PsoC Eve Library

FTDI EVE display controller library for Cypress PSoC 4 and PsoC 5.

This Doc. version: 23/05/16

Index

1 Introduction.

The PSoC Eve Library is an open source library to use some FTDI graphics controllers chips (EVE) with Cypress PSoC 4 and PSoC 5 microcontrollers.

I have started the development of this library for my own purpouses and for some projects related with machine control and industrial control using small microcontrollers. Actually my needs are static graphics and not very complex animations, so i am not sure if i will provide support inside the library for complex sounds and new features found in newer FT chips like video playing. I want to do it but it will depend on time availability and my work on other projects.

Actually, the library its been developed using two dev kits from FTDI: one with FT800 chip and a 4,3” display; and the other with FT810 chip and a 5” display. Functions related to capacitive touch are not supported or not tested because i don´t own FT chips with capacitive touch capabilities. Also, you can see in library code, some values related to QVGA (320x240) display, but it is not tested. I don´t have QVGA displays.

English is not my native lenguaje, so probably you will find some or a lot of grammar mistakes in this document, but i will do my best to give enough useful information to make using the library as easy as possible.

At the time of writting this, the library is in active development and in a very early phase. Version is 0.1 alpha. During development, i will keep this document updated. If you want to compare this document with previous versions to learn about new features, corrected bugs, etc. i will update the date shown in the first page every time i make a change.

In section 5 (Library State) you can read about the actual state of every function in the library. This section, and ‘issues’ section in Github will give the most up to date information about the library.

About examples and hardware used you will find more information in examples section.

2 Using PSoC Eve Library.

2.1. SPI bus.

Using PSoC Eve Library, the conection between the PSoC micro and the FT chip (or dev kit) is made using the SPI bus.

Add a SPI master bus module to your PSoC project. Also, add a digital output pin to be used as the SS signal for the SPI bus. SS signal is managed by the library and not by PSoC Creator API's. An other digital output pin for the FT PD_N signal.

Actually, the library does not support FT chip interrupts.

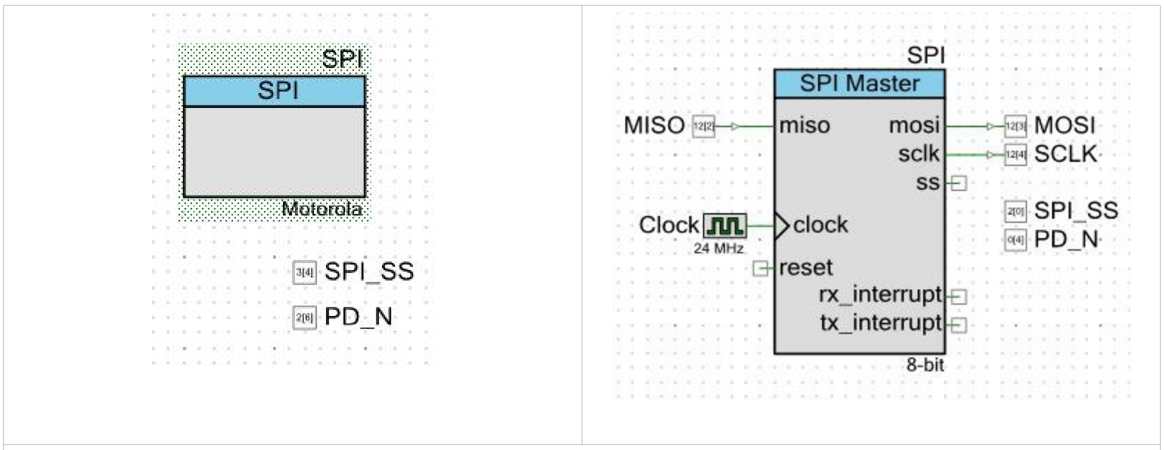
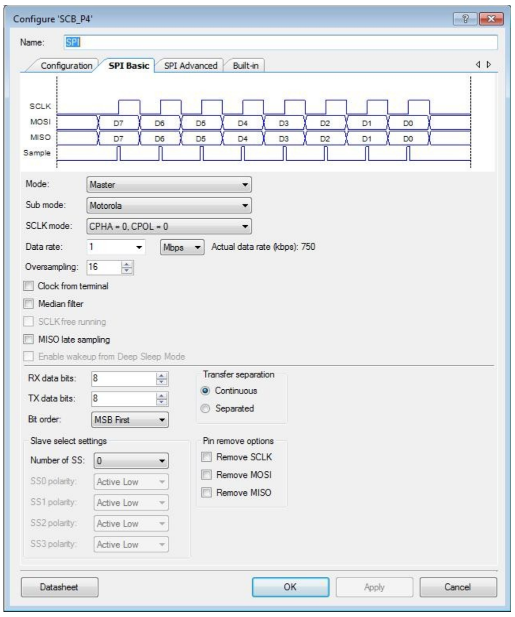


Fig.1 – SPI modules. Left = PSoC 4, right = PSoC 5.

Figure 2a and figure 2b shows my actual configuration settings for SPI modules during development. In PSoC 4 settings, 'Number of SS' option is 0, this way, we can reuse the same pin for our own purpouses.


Configure 'SCB_P4'

Name: SPI

Configuration | **SPI Basic** | SPI Advanced | Built-in

Mode: Master
Sub mode: Motorola
SCLK mode: CPHA = 0, CPOL = 0
Data rate: 1 Mbps Actual data rate (kbps): 750
Oversampling: 16
☐ Clock from terminal
☐ Median filter
☐ SCLK free running
☐ MISO late sampling
☐ Enable wakeup from Deep Sleep Mode

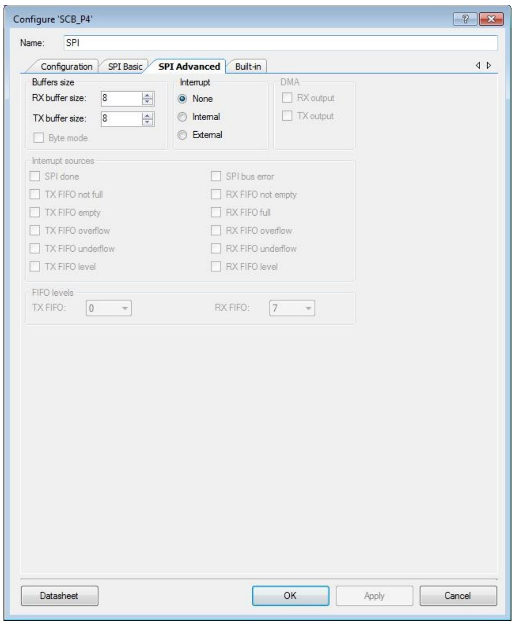
RX data bits: 8
TX data bits: 8
Bit order: MSB First

Transfer separation
● Continuous
○ Separated

Slave select settings
Number of SS: 0
SS0 polarity: Active Low
SS1 polarity: Active Low
SS2 polarity: Active Low
SS3 polarity: Active Low

Pin remove options
☐ Remove SCLK
☐ Remove MOSI
☐ Remove MISO

Datasheet | OK | Apply | Cancel


Configure 'SCB_P4'

