### Ambarella A7 Hardware Abstraction Layer

Generated by Doxygen 1.7.6.1

Mon Aug 5 2013 23:32:20

# **Contents**

1	Ambarella A7S Hardware Abstraction Layer					1
2	HAL Loading, Initialization & Usage					
3	Cha	nging C	perating I	Mode		5
	3.1	Operat	ting Mode	Settings		5
4	Cha	nging P	LL Freque	ency		9
5	Cha	nging C	lock Sour	ces		11
6	Mod	ule Inde	ex			13
	6.1	Module	es			13
7	Data	Struct	ure Index			15
	7.1					15
8	Mod	ule Doc	umentatio	on		17
	8.1	Initializ	ation			17
		8.1.1	Detailed	Description		17
		8.1.2	Enumera	tion Type Documentation		18
			8.1.2.1	amb_dram_arbiter_priority_t		18
		8.1.3	Function	Documentation		18
			8.1.3.1	amb_hal_init		18
			8.1.3.2	amb_get_chip_name		19
			8.1.3.3	amb_get_version		19
			8.1.3.4	amb_set_peripherals_base_address		20
			8.1.3.5	amb_set_dram_arbiter_priority		20

ii CONTENTS

8.2		Command Status				
		8.2.1	Detailed [	Description	21	
		8.2.2	Enumerat	tion Type Documentation	21	
			8.2.2.1	$amb\_hal\_success\_t \ \dots \ \dots \ \dots \ \dots$	21	
	8.3	Operat	ing Mode		22	
		8.3.1	Detailed [	Description	23	
		8.3.2	Enumerat	tion Type Documentation	23	
			8.3.2.1	$amb\_vidcap\_window\_size\_t\ .\ .\ .\ .\ .\ .\ .\ .\ .\ .$	23	
			8.3.2.2	$amb\_performance\_t \ \dots \ \dots \ \dots \ \dots \ \dots$	23	
			8.3.2.3	$amb\_mode\_t \ . \ . \ . \ . \ . \ . \ . \ . \ . \ $	24	
			8.3.2.4	$amb\_dual\_stream\_state\_t\ .\ .\ .\ .\ .\ .\ .\ .\ .\ .$	24	
			8.3.2.5	$amb\_digital\_gamma\_mode\_t \ \dots \dots \dots \dots$	24	
		8.3.3	Function	Documentation	24	
			8.3.3.1	amb_set_operating_mode	24	
			8.3.3.2	amb_get_operating_mode	25	
			8.3.3.3	amb_get_operating_mode_status	25	
		8.3.4	Variable [	Documentation	26	
			8.3.4.1	$amb\_idsp\_frequency\_parameters\_table \ . \ . \ . \ . \ .$	26	
	8.4	System	n Configura	ation	27	
		8.4.1	Detailed [	Description	27	
		8.4.2	Enumerat	tion Type Documentation	27	
			8.4.2.1	$amb\_system\_configuration\_t  .  .  .  .  .  .$	27	
			8.4.2.2	amb_boot_type_t	28	
			8.4.2.3	$amb\_hif\_type\_t \dots \dots$	28	
		8.4.3	Function	Documentation	29	
			8.4.3.1	$amb\_get\_reference\_clock\_frequency \ . \ . \ . \ . \ . \ .$	29	
			8.4.3.2	amb_get_system_configuration	29	
			8.4.3.3	amb_get_boot_type	29	
			8.4.3.4	amb_get_hif_type	30	
	8.5	Divided	d Clocks .		31	
		8.5.1	Detailed [	Description	31	
	8.6	PLL .			32	
		8.6.1	Detailed [	Description	32	
		8.6.2	Typedef D	Occumentation	32	

CONTENTS iii

		8.6.2.1 amb_pll_fractional_divisor_t
	8.6.3	Enumeration Type Documentation
		8.6.3.1 amb_clock_source_t
	8.6.4	Function Documentation
		8.6.4.1 amb_disable_clock_observation
8.7	ADC .	
	8.7.1	Detailed Description
	8.7.2	Function Documentation
		8.7.2.1 amb_set_adc_clock_frequency
		8.7.2.2 amb_get_adc_clock_frequency
		8.7.2.3 amb_get_adc_clock_configuration
8.8	ARM .	
	8.8.1	Detailed Description
	8.8.2	Function Documentation
		8.8.2.1 amb_get_arm_clock_frequency
8.9	AHB .	
	8.9.1	Detailed Description
	8.9.2	Function Documentation
		8.9.2.1 amb_get_ahb_clock_frequency
8.10	APB .	
	8.10.1	Detailed Description
	8.10.2	Function Documentation
		8.10.2.1 amb_get_apb_clock_frequency 39
8.11	Audio	
	8.11.1	Detailed Description
	8.11.2	Function Documentation
		8.11.2.1 amb_set_audio_clock_source 40
		8.11.2.2 amb_set_audio_clock_frequency 41
		8.11.2.3 amb_get_audio_pll_configuration 41
		8.11.2.4 amb_get_audio_clock_frequency 42
		8.11.2.5 amb_get_audio_pll_lock_status 42
		8.11.2.6 amb_enable_audio_clock_observation 42
8.12	Audio 1	CH
	8.12.1	Detailed Description

iv CONTENTS

	8.12.2	Function Documentation	14
		8.12.2.1 amb_set_au1ch_clock_frequency	14
		8.12.2.2 amb_get_au1ch_clock_frequency	14
		8.12.2.3 amb_get_au1ch_clock_configuration	<del>1</del> 5
8.13	Core .		16
	8.13.1	Detailed Description	16
	8.13.2	Function Documentation	16
		8.13.2.1 amb_get_core_pll_configuration	16
		8.13.2.2 amb_get_core_clock_frequency	16
		8.13.2.3 amb_get_core_pll_lock_status	17
		8.13.2.4 amb_enable_core_clock_observation	17
8.14	DDR .		19
	8.14.1	Detailed Description	19
	8.14.2	Function Documentation	19
		8.14.2.1 amb_get_ddr_pll_configuration	19
		8.14.2.2 amb_get_ddr_clock_frequency	19
		8.14.2.3 amb_get_ddr_pll_lock_status 5	50
		8.14.2.4 amb_enable_ddr_clock_observation 5	50
8.15	Cortex		52
	8.15.1	Detailed Description	52
	8.15.2	Function Documentation	52
		8.15.2.1 amb_set_cortex_clock_frequency 5	52
		8.15.2.2 amb_get_cortex_pll_configuration	53
		8.15.2.3 amb_get_cortex_clock_frequency 5	53
		8.15.2.4 amb_get_cortex_pll_lock_status	53
		8.15.2.5 amb_enable_cortex_clock_observation 5	54
		8.15.2.6 amb_get_axi_clock_frequency	54
8.16	Face D	etection	56
	8.16.1	Detailed Description	56
	8.16.2	Function Documentation	56
		8.16.2.1 amb_set_fdet_clock_frequency 5	56
		8.16.2.2 amb_get_fdet_clock_frequency 5	56
		8.16.2.3 amb_get_fdet_clock_configuration	57
8.17	Flash I	0	58

CONTENTS v

	8.17.1	Detailed I	Description	. 58
	8.17.2	Function	Documentation	. 58
		8.17.2.1	amb_reset_flash	. 58
		8.17.2.2	amb_reset_xd	. 58
		8.17.2.3	amb_reset_cf	. 59
		8.17.2.4	amb_reset_fio	. 59
		8.17.2.5	amb_reset_all	. 59
8.18	Gigabit	Ethernet		. 61
	8.18.1	Detailed I	Description	. 61
	8.18.2	Function	Documentation	. 61
		8.18.2.1	amb_set_gtx_clock_frequency	. 61
		8.18.2.2	amb_get_gtx_clock_frequency	. 61
		8.18.2.3	amb_get_gtx_clock_configuration	. 62
		8.18.2.4	amb_set_gtx_clock_source	. 62
		8.18.2.5	amb_get_gtx_clock_source	. 63
8.19	HDMI			. 64
	8.19.1	Detailed I	Description	. 64
	8.19.2	Enumera	tion Type Documentation	. 64
		8.19.2.1	amb_hdmi_interface_state_t	. 64
	8.19.3	Function	Documentation	. 64
		8.19.3.1	amb_get_hdmi_clock_frequency	. 64
		8.19.3.2	amb_enable_hdmi_clock_observation	. 65
		8.19.3.3	amb_get_hdmi4k_clock_frequency	. 65
		8.19.3.4	amb_enable_hdmi4k_clock_observation	. 65
		8.19.3.5	amb_auto_select_hdmi4k_range	. 66
8.20	IDSP.			. 67
	8.20.1	Detailed I	Description	. 67
	8.20.2	Function	Documentation	. 67
		8.20.2.1	amb_get_idsp_pll_configuration	. 67
		8.20.2.2	amb_get_idsp_clock_frequency	. 68
		8.20.2.3	amb_get_idsp_pll_lock_status	. 68
		8.20.2.4	amb_enable_idsp_clock_observation	. 68
		8.20.2.5	amb_get_vout_clock_frequency	. 69
8.21	Infrared	1		. 70

vi CONTENTS

8.21.1	Detailed	Description	70
8.21.2	Function	Documentation	70
	8.21.2.1	amb_set_ir_clock_frequency	70
	8.21.2.2	amb_get_ir_clock_frequency	70
	8.21.2.3	amb_get_ir_clock_configuration	71
8.22 IO Pac	ls Control		72
8.22.1	Detailed	Description	74
8.22.2	Enumera	tion Type Documentation	74
	8.22.2.1	$amb\_ioctrl\_drive\_strength\_t \ . \ . \ . \ . \ . \ . \ . \ . \ . \ $	74
	8.22.2.2	$amb\_ioctrl\_pullupdown\_t  .  .  .  .  .  .  .  .$	74
	8.22.2.3	$amb\_ioctrl\_input\_type\_t \ \dots \ \dots \ \dots \ \dots \ \dots$	74
	8.22.2.4	amb_ioctrl_slew_rate_t	74
8.22.3	Function	Documentation	75
	8.22.3.1	amb_set_misc1_ioctrl_drive_strength	75
	8.22.3.2	amb_set_misc1_ioctrl_pullupdown	75
	8.22.3.3	amb_set_misc1_ioctrl_input_type	76
	8.22.3.4	amb_get_misc1_ioctrl_configuration	76
	8.22.3.5	amb_set_misc2_ioctrl_drive_strength	76
	8.22.3.6	amb_set_misc2_ioctrl_pullupdown	77
	8.22.3.7	amb_set_misc2_ioctrl_input_type	77
	8.22.3.8	amb_get_misc2_ioctrl_configuration	78
	8.22.3.9	amb_set_smioa_ioctrl_drive_strength	78
	8.22.3.10	amb_set_smioa_ioctrl_pullupdown	78
	8.22.3.11	amb_set_smioa_ioctrl_input_type	79
	8.22.3.12	amb_get_smioa_ioctrl_configuration	79
	8.22.3.13	amb_set_smiob_ioctrl_drive_strength	80
	8.22.3.14	amb_set_smiob_ioctrl_pullupdown	80
	8.22.3.15	amb_set_smiob_ioctrl_input_type	80
	8.22.3.16	amb_get_smiob_ioctrl_configuration	81
	8.22.3.17	amb_set_smioc_ioctrl_drive_strength	81
	8.22.3.18	amb_set_smioc_ioctrl_pullupdown	82
	8.22.3.19	amb_set_smioc_ioctrl_input_type	82
	8.22.3.20	amb_get_smioc_ioctrl_configuration	82
	8.22.3.21	amb_set_smiod_ioctrl_drive_strength	83

CONTENTS vii

		8.22.3.22 amb_set_smiod_ioctrl_pullupdown	83
		8.22.3.23 amb_set_smiod_ioctrl_input_type	84
		8.22.3.24 amb_get_smiod_ioctrl_configuration	84
		8.22.3.25 amb_set_vd1_ioctrl_drive_strength	84
		8.22.3.26 amb_set_vd1_ioctrl_pullupdown	85
		8.22.3.27 amb_set_vd1_ioctrl_input_type	85
		8.22.3.28 amb_get_vd1_ioctrl_configuration	86
		8.22.3.29 amb_set_sensor_ioctrl_drive_strength	86
		8.22.3.30 amb_set_sensor_ioctrl_pullupdown	86
		8.22.3.31 amb_set_sensor_ioctrl_input_type	87
		8.22.3.32 amb_get_sensor_ioctrl_configuration	87
8.23	LCD .		88
	8.23.1	Detailed Description	88
	8.23.2	Function Documentation	88
		8.23.2.1 amb_set_lcd_clock_source	88
		8.23.2.2 amb_set_lcd_clock_frequency	89
		8.23.2.3 amb_get_lcd_pll_configuration	89
		8.23.2.4 amb_get_lcd_clock_frequency	90
		8.23.2.5 amb_get_lcd_pll_lock_status	90
		8.23.2.6 amb_enable_lcd_clock_observation	90
8.24	LVDS		92
	8.24.1	Detailed Description	92
	8.24.2	Function Documentation	92
		8.24.2.1 amb_set_lvds_pad_mode	92
		8.24.2.2 amb_get_lvds_pad_mode	92
8.25	Motor		94
	8.25.1	Detailed Description	94
	8.25.2	Function Documentation	94
		8.25.2.1 amb_set_motor_clock_frequency	94
		8.25.2.2 amb_get_motor_clock_frequency	94
		8.25.2.3 amb_get_motor_clock_configuration	95
8.26	Memor	y Stick	96
	8.26.1	Detailed Description	96
	8.26.2	Enumeration Type Documentation	97

viii CONTENTS

	8.26.2.1 amb_ms_status_t
8.26.3	Function Documentation
	8.26.3.1 amb_set_ms_clock_frequency
	8.26.3.2 amb_get_ms_clock_frequency 97
	8.26.3.3 amb_set_ms_sclk_delay
	8.26.3.4 amb_set_ms_sd_input_delay 98
	8.26.3.5 amb_set_ms_sd_output_delay 98
	8.26.3.6 amb_set_ms_read_delay
	8.26.3.7 amb_get_ms_delay_configuration 99
	8.26.3.8 amb_set_ms_status
	8.26.3.9 amb_get_ms_status
8.27 PWM.	
8.27.1	Detailed Description
8.27.2	Function Documentation
	8.27.2.1 amb_set_pwm_clock_frequency 101
	8.27.2.2 amb_get_pwm_clock_frequency 101
	8.27.2.3 amb_get_pwm_clock_configuration 102
	8.27.2.4 amb_set_pwm_clock_source
8.28 PWM .	
8.28.1	Detailed Description
8.28.2	Function Documentation
	8.28.2.1 amb_set_pwmois_clock_frequency 104
	8.28.2.2 amb_get_pwmois_clock_frequency 104
	8.28.2.3 amb_get_pwmois_clock_configuration 105
	8.28.2.4 amb_set_pwmois_clock_source 105
8.29 Reset	
8.29.1	Function Documentation
	8.29.1.1 amb_reset_chip
8.30 SD	
8.30.1	Detailed Description
8.30.2	Function Documentation
	8.30.2.1 amb_set_sd_clock_frequency
	8.30.2.2 amb_get_sd_clock_frequency
	8.30.2.3 amb_get_sd_clock_configuration 109

