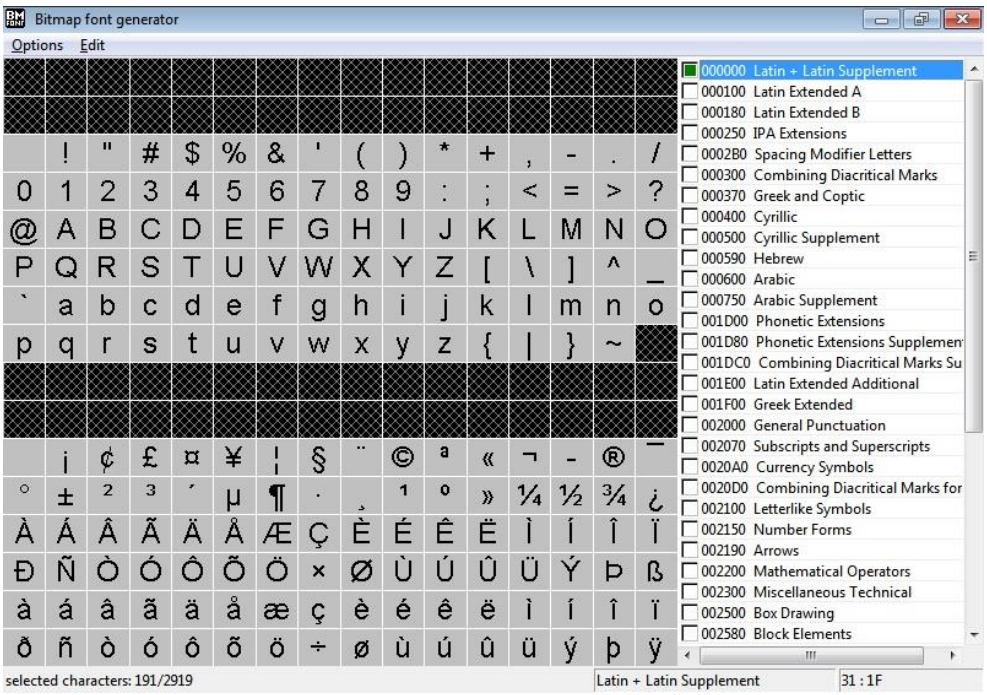
```
#include "adStar-L.h"

int main()
{
        boardinit();
        uart_config(1,115200,DATABITS_8,STOPBITS_1,UART_PARNONE);
        debugstring("================================================₩r₩n");
        debugprintf("ADSTAR-L FONT Example. System Clock(%dMhz)₩r₩n",get_ahb_clock()/1000000);
        debugstring("================================================₩r₩n");

        FATFS fs;
        f_mount(DRIVE_NAND,&fs);
        f_chdrive(DRIVE_NAND);

        crtc_clock_init();
        SURFACE*  frame = createframe(800,480,16);
        setscreen(SCREEN_800x480,SCREENMODE_RGB565);
        lcdon();

        setframebuffer(frame);
        setdrawtarget(frame);
        drawrectfill(0,0,getscreenwidth(),getscreenheight(),MAKE_COLORREF(255,255,255));

        EGL_FONT* pFont = create_bitfont();          // set to use the bit type font.
        egl_font_set_color(pFont, MAKE_COLORREF(255,0,0);
                                        // change the color of the bit type font as red.
        bitfont_draw(pFont, 100,100,"bit type font test");
                            // display the characters onto coordinate (100, 100) using the bit type font.
        SURFACE*  fontsurf = bitfont_makesurface(pFont, "bf_makesurface surface");
                        // convert the characters into an image. It could be used as an image using
                        //drawsurface function.
        drawsurface(fontsurf,100,300);
                        // display converted image onto coordinate (100, 300).
        while(1);
        return 0;
}
```

< how to create the image font >

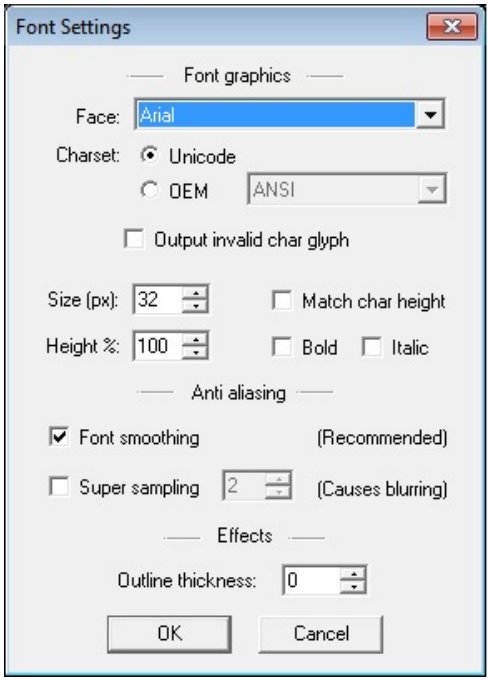
1. Download and install the Bitmap Font Generator v1.12 from http://www.angelcode.com/products/bmfont.



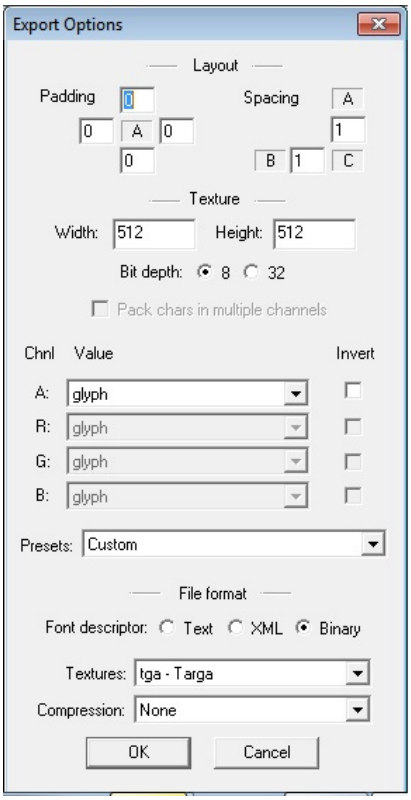
2. Next, execute the program shows up the below window.