Name: SPI

Configuration | SPI Basic | **SPI Advanced** | Built-in

Buffers size
RX buffer size: 8
TX buffer size: 8
☐ Byte mode

Interrupt
● None
○ Internal
○ External

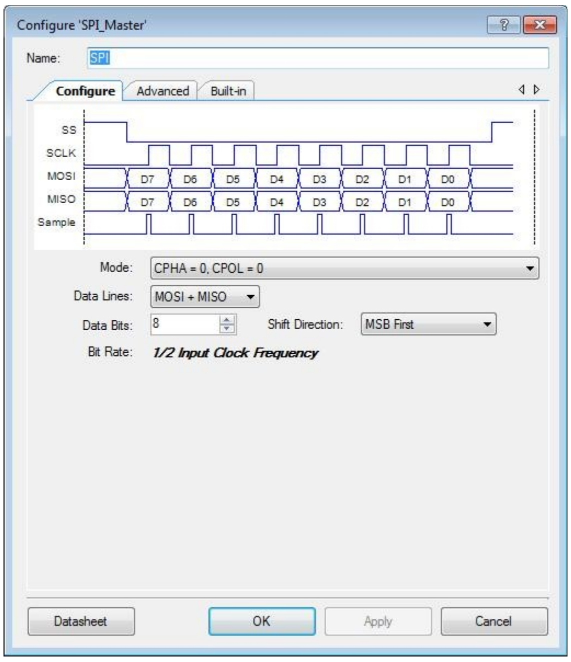
DMA
☐ RX output
☐ TX output

Interrupt sources
☐ SPI done
☐ TX FIFO not full
☐ TX FIFO empty
☐ TX FIFO overflow
☐ TX FIFO underflow
☐ TX FIFO level
☐ SPI bus error
☐ RX FIFO not empty
☐ RX FIFO full
☐ RX FIFO overflow
☐ RX FIFO underflow
☐ RX FIFO level

FIFO levels
TX FIFO: 0
RX FIFO: 7

Datasheet | OK | Apply | Cancel

Fig 2a. PSoC 4 SPI bus settings.

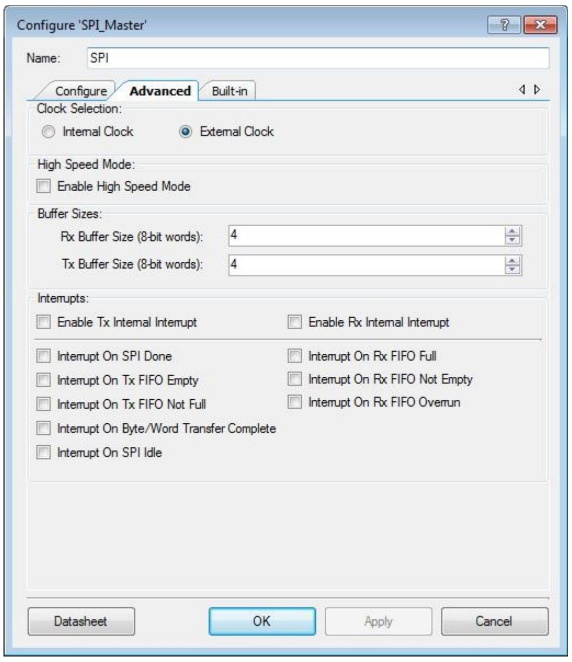

Configure 'SPI_Master'

Name: SPI

Configure | Advanced | Built-in

Mode: CPHA = 0, CPOL = 0
Data Lines: MOSI + MISO
Data Bits: 8
Shift Direction: MSB First
Bit Rate: *1/2 Input Clock Frequency*

Datasheet | OK | Apply | Cancel


Configure 'SPI_Master'

Name: SPI

Configure | **Advanced** | Built-in

Clock Selection:
○ Internal Clock ● External Clock

High Speed Mode:
☐ Enable High Speed Mode

Buffer Sizes:
Rx Buffer Size (8-bit words): 4
Tx Buffer Size (8-bit words): 4

Interrupts:
☐ Enable Tx Internal Interrupt
☐ Enable Rx Internal Interrupt
☐ Interrupt On SPI Done
☐ Interrupt On Tx FIFO Empty
☐ Interrupt On Tx FIFO Not Full
☐ Interrupt On Byte/Word Transfer Complete
☐ Interrupt On SPI Idle
☐ Interrupt On Rx FIFO Full
☐ Interrupt On Rx FIFO Not Empty
☐ Interrupt On Rx FIFO Overrun

Datasheet | OK | Apply | Cancel

Fig 2b. PSoC 5 SPI bus settings.

2.2 Adding the library.

The PsoC Eve Library is a usual library and not a module to be used in PsoC Creator.

For using the library, you only will need to add the library files to you own project in PSoC Creator and then add needed '#include" to your code.

Library files are:

- PSoCEve.c
- PSoCEve.h
- PSocEve_Hal.c
- PSoCEve_Hal.h
- PSoCEve_List.c
- PSocEve_List.h
- PSoCEve_TouchPanel.c
- PSoCEve_TouchPanel.h
- PSoCEve_Audio.c
- PSocEve_Audio.h
- PSoCEve_Config.h

Add these files to your project and then “#include” the file ‘PSoC_Eve.h’ in those files in wich you will use functions from the library.

2.3Library configuration.

The library gives support for FT800 and FT810 chips and also for different displays. You will have to configure the hardware in the PSoCEve_Config.h file.

2.3.1 Define SPI_NAME

Put here the same name you have given to the SPI module in the schematic. The library uses text substitution macros to avoid changing the name of every function related to SPI bus every time you want change the name of the module.

2.3.2 Define SPI_SS_NAME

In the same way, put here the name you have given in the schematic to the digital pin used for the SS signal.

2.3.3 FT Chip.

```
//#define EVE_FT800
#define EVE_FT810
```

Select to FT chip you will use. Uncomment the line of the selected chip, and comment the other.

2.3.4 Select the display.

The library gives support fot the displays used in FTDI development kits.
Uncomment the line of the selected display and comment the others.

- //#define LCD_QVGA   for display TFT 3,5” 320x240
- //#define LCD_WQVGA   for display TFT 4,3” 480x272

- #define LCD_WVVGA   for display TFT 5”   800x480

QVGA display values are not teste beacuase i don´t own one of these displays. Values where copied from FTDI EVE examples.

If you want to add another display, application note “ **AN_240 FT800 From the Ground Up” from FTDI gives you information about calculation of values needed to control it.**

2.3.5 Touch panel

Define “USE_TOUCHPANEL” lets you remove code related to the touch panel if you will not use one. You will not save to much memory, but… it there…

2.3.6 Audio

Define “USE_AUDIO” lets you remove code realted to audio.

2.3.7 GPIO Audio

Development kits from FTDI use the GPIO1 signal to enable/disable the sound amplifier chip. This enables the funcion MUTE and UNMUTE of the library.

If your own hardware doesn´t use the same configuration you can comment this line.

TODO: cointue documment. More info about starting developing with the library.