CONTENTS ix

8.31	SDIO.	
	8.31.1	Detailed Description
	8.31.2	Function Documentation
		8.31.2.1 amb_set_sdio_clock_frequency
		8.31.2.2 amb_get_sdio_clock_frequency
		8.31.2.3 amb_get_sdio_clock_configuration
8.32	Sensor	
	8.32.1	Detailed Description
	8.32.2	Enumeration Type Documentation
		8.32.2.1 amb_sensor_clock_pad_mode_t
	8.32.3	Function Documentation
		8.32.3.1 amb_set_sensor_clock_frequency
		8.32.3.2 amb_get_sensor_pll_configuration
		8.32.3.3 amb_get_sensor_clock_frequency
		8.32.3.4 amb_get_sensor_pll_lock_status
		8.32.3.5 amb_enable_sensor_clock_observation
		8.32.3.6 amb_set_sensor_clock_pad_mode
		8.32.3.7 amb_get_sensor_clock_pad_mode
8.33	SSI	
	8.33.1	Detailed Description
	8.33.2	Function Documentation
		8.33.2.1 amb_set_ssi_clock_frequency
		8.33.2.2 amb_get_ssi_clock_frequency
		8.33.2.3 amb_get_ssi_clock_configuration
		8.33.2.4 amb_set_ssi_clock_source
8.34	SSI2.	
	8.34.1	Detailed Description
	8.34.2	Function Documentation
		8.34.2.1 amb_set_ssi2_clock_frequency
		8.34.2.2 amb_get_ssi2_clock_frequency
		8.34.2.3 amb_set_ssi2_clock_source
8.35	UART	
	8.35.1	Detailed Description
	8.35.2	Function Documentation

x CONTENTS

			8.35.2.1 amb_set_uart_clock_frequency
			8.35.2.2 amb_get_uart_clock_frequency
			8.35.2.3 amb_uart_init
			8.35.2.4 amb_get_uart_clock_configuration
			8.35.2.5 amb_set_uart_clock_source
	8.36	VIN .	
		8.36.1	Detailed Description
		8.36.2	Function Documentation
			8.36.2.1 amb_set_vin_clock_frequency
			8.36.2.2 amb_get_vin_clock_frequency
			8.36.2.3 amb_get_vin_clock_configuration
	8.37	USB .	
		8.37.1	Detailed Description
		8.37.2	Enumeration Type Documentation
			8.37.2.1 amb_usb_port_state_t
		8.37.3	Function Documentation
			8.37.3.1 amb_usb_device_soft_reset
			8.37.3.2 amb_usb_host_soft_reset
			8.37.3.3 amb_set_usb_port_state
			8.37.3.4 amb_get_usb_port_state
	8.38	Video (	Out
		8.38.1	Detailed Description
		8.38.2	Function Documentation
			8.38.2.1 amb_set_vout_clock_source
			8.38.2.2 amb_set_vout_clock_frequency
			8.38.2.3 amb_get_vout_pll_configuration
			8.38.2.4 amb_get_vout_pll_lock_status
			8.38.2.5 amb_enable_vout_clock_observation
9	Data	Structu	re Documentation 133
	9.1		vider configuration t Struct Reference
		9.1.1	Detailed Description
	9.2		ctrl_configuration_t Struct Reference
		9.2.1	Detailed Description

CONTENTS	xi

9.3	amb_m	s_delay_configuration	_t Struct Refe	rence					. 13	34
	9.3.1	<b>Detailed Description</b>			 				. 13	34
9.4	amb_o	perating_mode_t Struc	t Reference		 				. 13	34
	9.4.1	Detailed Description			 				. 13	34
9.5	amb_p	ll_configuration_t Struc	t Reference		 				. 13	34
	9.5.1	Detailed Description			 				. 1:	35

# Ambarella A7S Hardware Abstraction Layer

### Author

Mahendra Lodha <mlodha@ambarella.com>
Rudi Rughoonundon <rudir@ambarella.com>

### Date

January 2012

### Version

7195

### Introduction:

The Ambarella A7S Hardware Abstraction Layer (ambhal) provides an API between high level software and the low level hardware registers of the A7S chip.

### Objectives:

- Ease of Use
- Stability
- Low Power

### Topics:

- · HAL Loading, Initialization & Usage
- · Changing Operating Mode
- Changing PLL Frequency
- Changing Clock Sources

### Modules:

- Initialization
- · Command Status
- Operating Mode
- System Configuration
- Divided Clocks
- PLL
- ADC
- ARM
- AHB
- APB
- Audio
- Audio 1CH
- Core
- DDR
- Face Detection
- Flash IO
- · Gigabit Ethernet
- HDMI
- Cortex
- IDSP
- Infrared
- IO Pads Control
- LCD
- LVDS
- Motor
- Memory Stick
- PWM
- PWM
- Reset
- SD
- SDIO
- Sensor
- SSI
- SSI2
- UART
- VIN
- USB
- · Video Out

# **HAL Loading, Initialization & Usage**

#### Introduction

The ambhal image must be loaded into a 4 byte aligned region of memory by the OS or the boot loader. The binary image may be relocated in physical memory (by copying the entire image to a new location) or in virtual memory (by using the mmu) **before** it is initialized.

### Initialization

The function amb\_hal\_init() must be invoked first before any other ambhal API calls are made. This function implements a simple dynamic loader to initialize the ambhal global offset table and to initialize the hardware to a known state. This function must be called after the mmu has been initialized.

### Warning

The hal image must not be relocated in physical or virtual memory after initialization. The function amb\_hal\_init() should only be called once.

### Usage

The api functions may be called by including the ambhal.h header file.

All ambhal functions are implemented in C using the ARM APCS32 ABI and they use the ARM instruction set only.

# **Changing Operating Mode**

#### Introduction

The hal operating mode represents the current operating status of all the hardware under the control of hal.

The operating mode is defined using the structure amb\_operating\_mode\_t.

### **Mode Switch**

When the operating mode is changed the status of the hardware under the control of hal is changed. For example changing the performance changes the pll settings to increase/decrease clock frequencies to meet the required new performance setting.

An operating mode switch is performed by using the following sequence.

```
amb_hal_success_t success ;
amb_operating_mode_t operating_mode ;
operating_mode.vidcap_size = AMB_VIDCAP_4000X2250 ;
operating_mode.performance = AMB_PERFORMANCE_720P30 ;
operating_mode.mode = AMB_OPERATING_MODE_CAPTURE ;
success =
   amb_set_operating_mode (amb_hal_base_address, &operating_mode) ;
K_ASSERT (success == AMB_HAL_SUCCESS) ;
```

See also Operating Mode Settings.

### 3.1 Operating Mode Settings.

The operating mode settings are hardcoded into a table inside of HAL.

The settings define the clock frequencies of various plls when using a 24 MHz reference clock.

```
amb\_operating\_mode\_parameters\_table[][] = \{idsp, postscaler_{idsp}, postscaler_{arm}, core, dram\}
                   idsp\_clock\_frequency = -
                                          postscaler<sub>idsp</sub>
                                              idsp
                    arm\_clock\_frequency =
                                          post scaler<sub>arm</sub>
                        core\_clock\_frequency = core
                       dram\_clock\_frequency = dram
#include "operating_mode_parameters.h"
// amb_operating_mode_parameters_table [][] = { idsp, postscaler_idsp,
       postscaler arm, core, dram }
amb_operating_mode_parameters_t amb_operating_mode_parameters_table [][10] = {
[AMB_OPERATING_MODE_PREVIEW] [AMB_PERFORMANCE_480P30] = {0, 0, 0, 216000000,
      336000000},
[AMB_OPERATING_MODE_PREVIEW] [AMB_PERFORMANCE_720P30] = {0, 0, 0, 144000000,
      2160000001.
[AMB_OPERATING_MODE_PREVIEW] [AMB_PERFORMANCE_720P60] = {0, 0, 0, 216000000,
      336000000),
[AMB_OPERATING_MODE_PREVIEW] [AMB_PERFORMANCE_1080160] = {0, 0, 0, 144000000,
      216000000),
[AMB_OPERATING_MODE_PREVIEW] [AMB_PERFORMANCE_1080P30] = {0, 0, 0, 144000000,
      2160000001.
[AMB_OPERATING_MODE_PREVIEW] [AMB_PERFORMANCE_1080P60] = {0, 0, 0, 216000000,
      2160000001.
[AMB_OPERATING_MODE_PREVIEW] [AMB_PERFORMANCE_4KP30] = {0, 0, 0, 216000000,
      336000000),
[AMB_OPERATING_MODE_CAPTURE] [AMB_PERFORMANCE_480P30] = {0, 0, 0, 216000000,
      3360000001.
[AMB_OPERATING_MODE_CAPTURE] [AMB_PERFORMANCE_720P30] = {0, 0, 0, 144000000,
      216000000),
[AMB_OPERATING_MODE_CAPTURE] [AMB_PERFORMANCE_720P60] = {0, 0, 0, 216000000,
      3360000001,
[AMB_OPERATING_MODE_CAPTURE] [AMB_PERFORMANCE_1080160] = {0, 0, 0, 144000000,
      216000000).
[AMB_OPERATING_MODE_CAPTURE] [AMB_PERFORMANCE_1080P30] = {0, 0, 0, 240000000,
      528000000},
[AMB_OPERATING_MODE_CAPTURE] [AMB_PERFORMANCE_1080P60] = {0, 0, 0, 216000000,
      528000000},
[AMB_OPERATING_MODE_CAPTURE] [AMB_PERFORMANCE_4KP30] = {0, 0, 0, 432000000,
      528000000},
[AMB_OPERATING_MODE_STILL_CAPTURE] [AMB_PERFORMANCE_480P30] = {576000000, 2, 1
      , 216000000, 336000000),
[AMB_OPERATING_MODE_STILL_CAPTURE] [AMB_PERFORMANCE_720P30] = {576000000, 2, 1
      , 216000000, 336000000},
[AMB_OPERATING_MODE_STILL_CAPTURE] [AMB_PERFORMANCE_720P60] = {576000000, 2, 1
      , 216000000, 336000000},
[AMB_OPERATING_MODE_STILL_CAPTURE] [AMB_PERFORMANCE_1080160] = {576000000, 2, 1
      , 216000000, 336000000),
```

```
[AMB_OPERATING_MODE_STILL_CAPTURE] [AMB_PERFORMANCE_1080P30] = {576000000, 2, 1
      , 216000000, 336000000},
[AMB_OPERATING_MODE_STILL_CAPTURE] [AMB_PERFORMANCE_1080P60] = {576000000, 2, 1
     , 216000000, 336000000),
[AMB_OPERATING_MODE_STILL_CAPTURE] [AMB_PERFORMANCE_4KP30] = {576000000, 2, 1
     , 216000000, 336000000),
[AMB_OPERATING_MODE_STILL_PREVIEW] [AMB_PERFORMANCE_480P30] = {0, 0, 0,
     216000000, 336000000),
[AMB_OPERATING_MODE_STILL_PREVIEW] [AMB_PERFORMANCE_720P30] = {0, 0, 0,
     144000000, 336000000),
[AMB_OPERATING_MODE_STILL_PREVIEW] [AMB_PERFORMANCE_720P60] = {0, 0, 0,
     216000000, 336000000},
[AMB_OPERATING_MODE_STILL_PREVIEW] [AMB_PERFORMANCE_1080160] = {0, 0, 0,
     144000000, 336000000),
[AMB_OPERATING_MODE_STILL_PREVIEW] [AMB_PERFORMANCE_1080P30] = {0, 0, 0,
     144000000, 336000000),
[AMB_OPERATING_MODE_STILL_PREVIEW] [AMB_PERFORMANCE_1080P60] = {0, 0, 0,
     216000000, 336000000),
[AMB_OPERATING_MODE_STILL_PREVIEW] [AMB_PERFORMANCE_4KP30] = {0, 0, 0,
     216000000, 336000000),
[AMB_OPERATING_MODE_PLAYBACK] [AMB_PERFORMANCE_480P30] = {5040000000, 2, 1,
     216000000, 336000000),
[AMB_OPERATING_MODE_PLAYBACK] [AMB_PERFORMANCE_720P30] = {504000000, 2,
     216000000, 336000000),
[AMB_OPERATING_MODE_PLAYBACK] [AMB_PERFORMANCE_720P60] = {504000000, 2, 1,
     216000000, 336000000},
[AMB_OPERATING_MODE_PLAYBACK] [AMB_PERFORMANCE_1080160] = {504000000, 2, 1,
     216000000, 336000000},
[AMB_OPERATING_MODE_PLAYBACK] [AMB_PERFORMANCE_1080P30] = {504000000, 2,
                                                                         1.
     216000000, 336000000),
[AMB_OPERATING_MODE_PLAYBACK] [AMB_PERFORMANCE_1080P60] = {504000000, 2, 1,
     216000000, 336000000),
[AMB_OPERATING_MODE_PLAYBACK] [AMB_PERFORMANCE_4KP30] = {672000000, 2, 1,
     432000000, 528000000},
[AMB_OPERATING_MODE_DISPLAY_AND_ARM] [AMB_PERFORMANCE_480P30] = {504000000, 2,
      1, 216000000, 336000000},
[AMB_OPERATING_MODE_DISPLAY_AND_ARM] [AMB_PERFORMANCE_720P30] = {504000000, 2,
       1, 216000000, 336000000},
[AMB_OPERATING_MODE_DISPLAY_AND_ARM] [AMB_PERFORMANCE_720P60] = {504000000, 2,
       1, 216000000, 336000000},
[AMB_OPERATING_MODE_DISPLAY_AND_ARM] [AMB_PERFORMANCE_1080160] = {504000000, 2,
       1, 216000000, 336000000},
[AMB_OPERATING_MODE_DISPLAY_AND_ARM] [AMB_PERFORMANCE_1080P30] = {504000000, 2,
       1, 216000000, 336000000),
[AMB_OPERATING_MODE_DISPLAY_AND_ARM] [AMB_PERFORMANCE_1080P60] = {504000000, 2,
       1, 216000000, 336000000},
[AMB_OPERATING_MODE_DISPLAY_AND_ARM] [AMB_PERFORMANCE_4KP30] = {504000000, 2,
       1, 216000000, 336000000),
[AMB_OPERATING_MODE_STANDBY] [AMB_PERFORMANCE_480P30] = {48000000, 1, 1,
     48000000, 480000001,
[AMB_OPERATING_MODE_STANDBY] [AMB_PERFORMANCE_720P30] = {48000000, 1, 1,
     48000000, 48000000},
[AMB_OPERATING_MODE_STANDBY] [AMB_PERFORMANCE_720P60] = {48000000, 1, 1,
     48000000, 48000000),
[AMB_OPERATING_MODE_STANDBY] [AMB_PERFORMANCE_1080160] = {48000000, 1, 1,
     48000000, 48000000},
[AMB_OPERATING_MODE_STANDBY] [AMB_PERFORMANCE_1080P30] = {48000000, 1, 1,
     48000000, 48000000},
```

```
[AMB_OPERATING_MODE_STANDBY] [AMB_PERFORMANCE_1080P60] = {48000000, 1, 1,
      48000000, 48000000},
[AMB_OPERATING_MODE_STANDBY] [AMB_PERFORMANCE_4KP30] = {48000000, 1, 1,
     48000000, 48000000},
[AMB_OPERATING_MODE_IP_CAM] [AMB_PERFORMANCE_480P30]
                                                         = {540000000, 2, 1,
      228000000, 384000000},
[AMB_OPERATING_MODE_IP_CAM] [AMB_PERFORMANCE_720P30]
                                                         = {540000000, 2, 1,
      228000000, 384000000},
[AMB_OPERATING_MODE_IP_CAM] [AMB_PERFORMANCE_720P60]
                                                         = {540000000, 2, 1,
      228000000, 384000000},
[AMB_OPERATING_MODE_IP_CAM] [AMB_PERFORMANCE_1080160]
                                                         = {540000000, 2, 1,
      228000000, 384000000},
[AMB_OPERATING_MODE_IP_CAM] [AMB_PERFORMANCE_1080P30]
                                                         = \{540000000, 2, 1,
     228000000, 384000000},
[AMB_OPERATING_MODE_IP_CAM] [AMB_PERFORMANCE_1080P60]
                                                         = \{540000000, 2, 1,
     228000000, 384000000},
[AMB_OPERATING_MODE_IP_CAM] [AMB_PERFORMANCE_4KP30]
                                                         = {540000000, 2, 1,
      228000000, 384000000}
} ;
// amb_idsp_frequency_parameters_table [] = { idsp, postscaler_idsp,
      postscaler_arm }
// idsp must run at at least core frequency / 2
amb_idsp_frequency_parameters_t amb_idsp_frequency_parameters_table [] = {
[AMB_VIDCAP_4KP30]
                                = {672000000, 2, 1}, // { 672000000,
       336000000, 656000000 }
[AMB_VIDCAP_4096X2176_60FPS]
                                = {540000000, 2, 1}, // { 540000000,
      270000000, 540000000 }
                                = {504000000, 2, 1}, // { 504000000,
[AMB_VIDCAP_4096X3575]
      252000000, 504000000 }
[AMB_VIDCAP_4000X2250]
                                = \{432000000, 2, 1\}, // \{432000000,
       216000000, 432000000 }
[AMB_VIDCAP_2304X1296]
                                = \{408000000, 2, 1\}, // \{408000000,
      204000000, 408000000 }
[AMB_VIDCAP_1984X1116]
                                = {456000000, 3, 1}, // { 456000000,
      152000000, 456000000 }
[AMB_VIDCAP_2048X1536]
                                = {432000000, 4, 1}, // { 432000000,
       108000000, 432000000 }
                                = \{504000000, 7, 1\}, // \{504000000,
[AMB_VIDCAP_1312X984]
       72000000, 480000000 }
[AMB_VIDCAP_1536X384]
                                = \{504000000, 7, 1\}, // \{504000000,
      72000000, 432000000 }
[AMB\_VIDCAP\_1536X384\_SMALL\_VB] = \{504000000, 7, 1\} // \{504000000, 7, 1\}
      72000000, 432000000 }
```

# **Changing PLL Frequency**

#### Introduction

A number of phase locked loops (pll) are present in the device to generate various independent clocks.