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

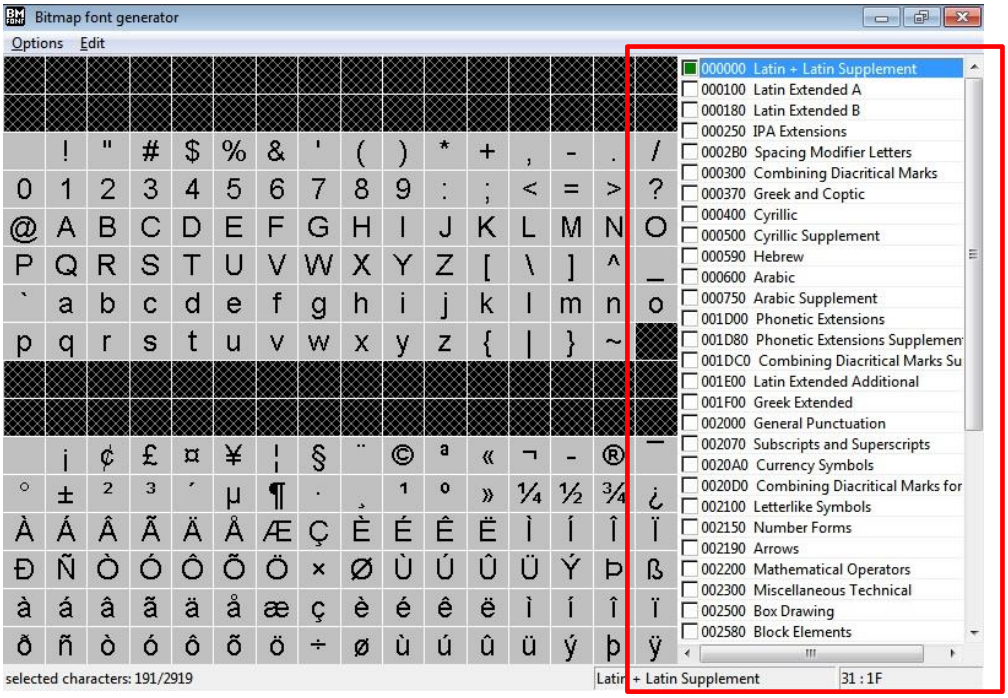


4. Set as the below window after executing 'Options' → 'Export options'.



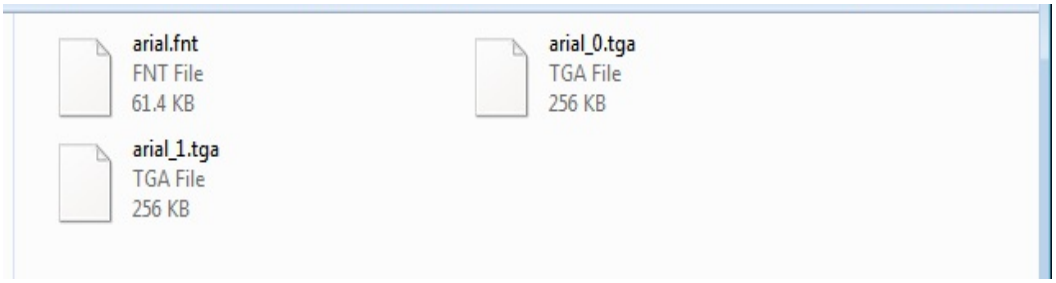
Width and Height are size of a single bitmap file's width and height. Bigger the size reduces total number of the bitmap files and wasted region could exist.

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 bmfont_init function.

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



13. SPI

13.1 spi_master_init

void spi_master_init(**int** ch)

Function to SPI master initial. Can select wanted channel.

Parameter

ch channel value. (0 or 1)

Result value

None.

13.2 spi_set_freq

int spi_set_freq(**int** ch, **int** mode, **U32** freq);

Function to set SPI baud rate.

Parameter

ch channel value. (0 or 1)

mode master / slave value. (SPI_MASTER / SPI_SLAVE)

freq frequency value.

Return value

Baud rate value.

13.3 spi_master_xfer

void spi_master_xfer(**int** ch, **U8** *wbuf, **int** wlength, **U8** *rbuf, **int** length, **int** continue_xfer)

Send wbuf data to the slave as many as wlength. And bring rbuf data to the master as many as rlength at slave.

Parameter

ch channel value. (0 or 1)

wbuf write data buffer.

wlength write data length.

rbuf	read data buffer.
rlength	read data length.
continue_xfer	whether transfer continue.

Return value

None.

13.4 spi_wait_empty_fifo

void spi_wait_empty_fifo(**int** ch)

Check whether tx fifo empty.

Parameter

ch	channel value.

Return value

None.

13.5 SPI Example

<SPI EEPROM Example>

=== w25xxx_flash.h ===

```
#ifndef __W25Xxx_H__
#define __W25Xxx_H__
int w25xxx_write_buffer (U32 addr, U8 *buffer, int length);
int w25xxx_read_buffer (U32 addr, U8 *buffer, int length);
int w25xxx_erase (U32 start_addr, int length);
int w25xxx_block_erase (U32 addr, int length);
int w25xxx_wait_ready (void);
int w25xxx_check_id (void);

#define W25XXX_PAGE_SIZE        (256)
#define W25XXX_SECTOR_SIZE      (4096)
#define W25XXX_BLOCK_SIZE       (16*W25XXX_SECTOR_SIZE)

#define W25X16_TOTAL_SIZE       (2*1024*1024)
#define W25X32_TOTAL_SIZE       (64*W25XXX_BLOCK_SIZE)
```

```
#define W25X64_TOTAL_SIZE       (128*W25XXX_BLOCK_SIZE)
#define W25X32_FLASH_MAXADDR    W25X32_TOTAL_SIZE

// W25Xxx: 3.3V -> Max ??MHz
 #define W25Xxx_SPI_FREQ    23000000    // 23MHz

 // W25Xxx Flash instruction
 #define W25Xxx_WREN         0x06    // Write Enable
 #define W25Xxx_WRDI         0x04    // Write Disable
 #define W25Xxx_RDSR         0x05    // Read Status Register
//#define W25Xxx_WRSR          0x01    // Write Status Register
 #define W25Xxx_READ         0x03      // Read Data Bytes
 #define W25Xxx_FAST_READ    0x0B    // Read Data Bytes at Higher Speed
 #define W25Xxx_DUAL_READ    0x3B    // Read Data Bytes at Higher Speed
 #define W25Xxx_Page_Program       0x02    // Page Program
 #define W25Xxx_BERASE       0xD8    // Block Erase
 #define W25Xxx_SERASE       0x20    // Sector Erase
 #define W25Xxx_CERASE       0xC7    // Sector Erase
 #define W25Xxx_READID       0x90    // Read ID /* ID: 0x00, 0xEF, (0x14, 0x15, 0x16) */
 #define W25Xxx_READJEDECID  0x9F    // Read ID /* ID: 0x00, 0xEF, (0x14, 0x15, 0x16) */

 #define W25XXX_MAKERID      0xEF
 #define W25X16_DEVID        0x14
 #define W25X32_DEVID        0x15

 #define W25Xxx_PDOWN     0xB9    // Power-down
 #define W25Xxx_PON         0xAB    // Release Power-down / Device ID

 #define NORMAL           W25Xxx_READ
 #define FAST             W25Xxx_FAST_READ
 #define BLOCK            W25Xxx_BERASE
 #define SECTOR           W25Xxx_SERASE
 #define ALL              W25Xxx_CERASE

 #define W25Xxx_SR_WIP  (1 << 0)    // Wait In Progress bit
 #define W25Xxx_SR_WEL  (1 << 1)    // Wait Enable Latch bit

 #define W25Xxx_ID1     0x00 // Manufacture ID
 #define W25Xxx_ID2     0xEF // Memory Type
 #define W25X16_ID3     0x14 // Memory Capacity
 #define W25X32_ID3     0x15 // Memory Capacity
```

```
#define W25X64_ID3      0x16 // Memory Capacity

#define W25Xxx_JEDECID1      0xEF // Manufacture ID
#define W25Xxx_JEDECID2      0x30 // Memory Type
#define W25X16_JEDECID3      0x15 // Memory Capacity
#define W25X32_JEDECID3      0x16 // Memory Capacity
#define W25X64_JEDECID3      0x17 // Memory Capacity

#endif //__W25Xxx_H__
```