3 PsoC Eve Library Functions & Macros.

3.1 Library functions.

3.1.1 PsocEve.h

3.1.1.1 FT_Register_Write

void FT_Register_Write(uint32 everegister, uint32 value)	
Write value into selected register of FT chip.	
Parameters	everegister – FT register in wich to write value. value - value to be written inside register.
Returns	none
Don´t use this function inside a display list.	

3.1.1.2 FT_Register_Read

uint32 FT_Register_Read(uint32 everegister)	
Write value into selected register of FT chip.	
Parameters	everegister – register we want to read value from.
Returns	Value readed from register.
Don´t use this function inside a display list.	

3.1.1.3 FT_Init

uint8 FT_Init()	
Initalizes the FT chip.	
Parameters	none
Returns	Returns 1 if the initialization was succesful. Returns 0 if the initialization fails.

3.1.1.4 FT_Display_ON

void FT_Display_ON()	
Turns on the display.	
Parameters	none
Returns	none
This functions manages the DISP hardware signal (LCD Display Enable).	

3.1.1.5 FT_Display_OFF

void FT_Display_OFF()
Turns off the display.

Parameters	none
Returns	none
This functions manages the DISP hardware signal (LCD Display Enable).	

3.1.2 PsocEve_List.h

3.1.2.1 FT_ListStart

FTERROR FT_ListStart(TRANSFERTYPE transfertype)	
Start a new display list.	
Parameters	transfertype – type of transfer for the list. Values for 'transfertype': NONE, DLIST, DATA
Returns	OK – if there is no problem. LIST_IN_PROGRESS – if a list was started previously and has not been close.
This function starts a display list. It activates the SS line of SPI bus, so after it was used; no other functions that use the SPI bus can be used until the list is finished. SS line will be active until 'FT_ListEnd' function is called. As the library is in a very early stage and in active development, use only "DLIST" as transfertype. Other transfer types are for internal experiments and is possible they will be removed in future.	

3.1.2.2 FT_ListEnd

FTERROR FT_ListEnd(SWAPACTION swap)	
Finish an active display list.	
Parameters	swap – select if the display list will be swaped or not. Values for 'swap': END_DL_NOSWAP – doesn´t swap END_DL_SWAP – swap to make changes visible
Returns	OK – if there is no problem. LIST_NOT_IN_PROGRESS – if there is no list currently open.
This function finishes a display list. Usually, at the end of a display list you will want to swap the list so the changes are visible in the display. But realize, that there are some commands that swaps the list by themselves, like the SPINNER command. In these type of commands, use END_DL_NOSWAP to avoid problems.	

3.1.2.3 FTIsCoproccesorReady

uint8 FTIsCoproccesorReady()	
Check if the FT coproccesor is busy or ready to process new commands.	
Parameters	none
Returns	0 – if the coproccesor is busy processing previous commands. 1 – if the coproccesor is ready to proccess new commands.

3.1.2.4 FTGetCMDFifoFreeSpace

uint16 FTGetCMDFifoFreeSpace()	
Get the free space in the FT fifo.	
Parameters	none
Returns	Fifo free space.
Used internally. Usually you will not need to use this function directly.	

3.1.2.5 FT_Write_ByteArray_4

void FT_Write_ByteArray_4(const uint8 *data, uint32 length)	
Send an array of bytes to the FT chip. If array size is not multiple of four this funcion will send some extra bytes until total bytes sent is multiple of four.	
Parameters	* data – pointer to the array of data to be sent. length – size of the array.
Returns	none
For some commands that use extra data (ex: INFLATE command) it is mandatory that the quantity of bytes sent is multiple of four. This functions sends the data and if the size is not multiple of four it will send extra bytes until the length of data sent is multiple of four. This function does not manage the SS SPI bus line. It have to be used combined with ‘FT_Transfer_Start’ and ‘FT_Transfer_End’ functions.	

3.1.2.6 Experimental & in development functions.

Following functions are accesible from the library, but at the time of writing this, they are experimental and/or in development so they will be documented when they are ready to be used.

```
FTERROR FT_InflateFromFlash(const uint8 *flashptr, uint32 ramgptr,
uint32 size);
FTERROR FT_InflateFromExternalFlash(uint32 flashptr, uint32 ramgptr,
uint32 size);
FTERROR FT_LoadImageFromExternalFlash(uint32 flashptr, uint32 ramgptr,
uint32 size, uint16 options);
FTERROR FT_TransferToRAMG(uint32 flashptr, uint32 size);
```

3.1.3 PsocEve_Hal.h

3.1.3.1 FTCommandWrite

void FTCommandWrite(uint8 command)	
Send a command to the FT chip.	
Parameters	command – command to be sent. Look at #defines section to know wich commands are available.
Returns	none
This function manages SS line by itself, so don´t use it inside a display list.	

3.1.3.2 FT_Write_Byte

void FT_Write_Byte(uint32 address, uint8 data)	
Send a byte to the FT chip.	
Parameters	address – address where to send the byte. data – byte to be sent.
Returns	none
This function manages SS line by itself, so don´t use it inside a display list.	

3.1.3.3 FT_Read_Byte

uint8 FT_Read_Byte(uint32 address)	
Send a byte to the FT chip.	
Parameters	address – address from wich to read the byte.
Returns	Byte readed from FT chip.
This function manages SS line by itself, so don´t use it inside a display list.	

3.1.3.4 FT_Write_UINT32

void FT_Write_UINT32(uint32 address, uint32 data)	
Send uint32 to the FT chip.	
Parameters	address data
Returns	none
This function manages SS line by itself, so don´t use it inside a display list.	

3.1.3.5 FT_Read_UINT32

uint32 FT_Read_UINT32(uint32 address)	
Read anuint32from the FT chip.	
Parameters	address
Returns	Data readed from the chip.
This function manages SS line by itself, so don´t use it inside a display list.	

3.1.3.6 FT_Send_ByteArray_S

void FT_Send_ByteArray_S(const uint8 *data, uint32 size)	
Send an array of bytes to the FT chip.	
Parameters	data – pointer to array size – size of the array
Returns	none
This function manages SS line by itself, so don´t use it inside a display list.	

3.1.3.7 FT_Transfer_Start

void FT_Transfer_Start(uint32 address)	
Start a trasnfer of data to the FT chip.	
Parameters	address -
Returns	none
Used to start sending a block of data to the FT chip. For example to send a bitmap to RAM-G or data for the INFLATE command. This function activates the SS line of SPI bus. SS line will stay active until the ‘FT_Transfer_End’ command.	

3.1.3.8 FT_Transfer_End

void FT_Transfer_End()	
Finish a transfer of data.	
Parameters	none
Returns	none
Deactivates the SS line at the end of a transfer of data.	

3.1.3.9 FT_Send_Byte

void FT_Send_Byte(uint8 data)	
Send a byte to FT chip.	
Parameters	data
Returns	none
To be used with ‘FT_Transfer_Start’ & ‘FT_Transfer_End’.	

3.1.3.10 FT_Send_UINT32

void FT_Send_UINT32(uint32 data)	
Send a unit32 to FT chip.	
Parameters	data
Returns	none
To be used with ‘FT_Transfer_Start’ & ‘FT_Transfer_End’.	