The api allows the frequencies of most of the plls to be set to discrete values. It also allows the frequencies to be changed in such a way that the pll remains locked during the change (this ensures that the clock is stable during the transition).

A clock frequency change is performed by using the following sequence.

## **Changing Clock Sources**

#### Introduction

Some of the plls (Video Out, Audio & LCD) in the design allow the reference clock source to be changed. The api to change the clock source takes the new clock source name and the new clock source frequency.

#### **External PLL Reference Clocks**

When the new clock source is AMB\_PLL\_REFERENCE\_CLOCK\_SOURCE\_CLK\_SI or AMB\_PLL\_REFERENCE\_CLOCK\_SOURCE\_LVDS\_IDSP\_SCLK the reference clock source of the pll is being changed. The api needs that reference clock frequency to be able to calculate the correct pll settings that will generate the output clock of the pll.

### **Internal PLL Reference Clock**

When the new clock source is AMB\_PLL\_REFERENCE\_CLOCK\_SOURCE\_CLK\_REF the api selects the reference clock frequency based on the system configuration pins (it is either 24 MHz or 27 MHz). In this case the application does not need to provide anything as the api will figure it out on its own and do the pll settings calculations accordingly.

### External Clock (No PLL)

When the new clock source is AMB\_EXTERNAL\_CLOCK\_SOURCE the pll is not used and so the api does not care what the reference clock frequency is. In fact the api will power down that pll when the application selects that option to save power.

# **Module Index**

### 6.1 Modules

Horo	ic a	lict of	lle :	modulac

Initialization
Command Status
Operating Mode
System Configuration
Divided Clocks
PLL 32
ADC
ARM
AHB
APB
Audio
Audio 1CH
Core
DDR
Cortex
Face Detection
Flash IO
Gigabit Ethernet
HDMI 64
IDSP
Infrared
IO Pads Control
LCD
LVDS 92
Motor
Memory Stick
PWM
PWM
Reset 107

14 Module Index

SD																					108
SDIO .																					110
Sensor																					112
SSI																					116
SSI2 .																					119
UART																					121
VIN																					124
USB .																					126
Video C	Эu	t																			129

# **Data Structure Index**

### 7.1 Data Structures

Lara	ara the	. data	ctructuroc	with	hrinf	descriptions:
пеге	are ure	: uaia	Siluciules	VVILLI	וטווטו	uescriptions.

amb_divider_configuration_t	
Clock Divider Configuration	. 133
amb_ioctrl_configuration_t	
IO pad configuration	. 133
amb_ms_delay_configuration_t	
Memory stick delay configuration	. 134
amb_operating_mode_t	
Operating mode	. 134
amb_pll_configuration_t	
All the fields that make up the oll frequency programming	. 134

### **Module Documentation**

### 8.1 Initialization

### **Enumerations**

enum amb\_dram\_arbiter\_priority\_t { AMB\_DRAM\_ARBITER\_DSP\_VERY\_LO-W\_PRIORITY, AMB\_DRAM\_ARBITER\_DSP\_LOW\_PRIORITY, AMB\_DRAM\_-ARBITER\_DSP\_NORMAL\_PRIORITY, AMB\_DRAM\_ARBITER\_DSP\_HIGH\_PRIORITY, HIGH\_THROTTLE, AMB\_DRAM\_ARBITER\_DSP\_HIGH\_PRIORITY, AMB\_DRAM\_ARBITER\_DSP\_VERY\_HIGH\_PRIORITY, AMB\_DRAM\_ARBITER\_DSP\_VERY\_HIGH\_PRIORITY, AMB\_DRAM\_ARBITER\_DSP\_HIGHEST\_PRIORITY}

### **Functions**

- static INLINE amb\_hal\_success\_t amb\_hal\_init (void \*amb\_hal\_base\_address, void \*amb\_apb\_peripherals\_base\_address, void \*amb\_ahb\_peripherals\_base\_address)
- static INLINE amb\_hal\_success\_t amb\_get\_chip\_name (void \*amb\_hal\_base\_-address, char \*\*amb hal chip name)
- static INLINE amb\_hal\_success\_t amb\_get\_version (void \*amb\_hal\_base\_-address, unsigned int \*amb\_hal\_version)
- static INLINE amb\_hal\_success\_t amb\_set\_peripherals\_base\_address (void \*amb\_hal\_base\_address, void \*amb\_apb\_peripherals\_base\_address, void \*amb\_ahb\_peripherals\_base\_address)
- static INLINE amb\_hal\_success\_t amb\_set\_dram\_arbiter\_priority (void \*amb\_hal\_base\_address, amb\_dram\_arbiter\_priority\_t amb\_dram\_arbiter\_priority)

### 8.1.1 Detailed Description

Set of api calls used to setup/query AMBHAL. See also HAL Loading, Initialization & Usage.

### 8.1.2 Enumeration Type Documentation

8.1.2.1 enum amb\_dram\_arbiter\_priority\_t

DRAM arbiter priority.

#### **Enumerator:**

- **AMB\_DRAM\_ARBITER\_DSP\_VERY\_LOW\_PRIORITY** Low priority for dsp clients (75 of total bandwidth)
- **AMB\_DRAM\_ARBITER\_DSP\_LOW\_PRIORITY** Low priority for dsp clients (81.-25% of total bandwidth)
- AMB\_DRAM\_ARBITER\_DSP\_NORMAL\_PRIORITY Normal priority for dsp clients (87.5% of total bandwidth)
- AMB\_DRAM\_ARBITER\_DSP\_HIGH\_PRIORITY\_HIGH\_THROTTLE High priority for dsp clients (93.75% of total bandwidth large arbiter throttle period)
- **AMB\_DRAM\_ARBITER\_DSP\_HIGH\_PRIORITY** High priority for dsp clients (93.75% of total bandwidth)
- **AMB\_DRAM\_ARBITER\_DSP\_VERY\_HIGH\_PRIORITY** High priority for dsp clients (96.8% of total bandwidth)
- **AMB\_DRAM\_ARBITER\_DSP\_HIGHEST\_PRIORITY** High priority for dsp clients (100% of total bandwidth)

### 8.1.3 Function Documentation

8.1.3.1 static INLINE amb\_hal\_success\_t amb\_hal\_init ( void \* amb\_hal\_base\_address, void \* amb\_apb\_peripherals\_base\_address, void \* amb\_ahb\_peripherals\_base\_address )

[static]

Initialize the ambhal.

### Note

This must be called before any other ambhal functions are invoked.

### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_apb	Virtual address of peripherals (corresponding to physical ad-
	peripherals-	dress 0x70000000)
	_base	
	address	
in	amb_ahb	Virtual address of peripherals (corresponding to physical ad-
	peripherals-	dress 0x60000000)
	_base	
	address	

Generated on Mon Aug 5 2013 23:32:18 for Ambarella A7 Hardware Abstraction Layer by Doxygen

8.1 Initialization 19

### Return values

AMB_HAL_SUCCE-	ambhal initialization was successful
SS	
AMB_HAL_FAIL	ambhal system failure

8.1.3.2 static INLINE amb\_hal\_success\_t amb\_get\_chip\_name ( void \* amb\_hal\_base\_address, char \*\* amb\_hal\_chip\_name ) [static]

Get the chip name.

### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
out	amb_hal	Pointer to the name of the device.
	chip_name	

### Return values

AMB_HAL_SUCCE-	Always returns success
SS	

8.1.3.3 static INLINE amb\_hal\_success\_t amb\_get\_version ( void \* amb\_hal\_base\_address, unsigned int \* amb\_hal\_version ) [static]

Get the hal version.

### Parameters

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
out	amb_hal	Pointer to the version of hal.
	version	

### **Return values**

AMB_HAL_SUCCE-	Always returns success
SS	

8.1.3.4 static INLINE amb\_hal\_success\_t amb\_set\_peripherals\_base\_address ( void \* amb\_hal\_base\_address, void \* amb\_apb\_peripherals\_base\_address, void \* amb\_ahb\_peripherals\_base\_address ) [static]

Change the base address of the apb and ahb peripherals.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_apb	Virtual address of peripherals (corresponding to physical ad-
	peripherals-	dress 0x70000000)
	_base	
	address	
in	amb_ahb	Virtual address of peripherals (corresponding to physical ad-
	peripherals-	dress 0x60000000)
	_base	
	address	

### Return values

	AMB_HAL_SUCCE-	ambhal initialization was successful
	SS	
Ī	AMB_HAL_FAIL	ambhal system failure

8.1.3.5 static INLINE amb\_hal\_success\_t amb\_set\_dram\_arbiter\_priority ( void \* amb\_hal\_base\_address, amb\_dram\_arbiter\_priority\_t amb\_dram\_arbiter\_priority )

[static]

Change the priority of dsp clients in DRAM arbiter.

### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_dram	Priority given to dsp clients by DRAM arbiter.
	arbiter	
	priority	

### Return values

AMB_HAL_SUCCE-	ambhal initialization was successful
SS	
AMB_HAL_FAIL	The amb_dram_arbiter_priority is not defined

8.2 Command Status 21

### 8.2 Command Status

### **Enumerations**

enum amb\_hal\_success\_t { AMB\_HAL\_SUCCESS = 0x00000000UL, AMB\_H-AL\_FAIL = 0xffffffUL, AMB\_HAL\_RETRY = 0xffffffeUL}

### 8.2.1 Detailed Description

Status values returned by AMBHAL api calls.

### 8.2.2 Enumeration Type Documentation

8.2.2.1 enum amb\_hal\_success\_t

Status returned by ambhal functions.

### **Enumerator:**

AMB\_HAL\_SUCCESS function succeeded.

AMB\_HAL\_FAIL function failed - check arguments.

AMB\_HAL\_RETRY function cannot complete right now - try again.

### 8.3 Operating Mode

### **Data Structures**

struct amb\_operating\_mode\_t
 Operating mode.

### **Files**

· file operating mode parameters.c

#### **Enumerations**

- enum amb\_vidcap\_window\_size\_t { AMB\_VIDCAP\_4KP30, AMB\_VIDCAP\_4096X2176\_60FPS, AMB\_VIDCAP\_4096X3575, AMB\_VIDCAP\_4000X2250, AMB\_VIDCAP\_2304X1296, AMB\_VIDCAP\_1984X1116, AMB\_VIDCAP\_2048-X1536, AMB\_VIDCAP\_1312X984, AMB\_VIDCAP\_1536X384, AMB\_VIDCAP\_1536X384\_SMALL\_VB }
- enum amb\_performance\_t { AMB\_PERFORMANCE\_480P30, AMB\_PERFORMANCE\_720P30, AMB\_PERFORMANCE\_720P60, AMB\_PERFORMANCE\_1080P60, AMB\_PERFORMANCE\_1080P30, AMB\_PERFORMANCE\_1080P60, AMB\_PERFORMANCE\_4KP30 }
- enum amb\_mode\_t { AMB\_OPERATING\_MODE\_PREVIEW, AMB\_OPERATING\_MODE\_STILL\_CAPTURE, AMB\_OPERATING\_MODE\_CAPTURE, AMB\_OPERATING\_MODE\_DISPLAY\_A-ND\_ARM, AMB\_OPERATING\_MODE\_STANDBY, AMB\_OPERATING\_MODE\_LCD\_BYPASS, AMB\_OPERATING\_MODE\_STILL\_PREVIEW, AMB\_OPERATING\_MODE\_LOW\_POWER, AMB\_OPERATING\_MODE\_IP\_CAM }
- enum amb\_dual\_stream\_state\_t { AMB\_DUAL\_STREAM\_OFF, AMB\_DUAL\_-STREAM\_ON }
- enum amb\_digital\_gamma\_mode\_t { AMB\_DIGITAL\_GAMMA\_MODE\_OFF, × AMB\_DIGITAL\_GAMMA\_MODE\_ON }

### **Functions**

- static INLINE amb\_hal\_success\_t amb\_set\_operating\_mode (void \*amb\_hal\_base\_address, amb\_operating\_mode\_t \*amb\_operating\_mode)
- static INLINE amb\_hal\_success\_t amb\_get\_operating\_mode (void \*amb\_hal\_base\_address, amb\_operating\_mode\_t \*amb\_operating\_mode)
- static INLINE amb\_hal\_success\_t amb\_get\_operating\_mode\_status (void \*amb-hal\_base\_address)

### **Variables**

• amb idsp frequency parameters tamb idsp frequency parameters table []

### 8.3.1 Detailed Description

API calls used to change and query the operating mode. See also Changing Operating Mode.

## 8.3.2 Enumeration Type Documentation

8.3.2.1 enum amb\_vidcap\_window\_size\_t

Video Capture Window Size.

### **Enumerator:**

```
AMB_VIDCAP_4KP30 4K p30

AMB_VIDCAP_4096X2176_60FPS 4K p60

AMB_VIDCAP_4096X3575 IMX 083: 1080p30.

AMB_VIDCAP_4000X2250 IMX 078: 1080p30, 720p30.

AMB_VIDCAP_2304X1296 Aptina 3135: 1080p60, 720p60, 1080p30, 720p30.

AMB_VIDCAP_1984X1116 IMX 078: 1080p60, 720p60.

AMB_VIDCAP_2048X1536 Aptina 3135: 4:3 photo preview.

AMB_VIDCAP_1312X984 IMX 078: 4:3 photo preview.

AMB_VIDCAP_1536X384 IMX 083: 720p30.
```

AMB\_VIDCAP\_1536X384\_SMALL\_VB IMX 083: photo preview.

## 8.3.2.2 enum amb\_performance\_t

## Performance.

## Enumerator:

```
AMB_PERFORMANCE_480P30 480p30
AMB_PERFORMANCE_720P30 720p30
AMB_PERFORMANCE_720P60 720p60
AMB_PERFORMANCE_1080I60 1080i
AMB_PERFORMANCE_1080P30 1080p30
AMB_PERFORMANCE_1080P60 1080p60
AMB_PERFORMANCE_4KP30 4K p30
```

8.3.2.3 enum amb\_mode\_t

Operating Mode.

#### **Enumerator:**

**AMB\_OPERATING\_MODE\_PREVIEW** Camera is on but no video being captured.

AMB\_OPERATING\_MODE\_STILL\_CAPTURE Still picture capture.

**AMB\_OPERATING\_MODE\_CAPTURE** Camera is on and video is being captured.

AMB\_OPERATING\_MODE\_PLAYBACK Video playback.

AMB OPERATING MODE DISPLAY AND ARM GUI only.

AMB\_OPERATING\_MODE\_STANDBY Low power mode.

AMB\_OPERATING\_MODE\_LCD\_BYPASS LCD off.

AMB\_OPERATING\_MODE\_STILL\_PREVIEW Still picture preview.

AMB OPERATING MODE LOW POWER Low power.

AMB\_OPERATING\_MODE\_IP\_CAM IP Cam.

8.3.2.4 enum amb\_dual\_stream\_state\_t

Dual Stream state.

**Enumerator:** 

**AMB\_DUAL\_STREAM\_OFF** Dual Stream is off. **AMB\_DUAL\_STREAM\_ON** Dual Stream is on.

8.3.2.5 enum amb\_digital\_gamma\_mode\_t

Digital Gamma Mode.