=== w25xxx_flash.c ===

```
#include "adStar-L.h"
#include "w25xxx_flash.h"
#include "spi.h"

int w25xxx_write_buffer (U32 addr, U8 *buffer, int length)
{
    int wlen;
    U8 wbuf[4];
    int ret = 0;
        int i;

        U32 startaddr = addr;
        U32 endaddr = addr+length-1;
        U32 startpage;
        U32 endpage;

        startpage = startaddr/W25XXX_PAGE_SIZE;
        endpage = endaddr/W25XXX_PAGE_SIZE;

        for(i=startpage;i<endpage+1;i++)
        {
                U32 offset = addr%W25XXX_PAGE_SIZE;
                if(offset + length > W25XXX_PAGE_SIZE)
                {
                        wlen = W25XXX_PAGE_SIZE  - offset;
                }
                else
                {
                        wlen = length;
```

```
                }

                asm("clr 13");

                wbuf[0] = W25Xxx_WREN;
                spi_master_xfer(SPI_CH1,wbuf, 1, NULL, 0, 0);
                ret = w25xxx_wait_ready();
                if (ret != 0)
                        break;

                wbuf[0] = W25Xxx_Page_Program;

                wbuf[1] = (addr>>16);
                wbuf[2] = (addr>>8);
                wbuf[3] = addr;

                spi_master_xfer(SPI_CH1,wbuf, 4, NULL, 0, 1); // Write Command & Address
                                                              // Continue to write data next
                spi_master_xfer(SPI_CH1,buffer, wlen, NULL, 0, 0); // Write Data

                /*  Write Cycle Time of W25Xxx FLASH: 3.3V -> 11ms, 1.2ms */
                ret = w25xxx_wait_ready();
                if (ret != 0)
                        break;

                buffer += wlen;
                addr += wlen;
                length -=wlen;

                wbuf[0] = W25Xxx_WRDI;
                spi_master_xfer(SPI_CH1,wbuf, 1, NULL, 0, 0);
                ret = w25xxx_wait_ready();
                if (ret != 0)
                        break;

                asm("set 13");
        }
        return ret;
}
```

```
/*
 * mode: NORMAL, FAST
 */
int w25xxx_read_buffer (U32 addr, U8 *buffer, int length)
{
    int ret = 0;
    U8 wbuf[4];

        wbuf[0] = W25Xxx_READ;
        wbuf[1] = (addr>>16);
        wbuf[2] = (addr>>8);
        wbuf[3] = addr;
        asm("clr 13");
        spi_master_xfer(SPI_CH1,wbuf, 4, buffer, length, 0);

        ret = w25xxx_wait_ready();
        asm("set 13");

    return ret;
}

/*
 * SECTOR erase
 */
int w25xxx_erase (U32 addr, int length)
{
    U8 wbuf[4];
    int ret = 0;
    unsigned int esize = 0;
    unsigned int sector_size = W25XXX_SECTOR_SIZE;
    unsigned int block_size = W25XXX_BLOCK_SIZE;

    length += sector_size - 1;
    length &= ~(sector_size - 1);

    addr &= ~(sector_size - 1);

    while (length) {

        if (length >= block_size) {
            esize = block_size;
```

```
        }
        else if (length >= sector_size) {
            esize = sector_size;
        }

        wbuf[0] = W25Xxx_WREN;
        spi_master_xfer(SPI_CH1, wbuf, 1, NULL, 0, 0);
        ret = w25xxx_wait_ready();
        if (ret != 0)
            break;

        if (esize == block_size) {
            wbuf[0] = W25Xxx_BERASE;
        }
        else if (esize == sector_size) {
            wbuf[0] = W25Xxx_SERASE;
        }

        wbuf[1] = (addr>>16);
        wbuf[2] = (addr>>8);
        wbuf[3] = addr;

        spi_master_xfer(SPI_CH1, wbuf, 4, NULL, 0, 0);

        ret = w25xxx_wait_ready();
        if (ret != 0)
            break;

        addr += esize;
        length -= esize;

        wbuf[0] = W25Xxx_WRDI;
        spi_master_xfer(SPI_CH1,wbuf, 1, NULL, 0, 0);
        ret = w25xxx_wait_ready();
        if (ret != 0)
            break;
    }
    return 0;
}

int w25xxx_wait_ready (void)
```

```
{
    U8 inst = W25Xxx_RDSR;
    U8 status;
    int delay = 0;
    int ret = 0;

    do {
        spi_master_xfer (SPI_CH1, &inst, 1, &status, 1, 0);
        delay++;
        if (delay > 0x80000) {
            ret = -1;
            break;
        }
    } while (status & W25Xxx_SR_WIP);
    return ret;
}

int w25xxx_check_id (void)
{
    U8 buf[4];

    buf[0] = W25Xxx_READID;
    buf[1] = 0;
    buf[2] = 0;
    buf[3] = 0;

    spi_master_xfer(SPI_CH1, buf, 4, buf, 2, 0);

    w25xxx_wait_ready();

    if ((buf[0] != W25XXX_MAKERID) || (buf[1] != W25X32_DEVID)) {
        debugprintf(" 0x%x ", buf[0] );
        debugprintf("0x%x ₩r₩n", buf[1] ); // W25X32_ID3
       return -1;
    }
    return 0;
}
```