3.1.3.11 FT_Send_ByteArray

void FT_Send_ByteArray(const uint8 *data, uint32 size)	
Send an array of bytes to the FT chip.	
Parameters	data
Returns	none
To be used with ‘FT_Transfer_Start’ & ‘FT_Transfer_End’.	

3.1.4 PsocEve_TouchPanel.h

3.1.4.1 FT_Touch_Enable

void FT_Touch_Enable()	
Enable the touch panel.	
Parameters	none
Returns	none

3.1.4.2 FT_Touch_Disable

void FT_Touch_Disable()	
Disable the touch panel.	
Parameters	none
Returns	none

3.1.4.3 FT_Touch_Calibrate()

void FT_Touch_Disable()	
Calibrate the touch panel.	
Parameters	none
Returns	none
Internally, this function starts a display list, sends the CMDCalibrate command to the FT whip and waits until calibration finish.	

3.1.4.4 FT_Touch_WaitCalibrationEnds

void FT_Touch_Disable()	
Wait until the calibration finish.	
Parameters	none
Returns	none
This function is intended to be used in your own calibration procedure. Look at the examples to know different ways of calibration.	

TODO: document here some examples about calibration.

3.1.4.5 FT_Touch_ReadCalibrationValues

void FT_Touch_ReadCalibrationValues(TouchCalibrationValues* values)	
Read calibration values from FT chip after calibration procedure.	
Parameters	**values – pointer to structure to store calibration values.**
Returns	none
TODO: document here how to read calibraton values.	

3.1.4.6 FT_Touch_WriteCalibrationValues

void FT_Touch_WriteCalibrationValues(TouchCalibrationValues* values)	
Write prevously stored calibration values to the FT chip.	
Parameters	**values – pointer to structure with calibration values.**
Returns	none
TODO: document here how to write calibraton values.	

3.1.5 PsocEve_Audio.h

3.1.5.1 FT_Audio_Mute

void FT_Audio_Mute()	
Write prevously stored calibration values to the FT chip.	
Parameters	none
Returns	none
Function accesible depending on '#define USE_GPIO1_AUDIO' in configuration header file. This function manages the signal used in FTDI development kits to enable/disable the audio amplifier.	

3.1.5.2 FT_Audio_Unmute

void FT_Audio_Unmute()	
Write prevously stored calibration values to the FT chip.	
Parameters	none
Returns	none
Function accesible depending on '#define USE_GPIO1_AUDIO' in configuration header file. This function manages the signal used in FTDI development kits to enable/disable the audio amplifier.	

3.1.5.3 FT_Sound_Volume

void FT_Sound_Volume(uint8 volume)	
Set volume	
Parameters	volume
Returns	none
Manages REG_VOL_SOUND register.	

3.1.5.4 FT_Sound_Play

void FT_Sound_Play(uint8 sound, uint8 pitch)	
Play sound.	
Parameters	sound pitch
Returns	none

Manages REG_SOUND & REG_PLAY registers.

3.1.5.5 FT_Sound_Stop

void FT_Sound_Stop()	
Stop sound.	
Parameters	none
Returns	none
Manages REG_SOUND & REG_PLAY registers.	

3.2 Display list functions.

3.2.1 BITMAP_SOURCE

Macro	`_DLBitmapSource(address)`
Function	`void DLBitmapSource(uint32 address)`
Notes	

3.2.2CLEAR_COLOR_RGB

Macro	`_DLClearColorRGB(red, green, blue)`
Function	`void DLClearColorRGB(uint8 red, uint8 green, uint8 blue)`
Notes	

3.2.3TAG

Macro	`_DLTag(s)`
Function	`void DLTag(uint8 tag)`
Notes	

3.2.4COLOR_RGB

Macro	`_DLColorRGB(red, green, blue)`
Function	`void DLColorRGB(uint8 red, uint8 green, uint8 blue)`
Notes	

3.2.5BITMAP_HANDLE

Macro	_DLBitmapHandle(handle)
Function	void DLBitmapHandle(uint8 handle)
Notes	

3.2.6CELL

Macro	_DLCell(cell)
Function	void DLCell(uint8 cell)
Notes	

3.2.7BITMAP_LAYOUT

Macro	_DLBitmapLayout(format, linestride, height)
Function	void DLBitmapLayout(uint8 format, uint16 linestride, uint16 height)
	Values for ‘format’: *** For FT800 & FT810 *** #define BITMAP_LAYOUT_ARGB1555 0 #define BITMAP_LAYOUT_L1 1 #define BITMAP_LAYOUT_L4 2 #define BITMAP_LAYOUT_L8 3 #define BITMAP_LAYOUT_RGB332 4 #define BITMAP_LAYOUT_ARGB2 5 #define BITMAP_LAYOUT_ARGB4 6 #define BITMAP_LAYOUT_RGB565 7 #define BITMAP_LAYOUT_TEXT8X8 9 #define BITMAP_LAYOUT_TEXTVGA 10 #define BITMAP_LAYOUT_BARGRAPH 11 *** Only for FT800 *** #define BITMAP_LAYOUT_PALETTED 8 *** Onlyt for FT810 *** #define BITMAP_LAYOUT_PALETTED565 14 #define BITMAP_LAYOUT_PALETTED4444 1 5 #define BITMAP_LAYOUT_PALETTED8 16 #define BITMAP_LAYOUT_L2 17
Notes	

3.2.8BITMAP_SIZE

Macro	_DLBitmapSize(filter, wrapx, wrapy, width, height)
Function	void DLBitmapSize(uint8 filter, uint8 wrapx, uint8 wrapy, uint16 width, uint16 height)

	Values for ‘filter’: #define BITMAP_SIZE_FILTER_NEAREST 0 #define BITMAP_SIZE_FILTER_BILINEAR 1 Values for ‘wrap’: #define BITMAP_SIZE_WRAP_BORDER 0 #define BITMAP_SIZE_WRAP_REPEAT 1
Notes	

3.2.9ALPHA_FUNC

Macro	_DLAlphaFunc(func, ref)
Function	void DLAlphaFunc(uint8 func, uint8 ref)
	Values for ‘func’: #define ALPHA_FUNC_NEVER 0 #define ALPHA_FUNC_LESS 1 #define ALPHA_FUNC_LEQUAL 2 #define ALPHA_FUNC_GREATER 3 #define ALPHA_FUNC_GEQUAL 4 #define ALPHA_FUNC_EQUAL 5 #define ALPHA_FUNC_NOTEQUAL 6 #define ALPHA_FUNC_ALLWAYS 7
Notes	

3.2.10STENCIL_FUNC

Macro	_DLStencilFunc(func, ref, mask)
Function	void DLStencilFunc(uint8 func, uint8 ref, uint8 mask)
	Values for ‘func’: #define STENCIL_FUNC_NEVER 0 #define STENCIL_FUNC_LESS 1 #define STENCIL_FUNC_LEQUAL 2 #define STENCIL_FUNC_GREATER 3 #define STENCIL_FUNC_GEQUAL 4 #define STENCIL_FUNC_EQUAL 5 #define STENCIL_FUNC_NOTEQUAL 6 #define STENCIL_FUNC_ALLWAYS 7
Notes	