Turning this on forces the core clock frequency to be multiple of 36 MHz.

**Enumerator:** 

AMB\_DIGITAL\_GAMMA\_MODE\_OFF Digital Gamma Mode is off.

AMB\_DIGITAL\_GAMMA\_MODE\_ON Digital Gamma Mode is on.

### 8.3.3 Function Documentation

8.3.3.1 static INLINE amb\_hal\_success\_t amb\_set\_operating\_mode ( void \* amb\_hal\_base\_address, amb\_operating\_mode\_t \* amb\_operating\_mode ) [static]

Set the current operating mode for the system.

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb	New operating mode.
	operating	
	mode	

#### Return values

AMB_HAL_SUCCE-	The new operating mode has been programmed.
SS	

amb\_get\_operating\_mode\_status() must be called after this to check whether the new operating mode has taken effect.

### Return values

AMB_HAL_RETRY	Another operation is in progress. Try later
AMB_HAL_FAIL	The new operating mode was not set because of invalid arguments.

8.3.3.2 static INLINE amb\_hal\_success\_t amb\_get\_operating\_mode ( void \* amb\_hal\_base\_address, amb\_operating\_mode\_t \* amb\_operating\_mode )  $[static] \label{eq:static}$ 

Get the current operating mode for the system.

### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
out	amb	Current operating mode.
	operating	
	mode	

### Return values

AMB_HAL_SUCCE-	Always returns success.
SS	

8.3.3.3 static INLINE amb\_hal\_success\_t amb\_get\_operating\_mode\_status ( void \*  $amb\_hal\_base\_address \ ) \quad [static]$ 

Check whether a previous amb\_set\_operating\_mode() call has completed.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

#### Return values

AMB_HAL_SUCCE-	The new operating mode has been set.
SS	
AMB_HAL_RETRY	The new operating mode has not been set yet.
AMB_HAL_FAIL	The new operating mode has failed.

#### 8.3.4 Variable Documentation

### 8.3.4.1 amb\_idsp\_frequency\_parameters\_t amb\_idsp\_frequency\_parameters\_table[]

#### Initial value:

The idsp clock frequencies are based on the video capture window and are hardcoded into a table inside of HAL.

The settings define the clock frequency of the idsp pll, idsp clock and arm clock when using a 24 MHz reference clock.

 $amb\_operating\_mode\_parameters\_table[][] = \{idsp, postscaler_{idsp}, postscaler_{arm}, core, dram\}$ 

$$idsp\_clock\_frequency = \frac{idsp}{postscaler_{idsp}}$$
 
$$arm\_clock\_frequency = \frac{idsp}{postscaler_{arm}}$$

#### Note

When hdmi is on and operating mode is still preview, video preview, video capture or video playback the idsp clock frequency is forced to be a minimum of 93 MHz and arm clock frequency is forced to be 468 MHz (at 24 MHz reference clock)

# 8.4 System Configuration

#### **Enumerations**

- enum amb\_system\_configuration\_t { AMB\_SYSTEM\_CONFIGURATION\_NAND\_FLASH\_TYPE = 0x1UL, AMB\_SYSTEM\_CONFIGURATION\_NAND\_FLASH\_2048\_PAGE\_SIZE = 0x10UL, AMB\_SYSTEM\_CONFIGURATION\_NAND\_FLASH\_READ\_CONFIRM = 0x20UL, AMB\_SYSTEM\_CONFIGURATION\_NAND\_FLASH\_ECC = 0x400UL, AMB\_SYSTEM\_CONFIGURATION\_NAND\_FLASH\_SPARE\_CELL = 0x800UL, AMB\_SYSTEM\_CONFIGURATION\_ETHERNET\_SELECTED = 0x00080000UL, AMB\_SYSTEM\_CONFIGURATION\_RMII\_SELECTED = 0x8000UL, AMB\_SYSTEM\_CONFIGURATION\_HIF\_SECUREMODE = 0x200000UL}
- enum amb\_boot\_type\_t { AMB\_USB\_BOOT, AMB\_SD\_BOOT, AMB\_NAND\_-BOOT, AMB\_NOR\_BOOT, AMB\_SSI\_BOOT, AMB\_HIF\_BOOT, AMB\_XIP\_-BOOT }
- enum amb\_hif\_type\_t { AMB\_HIF\_DISABLE, AMB\_INTEL\_READY\_ACTIVE\_-HIGH, AMB\_INTEL\_READY\_ACTIVE\_LOW, AMB\_MOTOROLA\_DACK\_ACT-IVE\_HIGH, AMB\_MOTOROLA\_DACK\_ACTIVE\_LOW }

#### **Functions**

- static INLINE amb\_clock\_frequency\_t amb\_get\_reference\_clock\_frequency (void \*amb hal base address)
- static INLINE amb\_system\_configuration\_t amb\_get\_system\_configuration (void \*amb\_hal\_base\_address)
- static INLINE amb\_boot\_type\_t amb\_get\_boot\_type (void \*amb\_hal\_base\_-address)
- static INLINE amb\_hif\_type\_t amb\_get\_hif\_type (void \*amb\_hal\_base\_address)

### 8.4.1 Detailed Description

These api calls query the system configuration pins and return how the chip is configured.

## 8.4.2 Enumeration Type Documentation

8.4.2.1 enum amb\_system\_configuration\_t

System configuration settings.

### **Enumerator:**

AMB\_SYSTEM\_CONFIGURATION\_NAND\_FLASH\_TYPE 1.8V NAND Flash - Selected (1) or 3.3V NAND Flash Selected (0)

AMB\_SYSTEM\_CONFIGURATION\_NAND\_FLASH\_2048\_PAGE\_SIZE 2048
Bytes Flash Page Size (1) or 512 Bytes Flash Page Size (0)

**AMB\_SYSTEM\_CONFIGURATION\_NAND\_FLASH\_READ\_CONFIRM** NAND - Read Confirm.

AMB\_SYSTEM\_CONFIGURATION\_NAND\_FLASH\_ECC NAND ECC.

AMB\_SYSTEM\_CONFIGURATION\_NAND\_FLASH\_SPARE\_CELL NAND Spare Cell.

AMB\_SYSTEM\_CONFIGURATION\_ETHERNET\_SELECTED Ethernet Selected.

AMB\_SYSTEM\_CONFIGURATION\_RMII\_SELECTED RMII Selected.

AMB\_SYSTEM\_CONFIGURATION\_HIF\_SECURE\_MODE Host Interface Secure Mode.

8.4.2.2 enum amb\_boot\_type\_t

Boot type select.

#### **Enumerator:**

AMB\_USB\_BOOT USB Boot.

AMB\_SD\_BOOT SD Boot.

AMB\_NAND\_BOOT Flash Boot.

AMB\_NOR\_BOOT Flash Boot.

AMB\_SSI\_BOOT SSI Boot.

AMB\_HIF\_BOOT Host Interface Boot.

AMB\_XIP\_BOOT XIP Boot.

8.4.2.3 enum amb\_hif\_type\_t

Host interface type select.

# **Enumerator:**

AMB\_HIF\_DISABLE Host Interface Disabled.

AMB\_INTEL\_READY\_ACTIVE\_HIGH Intel Ready Asserted High.

AMB\_INTEL\_READY\_ACTIVE\_LOW Intel Ready Asserted Low.

AMB\_MOTOROLA\_DACK\_ACTIVE\_HIGH Motorola Data Acknowledge Asserted High.

AMB\_MOTOROLA\_DACK\_ACTIVE\_LOW Motorola Data Acknowledge - Asserted Low.

## 8.4.3 Function Documentation

8.4.3.1 static INLINE amb\_clock\_frequency\_t amb\_get\_reference\_clock\_frequency ( void \* amb\_hal\_base\_address ) [static]

Get the reference clock frequency.

### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

### Return values

amb_clock	The reference clock frequency from the system configuration pins.
frequency_t	

8.4.3.2 static INLINE amb\_system\_configuration\_t amb\_get\_system\_configuration (void \* amb\_hal\_base\_address ) [static]

Get the system configuration.

## Parameters

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

## Note

Use the flags defined in <a href="mailto:amb\_system\_configuration\_t">amb\_system\_configuration\_t</a> to determine what system configuration was set.

#### Return values

```
amb_system_- The system configuration.

configuration_t
```

8.4.3.3 static INLINE amb\_boot\_type\_t amb\_get\_boot\_type ( void \* amb\_hal\_base\_address ) [static]

Get the boot type selection.

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

## **Return values**

```
amb_boot_type_t | The boot type selected.
```

8.4.3.4 static INLINE amb\_hif\_type\_t amb\_get\_hif\_type ( void \* amb\_hal\_base\_address ) [static]

Get the host interface type.

### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

## Return values

amb\_hif\_type\_t | The host interface type selected.

8.5 Divided Clocks 31

# 8.5 Divided Clocks

# **Data Structures**

• struct amb\_divider\_configuration\_t Clock Divider Configuration.

# **Typedefs**

• typedef unsigned int amb\_clock\_frequency\_t

# 8.5.1 Detailed Description

Defines the type used to specify and return clock frequencies.

# 8.6 PLL

# **Data Structures**

struct amb\_pll\_configuration\_t

All the fields that make up the pll frequency programming.

## **Typedefs**

• typedef unsigned int amb\_pll\_fractional\_divisor\_t

#### **Enumerations**

enum amb\_clock\_source\_t { AMB\_REFERENCE\_CLOCK\_SOURCE\_CLK\_REF, AMB\_REFERENCE\_CLOCK\_SOURCE\_CLK\_SI, AMB\_REFERENCE\_CLOCK\_SOURCE\_LVDS\_IDSP\_SCLK, AMB\_EXTERNAL\_CLOCK\_SOURCE, AMB\_SHARE\_VOUT\_CLOCK, AMB\_REFERENCE\_CLOCK\_SOURCE\_CORE\_PLL\_VCO, AMB\_REFERENCE\_CLOCK\_SOURCE\_CORE\_PLL, AMB\_REFERENCE\_CLOCK\_SOURCE\_AUDIO\_PLL\_VCO, AMB\_REFERENCE\_CLOCK\_SOURCE\_IDSP\_PLL\_VCO, AMB\_REFERENCE\_CLOCK\_SOURCE\_CORTEX\_PLL\_VCO, AMB\_REFERENCE\_CLOCK\_SOURCE\_APB, AMB\_REFERENCE\_CLOCK\_SOURCE\_CORE\_CLOCK\_SOURCE\_CORE, AMB\_REFERENCE\_CLOCK\_SOURCE\_ARM, AMB\_REFERENCE\_CLOCK\_SOURCE\_CORE\_SOURCE\_IDSP\_S

### **Functions**

static INLINE amb\_hal\_success\_t amb\_disable\_clock\_observation (void \*amb\_hal\_base\_address)

### 8.6.1 Detailed Description

Various type definitions related to pll programming/query. Also defines an api call to disable observation of pll through xx clk si pin.

# 8.6.2 Typedef Documentation

8.6.2.1 typedef unsigned int amb\_pll\_fractional\_divisor\_t

PLL fractional frequency setting.

#### Note

This value is limited so that only a fractional change of up to  $\sim\!\!50$  KHz may be requested.

8.6 PLL 33

### 8.6.3 Enumeration Type Documentation

8.6.3.1 enum amb\_clock\_source\_t

PLL/Divider Reference Clock Source.

#### **Enumerator:**

- **AMB\_REFERENCE\_CLOCK\_SOURCE\_CLK\_REF** Reference clock from crystal oscillator either 24 MHz or 27 MHz.
- AMB\_REFERENCE\_CLOCK\_SOURCE\_CLK\_SI Use clk\_si as reference for the pll.
- **AMB\_REFERENCE\_CLOCK\_SOURCE\_LVDS\_IDSP\_SCLK** Use lvds\_idsp\_-sclk as reference for the pll/divider.
- AMB\_EXTERNAL\_CLOCK\_SOURCE Use external clock source no pll.
- AMB\_SHARE\_VOUT\_CLOCK Use the vout pll clock for lcd/hdmi pll.
- **AMB\_REFERENCE\_CLOCK\_SOURCE\_CORE\_PLL\_VCO** Use the core pll vco output for divider.
- **AMB\_REFERENCE\_CLOCK\_SOURCE\_CORE\_PLL** Use the core pll output for divider.
- **AMB\_REFERENCE\_CLOCK\_SOURCE\_AUDIO\_PLL\_VCO** Use the audio pll vco output for divider.
- **AMB\_REFERENCE\_CLOCK\_SOURCE\_AUDIO\_PLL** Use the audio pll output for divider.
- AMB\_REFERENCE\_CLOCK\_SOURCE\_IDSP\_PLL\_VCO Use the idsp pll vco output for divider.
- **AMB\_REFERENCE\_CLOCK\_SOURCE\_CORTEX\_PLL\_VCO** Use the cortex pll vco output for divider.
- AMB\_REFERENCE\_CLOCK\_SOURCE\_APB Use the apb clock for divider.
- AMB\_REFERENCE\_CLOCK\_SOURCE\_CORE Use the core clock for divider.
- AMB\_REFERENCE\_CLOCK\_SOURCE\_ARM Use the arm clock for divider.
- AMB\_REFERENCE\_CLOCK\_SOURCE\_IDSP Use the idsp clock for divider.

# 8.6.4 Function Documentation

8.6.4.1 static INLINE amb\_hal\_success\_t amb\_disable\_clock\_observation ( void \* amb\_hal\_base\_address ) [static]

Disable clock observation.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

AMB_HAL_SUCCE-	Always returns success.
SS	

8.7 ADC 35

# 8.7 ADC

## **Functions**

static INLINE amb\_hal\_success\_t amb\_set\_adc\_clock\_frequency (void \*amb\_hal\_base\_address, amb\_clock\_frequency\_t amb\_clock\_frequency)

- static INLINE amb\_clock\_frequency\_t amb\_get\_adc\_clock\_frequency (void \*amb\_hal\_base\_address)
- static INLINE amb\_clock\_frequency\_t amb\_get\_adc\_clock\_configuration (void \*amb\_hal\_base\_address, amb\_divider\_configuration\_t \*amb\_adc\_divider\_configuration)

# 8.7.1 Detailed Description

API calls to change/query the frequency of the ADC.

## 8.7.2 Function Documentation

8.7.2.1 static INLINE amb\_hal\_success\_t amb\_set\_adc\_clock\_frequency ( void \* amb\_hal\_base\_address, amb\_clock\_frequency\_t amb\_clock\_frequency ) [static]

Set the adc clock frequency.

#### **Parameters**

_			
	in	amb_hal	Virtual address where ambhal is loaded by OS.
		base	
		address	
	in	amb_clock	New adc clock frequency
		frequency	

#### **Return values**

	AMB_HAL_SUCCE-	The new frequency has been set
	SS	
Ī	AMB_HAL_FAIL	The new frequency is invalid and has not been set

8.7.2.2 static INLINE amb\_clock\_frequency\_t amb\_get\_adc\_clock\_frequency ( void \* amb\_hal\_base\_address ) [static]

Get the frequency of the adc clock.

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

## **Return values**

amb_clock	Requested clock frequency
frequency_t	

8.7.2.3 static INLINE amb\_clock\_frequency\_t amb\_get\_adc\_clock\_configuration ( void \* amb\_hal\_base\_address, amb\_divider\_configuration\_t \* amb\_adc\_divider\_configuration ) [static]

Get the configuration of the adc clock divider.

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
out	amb_adc	Clock configuration information read from divider.
	divider	
	configuration	

AMB_HAL_SUCCE-	Always returns success.
SS	

8.8 ARM 37

# 8.8 ARM

# **Functions**

 static INLINE amb\_clock\_frequency\_t amb\_get\_arm\_clock\_frequency (void \*amb\_hal\_base\_address)

# 8.8.1 Detailed Description

API calls to query the frequency of the arm clock. The arm clock frequency cannot be set directly. It is changed by AMBHAL when the operating mode is changed. See also Operating Mode.