=== main.c ===

```
#include "adStar-L.h"
#include "w25xxx_flash.h"

extern void boardinit();

void SPI_pin_init()
{
        *R_PAF4 &= ~(0xf<<12);
        *R_PAF4 |= 0x5<<12;        //spi_sck1, spi_cs1

        *R_PAF5 &= ~(0xf<<0);
        *R_PAF5 |= 0x5<<0;       //spi_miso1, spi_mosi1
}

int main()
{
        boardinit();
        uart_config(1,115200,DATABITS_8,STOPBITS_1,UART_PARNONE);

        debugstring("================================================₩r₩n");
        debugprintf("ADSTAR-L SPI_Flash.System Clock(%dMhz)₩r₩n",get_ahb_clock()/1000000);
        debugstring("================================================₩r₩n");

        SPI_pin_init();

        int ret;

        spi_master_init(SPI_CH1);
        int baud = spi_set_freq(SPI_CH1,SPI_MASTER, W25Xxx_SPI_FREQ);
        debugprintf("SPI SCK : %d KHz₩n₩r", SPI_SCK(baud)/1000 );

        ret = w25xxx_check_id();
    if (ret != 0) {
        debugstring("W25Xxx ID Check Fail₩r₩n!₩n");
                while(1);
    }
        U8 rbuf[256];
        U8 wbuf[256];
        int i;
```

```
	for(i=0;i<256;i++)
		wbuf[i]=i;
	for(i=0;i<256;i++)
		rbuf[i]=0;

	w25xxx_erase(0,W25XXX_SECTOR_SIZE);
	w25xxx_write_buffer(0,wbuf,256);
	w25xxx_read_buffer(0,rbuf,256);
	for(i=0;i<256;i++)
	{
		if(wbuf[i]!=rbuf[i])
		{
			debugprintf("wbuf[%d] = %d  !=  rbuf[%d] = %d ₩r₩n",i,wbuf[i],i,rbuf[i]);
			debugstring("Test Error₩r₩n");
			while(1);
		}
		else
		{
			debugprintf("wbuf[%d] = %d  ==  rbuf[%d] = %d ₩r₩n",i,wbuf[i],i,rbuf[i]);
		}
	}
	while(1);
	return 0;
}
```

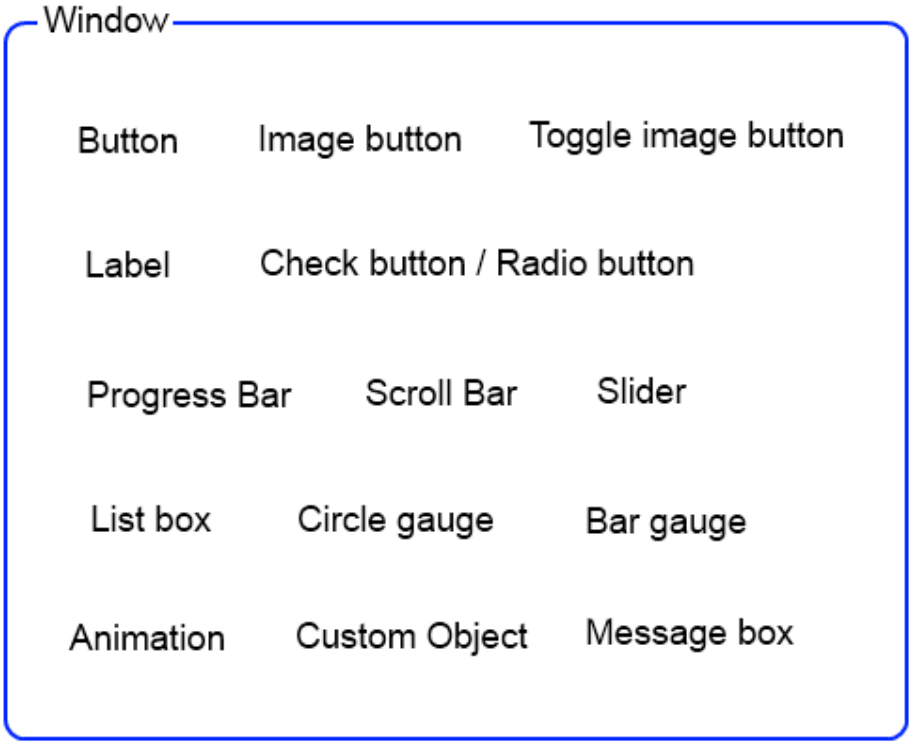
14. EGL Library

1) Itroduce

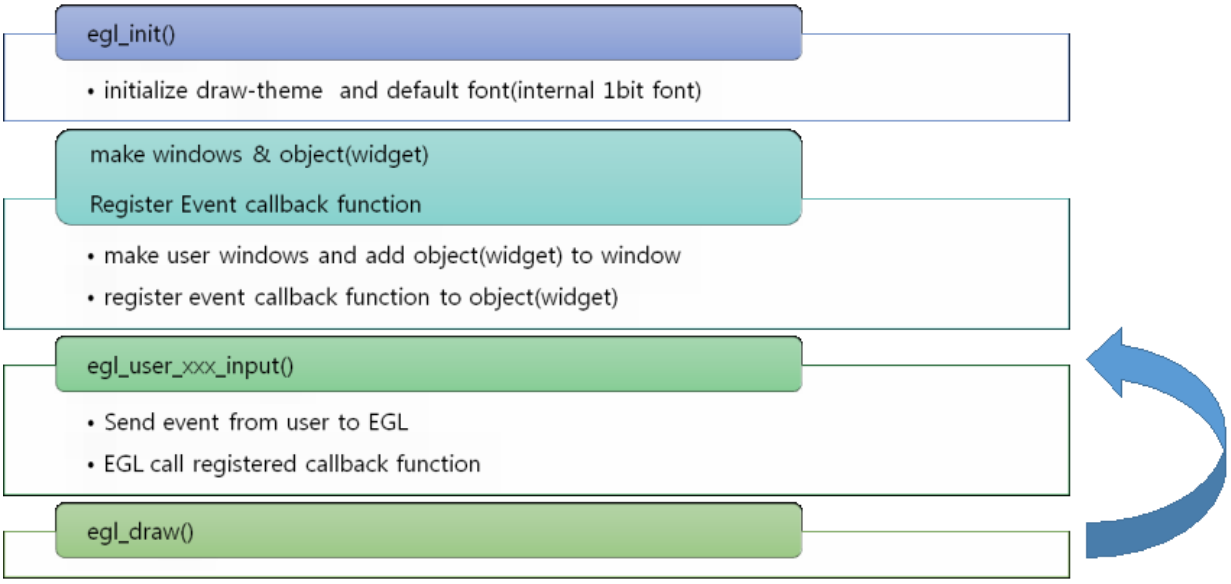
EGL(Embedded Graphic Library)is made for reduce time of graphic application development and use object by easy. EGL is offered function for use many object by easy. Event processing and object management is done in the EGL library. Use Callback function that created by developers for event action.