3.2.11BLEND_FUNC

Macro	_DLBlendFunc(src, dst)
Function	void DLBlendFunc(uint8 src, uint8 dst)
	#define BLEND_FUNC_ZERO 0 #define BLEND_FUNC_ONE 1 #define BLEND_FUNC_SRC_ALPHA 2 #define BLEND_FUNC_DST_ALPHA 3 #define BLEND_FUNC_ONE_MINUS_SRC_ALPHA 4 #define BLEND_FUNC_ONE_MINUS_DST_ALPHA 5

Notes	

3.2.12STENCIL_OP

Macro	_DLStencilOp(sfail, spass)
Function	void DLStencilOp(uint8 sfail, uint8 spass)
	#define STENCIL_OP_ZERO 0 #define STENCIL_OP_KEEP 1 #define STENCIL_OP_REPLACE 2 #define STENCIL_OP_INCR 3 #define STENCIL_OP_DECR 4 #define STENCIL_OP_INVERT 5
Notes	

3.2.13POINT_SIZE

Macro	_DLPointSize(size)
Function	void DLPointSize(uint32 size)
Notes	

3.2.14LINE_WIDTH

Macro	_DLLineWidth(width)
Function	void DLLineWidth(uint16 width)
Notes	

3.2.15COLOR_A

Macro	_DLColorA(alpha)
Function	void DLColorA(uint8 alpha)
Notes	

3.2.16CLEAR_STENCIL

Macro	_DLClearStencil(s)
Function	void DLClearStencil(uint8 s)
Notes	

3.2.17CLEAR_TAG

Macro	_DLClearTag(t)
Function	void DLClearTag(uint8 t)

Notes	

3.2.18STENCIL_MASK

Macro	_DLStencilMask(mask)
Function	void DLStencilMask(uint8 mask)
Notes	

3.2.19TAG_MASK

Macro	_DLTagMask(mask)
Function	void DLTagMask(uint8 mask)
Notes	

3.2.20BITMAP_TRANSFORM_A

Macro	_DLBitmapTransformF(a)
Function	void DLBitmapTransformF(uint32 a)
Notes	

3.2.21BITMAP_TRANSFORM_B

Macro	_DLBitmapTransformF(b)
Function	void DLBitmapTransformF(uint32 b)
Notes	

3.2.22BITMAP_TRANSFORM_C

Macro	_DLBitmapTransformF(c)
Function	void DLBitmapTransformF(uint32 c)
Notes	

3.2.23BITMAP_TRANSFORM_D

Macro	_DLBitmapTransformF(d)
Function	void DLBitmapTransformF(uint32 d)
Notes	

3.2.24BITMAP_TRANSFORM_E

Macro	_DLBitmapTransformF(e)
Function	void DLBitmapTransformF(uint32 e)
Notes	

3.2.25BITMAP_TRANSFORM_F

Macro	_DLBitmapTransformF(f)
Function	void DLBitmapTransformF(uint32 f)
Notes	

3.2.26SCISSOR_XY

Macro	_DLScissorXY(x, y)
Function	void DLScissorXY(uint16 x, uint16 y)
Notes	

3.2.27SCISSOR_SIZE

Macro	_DLScissorSize(width, height)
Function	void DLScissorSize(uint16 width, uint16 height)
Notes	

3.2.28JUMP

Macro	_DLJump(address)
Function	void DLJump(uint16 address)
Notes	

3.2.29BEGIN

Macro	_DLBegin(primitive)
Function	void DLBegin(uint8 primitive)
	Values for ‘primitive’: #define PRIMITIVE_BITMAP 1 #define PRIMITIVE_POINT 2 #define PRIMITIVE_LINE 3 #define PRIMITIVE_LINE_STRIP 4 #define PRIMITIVE_EDGE_STRIP_R 5 #define PRIMITIVE_EDGE_STRIP_L 6

	#define PRIMITIVE_EDGE_STRIP_A 7 #define PRIMITIVE_EDGE_STRIP_B 8 #define PRIMITIVE_RECTANGLE 9
Notes	

3.2.30CALL

Macro	_DLCall(address)
Function	void DLCall(uint16 address)
Notes	

3.2.31COLOR_MASK

Macro	_DLColorMask(r, g, b, a)
Function	void DLColorMask(uint8 r, uint8 g, uint8 b, uint8 a)
Notes	

3.2.32END

Macro	_DLEnd()
Function	void DLEnd()
Notes	

3.2.33SAVE_CONTEXT

Macro	_DLSaveContext()
Function	void DLSaveContext()
Notes	

3.2.34RESTORE_CONTEXT

Macro	_DLRestoreContext()
Function	void DLRestoreContext()
Notes	

3.2.35RETURN

Macro	_DLReturn()
Function	void DLReturn()

Notes	

3.2.36MACRO

Macro	_DLMacro(macro)
Function	void DLMacro(uint8 macro)
Notes	

3.2.37CLEAR

Macro	_DLClear(color, stencil, tag)
Function	void DLClear(uint8 color, uint8 stencil, uint8 tag)
	#define CLEAR_TAG 0x01 #define CLEAR_STENCIL 0x02 #define CLEAR_COLOR 0x04
Notes	

3.2.38VERTEX2F

Macro	_DLVertex2F(x, y)
Function	void DLVertex2F(int16 x, int16 y)
Notes	

3.2.39VERTEX2II

Macro	_DLVertex2II(x, y, handle, cell)
Function	void DLVertex2II(uint16 x, uint16 y, uint8 handle, uint8 cell)
Notes	

3.2.40VERTEX_FORMAT

Macro	_DLVertexFormat(format)
Function	void DLVertexFormat(uint8 format)
	Values for ‘format’: #define VERTEX_FORMAT_1 0 #define VERTEX_FORMAT_1_2 1 #define VERTEX_FORMAT_1_4 2 #define VERTEX_FORMAT_1_8 3 #define VERTEX_FORMAT_1_16 4
Notes	New command in FT81x This is a new command supported internally by FT81x chips but not by FT800. You can use it anyways. It is implemented in software for FT800 chip.

3.2.41BITMAP_LAYOUT_H

Macro	_DLBitmapLayout_H(linestride, height)
Function	
Notes	New command in FT81x not supported by FT800 Not to be used directly. It is managed by the library when configured for FT810

3.2.42BITMAP_SIZE_H

Macro	_DLBitmapSize_H(width, height)
Function	
Notes	New command in FT81x not supported by FT800 Not to be used directly. It is managed by the library when configured for FT810

3.2.43PALETTE_SOURCE

Macro	_DLPaletteSource(address)
Function	void DLPaletteSource(uint32 address)
Notes	New command in FT81x not supported by FT800

3.2.44VERTEX_TRANSLATE_X

Macro	_DLVertexTranslateX(translation)
Function	void DLVertexTranslateX(int32 translation)
Notes	New command in FT81x not supported by FT800

3.2.45VERTEX_TRANSLATE_Y

Macro	_DLVertexTranslateY(translation)
Function	void DLVertexTranslateY(int32 translation)
Notes	New command in FT81x not supported by FT800

3.2.46NOP

Macro	_ NOP
Function	void DLNop()
Notes	New command in FT81x not supported by FT800

3.3 Coprocessor functions.

3.3.1 #Defines.