### 8.8.2 Function Documentation

8.8.2.1 static INLINE amb\_clock\_frequency\_t amb\_get\_arm\_clock\_frequency ( void \* amb\_hal\_base\_address ) [static]

Get the arm clock frequency.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

amb_clock	The arm clock frequency.
frequency_t	

# 8.9 AHB

# **Functions**

 static INLINE amb\_clock\_frequency\_t amb\_get\_ahb\_clock\_frequency (void \*amb\_hal\_base\_address)

# 8.9.1 Detailed Description

API calls to query the frequency of the ahb clock. The ahb clock frequency cannot be set directly. It is changed by AMBHAL when the operating mode is changed. See also Operating Mode.

### 8.9.2 Function Documentation

8.9.2.1 static INLINE amb\_clock\_frequency\_t amb\_get\_ahb\_clock\_frequency( void \* amb\_hal\_base\_address) [static]

Get the AHB clock frequency.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

amb_clock	The AHB clock frequency.
frequency_t	

8.10 APB 39

# 8.10 APB

# **Functions**

 static INLINE amb\_clock\_frequency\_t amb\_get\_apb\_clock\_frequency (void \*amb\_hal\_base\_address)

# 8.10.1 Detailed Description

API calls to query the frequency of the apb clock. The apb clock frequency cannot be set directly. It is changed by AMBHAL when the operating mode is changed. See also Operating Mode.

### 8.10.2 Function Documentation

8.10.2.1 static INLINE amb\_clock\_frequency\_t amb\_get\_apb\_clock\_frequency( void \* amb\_hal\_base\_address) [static]

Get the APB clock frequency.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

amb_clock	The APB clock frequency.
frequency_t	

## 8.11 **Audio**

### **Functions**

- static INLINE amb\_hal\_success\_t amb\_set\_audio\_clock\_source (void \*amb\_hal\_base\_address, amb\_clock\_source\_t amb\_clock\_source, amb\_clock\_frequency-t amb\_clock\_frequency)
- static INLINE amb\_hal\_success\_t amb\_set\_audio\_clock\_frequency (void \*amb\_hal\_base\_address, amb\_clock\_frequency\_t amb\_audio\_clock\_frequency)
- static INLINE amb\_hal\_success\_t amb\_get\_audio\_pll\_configuration (void \*amb-hal base address, amb pll configuration t \*amb audio pll configuration)
- static INLINE amb\_clock\_frequency\_t amb\_get\_audio\_clock\_frequency (void \*amb\_hal\_base\_address)
- static INLINE amb\_hal\_success\_t amb\_get\_audio\_pll\_lock\_status (void \*amb\_hal base address)
- static INLINE amb\_hal\_success\_t amb\_enable\_audio\_clock\_observation (void \*amb\_hal\_base\_address)

# 8.11.1 Detailed Description

API calls to change/query frequency of the Audio pll. See also Changing PLL Frequency and Changing Clock Sources.

## 8.11.2 Function Documentation

Set the clock source for audio.

# Parameters

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_clock	the new clock source.
	source	
in	amb_clock	the clock frequency of the new source.
	frequency	

### Note

The amb\_clock\_frequency only needs to be specified for the clock sources AMB\_-PLL\_REFERENCE\_CLOCK\_SOURCE\_CLK\_SI and AMB\_PLL\_REFERENCE\_C-LOCK\_SOURCE\_LVDS\_IDSP\_SCLK. Specify an amb\_clock\_frequency of 0 for all other clock sources. The topic Changing Clock Sources covers this in more details.

8.11 Audio 41

## Return values

ĺ	AMB_HAL_SUCCE-	a new clock source has been set.
	SS	
ĺ	AMB_HAL_FAIL	the clock source is not supported.

8.11.2.2 static INLINE amb\_hal\_success\_t amb\_set\_audio\_clock\_frequency ( void \* amb\_hal\_base\_address, amb\_clock\_frequency\_t amb\_audio\_clock\_frequency ) [static]

Set the audio pll frequency.

### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_audio-	The requested frequency.
	_clock	
	frequency	

## **Return values**

AMB_HAL_SUCCE-	the new requested pll frequency is valid and it has been pro-
SS	grammed.
AMB_HAL_FAIL	the new pll frequency requested is not supported.
AMB_HAL_RETRY	a previous pll frequency change request is still outstanding.

8.11.2.3 static INLINE amb\_hal\_success\_t amb\_get\_audio\_pll\_configuration ( void \*  $amb\_hal\_base\_address$ , amb\_pll\_configuration\_t \*  $amb\_audio\_pll\_configuration$  ) [static]

Get the current audio pll configuration.

# Parameters

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
out	amb_audio-	Sensor pll configuration information read from pll registers.
	_pll	
	configuration	

AMB HAL SUCCE-	always returns success.
, <u>_</u> <u>_</u> _00000	
SS	
00	

8.11.2.4 static INLINE amb\_clock\_frequency\_t amb\_get\_audio\_clock\_frequency( void \* amb\_hal\_base\_address ) [static]

Get the current audio pll frequency.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

### **Return values**

amb_clock	Requested clock frequency.
frequency_t	

8.11.2.5 static INLINE amb\_hal\_success\_t amb\_get\_audio\_pll\_lock\_status ( void \* amb\_hal\_base\_address ) [static]

Get the status of the previous requested audio pll frequency change.

### Note

A new audio pll frequency change may be requested after this function returns A-MB\_HAL\_SUCCESS.

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

### Return values

AMB_HAL_SUCCE-	the pll has locked to the new frequency.
SS	
AMB_HAL_FAIL	the pll lock has failed to lock in a reasonable amount of time. some-
	thing is wrong.
AMB_HAL_RETRY	the pll has not locked yet. try again.

8.11.2.6 static INLINE amb\_hal\_success\_t amb\_enable\_audio\_clock\_observation(
 void \* amb\_hal\_base\_address) [static]

Enable observation of audio clock.

8.11 Audio 43

# Note

A divided by 16 version of the clock may be observed on the xx\_clk\_si pin.

# **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

AMB_HAL_SUCCE-	Always returns success
SS	

# 8.12 Audio 1CH

### **Functions**

- static INLINE amb\_hal\_success\_t amb\_set\_au1ch\_clock\_frequency (void \*amb-hal\_base\_address, amb\_clock\_frequency\_t amb\_clock\_frequency)
- static INLINE amb\_clock\_frequency\_t amb\_get\_au1ch\_clock\_frequency (void \*amb\_hal\_base\_address)
- static INLINE amb\_clock\_frequency\_t amb\_get\_au1ch\_clock\_configuration (void \*amb\_hal\_base\_address, amb\_divider\_configuration\_t \*amb\_au1ch\_divider\_configuration)

# 8.12.1 Detailed Description

API calls to change/query the frequency of the Audio 1CH block.

## 8.12.2 Function Documentation

8.12.2.1 static INLINE amb\_hal\_success\_t amb\_set\_au1ch\_clock\_frequency ( void \* amb\_hal\_base\_address, amb\_clock\_frequency\_t amb\_clock\_frequency )

[static]

Set the clock frequency of the single channel audio controller.

### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_clock	New au1ch frequency
	frequency	

#### **Return values**

AMB_HAL_SUCCE-	the new frequency has been set
SS	
AMB_HAL_FAIL	the new requested frequency is not valid

8.12.2.2 static INLINE amb\_clock\_frequency\_t amb\_get\_au1ch\_clock\_frequency( void \* amb\_hal\_base\_address ) [static]

Get the frequency of the au1ch clock.

8.12 Audio 1CH 45

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

## Return values

amb_clock	Requested clock frequency
frequency_t	

8.12.2.3 static INLINE amb\_clock\_frequency\_t amb\_get\_au1ch\_clock\_configuration ( void \* amb\_hal\_base\_address, amb\_divider\_configuration\_t \*  $amb\_au1ch\_divider\_configuration \ ) \quad [static]$ 

Get the configuration of the au1ch clock divider.

## **Parameters**

-			
	in	amb_hal	Virtual address where ambhal is loaded by OS.
		base	
		address	
Ī	out	amb_au1ch-	Clock configuration information read from divider.
		_divider	
		configuration	

AMB_HAL_SUCCE-	Always returns success.
SS	

## 8.13 Core

## **Functions**

- static INLINE amb\_hal\_success\_t amb\_get\_core\_pll\_configuration (void \*amb\_hal\_base\_address, amb\_pll\_configuration\_t \*amb\_core\_pll\_configuration)
- static INLINE amb\_clock\_frequency\_t amb\_get\_core\_clock\_frequency (void \*amb\_hal\_base\_address)
- static INLINE amb\_hal\_success\_t amb\_get\_core\_pll\_lock\_status (void \*amb\_hal\_base\_address)
- static INLINE amb\_hal\_success\_t amb\_enable\_core\_clock\_observation (void \*amb hal base address)

### 8.13.1 Detailed Description

API calls to query the frequency of the core clock. The core clock frequency cannot be set directly. It is changed by AMBHAL when the operating mode is changed. See also Operating Mode.

### 8.13.2 Function Documentation

8.13.2.1 static INLINE amb\_hal\_success\_t amb\_get\_core\_pll\_configuration ( void \*  $amb\_hal\_base\_address, \ amb\_pll\_configuration\_t * amb\_core\_pll\_configuration \ )$  [static]

Get the current core pll configuration.

#### **Parameters**

	in	amb_hal	Virtual address where ambhal is loaded by OS.
		base	
		address	
Ī	out	amb_core	pll configuration information read from pll registers.
		pII	
		configuration	

#### Return values

```
AMB_HAL_SUCCE- always returns success.

SS
```

8.13.2.2 static INLINE amb\_clock\_frequency\_t amb\_get\_core\_clock\_frequency(
 void \* amb\_hal\_base\_address ) [static]

Get the current core pll frequency.

8.13 Core 47

### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

## Return values

amb_clock	Requested clock frequency.
frequency_t	

8.13.2.3 static INLINE amb\_hal\_success\_t amb\_get\_core\_pll\_lock\_status ( void \* amb\_hal\_base\_address ) [static]

Get the status of the previous requested core pll frequency change.

#### Note

A new core pll frequency change may be requested after this function returns AM-B HAL SUCCESS.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

## Return values

AMB_HA	AL_SUCCE-	the pll has locked to the new frequency.
	SS	
AMB	_HAL_FAIL	the pll lock has failed to lock in a reasonable amount of time. some-
		thing is wrong.
AMB_H	AL_RETRY	the pll has not locked yet. try again.

8.13.2.4 static INLINE amb\_hal\_success\_t amb\_enable\_core\_clock\_observation ( void \* amb\_hal\_base\_address ) [static]

Enable observation of core clock.

### Note

A divided by 16 version of the clock may be observed on the xx\_clk\_si pin.

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

Generated on Mon Aug 5 2013 23:32:18 for Ambarella A7 Hardware Abstraction Layer by Doxygen

AMB_HAL_SUCCE-	Always returns success
SS	

8.14 DDR 49

## 8.14 DDR

### **Functions**

static INLINE amb\_hal\_success\_t amb\_get\_ddr\_pll\_configuration (void \*amb\_hal\_base\_address, amb\_pll\_configuration\_t \*amb\_ddr\_pll\_configuration)

- static INLINE amb\_clock\_frequency\_t amb\_get\_ddr\_clock\_frequency (void \*amb\_hal\_base\_address)
- static INLINE amb\_hal\_success\_t amb\_get\_ddr\_pll\_lock\_status (void \*amb\_hal-base address)
- static INLINE amb\_hal\_success\_t amb\_enable\_ddr\_clock\_observation (void \*amb hal base address)

## 8.14.1 Detailed Description

API calls to query the frequency of the ddr clock. The ddr clock frequency cannot be set directly. It is changed by AMBHAL when the operating mode is changed. See also Operating Mode.

### 8.14.2 Function Documentation

8.14.2.1 static INLINE amb\_hal\_success\_t amb\_get\_ddr\_pll\_configuration ( void \* amb\_hal\_base\_address, amb\_pll\_configuration\_t \* amb\_ddr\_pll\_configuration ) [static]

Get the current ddr pll configuration.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
out	amb_ddr	pll configuration information read from pll registers.
	pll	
	configuration	

### Return values

AMB_HAL_SUCCE-	always returns success.
SS	

8.14.2.2 static INLINE amb\_clock\_frequency\_t amb\_get\_ddr\_clock\_frequency ( void \* amb\_hal\_base\_address ) [static]

Get the current ddr pll frequency.

### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

## Return values

amb_clock	Requested clock frequency.
frequency_t	

8.14.2.3 static INLINE amb\_hal\_success\_t amb\_get\_ddr\_pll\_lock\_status ( void \* amb\_hal\_base\_address ) [static]

Get the status of the previous requested ddr pll frequency change.

#### Note

A new ddr pll frequency change may be requested after this function returns AMB\_HAL\_SUCCESS.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

## Return values

AMB_HAL_SUCCE-	the pll has locked to the new frequency.
SS	
AMB_HAL_FAIL	the pll lock has failed to lock in a reasonable amount of time. some-
	thing is wrong.
AMB_HAL_RETRY	the pll has not locked yet. try again.

8.14.2.4 static INLINE amb\_hal\_success\_t amb\_enable\_ddr\_clock\_observation ( void \* amb\_hal\_base\_address ) [static]

Enable observation of ddr clock.

### Note

A divided by 16 version of the clock may be observed on the xx\_clk\_si pin.

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
	Generated on Mo	on Aug 5 2013 23:32:18 for Ambarella A7 Hardware Abstraction Layer by Doxygen

8.14 DDR 51

AMB_HAL_SUCCE-	Always returns success
SS	

# 8.15 Cortex

### **Functions**

- static INLINE amb\_hal\_success\_t amb\_set\_cortex\_clock\_frequency (void \*amb-hal\_base\_address, amb\_clock\_frequency\_t amb\_cortex\_clock\_frequency)
- static INLINE amb\_hal\_success\_t amb\_get\_cortex\_pll\_configuration (void \*amb-hal\_base\_address, amb\_pll\_configuration\_t \*amb\_cortex\_pll\_configuration)
- static INLINE amb\_clock\_frequency\_t amb\_get\_cortex\_clock\_frequency (void \*amb\_hal\_base\_address)
- static INLINE amb\_hal\_success\_t amb\_get\_cortex\_pll\_lock\_status (void \*amb\_hal base address)
- static INLINE amb\_hal\_success\_t amb\_enable\_cortex\_clock\_observation (void \*amb\_hal\_base\_address)
- static INLINE amb\_clock\_frequency\_t amb\_get\_axi\_clock\_frequency (void \*amb\_hal\_base\_address)

## 8.15.1 Detailed Description

API calls to set/query the frequency of the cortex clock.

#### 8.15.2 Function Documentation

8.15.2.1 static INLINE amb\_hal\_success\_t amb\_set\_cortex\_clock\_frequency ( void \* amb\_hal\_base\_address, amb\_clock\_frequency\_t amb\_cortex\_clock\_frequency )
[static]

Set the cortex pll frequency.

#### **Parameters**

_			
	in	amb_hal	Virtual address where ambhal is loaded by OS.
		base	
		address	
	in	amb_cortex-	The requested frequency.
		_clock	
		frequency	

AMB_HAL_SUCCE-	the new requested pll frequency is valid and it has been pro-
SS	grammed.
AMB_HAL_FAIL	the new pll frequency requested is not supported.
AMB_HAL_RETRY	a previous pll frequency change request is still outstanding.

8.15 Cortex 53

8.15.2.2 static INLINE amb\_hal\_success\_t amb\_get\_cortex\_pll\_configuration (void \* amb\_hal\_base\_address, amb\_pll\_configuration\_t \* amb\_cortex\_pll\_configuration
) [static]

Get the current cortex pll configuration.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
out	amb_cortex-	pll configuration information read from pll registers.
	_pll	
	configuration	

### Return values

AMB_HAL_SUCCE-	Always returns success.
SS	

8.15.2.3 static INLINE amb\_clock\_frequency\_t amb\_get\_cortex\_clock\_frequency( void \* amb\_hal\_base\_address ) [static]

Get the current cortex pll frequency.

## Parameters

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

### Return values

amb clock -	Requested clock frequency.
frequency_t	

8.15.2.4 static INLINE amb\_hal\_success\_t amb\_get\_cortex\_pll\_lock\_status ( void \* amb\_hal\_base\_address ) [static]

Get the status of the previous requested cortex pll frequency change.