2) Composition

EGL is composed of follow objects.



3) Program base



< Source Code >

```
#include "adStar-L.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. Apply to screen, after egl_window_invalidate() function called.
(or call egl_window_invalidate_rect( ))

Parameter

EGL_HANDLE h	object handle.
bool b	visible status. TRUE : visible, FALSE : invisible

Return Value

None.

Example

```
egl_visible_object( button, FALSE );
egl_window_invalidate( );
…
egl_draw( );
```

▶egl_window_delete_object function

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

Overview

Delete object of window. Apply to screen, after egl_window_invalidate() function called.
(or call egl_window_invalidate_rect( ))

Parameter

EGL_HANDLE hWin	window handle.
EGL_HANDLE hObj	object handle.

Return Value

TRUE or FALSE.

Example

```
egl_window_delete_object( hWin, button );
egl_window_invalidate( );
...
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-L.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

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

TRUE or FALSE

Example

▶ 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

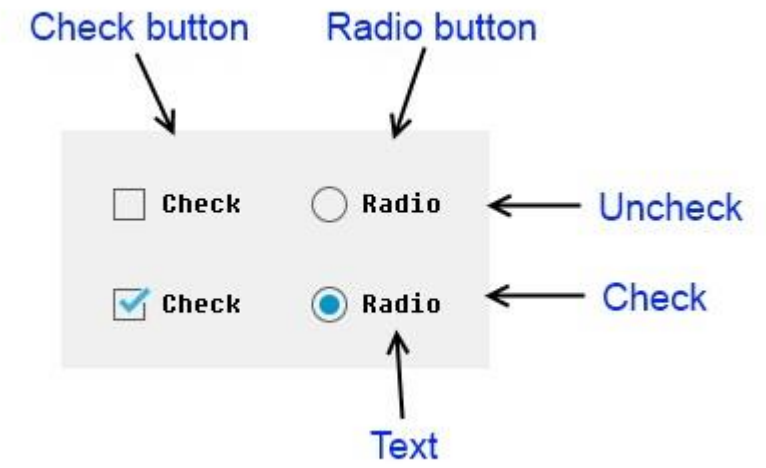
```
if( label != NULL )
{
    egl_window_delete_object( hWin, label );
    egl_release_label( label );
    label = NULL;
}
```

▶ 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;
}
```

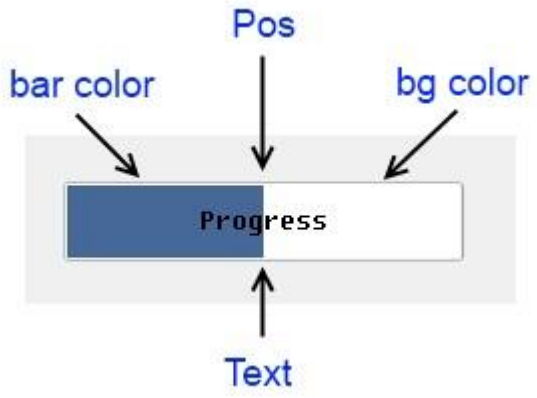
▶ Check / Radio button Example.

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 vaule. Use MAKE_COLORREF(r,b,g) 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

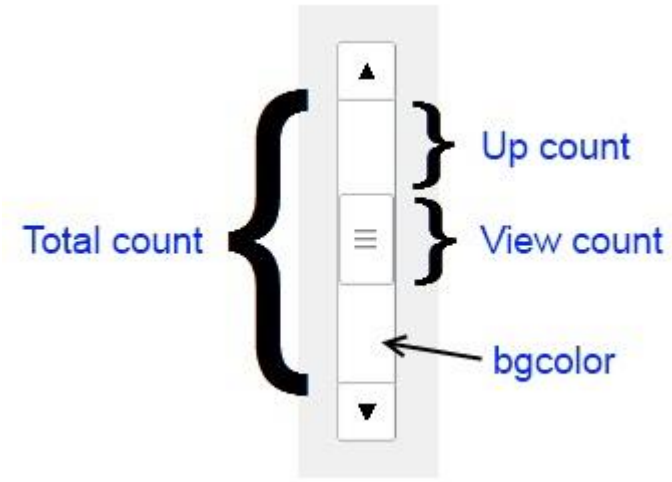
```
if(pgbar != NULL)
{
    egl_window_delete_object( hWin, pgbar );
    egl_release_progressbar(pgbar);
    pgbar = NULL;
}
```

▶ 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

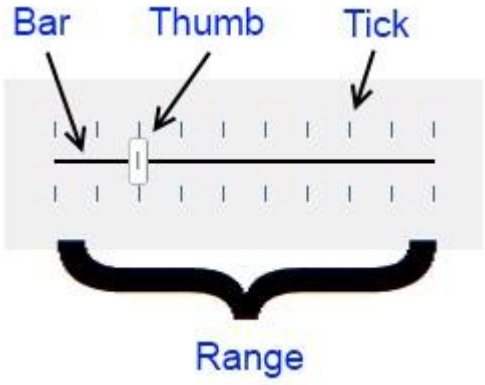
```
  If( scroll != NULL )
{
    egl_window_delete_object(  hWin, scroll );
    egl_release_scrollbar( scroll );
    scroll = NULL;
 }
```

▶ 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

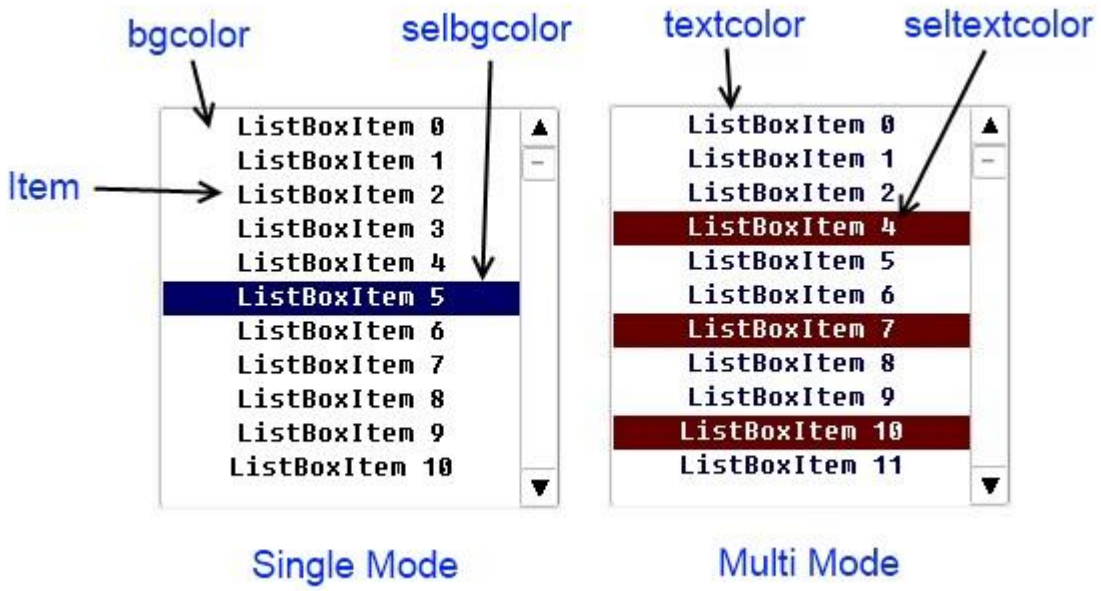
```
if( slider != NULL)
{
    egl_window_delete_object( hWin, slider );
    egl_release_slider( slider );
    slider = NULL;
}
```

▶ 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 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_event_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

List box 에서 text 에 해당되는 Item 을 제거한다.

Parameter

EGL_HANDLE hObj	Item 을 제거할 List box 의 handle
const char* text	List box 에서 제거할 Item 의 text

Return Value

Item 제거의 성공 여부를 반환

Example