These are some ‘options’ defines for those commands that use ‘options’ parameter.

```
#define OPT_3D              0
#define OPT_RGB565          0
#define OPT_MONO            1
#define OPT_NODL            2
#define OPT_FLAT            256
#define OPT_SIGNED          256
#define OPT_CENTERX         512
#define OPT_CENTERY         1024
#define OPT_CENTER          1536
#define OPT_RIGHTX          2048
#define OPT_NOBACK          4096
#define OPT_NOTICKS         8192
#define OPT_NOHM            16384
#define OPT_NOPOINTER       16384
#define OPT_NOSECS          32768
#define OPT_NOHANDS         49152
```

3.3.2CMD_DLSTART

Macro	_CMDDLStart()
Function	void CMDDLStart()
Notes	Usually you will not need to use this. It is used internally by library functions.

3.3.3CMD_SWAP

Macro	_CMDSwap
Function	void CMD Swap ()
Notes	Usually you will not need to use this. It is used internally by library functions.

3.3.4CMD_SWAP

Macro	_CMDInterrupt(milliseconds)
Function	void CMDInterrupt(uint32 milliseconds)
Notes	

3.3.5CMD_BGCOLOR

Macro	_CMDBgcolor(red, green, blue)

Function	void CMDBgcolor(int8 red, int8 green, int8 blue)
Notes	

3.3.6CMD_FGCOLOR

Macro	_CMD F gcolor(red, green, blue)
Function	void CMD F gcolor(int8 red, int8 green, int8 blue)
Notes	

3.3.7CMD_GRADIENT

Macro	_CMDGradient(x0, y0, red0, green0, blue0, x1, y1, red1, green1, blue1)
Function	void CMDGradient(int16 x0, int16 y0, int8 red0, int8 green0, int8 blue0, int16 x1, int16 y1, int8 red1, int8 green1, int8 blue1)
Notes	

3.3.8CMD_TEXT

Macro	_CMDTextNew(x, y, font, options, text)
Function	void CMDText(int16 x, int16 y, int16 font, int16 options, char* text)
Notes	

3.3.9CMD_BUTTON

Macro	_CMDButton(x, y, width, height, font, options, text)
Function	void CMDButton(int16 x, int16 y, int16 width, int16 height, int16 font, int16 options, char* text)
Notes	

3.3.10CMD_KEYS

Macro	_CMDKeys(x, y, width, height, font, options, text)
Function	void CMDKeys(int16 x, int16 y, int16 width, int16 height, int16 font, int16 options, char* text)
Notes	

3.3.11CMD_PROGRESS

Macro	_CMDProgressBar(x, y, width, height, options, value, range)

Function	void CMDProgressBar(int16 x, int16 y, int16 width, int16 height, int16 options, int16 value, int16 range)
Notes	

3.3.12CMD_SLIDER

Macro	_CMDSlider(x, y, width, height, options, value, range)
Function	void CMDSlider(int16 x, int16 y, int16 width, int16 height, int16 options, int16 value, int16 range)
Notes	

3.3.13CMD_SCROLLBAR

Macro	`_CMDScrollBar(x, y, width, height, options, value, size, range)`
Function	`void CMDScrollBar(int16 x, int16 y, int16 width, int16 height, int16 options, int16 value, int16 size, int16 range)`
Notes	

3.3.14CMD_TOGGLE

Macro	_CMDToggle(x, y, width, font, options, state, text)
Function	void CMDToggle(int16 x, int16 y, int16 width, int16 font, int16 options, int16 state, char* text)
	Values for ‘state’: #define TOGGLE_STATE_OFF 0 #define TOGGLE_STATE_ON 65535
Notes	

3.3.15CMD_GAUGE

Macro	_CMDGauge(x, y, radius, options, major, minor, value, range)
Function	void CMDGauge(int16 x, int16 y, int16 radius, int16 options, int16 major, int16 minor, int16 value, int16 range)
Notes	

3.3.16CMD_CLOCK

Macro	_CMDClock(x, y, radius, options, hours, minutes, seconds, milliseconds)
Function	void CMDClock(int16 x, int16 y, int16 radius, int16 options, int16 hours, int16 minutes, int16 seconds, int16 milliseconds)

Notes	

3.3.17CMD_CALIBRATE

Macro	_CMDCalibrate()
Function	void CMDCalibrate()
Notes	

3.3.18CMD_SPINNER

Macro	_CMDSpinner(x, y, style, scale)
Function	void CMDSpinner(int16 x, int16 y, int16 style, int16 scale)
	Values for ‘style’: #define SPINNER_ROUND 0 #define SPINNER_LINEAR 1 #define SPINNER_CLOCK 2 #define SPINNER_ORBIT 3
Notes	

3.3.19CMD_STOP

Macro	_CMDStop()
Function	void CMDStop()
Notes	

3.3.20CMD_MEMCRC

Macro	
Function	
Notes	

3.3.21CMD_REGREAD

Macro	
Function	
Notes	

3.3.22CMD_MEMWRITE

Macro	
Function	

Notes	

3.3.23CMD_MEMSET

Macro	_CMDMemSet(ptr, value, size)
Function	void CMDMemSet(int32 ptr, int32 value, int32 size)
Notes	

3.3.24CMD_MEMZERO

Macro	_CMDMemZero(ptr, size)
Function	void CMDMemZero(int32 ptr, int32 size)
Notes	

3.3.25CMD_MEMCPY

Macro	_CMDMemCopy(dest, src, size)
Function	void CMDMemCopy(int32 dest, int32 src, int32 size)
Notes	

3.3.26CMD_APPEND

Macro	`_CMDAppend(ptr, size)`
Function	`void CMDAppend(int32 ptr, int32 size)`
Notes	

3.3.27CMD_SNAPSHOT

Macro	_CMDSnapshot(ptr)
Function	void CMDSnapshot(int32 ptr)
Notes	

3.3.28CMD_INFLATE

Macro	_CMDInflate(ptr)
Function	void CMDInflate(int32 ptr)
Notes	

3.3.29CMD_GETPTR

Macro	_CMDGETPTR(dummy)
Function	int32 CMDGetPtr()
Notes	Although this is a list command, don´t use it inside a list. In PsoC Eve Library, this command manages the starting and ending of a display list by itself.