### Note

A new cortex pll frequency change may be requested after this function returns AMB\_HAL\_SUCCESS.

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

## Return values

AMB_HAL_SUCCE-	the pll has locked to the new frequency.
SS	
AMB_HAL_FAIL	the pll lock has failed to lock in a reasonable amount of time. some-
	thing is wrong.
AMB_HAL_RETRY	the pll has not locked yet. try again.

8.15.2.5 static INLINE amb\_hal\_success\_t amb\_enable\_cortex\_clock\_observation ( void \* amb\_hal\_base\_address ) [static]

Enable observation of cortex clock.

### Note

A divided by 16 version of the clock may be observed on the xx\_clk\_si pin.

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

## Return values

AMB_HAL_SUCCE-	Always returns success
SS	

8.15.2.6 static INLINE amb\_clock\_frequency\_t amb\_get\_axi\_clock\_frequency( void \* amb\_hal\_base\_address ) [static]

Get the AXI clock frequency.

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

8.15 Cortex 55

amb_clock	The AXI clock frequency.
frequency_t	

# 8.16 Face Detection

### **Functions**

- static INLINE amb\_hal\_success\_t amb\_set\_fdet\_clock\_frequency (void \*amb\_hal\_base\_address, amb\_clock\_frequency\_t amb\_clock\_frequency)
- static INLINE amb\_clock\_frequency\_t amb\_get\_fdet\_clock\_frequency (void \*amb\_hal\_base\_address)
- static INLINE amb\_clock\_frequency\_t amb\_get\_fdet\_clock\_configuration (void \*amb\_hal\_base\_address, amb\_divider\_configuration\_t \*amb\_fdet\_divider\_configuration)

# 8.16.1 Detailed Description

API calls to change/query the frequency of the face detection unit.

## 8.16.2 Function Documentation

8.16.2.1 static INLINE amb\_hal\_success\_t amb\_set\_fdet\_clock\_frequency ( void \* amb\_hal\_base\_address, amb\_clock\_frequency\_t amb\_clock\_frequency )

[static]

Set the clock frequency of the face detection unit.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_clock	New fdet frequency
	frequency	

#### **Return values**

AMB_HAL_SUCCE-	the new frequency has been set
SS	
AMB_HAL_FAIL	the new requested frequency is not valid

8.16.2.2 static INLINE amb\_clock\_frequency\_t amb\_get\_fdet\_clock\_frequency ( void \* amb\_hal\_base\_address ) [static]

Get the frequency of the fdet clock.

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

## Return values

amb_clock	Requested clock frequency
frequency_t	

8.16.2.3 static INLINE amb\_clock\_frequency\_t amb\_get\_fdet\_clock\_configuration ( void \* amb\_hal\_base\_address, amb\_divider\_configuration\_t \*  $amb\_fdet\_divider\_configuration \ ) \quad [ \ \texttt{static} ]$ 

Get the configuration of the fdet clock divider.

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
out	amb_fdet	Clock configuration information read from divider.
	divider	
	configuration	

AMB_HAL_SUCCE-	Always returns success.
SS	

# 8.17 Flash IO

## **Functions**

- static INLINE amb\_hal\_success\_t amb\_reset\_flash (void \*amb\_hal\_base\_-address)
- static INLINE amb\_hal\_success\_t amb\_reset\_xd (void \*amb\_hal\_base\_address)
- static INLINE amb\_hal\_success\_t amb\_reset\_cf (void \*amb\_hal\_base\_address)
- static INLINE amb\_hal\_success\_t amb\_reset\_fio (void \*amb\_hal\_base\_address)
- static INLINE amb\_hal\_success\_t amb\_reset\_all (void \*amb\_hal\_base\_address)

## 8.17.1 Detailed Description

API calls to reset the various Flash controllers.

#### 8.17.2 Function Documentation

```
8.17.2.1 static INLINE amb_hal_success_t amb_reset_flash ( void * amb_hal_base_address ) [static]
```

Reset the flash controller.

### **Parameters**

in	amb_hal Virtual address where ambhal is loaded by OS.	
	base	
	address	

## Return values

AMB_HAL_SUCCE-	always returns success.
SS	

```
8.17.2.2 static INLINE amb_hal_success_t amb_reset_xd ( void * amb_hal_base_address ) [static]
```

Reset the xd controller.

### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

8.17 Flash IO 59

## **Return values**

AMB_HAL_SUCCE-	always returns success.
SS	

8.17.2.3 static INLINE amb\_hal\_success\_t amb\_reset\_cf ( void \* amb\_hal\_base\_address ) [static]

Reset the cf controller.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

## Return values

8.17.2.4 static INLINE amb\_hal\_success\_t amb\_reset\_fio ( void \* amb\_hal\_base\_address ) [static]

Reset the fio controller.

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

# Return values

```
AMB_HAL_SUCCE- always returns success.
SS
```

8.17.2.5 static INLINE amb\_hal\_success\_t amb\_reset\_all ( void \* amb\_hal\_base\_address ) [static]

Reset the fio, cf, xd & flash controller all at once.

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

# Return values

AMB\_HAL\_SUCCE- always returns success.

SS

# 8.18 Gigabit Ethernet

### **Functions**

- static INLINE amb\_hal\_success\_t amb\_set\_gtx\_clock\_frequency (void \*amb\_hal\_base\_address, amb\_clock\_frequency\_t amb\_clock\_frequency)
- static INLINE amb\_clock\_frequency\_t amb\_get\_gtx\_clock\_frequency (void \*amb\_hal\_base\_address)
- static INLINE amb\_clock\_frequency\_t amb\_get\_gtx\_clock\_configuration (void \*amb\_hal\_base\_address, amb\_divider\_configuration\_t \*amb\_gtx\_divider\_configuration)
- static INLINE amb\_hal\_success\_t amb\_set\_gtx\_clock\_source (void \*amb\_hal\_base\_address, amb\_clock\_source\_t amb\_gtx\_clock\_source)
- static INLINE amb\_clock\_source\_t amb\_get\_gtx\_clock\_source (void \*amb\_hal\_base address)

# 8.18.1 Detailed Description

API calls to change/query the frequency of the gigabit ethernet block.

### 8.18.2 Function Documentation

8.18.2.1 static INLINE amb\_hal\_success\_t amb\_set\_gtx\_clock\_frequency ( void \* amb\_hal\_base\_address, amb\_clock\_frequency\_t amb\_clock\_frequency )
[static]

Set the clock frequency of the gigabit ethernet controller.

#### **Parameters**

_			
	in	amb_hal	Virtual address where ambhal is loaded by OS.
		base	
		address	
Ī	in	amb_clock	New gtx frequency
		frequency	

#### Return values

AMB_HAL_SUCCE-	the new frequency has been set
SS	
AMB_HAL_FAIL	the new requested frequency is not valid

8.18.2.2 static INLINE amb\_clock\_frequency\_t amb\_get\_gtx\_clock\_frequency(void \* amb\_hal\_base\_address) [static]

Get the frequency of the gtx clock.

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

## Return values

amb_clock	Requested clock frequency
frequency_t	

8.18.2.3 static INLINE amb\_clock\_frequency\_t amb\_get\_gtx\_clock\_configuration ( void \* amb\_hal\_base\_address, amb\_divider\_configuration\_t \* amb\_gtx\_divider\_configuration ) [static]

Get the configuration of the gtx clock divider.

### **Parameters**

=	in	amb_hal	Virtual address where ambhal is loaded by OS.
		base	
		address	
0	ut	amb_gtx	Clock configuration information read from divider.
		divider	
		configuration	

### Return values

AMB_HAL_SUCCE-	Always returns success.
SS	

8.18.2.4 static INLINE amb\_hal\_success\_t amb\_set\_gtx\_clock\_source ( void 
\* amb\_hal\_base\_address, amb\_clock\_source\_t amb\_gtx\_clock\_source )

[static]

Set the GTX clock source.

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_gtx	The gtx clock source.
	clock	
	source	

## Note

: Valid clock sources are: AMB\_REFERENCE\_CLOCK\_SOURCE\_CORTEX\_PL-L\_VCO and AMB\_EXTERNAL\_CLOCK\_SOURCE.

# Return values

ĺ	AMB_HAL_SUCCE-	The gtx clock source was set properly.
	SS	
	AMB_HAL_FAIL	The requested gtx clock source is not valid.

8.18.2.5 static INLINE amb\_clock\_source\_t amb\_get\_gtx\_clock\_source ( void \* amb\_hal\_base\_address ) [static]

Get the GTX clock source.

### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

amb_clock_source_t	The gtx clock source.

# 8.19 HDMI

## **Enumerations**

enum amb\_hdmi\_interface\_state\_t { AMB\_HDMI\_OFF, AMB\_HDMI\_ON }

#### **Functions**

- static INLINE amb\_clock\_frequency\_t amb\_get\_hdmi\_clock\_frequency (void \*amb\_hal\_base\_address)
- static INLINE amb\_hal\_success\_t amb\_enable\_hdmi\_clock\_observation (void \*amb hal base address)
- static INLINE amb\_clock\_frequency\_t amb\_get\_hdmi4k\_clock\_frequency (void \*amb\_hal\_base\_address)
- static INLINE amb\_hal\_success\_t amb\_enable\_hdmi4k\_clock\_observation (void \*amb\_hal\_base\_address)
- static INLINE amb\_hal\_success\_t amb\_auto\_select\_hdmi4k\_range (void \*amb\_hal\_base\_address)

## 8.19.1 Detailed Description

API calls to change/query frequency of the HDMI pll. See also Changing PLL Frequency and Changing Clock Sources.

## 8.19.2 Enumeration Type Documentation

8.19.2.1 enum amb\_hdmi\_interface\_state\_t

State of HDMI Interface.

#### **Enumerator:**

```
AMB_HDMI_OFF HDMI phy is off. 
AMB_HDMI_ON HDMI phy is on.
```

### 8.19.3 Function Documentation

8.19.3.1 static INLINE amb\_clock\_frequency\_t amb\_get\_hdmi\_clock\_frequency( void \* amb\_hal\_base\_address ) [static]

Get the hdmi clock frequency.

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

8.19 HDMI 65

### **Return values**

amb_clock	The hdmi clock frequency.
frequency_t	

8.19.3.2 static INLINE amb\_hal\_success\_t amb\_enable\_hdmi\_clock\_observation (  $void * amb\_hal\_base\_address$  ) [static]

Enable observation of hdmi clock.

Note

A divided by 16 version of the clock may be observed on the xx\_clk\_si pin.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

### Return values

AMB_HAL_SUCCE-	Always returns success
SS	

8.19.3.3 static INLINE amb\_clock\_frequency\_t amb\_get\_hdmi4k\_clock\_frequency( void \* amb\_hal\_base\_address ) [static]

Get the hdmi4k clock frequency.

## Parameters

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

## Return values

```
amb_clock_- The hdmi4k clock frequency.

frequency_t
```

8.19.3.4 static INLINE amb\_hal\_success\_t amb\_enable\_hdmi4k\_clock\_observation (void \* amb\_hal\_base\_address) [static]

Enable observation of hdmi4k clock.

# Note

A divided by 16 version of the clock may be observed on the xx\_clk\_si pin.

# **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

### Return values

AMB_HAL_SUCCE-	Always returns success
SS	

8.19.3.5 static INLINE amb\_hal\_success\_t amb\_auto\_select\_hdmi4k\_range ( void \* amb\_hal\_base\_address ) [static]

Auto select range of hdmi4k pll.

Automatically set the hdmi4k pll range.

### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

AMB_HAL_SUCCE-	Returns success if a valid range was found
SS	
AMB_HAL_FAIL	Returns fail if a valid range was not found

8.20 IDSP 67

## 8.20 IDSP

### **Functions**

• static INLINE amb\_hal\_success\_t amb\_get\_idsp\_pll\_configuration (void \*amb\_hal\_base\_address, amb\_pll\_configuration\_t \*amb\_idsp\_pll\_configuration)

- static INLINE amb\_clock\_frequency\_t amb\_get\_idsp\_clock\_frequency (void \*amb\_hal\_base\_address)
- static INLINE amb\_hal\_success\_t amb\_get\_idsp\_pll\_lock\_status (void \*amb\_hal\_base\_address)
- static INLINE amb\_hal\_success\_t amb\_enable\_idsp\_clock\_observation (void \*amb hal base address)
- static INLINE amb\_clock\_frequency\_t amb\_get\_vout\_clock\_frequency (void \*amb\_hal\_base\_address)

### 8.20.1 Detailed Description

API calls to query the frequency of the idsp clock. The idsp clock frequency cannot be set directly. It is changed by AMBHAL when the operating mode is changed. See also Operating Mode.

### 8.20.2 Function Documentation

8.20.2.1 static INLINE amb\_hal\_success\_t amb\_get\_idsp\_pll\_configuration ( void \* amb\_hal\_base\_address, amb\_pll\_configuration\_t \* amb\_idsp\_pll\_configuration ) [static]

Get the current idsp pll configuration.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
out	amb_idsp	pll configuration information read from pll registers.
	pll	
	configuration	

AMB_HAL_SUCCE-	Always returns success.
SS	

8.20.2.2 static INLINE amb\_clock\_frequency\_t amb\_get\_idsp\_clock\_frequency( void \* amb\_hal\_base\_address) [static]

Get the current idsp pll frequency.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

### Return values

amb_clock	Requested clock frequency.
frequency_t	

8.20.2.3 static INLINE amb\_hal\_success\_t amb\_get\_idsp\_pll\_lock\_status ( void \* amb\_hal\_base\_address ) [static]

Get the status of the previous requested idsp pll frequency change.

### Note

A new idsp pll frequency change may be requested after this function returns AMB\_HAL\_SUCCESS.

# **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

### Return values

AMB_HAL_SUCCE-	the pll has locked to the new frequency.
SS	
AMB_HAL_FAIL	the pll lock has failed to lock in a reasonable amount of time. some-
	thing is wrong.
AMB_HAL_RETRY	the pll has not locked yet. try again.

8.20.2.4 static INLINE amb\_hal\_success\_t amb\_enable\_idsp\_clock\_observation ( void \* amb\_hal\_base\_address ) [static]

Enable observation of idsp clock.

8.20 IDSP 69

# Note

A divided by 16 version of the clock may be observed on the xx\_clk\_si pin.

# **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

### Return values

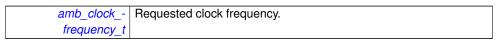
AMB_HAL_SUCCE-	Always returns success
SS	

8.20.2.5 static INLINE amb\_clock\_frequency\_t amb\_get\_vout\_clock\_frequency( void \* amb\_hal\_base\_address ) [static]

Get the current vout pll frequency.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	



# 8.21 Infrared

### **Functions**

- static INLINE amb\_hal\_success\_t amb\_set\_ir\_clock\_frequency (void \*amb\_hal\_base\_address, amb\_clock\_frequency\_t amb\_clock\_frequency)
- static INLINE amb\_clock\_frequency\_t amb\_get\_ir\_clock\_frequency (void \*amb\_hal\_base\_address)
- static INLINE amb\_clock\_frequency\_t amb\_get\_ir\_clock\_configuration (void \*amb\_hal\_base\_address, amb\_divider\_configuration\_t \*amb\_ir\_divider\_configuration)

# 8.21.1 Detailed Description

API calls to change/query the frequency of the IR controller.

# 8.21.2 Function Documentation

8.21.2.1 static INLINE amb\_hal\_success\_t amb\_set\_ir\_clock\_frequency ( void \* amb\_hal\_base\_address, amb\_clock\_frequency\_t amb\_clock\_frequency )
[static]

Set the infrared clock frequency.

#### **Parameters**

	in	amb_hal	Virtual address where ambhal is loaded by OS.
		base	
		address	
Ī	in	amb_clock	New infrared clock frequency
		frequency	

#### **Return values**

AMB_HAL_SUCCE-	The new frequency has been set
SS	
AMB_HAL_FAIL	The new frequency is invalid and has not been set

8.21.2.2 static INLINE amb\_clock\_frequency\_t amb\_get\_ir\_clock\_frequency ( void \* amb\_hal\_base\_address ) [static]

Get the frequency of the ir clock.