```
EGL_HANDLE listbox;

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

▶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 value. 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_num (
      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_num( 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.

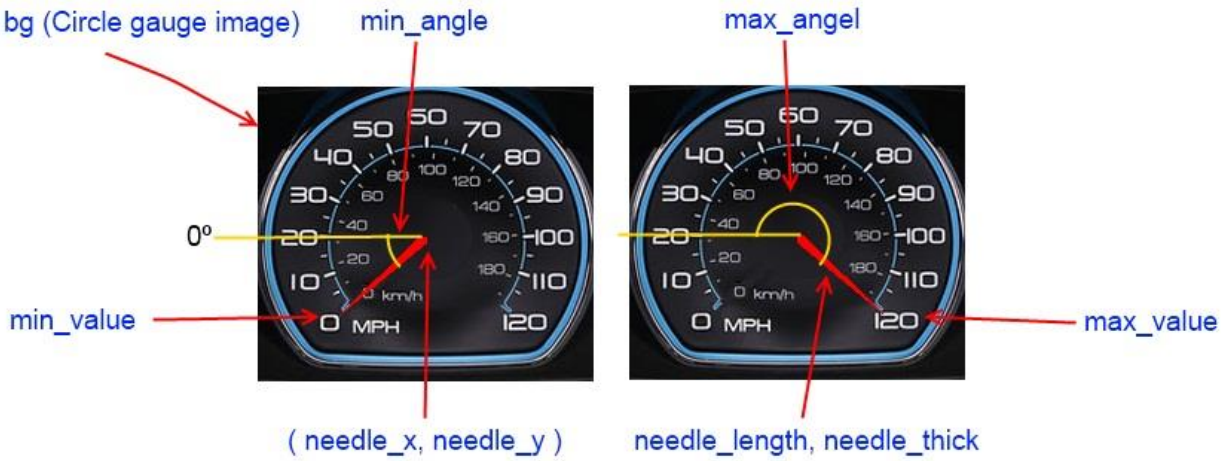
Example

```
#include "adStar-L.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], 0xff, 0xff, 0xff);
	// Set selected background color
	egl_listbox_set_selbgcolor(listbox[1], 0x66, 0x0, 0x0);
	// Set text color
	egl_listbox_set_textcolor(listbox[1], 0, 0, 0x66);
```

```
        // Set selected text color
        egl_listbox_set_seltextcolor(listbox[1], 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;
}
```

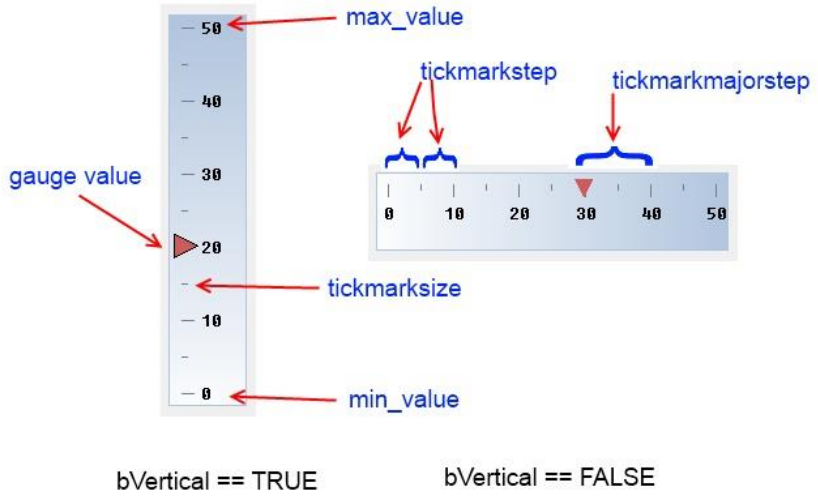
▶ Circle Gauge Example.

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



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_windwo_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 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 surface of picture object.
int x	Picture x-coordinate.
int y	Picture y-coordinate..
int w	Picture width.
int h	Picture height.

Return Value

Handle of created picture.

Example

```
SURFACE* image = loadbmp("image.bmp");
EGL_HANDLE picture;
picture  = egl_create_picture(image,50, 50, 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 list box.
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 image of Picture object.

Parameter

EGL_HANDLE hObj	Handle of picture.
SURFACE* surf	Image surface.

Return Value

Old image surface.

Example

```
…
SURFACE* oldimage;
SURFACE* image2 = loadbmp(“image2.bmp”);
lodimage = egl_picture_set(picture, image2);
…
```

▶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;
}
```

▶ 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();
    }
}
```

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_picture (
    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);
...
}
```

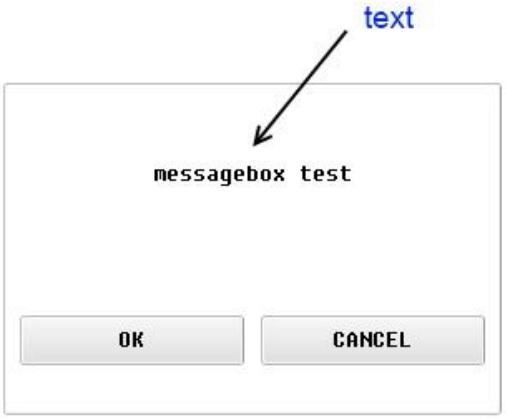
▶ Custom Example.