3.3.30CMD_LOADIMAGE

Macro	_CMDLoadImage(ptr, options)
Function	void CMDLoadImage(int32 ptr, int32 options)
Notes	

3.3.31CMD_LOADIDENTITY

Macro	_CMDLoadIdentity()
Function	void CMDLoadIdentity()
Notes	

3.3.32CMD_TRANSLATE

Macro	_CMDTranslate(tx, ty)
Function	void CMDTranslate(int32 tx, int32 ty)
Notes	

3.3.33CMD_SCALE

Macro	_CMDScale(x, y)
Function	void CMDScale(int32 x, int32 y)
Notes	

3.3.34CMD_ROTATE

Macro	_CMDRotate(angle)
Function	void CMDRotate(int32 angle)
Notes	

3.3.35CMD_SETMATRIX

Macro	_CMDSetmatrix()
Function	void CMDSetmatrix()
Notes	

3.3.36CMD_SETFONT

Macro	_CMDSetfont(font, ptr)
Function	void CMDSetfont(int32 font, int32 ptr)
Notes	

3.3.37CMD_TRACK

Macro	_CMDTrack(x, y, width, height, tag)
Function	void CMDTrack(int16 x, int16 y, int16 width, int16 height, int16 tag)
Notes	

3.3.38CMD_DIAL

Macro	_CMDDial(x, y, radius, options, value)
Function	void CMDDial(int16 x, int16 y, int16 radius, int16 options, int16 value)
Notes	

3.3.39CMD_NUMBER

Macro	_CMDNumber(x, y, font, options, number)
Function	void CMDNumber(int16 x, int16 y, int16 font, int16 options, int32 number)
Notes	

3.3.40CMD_SCREENSAVER

Macro	_CMDScreenSaver()
Function	void CMDScreenSaver()
Notes	

3.3.41CMD_SKETCH

Macro	_CMDSketch(x, y, width, height, ptr, format)

Function	void CMDSketch(int16 x, int16 y, int16 width, int16 height, int32 ptr, int16 format)
Notes	

3.3.42CMD_LOGO

Macro	_CMDLogo()
Function	void CMDLogo()
Notes	

3.3.43CMD_COLDSTART

Macro	_CMDColdstart()
Function	void CMDColdstart()
Notes	

3.3.44CMD_GETMATRIX

Macro	
Function	
Notes	

3.3.45CMD_GRADCOLOR

Macro	`_CMDGradcolor(red, green, blue)`
Function	`void CMDGradcolor(int8 red, int8 green, int8 blue)`
Notes	

3.3.46CMD_CSKETCH

Macro	
Function	
Notes	New command in FT81x not supported by FT800

3.3.47CMD_SETROTATE

Macro	
Function	
Notes	New command in FT81x not supported by FT800

3.3.48CMD_SNAPSHOT2

Macro	
Function	
Notes	New command in FT81x not supported by FT800

3.3.49CMD_SETBASE

Macro	_CMDSETBASE(base)
Function	void CMDSetBase(int32 base)
	Values for ‘base’: #define NUMBER_BASE_BINARY 2 #define NUMBER_BASE_OCTAL 8 #define NUMBER_BASE_DECIMAL 10 #define NUMBER_BASE_HEXADECIMAL 16
Notes	New command in FT81x not supported by FT800

3.3.50CMD_MEDIAFIFO

Macro	
Function	
Notes	New command in FT81x not supported by FT800

3.3.51CMD_PLAYVIDEO

Macro	
Function	
Notes	New command in FT81x not supported by FT800

3.3.52CMD_SETFONT2

Macro	
Function	
Notes	New command in FT81x not supported by FT800

3.3.53CMD_SETSCRATCH

Macro	
Function	

Notes	New command in FT81x not supported by FT800

3.3.54CMD_ROMFONT

Macro	_CMDROMFONT(handle, font)
Function	void CMDRomfont(int32 font, int32 handle)
Notes	New command in FT81x not supported by FT800

3.3.55CMD_VIDEOSTART

Macro	
Function	
Notes	New command in FT81x not supported by FT800

3.3.56CMD_VIDEOFRAME

Macro	
Function	
Notes	New command in FT81x not supported by FT800

3.3.57CMD_SETBITMAP

Macro	`_CMDSetBitmap(address, format, width, height)`
Function	`void CMDSetBitmap(int32 address, int16 format, int16 width, int16 height)`
Notes	New command in FT81x This is a new command supported internally by FT81x chips but not by FT800. You can use it anyways. It is implemented in software for FT800 chip.

4 #Defines

4.1 EVE chip memory map.

4.1.1Definition of memory map for FT800 (#if defined EVE_FT800)

```
#define RAM_G               0x000000   // Main graphics RAM.
#define ROM_CHIPID          0x0C0000   // Chip ID and revision.
#define ROM_FONT            0x0BB23C   // Font table and bitmaps.
#define ROM_FONT_ADDR       0x0FFFFC   // Font table pointer address,
#define RAM_DL              0x100000   // Display list RAM.
#define RAM_PAL             0x102000   // Palette RAM.
#define RAM_REG             0x102400   // Registers.
#define RAM_CMD             0x108000   // Coprocessor command buffer.
```

4.1. 2 Definition of m emory map for FT810 (#if defined EVE_FT810)

```
#define RAM_G               0x000000   // Main graphics RAM.
#define ROM_CHIPID          0x0C0000   // Chip ID and revision.
#define ROM_FONT            0x1E0000
#define ROM_FONT_ADDR       0x2FFFFC   // Font table pointer address,
#define RAM_DL              0x300000   // Display list RAM.
#define RAM_REG             0x302000   // Registers.
#define RAM_CMD             0x308000   // Coprocessor command buffer.
```

4.2 EVE chip registers.

4.2.1 Registers definition for FT800 (#if defined EVE_FT800)

```
#define REG_ID                      0x102400
#define REG_FRAMES                  0x102404
#define REG_CLOCK                   0x102408
#define REG_FREQUENCY               0x10240c
#define REG_RENDERMODE              0x102410
#define REG_SNAPY                   0x102414
#define REG_SNAPSHOT                0x102418
#define REG_CPURESET                0x10241c
#define REG_TAP_CRC                 0x102420
#define REG_TAP_MASK                0x102424
#define REG_HCYCLE                  0x102428
#define REG_HOFFSET                 0x10242c
#define REG_HSIZE                   0x102430
#define REG_HSYNC0                  0x102434
#define REG_HSYNC1                  0x102438
#define REG_VCYCLE                  0x10243c
#define REG_VOFFSET                 0x102440
#define REG_VSIZE                   0x102444
#define REG_VSYNC0                  0x102448
#define REG_VSYNC1                  0x10244c
#define REG_DLSWAP                  0x102450
```

```
#define REG_ROTATE                    0x102454
#define REG_OUTBITS                   0x102458
#define REG_DITHER                    0x10245c
#define REG_SWIZZLE                   0x102460
#define REG_CSPREAD                   0x102464
#define REG_PCLK_POL                  0x102468
#define REG_PCLK                      0x10246c
#define REG_TAG_X                     0x102470
#define REG_TAG_Y                     0x102474
#define REG_TAG                       0x102478
#define REG_VOL_PB                    0x10247c
#define REG_VOL_SOUND                 0x102480
#define REG_SOUND                     0x102484
#define REG_PLAY                      0x102488
#define REG_GPIO_DIR                  0x10248c
#define REG_GPIO                      0x102490
#define REG_INT_FLAGS                 0x102498
#define REG_INT_EN                    0x10249c
#define REG_INT_MASK                  0x1024a0
#define REG_PLAYBACK_START            0x1024a4
#define REG_PLAYBACK_LENGTH           0x1024a8
#define REG_PLAYBACK_READPTR          0x1024ac
#define REG_PLAYBACK_FREQ             0x1024b0
#define REG_PLAYBACK_FORMAT           0x1024b4
#define REG_PLAYBACK_LOOP             0x1024b8
#define REG_PLAYBACK_PLAY             0x1024bc
#define REG_PWM_HZ                    0x1024c0
#define REG_PWM_DUTY                  0x1024c4
#define REG_MACRO_0                   0x1024c8
#define REG_MACRO_1                   0x1024cc
#define REG_CMD_READ                  0x1024e4
#define REG_CMD_WRITE                 0x1024e8
#define REG_CMD_DL                    0x1024ec
#define REG_TOUCH_MODE                0x1024f0
#define REG_TOUCH_ADC_MODE            0x1024f4
#define REG_TOUCH_CHARGE              0x1024f8
#define REG_TOUCH_SETTLE              0x1024fc
#define REG_TOUCH_OVERSAMPLE          0x102500
#define REG_TOUCH_RZTHRESH            0x102504
#define REG_TOUCH_RAW_XY              0x102508
#define REG_TOUCH_RZ                  0x10250c
#define REG_TOUCH_SCREEN_XY           0x102510
#define REG_TOUCH_TAG_XY              0x102514
#define REG_TOUCH_TAG                 0x102518
#define REG_TOUCH_TRANSFORM_A         0x10251c
#define REG_TOUCH_TRANSFORM_B         0x102520
#define REG_TOUCH_TRANSFORM_C         0x102524
#define REG_TOUCH_TRANSFORM_D         0x102528
#define REG_TOUCH_TRANSFORM_E         0x10252c
#define REG_TOUCH_TRANSFORM_F         0x102530
#define REG_TOUCH_DIRECT_XY           0x102574
#define REG_TOUCH_DIRECT_Z1Z2         0x102578
```

```
#define REG_TRACKER                      0x109000
```