8.21 Infrared 71

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

## Return values

amb_clock	Requested clock frequency.
frequency_t	

8.21.2.3 static INLINE amb\_clock\_frequency\_t amb\_get\_ir\_clock\_configuration ( void \* amb\_hal\_base\_address, amb\_divider\_configuration\_t \* amb\_ir\_divider\_configuration ) [static]

Get the configuration of the ir clock divider.

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
out	amb_ir	Clock configuration information read from divider.
	divider	
	configuration	

AMB_HAL_SUCCE-	Always returns success.
SS	

### 8.22 IO Pads Control

### **Data Structures**

struct amb\_ioctrl\_configuration\_t
 IO pad configuration.

### **Enumerations**

- enum amb\_ioctrl\_drive\_strength\_t { AMB\_IOCTRL\_DRIVE\_STRENGTH\_2MA, AMB\_IOCTRL\_DRIVE\_STRENGTH\_8MA, AMB\_IOCTRL\_DRIVE\_STRENGT-H 4MA, AMB\_IOCTRL\_DRIVE\_STRENGTH\_12MA }
- enum amb\_ioctrl\_pullupdown\_t { AMB\_IOCTRL\_PULLUPDOWN\_DISABLED, AMB\_IOCTRL\_PULLUP\_ENABLED, AMB\_IOCTRL\_PULLDOWN\_ENABLED}
- enum amb\_ioctrl\_input\_type\_t { AMB\_IOCTRL\_CMOS\_INPUT\_TYPE, AMB\_I-OCTRL\_SCHMITT\_INPUT\_TYPE }
- enum amb\_ioctrl\_slew\_rate\_t { AMB\_IOCTRL\_FAST\_SLEW\_RATE, AMB\_IOCTRL SLOW SLEW RATE}

#### **Functions**

- static INLINE amb\_hal\_success\_t amb\_set\_misc1\_ioctrl\_drive\_strength (void \*amb\_hal\_base\_address, amb\_ioctrl\_drive\_strength\_t amb\_ioctrl\_drive\_strength)
- static INLINE amb\_hal\_success\_t amb\_set\_misc1\_ioctrl\_pullupdown (void \*amb\_hal\_base\_address, amb\_ioctrl\_pullupdown\_t amb\_ioctrl\_pullupdown)
- static INLINE amb\_hal\_success\_t amb\_set\_misc1\_ioctrl\_input\_type (void \*amb-hal\_base\_address, amb\_ioctrl\_input\_type\_t amb\_ioctrl\_input\_type)
- static INLINE amb\_hal\_success\_t amb\_get\_misc1\_ioctrl\_configuration (void \*amb\_hal\_base\_address, amb\_ioctrl\_configuration\_t \*amb\_ioctrl\_configuration)
- static INLINE amb\_hal\_success\_t amb\_set\_misc2\_ioctrl\_drive\_strength (void \*amb\_hal\_base\_address, amb\_ioctrl\_drive\_strength\_t amb\_ioctrl\_drive\_strength)
- static INLINE amb\_hal\_success\_t amb\_set\_misc2\_ioctrl\_pullupdown (void \*amb\_hal\_base\_address, amb\_ioctrl\_pullupdown\_t amb\_ioctrl\_pullupdown)
- static INLINE amb\_hal\_success\_t amb\_set\_misc2\_ioctrl\_input\_type (void \*amb-hal\_base\_address, amb\_ioctrl\_input\_type\_t amb\_ioctrl\_input\_type)
- static INLINE amb\_hal\_success\_t amb\_get\_misc2\_ioctrl\_configuration (void \*amb hal base address, amb ioctrl configuration t \*amb ioctrl configuration)
- static INLINE amb\_hal\_success\_t amb\_set\_smioa\_ioctrl\_drive\_strength (void \*amb\_hal\_base\_address, amb\_ioctrl\_drive\_strength\_t amb\_ioctrl\_drive\_strength)
- static INLINE amb\_hal\_success\_t amb\_set\_smioa\_ioctrl\_pullupdown (void \*amb\_hal\_base\_address, amb\_ioctrl\_pullupdown\_t amb\_ioctrl\_pullupdown)
- static INLINE amb\_hal\_success\_t amb\_set\_smioa\_ioctrl\_input\_type (void \*amb-hal base address, amb ioctrl input type t amb ioctrl input type)

- static INLINE amb\_hal\_success\_t amb\_get\_smioa\_ioctrl\_configuration (void \*amb\_hal\_base\_address, amb\_ioctrl\_configuration\_t \*amb\_ioctrl\_configuration)
- static INLINE amb\_hal\_success\_t amb\_set\_smiob\_ioctrl\_drive\_strength (void \*amb\_hal\_base\_address, amb\_ioctrl\_drive\_strength\_t amb\_ioctrl\_drive\_strength)
- static INLINE amb\_hal\_success\_t amb\_set\_smiob\_ioctrl\_pullupdown (void \*amb\_hal\_base\_address, amb\_ioctrl\_pullupdown\_t amb\_ioctrl\_pullupdown)
- static INLINE amb\_hal\_success\_t amb\_set\_smiob\_ioctrl\_input\_type (void \*amb-hal\_base\_address, amb\_ioctrl\_input\_type\_t amb\_ioctrl\_input\_type)
- static INLINE amb\_hal\_success\_t amb\_get\_smiob\_ioctrl\_configuration (void \*amb hal base address, amb ioctrl configuration t \*amb ioctrl configuration)
- static INLINE amb\_hal\_success\_t amb\_set\_smioc\_ioctrl\_drive\_strength (void \*amb\_hal\_base\_address, amb\_ioctrl\_drive\_strength\_t amb\_ioctrl\_drive\_strength)
- static INLINE amb\_hal\_success\_t amb\_set\_smioc\_ioctrl\_pullupdown (void \*amb\_hal\_base\_address, amb\_ioctrl\_pullupdown\_t amb\_ioctrl\_pullupdown)
- static INLINE amb\_hal\_success\_t amb\_set\_smioc\_ioctrl\_input\_type (void \*amb-hal\_base\_address, amb\_ioctrl\_input\_type\_t amb\_ioctrl\_input\_type)
- static INLINE amb\_hal\_success\_t amb\_get\_smioc\_ioctrl\_configuration (void \*amb\_hal\_base\_address, amb\_ioctrl\_configuration\_t \*amb\_ioctrl\_configuration)
- static INLINE amb\_hal\_success\_t amb\_set\_smiod\_ioctrl\_drive\_strength (void \*amb\_hal\_base\_address, amb\_ioctrl\_drive\_strength\_t amb\_ioctrl\_drive\_strength)
- static INLINE amb\_hal\_success\_t amb\_set\_smiod\_ioctrl\_pullupdown (void \*amb hal base address, amb ioctrl pullupdown t amb ioctrl pullupdown)
- static INLINE amb\_hal\_success\_t amb\_set\_smiod\_ioctrl\_input\_type (void \*amb-hal\_base\_address, amb\_ioctrl\_input\_type\_t amb\_ioctrl\_input\_type)
- static INLINE amb\_hal\_success\_t amb\_get\_smiod\_ioctrl\_configuration (void \*amb\_hal\_base\_address, amb\_ioctrl\_configuration\_t \*amb\_ioctrl\_configuration)
- static INLINE amb\_hal\_success\_t amb\_set\_vd1\_ioctrl\_drive\_strength (void \*amb\_hal\_base\_address, amb\_ioctrl\_drive\_strength\_t amb\_ioctrl\_drive\_strength)
- static INLINE amb\_hal\_success\_t amb\_set\_vd1\_ioctrl\_pullupdown (void \*amb\_hal\_base address, amb\_ioctrl\_pullupdown t amb\_ioctrl\_pullupdown)
- static INLINE amb\_hal\_success\_t amb\_set\_vd1\_ioctrl\_input\_type (void \*amb\_-hal\_base\_address, amb\_ioctrl\_input\_type\_t amb\_ioctrl\_input\_type)
- static INLINE amb\_hal\_success\_t amb\_get\_vd1\_ioctrl\_configuration (void \*amb-hal\_base\_address, amb\_ioctrl\_configuration\_t \*amb\_ioctrl\_configuration)
- static INLINE amb\_hal\_success\_t amb\_set\_sensor\_ioctrl\_drive\_strength (void \*amb\_hal\_base\_address, amb\_ioctrl\_drive\_strength\_t amb\_ioctrl\_drive\_strength)
- static INLINE amb\_hal\_success\_t amb\_set\_sensor\_ioctrl\_pullupdown (void \*amb hal base address, amb ioctrl pullupdown t amb ioctrl pullupdown)
- static INLINE amb\_hal\_success\_t amb\_set\_sensor\_ioctrl\_input\_type (void \*amb\_hal\_base\_address, amb\_ioctrl\_input\_type\_t amb\_ioctrl\_input\_type)
- static INLINE amb\_hal\_success\_t amb\_get\_sensor\_ioctrl\_configuration (void \*amb\_hal\_base\_address, amb\_ioctrl\_configuration\_t \*amb\_ioctrl\_configuration)

### 8.22.1 Detailed Description

API calls to change/query the characteristics of the GPIO pads (such as drive strength/pullup/pulldown/etc).

# 8.22.2 Enumeration Type Documentation

```
8.22.2.1 enum amb ioctrl drive strength t
```

IO pad drive strength.

#### **Enumerator:**

```
AMB_IOCTRL_DRIVE_STRENGTH_2MA 2 mA Driver
AMB_IOCTRL_DRIVE_STRENGTH_8MA 8 mA Driver
AMB_IOCTRL_DRIVE_STRENGTH_4MA 4 mA Driver
AMB_IOCTRL_DRIVE_STRENGTH_12MA 12 mA Driver
```

8.22.2.2 enum amb ioctrl pullupdown t

IO pad pull up/pull down.

### **Enumerator:**

```
AMB_IOCTRL_PULLUPDOWN_DISABLED Pullup/Pulldown disabled.
AMB_IOCTRL_PULLUP_ENABLED Pullup enabled.
AMB_IOCTRL_PULLDOWN_ENABLED Pulldown enabled.
```

8.22.2.3 enum amb\_ioctrl\_input\_type\_t

IO pad type.

## **Enumerator:**

```
AMB_IOCTRL_CMOS_INPUT_TYPE cmos input pad

AMB_IOCTRL_SCHMITT_INPUT_TYPE schmitt trigger input pad
```

8.22.2.4 enum amb\_ioctrl\_slew\_rate\_t

IO pad slew rate.

## **Enumerator:**

```
AMB_IOCTRL_FAST_SLEW_RATE fast slew rate

AMB_IOCTRL_SLOW_SLEW_RATE slow slew rate
```

8.22 IO Pads Control 75

## 8.22.3 Function Documentation

8.22.3.1 static INLINE amb\_hal\_success\_t amb\_set\_misc1\_ioctrl\_drive\_strength ( void \* amb\_hal\_base\_address, amb\_ioctrl\_drive\_strength\_t amb\_ioctrl\_drive\_strength ) [static]

Set the misc1 io pad drive strength.

Note

These are the pads not covered by the other api calls.

### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_ioctrl	The drive strength of the io pad
	drive	
	strength	

#### Return values

AMB_HAL_SUCCE-	The new drive strength was set.
SS	

8.22.3.2 static INLINE amb\_hal\_success\_t amb\_set\_misc1\_ioctrl\_pullupdown ( void 
\* amb\_hal\_base\_address, amb\_ioctrl\_pullupdown\_t amb\_ioctrl\_pullupdown )

[static]

Set the misc1 io pad pullup or pulldown.

Note

These are the pads not covered by the other api calls.

#### **Parameters**

_			
	in	amb_hal	Virtual address where ambhal is loaded by OS.
		base	
		address	
Ī	in	amb_ioctrl	The pullup/pulldown of the io pad
		pullupdown	

AMB_HAL_SUCCE-	The new pullup/pulldown was set.
SS	

8.22.3.3 static INLINE amb\_hal\_success\_t amb\_set\_misc1\_ioctrl\_input\_type ( void \* amb\_hal\_base\_address, amb\_ioctrl\_input\_type\_t amb\_ioctrl\_input\_type )
[static]

Set the misc1 io pad input type.

#### Note

These are the pads not covered by the other api calls.

### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_ioctrl	The input type of the io pad
	input_type	

### Return values

AMB_HAL_SUCCE-	The new input type was set.
SS	

8.22.3.4 static INLINE amb\_hal\_success\_t amb\_get\_misc1\_ioctrl\_configuration ( void \* amb\_hal\_base\_address, amb\_ioctrl\_configuration\_t \* amb\_ioctrl\_configuration ) [static]

Get the misc1 io pad configuration.

### Note

These are the pads not covered by the other api calls.

### **Parameters**

_			
	in	amb_hal	Virtual address where ambhal is loaded by OS.
		base	
		address	
Γ	out	amb_ioctrl	The current configuration of the io pad
		configuration	

8.22.3.5 static INLINE amb\_hal\_success\_t amb\_set\_misc2\_ioctrl\_drive\_strength ( void \* amb\_hal\_base\_address, amb\_ioctrl\_drive\_strength\_t amb\_ioctrl\_drive\_strength ) [static]

Set the misc2 io pad drive strength.

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_ioctrl	The drive strength of the io pad
	drive	
	strength	

#### **Return values**

AMB_HAL_SUCCE-	The new drive strength was set.
SS	

8.22.3.6 static INLINE amb\_hal\_success\_t amb\_set\_misc2\_ioctrl\_pullupdown ( void \* amb\_hal\_base\_address, amb\_ioctrl\_pullupdown\_t amb\_ioctrl\_pullupdown )
[static]

Set the misc2 io pad pullup or pulldown.

#### **Parameters**

i	n	amb_hal	Virtual address where ambhal is loaded by OS.
		base	
		address	
i	n	amb_ioctrl	The pullup/pulldown of the io pad
		pullupdown	

### **Return values**

AMB_HAL_SUCCE-	The new pullup/pulldown was set.
SS	

8.22.3.7 static INLINE amb\_hal\_success\_t amb\_set\_misc2\_ioctrl\_input\_type ( void \* amb\_hal\_base\_address, amb\_ioctrl\_input\_type\_t amb\_ioctrl\_input\_type )

[static]

Set the misc2 io pad input type.

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_ioctrl	The input type of the io pad
	input_type	

### Return values

```
AMB_HAL_SUCCE- SS The new input type was set.
```

8.22.3.8 static INLINE amb\_hal\_success\_t amb\_get\_misc2\_ioctrl\_configuration ( void \* amb\_hal\_base\_address, amb\_ioctrl\_configuration\_t \* amb\_ioctrl\_configuration ) [static]

Get the misc2 io pad configuration.

### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
out	amb_ioctrl	The current configuration of the io pad
	configuration	

8.22.3.9 static INLINE amb\_hal\_success\_t amb\_set\_smioa\_ioctrl\_drive\_strength ( void \* amb\_hal\_base\_address, amb\_ioctrl\_drive\_strength\_t amb\_ioctrl\_drive\_strength ) [static]

Set the smioa io pad drive strength.

# Parameters

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_ioctrl	The drive strength of the io pad
	drive	
	strength	

### Return values

```
AMB_HAL_SUCCE- The new drive strength was set.
```

Set the smioa io pad pullup or pulldown.

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_ioctrl	The pullup/pulldown of the io pad
	pullupdown	

#### Return values

AMB_HAL_SUCCE-	The new pullup/pulldown was set.
SS	

8.22.3.11 static INLINE amb\_hal\_success\_t amb\_set\_smioa\_ioctrl\_input\_type ( void \* amb\_hal\_base\_address, amb\_ioctrl\_input\_type\_t amb\_ioctrl\_input\_type ) [static]

Set the smioa io pad input type.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_ioctrl	The input type of the io pad
	input_type	

# Return values

```
AMB_HAL_SUCCE- The new input type was set.
```

8.22.3.12 static INLINE amb\_hal\_success\_t amb\_get\_smioa\_ioctrl\_configuration ( void \* amb\_hal\_base\_address, amb\_ioctrl\_configuration\_t \* amb\_ioctrl\_configuration ) [static]

Get the smioa io pad configuration.

in	amb_hal base -	Virtual address where ambhal is loaded by OS.
	address	
out	amb_ioctrl	The current configuration of the io pad
	configuration	

8.22.3.13 static INLINE amb\_hal\_success\_t amb\_set\_smiob\_ioctrl\_drive\_strength ( void \* amb\_hal\_base\_address, amb\_ioctrl\_drive\_strength\_t amb\_ioctrl\_drive\_strength ) [static]

Set the smiob io pad drive strength.