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



flags == MB_OKCANCEL

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

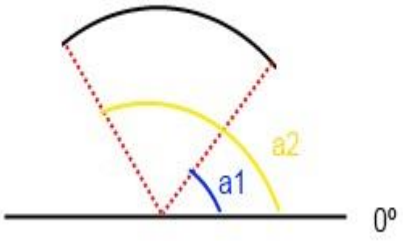
```
void draw_arc(
    int  x,
    int  y,
    int  rx,
    int  ry,
    int  a1,
    int  a2,
    EGL_COLOR  c
);
```

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

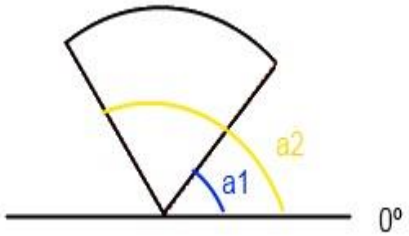
```
draw_pie(
    int  x,
    int  y,
    int  rx,
    int  ry,
    int  a1,
    int  a2,
    EGL_COLOR  c
);
```

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

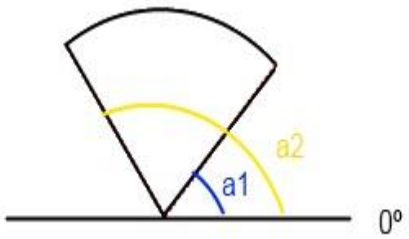
```
void  draw_piefill(
    int  x,
    int  y,
    int  rx,
    int  ry,
    int  a1,
    int  a2,
    EGL_COLOR  c
);
```

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

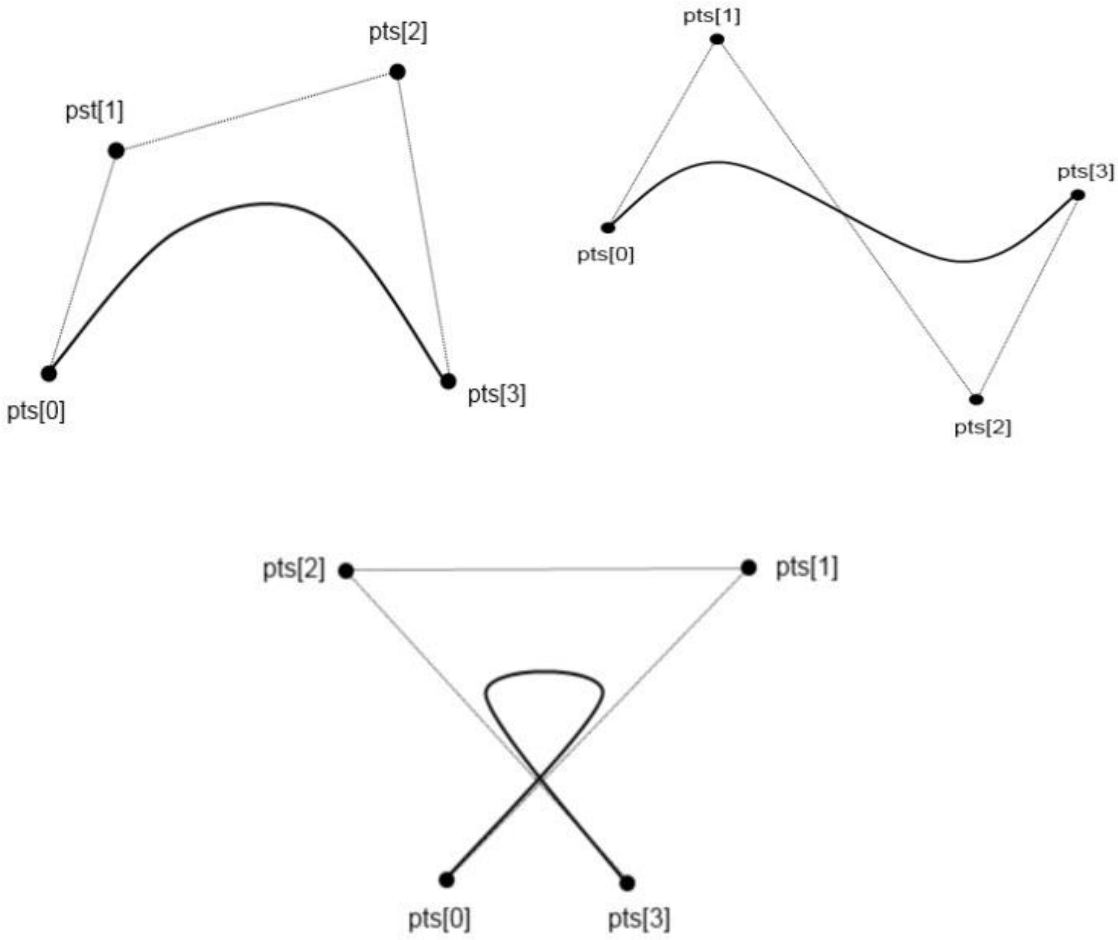
EGL_POINT* pts	Pointer to the coordinate array of Bezier curve.
int n	Number of coordinate.
int s	Decide Number of pointer between the coordinate and the coordinate.
EGL_COLOR c	Bezier curve color. (Use MAKE_COLORREF(r, g, b) macro function.)

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

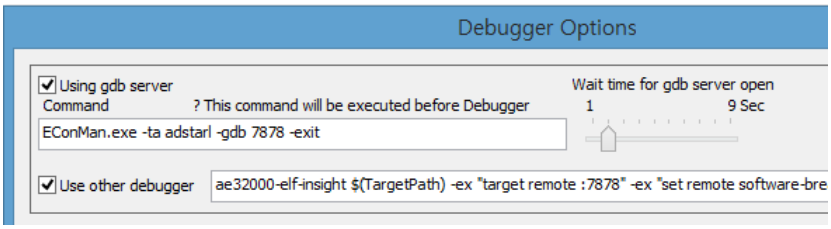
EGL_HANDLE handle	Handle of object.

Return Value

None.

15. Debugging

Open project to debugging and run project → properties. Set true on Build Uption → Tool Chain → Debug Info(-g). Set None(-O0) on Optimize Level(-O0). Run Build → Rebuild Project. And, Run Debug → Debug option. Write –ta adstarl in command like following image.