4. 2 . 2 Registers definition for FT810 (#if defined EVE_FT810)

```
#define REG_ID                           0x302000
#define REG_FRAMES                       0x302004
#define REG_CLOCK                        0x302008
#define REG_FREQUENCY                    0x30200C
#define REG_RENDERMODE                   0x302010
#define REG_SNAPY                        0x302014
#define REG_SNAPSHOT                     0x302018
#define REG_SNAPFORMAT                   0x30201C
#define REG_CPURESET                     0x302020
#define REG_TAP_CRC                      0x302024
#define REG_TAP_MASK                     0x302028
#define REG_HCYCLE                       0x30202C
#define REG_HOFFSET                      0x302030
#define REG_HSIZE                        0x302034
#define REG_HSYNC0                       0x302038
#define REG_HSYNC1                       0x30203C
#define REG_VCYCLE                       0x302040
#define REG_VOFFSET                      0x302044
#define REG_VSIZE                        0x302048
#define REG_VSYNC0                       0x30204C
#define REG_VSYNC1                       0x302050
#define REG_DLSWAP                       0x302054
#define REG_ROTATE                       0x302058
#define REG_OUTBITS                      0x30205C
#define REG_DITHER                       0x302060
#define REG_SWIZZLE                      0x302064
#define REG_CSPREAD                      0x302068
#define REG_PCLK_POL                     0x30206C
#define REG_PCLK                         0x302070
#define REG_TAG_X                        0x302074
#define REG_TAG_Y                        0x302078
#define REG_TAG                          0x30207C
#define REG_VOL_PB                       0x302080
#define REG_VOL_SOUND                    0x302084
#define REG_SOUND                        0x302088
#define REG_PLAY                         0x30208C
#define REG_GPIO_DIR                     0x302090
#define REG_GPIO                         0x302094
#define REG_GPIOX_DIR                    0x302098
#define REG_GPIOX                        0x30209C
#define REG_INT_FLAGS                    0x3020A8
#define REG_INT_EN                       0x3020AC
#define REG_INT_MASK                     0x3020B0
#define REG_PLAYBACK_START               0x3020B4
#define REG_PLAYBACK_LENGTH              0x3020B8
#define REG_PLAYBACK_READPTR             0x3020BC
#define REG_PLAYBACK_FREQ                0x3020C0
#define REG_PLAYBACK_FORMAT              0x3020C4
```

```
#define REG_PLAYBACK_LOOP                 0x3020C8
#define REG_PLAYBACK_PLAY                 0x3020CC
#define REG_PWM_HZ                        0x3020D0
#define REG_PWM_DUTY                      0x3020D4
#define REG_MACRO_0                       0x3020D8
#define REG_MACRO_1                       0x3020DC
#define REG_CMD_READ                      0x3020F8
#define REG_CMD_WRITE                     0x3020FC
#define REG_CMD_DL                        0x302100
#define REG_TOUCH_MODE                    0x302104
#define REG_TOUCH_ADC_MODE                0x302108
#define REG_TOUCH_CHARGE                  0x30210C
#define REG_TOUCH_SETTLE                  0x302110
#define REG_TOUCH_OVERSAMPLE              0x302114
#define REG_TOUCH_RZTHRESH                0x302118
#define REG_TOUCH_RAW_XY                  0x30211C
#define REG_TOUCH_RZ                      0x302120
#define REG_TOUCH_SCREEN_XY               0x302124
#define REG_TOUCH_TAG_XY                  0x302128
#define REG_TOUCH_TAG                     0x30212C
#define REG_TOUCH_TAG1_XY                 0x302130
#define REG_TOUCH_TAG1                    0x302134
#define REG_TOUCH_TAG2_XY                 0x302138
#define REG_TOUCH_TAG2                    0x30213C
#define REG_TOUCH_TAG3_XY                 0x302140
#define REG_TOUCH_TAG3                    0x302144
#define REG_TOUCH_TAG4_XY                 0x302148
#define REG_TOUCH_TAG4                    0x30214C
#define REG_TOUCH_TRANSFORM_A             0x302150
#define REG_TOUCH_TRANSFORM_B             0x302154
#define REG_TOUCH_TRANSFORM_C             0x302158
#define REG_TOUCH_TRANSFORM_D             0x30215C
#define REG_TOUCH_TRANSFORM_E             0x302160
#define REG_TOUCH_TRANSFORM_F             0x302164
#define REG_TOUCH_CONFIG                  0x302168
#define REG_CTOUCH_TOUCH4_X               0x30216C
#define REG_BIST_EN                       0x302174
#define REG_TRIM                          0x302180
#define REG_ANA_COMP                      0x302184
#define REG_SPI_WIDTH                     0x302188
#define REG_TOUCH_DIRECT_XY               0x30218C
#define REG_TOUCH_DIRECT_Z1Z2             0x302190
#define REG_DATESTAMP                     0x302564
#define REG_CMDB_SPACE                    0x302574
#define REG_CMDB_WRITE                    0x302578
#define REG_TRACKER                       0x309000
#define REG_TRACKER_1                     0x309004
#define REG_TRACKER_2                     0x309008
#define REG_TRACKER_3                     0x30900C
#define REG_TRACKER_4                     0x309010
```

5 Library State.

- W hite cells = command not supported internally by the chip.
- Ligh-Green cells = command supported by the chip.
- Dark-Green cells = command not supported by the chip but implemented in software.

(Look at features_text.pdf file)

5.1 Display list commands.

5.2 Coproccessor commands.