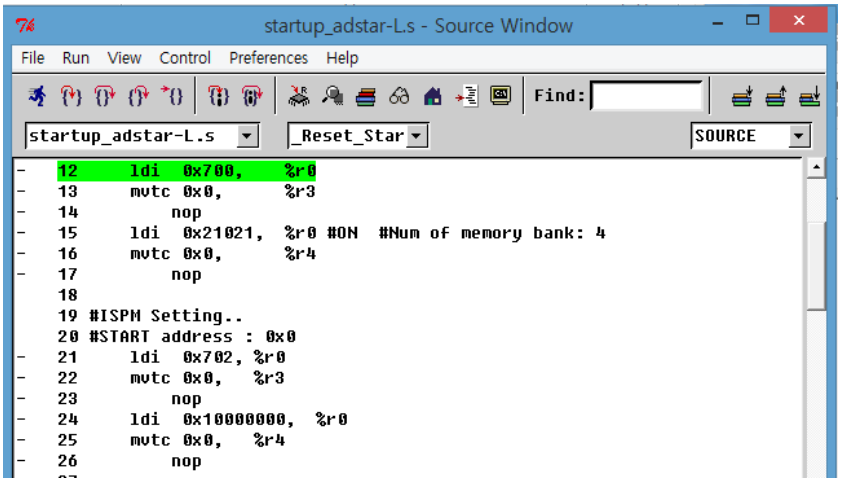


There is two case about debugging.

1. Case that start program at 0 without bootloader. (Use adStar-L.ld as Linker Script)
2. Case that start program at Ram using bootloader. (Use adStar-L_ram.ld as Linker Script)

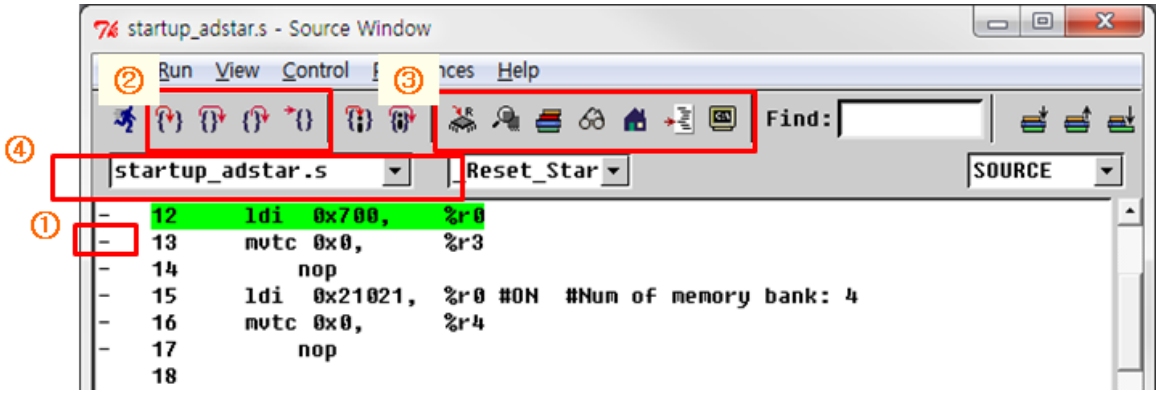
➔ Case that start program at 0

1) binary file download to 0 in flash.
2) After set debug mode (SW3 up), connect ECon and power on.
3) Run Debug → Start Debugger (F5)
4) When appeared following window, you can start debugging.



If the green code line is shown like above image, it means the connection is success.

5) Explain Debugger window



① : When Click '-', break point set / release

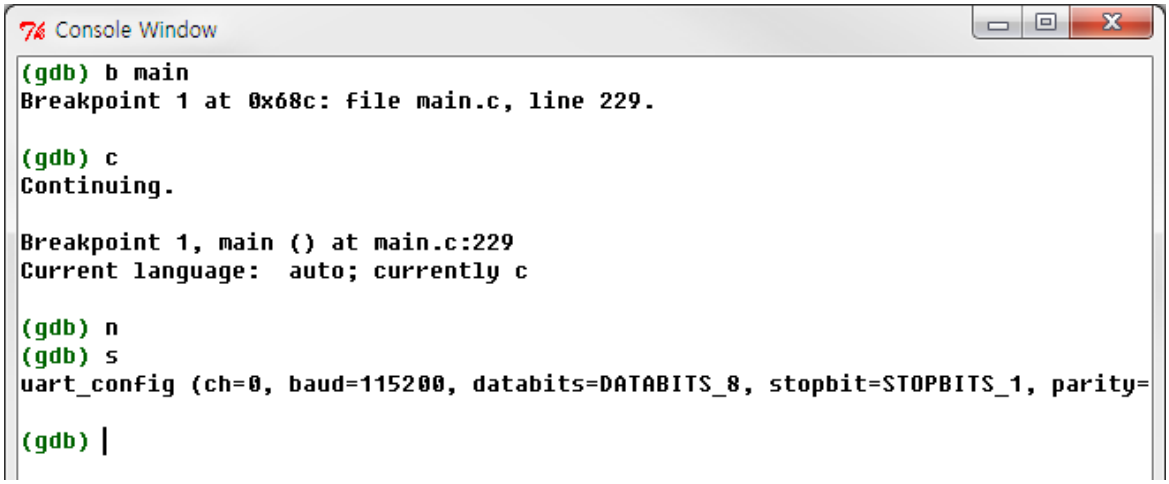
② : From the front, 'step', 'next', 'finish', 'continue'

③ : From the front, display 'register', 'memory', 'stack', 'watch expressions', 'local variable'. Last 'c: ' is that displaying 'console window.

Do debugging by using GDB command on console window.

④ : Move source code.

6) Debugging on console window.



b : Set break point.

c : Continue command. Run to next break point.

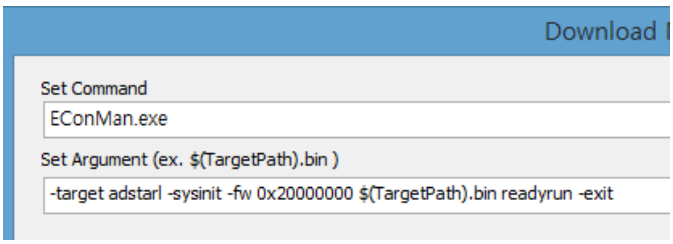
n : Next command. Run one code.

s : Step command. Run one code.

q : exit.

➔ Case that start program at Ram

1) After set debug mode (SW3 up), connect ECon and power on.
2) Run Build → Download Option and type argument like following.



-target adStar-L -sysinit -fw 0x20000000 $(TargetPath).bin readyrun -exit

3) After write download option, run Debug → Start Debugger (F5).
If the green code line is shown like following image, it means the connection is success. Start debugging like previous.