#### **Parameters**

i	n	amb_hal	Virtual address where ambhal is loaded by OS.
		base	
		address	
i	n	amb_ioctrl	The drive strength of the io pad
		drive	
		strength	

#### Return values

AMB_HAL_SUCCE-	The new drive strength was set.
SS	

Set the smiob io pad pullup or pulldown.

## **Parameters**

i	n	amb_hal	Virtual address where ambhal is loaded by OS.
		base	
		address	
i	n.	amb_ioctrl	The pullup/pulldown of the io pad
		pullupdown	

## Return values

```
AMB_HAL_SUCCE- The new pullup/pulldown was set.
```

8.22.3.15 static INLINE amb\_hal\_success\_t amb\_set\_smiob\_ioctrl\_input\_type ( void \* amb\_hal\_base\_address, amb\_ioctrl\_input\_type\_t amb\_ioctrl\_input\_type ) [static]

Set the smiob io pad input type.

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_ioctrl	The input type of the io pad
	input_type	

#### Return values

AMB_HAL_SUCCE-	The new input type was set.
SS	

8.22.3.16 static INLINE amb\_hal\_success\_t amb\_get\_smiob\_ioctrl\_configuration ( void \* amb\_hal\_base\_address, amb\_ioctrl\_configuration\_t \* amb\_ioctrl\_configuration ) [static]

Get the smiob io pad configuration.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
out	amb_ioctrl	The current configuration of the io pad
	configuration	

8.22.3.17 static INLINE amb\_hal\_success\_t amb\_set\_smioc\_ioctrl\_drive\_strength ( void \* amb\_hal\_base\_address, amb\_ioctrl\_drive\_strength\_t amb\_ioctrl\_drive\_strength ) [static]

Set the smioc io pad drive strength.

### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_ioctrl	The drive strength of the io pad
	drive	
	strength	

AMB_HAL_SUCCE-	The new drive strength was set.	
SS		

Set the smioc io pad pullup or pulldown.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_ioctrl	The pullup/pulldown of the io pad
	pullupdown	

#### Return values

```
AMB_HAL_SUCCE- The new pullup/pulldown was set.

SS
```

8.22.3.19 static INLINE amb\_hal\_success\_t amb\_set\_smioc\_ioctrl\_input\_type ( void \* amb\_hal\_base\_address, amb\_ioctrl\_input\_type\_t amb\_ioctrl\_input\_type )

[static]

Set the smioc io pad input type.

## **Parameters**

in	amb_hal base address	Virtual address where ambhal is loaded by OS.
in		The input type of the io pad

#### Return values

```
AMB_HAL_SUCCE- The new input type was set.

SS
```

8.22.3.20 static INLINE amb\_hal\_success\_t amb\_get\_smioc\_ioctrl\_configuration ( void \* amb\_hal\_base\_address, amb\_ioctrl\_configuration\_t \* amb\_ioctrl\_configuration ) [static]

Get the smioc io pad configuration.

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
out	amb_ioctrl	The current configuration of the io pad
	configuration	

8.22.3.21 static INLINE amb\_hal\_success\_t amb\_set\_smiod\_ioctrl\_drive\_strength ( void \* amb\_hal\_base\_address, amb\_ioctrl\_drive\_strength\_t amb\_ioctrl\_drive\_strength ) [static]

Set the smiod io pad drive strength.

### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_ioctrl	The drive strength of the io pad
	drive	
	strength	

## **Return values**

AMB_HAL_SUCCE-	The new drive strength was set.
SS	

Set the smiod io pad pullup or pulldown.

### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_ioctrl	The pullup/pulldown of the io pad
	pullupdown	

AMB_HAL_SUCCE-	The new pullup/pulldown was set.
SS	

8.22.3.23 static INLINE amb\_hal\_success\_t amb\_set\_smiod\_ioctrl\_input\_type ( void \* amb\_hal\_base\_address, amb\_ioctrl\_input\_type\_t amb\_ioctrl\_input\_type ) [static]

Set the smiod io pad input type.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_ioctrl	The input type of the io pad
	input_type	

### Return values

```
AMB_HAL_SUCCE- The new input type was set.

SS
```

8.22.3.24 static INLINE amb\_hal\_success\_t amb\_get\_smiod\_ioctrl\_configuration ( void \* amb\_hal\_base\_address, amb\_ioctrl\_configuration\_t \* amb\_ioctrl\_configuration ) [static]

Get the smiod io pad configuration.

## **Parameters**

in	base	Virtual address where ambhal is loaded by OS.
	address	
out	amb_ioctrl	The current configuration of the io pad
	configuration	

8.22.3.25 static INLINE amb\_hal\_success\_t amb\_set\_vd1\_ioctrl\_drive\_strength ( void \* amb\_hal\_base\_address, amb\_ioctrl\_drive\_strength\_t amb\_ioctrl\_drive\_strength ) [static]

Set the vd1 io pad drive strength.

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_ioctrl	The drive strength of the io pad
	drive	
	strength	

8.22 IO Pads Control 85

## Return values

AMB_HAL_SUCCE-	The new drive strength was set.
SS	

8.22.3.26 static INLINE amb\_hal\_success\_t amb\_set\_vd1\_ioctrl\_pullupdown ( void \* amb\_hal\_base\_address, amb\_ioctrl\_pullupdown\_t amb\_ioctrl\_pullupdown ) [static]

Set the vd1 io pad pullup or pulldown.

### **Parameters**

in	amb_hal base address	Virtual address where ambhal is loaded by OS.
in	amb_ioctrl	The pullup/pulldown of the io pad
	pullupdown	

#### Return values

8.22.3.27 static INLINE amb\_hal\_success\_t amb\_set\_vd1\_ioctrl\_input\_type ( void \* amb\_hal\_base\_address, amb\_ioctrl\_input\_type\_t amb\_ioctrl\_input\_type ) [static]

Set the vd1 io pad input type.

## **Parameters**

in	amb_hal base address	Virtual address where ambhal is loaded by OS.
in	amb_ioctrl	The input type of the io pad
	input_type	

AMB_HAL_SUCCE-	The new input type was set.
SS	

8.22.3.28 static INLINE amb\_hal\_success\_t amb\_get\_vd1\_ioctrl\_configuration ( void \* amb\_hal\_base\_address, amb\_ioctrl\_configuration\_t \* amb\_ioctrl\_configuration ) [static]

Get the vd1 io pad configuration.

#### **Parameters**

ſ	in	amb_hal	Virtual address where ambhal is loaded by OS.
		base	
		address	
Ī	out	amb_ioctrl	The current configuration of the io pad
		configuration	

8.22.3.29 static INLINE amb\_hal\_success\_t amb\_set\_sensor\_ioctrl\_drive\_strength ( void \* amb\_hal\_base\_address, amb\_ioctrl\_drive\_strength\_t amb\_ioctrl\_drive\_strength ) [static]

Set the sensor io pad drive strength.

### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_ioctrl	The drive strength of the io pad
	drive	
	strength	

### Return values

```
AMB_HAL_SUCCE- The new drive strength was set.

SS
```

8.22.3.30 static INLINE amb\_hal\_success\_t amb\_set\_sensor\_ioctrl\_pullupdown ( void \* amb\_hal\_base\_address, amb\_ioctrl\_pullupdown\_t amb\_ioctrl\_pullupdown ) [static]

Set the sensor io pad pullup or pulldown.

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_ioctrl	The pullup/pulldown of the io pad
	pullupdown	

8.22 IO Pads Control 87

## Return values

AMB_HAL_SUCCE-	The new pullup/pulldown was set.
SS	

Set the sensor io pad input type.

### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_ioctrl	The input type of the io pad
	input_type	

#### Return values

8.22.3.32 static INLINE amb\_hal\_success\_t amb\_get\_sensor\_ioctrl\_configuration ( void \* amb\_hal\_base\_address, amb\_ioctrl\_configuration\_t \* amb\_ioctrl\_configuration ) [static]

Get the sensor io pad configuration.

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
out	amb_ioctrl	The current configuration of the io pad
	configuration	

# 8.23 LCD

### **Functions**

- static INLINE amb\_hal\_success\_t amb\_set\_lcd\_clock\_source (void \*amb\_hal\_base\_address, amb\_clock\_source\_t amb\_clock\_source, amb\_clock\_frequency\_t amb\_clock\_frequency)
- static INLINE amb\_hal\_success\_t amb\_set\_lcd\_clock\_frequency (void \*amb\_hal\_base\_address, amb\_clock\_frequency\_t amb\_lcd\_clock\_frequency)
- static INLINE amb\_hal\_success\_t amb\_get\_lcd\_pll\_configuration (void \*amb\_hal\_base\_address, amb\_pll\_configuration\_t \*amb\_lcd\_pll\_configuration)
- static INLINE amb\_clock\_frequency\_t amb\_get\_lcd\_clock\_frequency (void \*amb\_hal\_base\_address)
- static INLINE amb\_hal\_success\_t amb\_get\_lcd\_pll\_lock\_status (void \*amb\_hal\_base\_address)
- static INLINE amb\_hal\_success\_t amb\_enable\_lcd\_clock\_observation (void \*amb\_hal\_base\_address)

# 8.23.1 Detailed Description

API calls to change/query frequency of the LCD pll. See also Changing PLL Frequency and Changing Clock Sources.

# 8.23.2 Function Documentation

Set the clock source for lcd.

# Parameters

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_clock	the new clock source.
	source	
in	amb_clock	the clock frequency of the new source.
	frequency	

### Note

The amb\_clock\_frequency only needs to be specified for the clock sources AMB\_-PLL\_REFERENCE\_CLOCK\_SOURCE\_CLK\_SI and AMB\_PLL\_REFERENCE\_C-LOCK\_SOURCE\_LVDS\_IDSP\_SCLK. Specify an amb\_clock\_frequency of 0 for all other clock sources. The topic Changing Clock Sources covers this in more details.

8.23 LCD 89

## Return values

ĺ	AMB_HAL_SUCCE-	a new clock source has been set.
	SS	
İ	AMB_HAL_FAIL	the new clock source is not supported.

8.23.2.2 static INLINE amb\_hal\_success\_t amb\_set\_lcd\_clock\_frequency ( void \* amb\_hal\_base\_address, amb\_clock\_frequency\_t amb\_lcd\_clock\_frequency ) [static]

Set the lcd pll frequency.

### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_lcd	The requested frequency.
	clock	
	frequency	

## **Return values**

AMB_HAL_SUCCE-	the new requested pll frequency is valid and it has been pro-
SS	grammed.
AMB_HAL_FAIL	the new pll frequency requested is not supported.
AMB_HAL_RETRY	a previous pll frequency change request is still outstanding.

8.23.2.3 static INLINE amb\_hal\_success\_t amb\_get\_lcd\_pll\_configuration ( void \*  $amb\_hal\_base\_address, \ amb\_pll\_configuration\_t * amb\_lcd\_pll\_configuration \ )$  [static]

Get the current lcd pll configuration.

# Parameters

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
out	amb_lcd_pll-	Sensor pll configuration information read from pll registers.
	configuration	

_		
	AMR HAL SUCCE-	always returns success.
	/IIVID_III/IL_COUCL	diways returns success.
	ee.	
	33	

8.23.2.4 static INLINE amb\_clock\_frequency\_t amb\_get\_lcd\_clock\_frequency ( void \* amb\_hal\_base\_address ) [static]

Get the current lcd pll frequency.

### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

### Return values

amb_clock	Requested clock frequency.
frequency_t	

8.23.2.5 static INLINE amb\_hal\_success\_t amb\_get\_lcd\_pll\_lock\_status ( void \* amb\_hal\_base\_address ) [static]

Get the status of the previous requested lcd pll frequency change.

### Note

A new lcd pll frequency change may be requested after this function returns AMB\_HAL\_SUCCESS.

# **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

### Return values

AMB_HAL_SUCCE-	the pll has locked to the new frequency.
SS	
AMB_HAL_FAIL	the pll lock has failed to lock in a reasonable amount of time. some-
	thing is wrong.
AMB_HAL_RETRY	the pll has not locked yet. try again.

8.23.2.6 static INLINE amb\_hal\_success\_t amb\_enable\_lcd\_clock\_observation ( void \* amb\_hal\_base\_address ) [static]

Enable observation of lcd clock.

8.23 LCD 91

# Note

A divided by 16 version of the clock may be observed on the xx\_clk\_si pin.

# **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

AMB_HAL_SUCCE-	Always returns success
SS	

# 8.24 LVDS

## **Functions**

- static INLINE amb\_hal\_success\_t amb\_set\_lvds\_pad\_mode (void \*amb\_hal\_base\_address, amb\_lvds\_pad\_mode\_t amb\_lvds\_pad\_mode)
- static INLINE amb\_lvds\_pad\_mode\_t amb\_get\_lvds\_pad\_mode (void \*amb\_hal-\_base\_address)

# 8.24.1 Detailed Description

API calls to change/query the mode of the LVDS pads.

## 8.24.2 Function Documentation

8.24.2.1 static INLINE amb\_hal\_success\_t amb\_set\_lvds\_pad\_mode ( void \* amb\_hal\_base\_address, amb\_lvds\_pad\_mode\_t amb\_lvds\_pad\_mode ) [static]

Set the lvds pad mode.

#### **Parameters**

	in	amb_hal	Virtual address where ambhal is loaded by OS.
		base	
		address	
ſ	in	amb_lvds	The mode for lvds pads
		pad_mode	

### Return values

AMB_HAL_SUCCE-	The new pad mode was set.
SS	
AMB_HAL_FAIL	The requested pad mode is not valid.

8.24.2.2 static INLINE amb\_lvds\_pad\_mode\_t amb\_get\_lvds\_pad\_mode ( void \* amb\_hal\_base\_address ) [static]

Get the lvds pad mode.

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

8.24 LVDS 93

amb_lvds_pad	The current pad mode setting.
mode_t	

# 8.25 Motor

## **Functions**

- static INLINE amb\_hal\_success\_t amb\_set\_motor\_clock\_frequency (void \*amb\_hal\_base\_address, amb\_clock\_frequency\_t amb\_clock\_frequency)
- static INLINE amb\_clock\_frequency\_t amb\_get\_motor\_clock\_frequency (void \*amb\_hal\_base\_address)
- static INLINE amb\_clock\_frequency\_t amb\_get\_motor\_clock\_configuration (void \*amb\_hal\_base\_address, amb\_divider\_configuration\_t \*amb\_motor\_divider\_configuration)

# 8.25.1 Detailed Description

API calls to change/query the frequency of the Motor controller.

## 8.25.2 Function Documentation

8.25.2.1 static INLINE amb\_hal\_success\_t amb\_set\_motor\_clock\_frequency ( void \* amb\_hal\_base\_address, amb\_clock\_frequency\_t amb\_clock\_frequency )

[static]

Set the motor clock frequency.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_clock	New motor clock frequency
	frequency	

#### **Return values**

AMB_HAL_SUCCE-	The new frequency has been set
SS	
AMB_HAL_FAIL	The new frequency is invalid and has not been set

8.25.2.2 static INLINE amb\_clock\_frequency\_t amb\_get\_motor\_clock\_frequency( void \* amb\_hal\_base\_address ) [static]

Get the frequency of the motor clock.

8.25 Motor 95

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

## Return values

amb_clock	Requested clock frequency.
frequency_t	

8.25.2.3 static INLINE amb\_clock\_frequency\_t amb\_get\_motor\_clock\_configuration ( void \* amb\_hal\_base\_address, amb\_divider\_configuration\_t \*  $amb\_motor\_divider\_configuration \ ) \quad [static]$ 

Get the configuration of the motor clock divider.

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
out	amb_motor-	Clock configuration information read from divider.
	_divider	
	configuration	

AMB_HAL_SUCCE-	Always returns success.
SS	

# 8.26 Memory Stick

#### **Data Structures**

struct amb\_ms\_delay\_configuration\_t
 Memory stick delay configuration.

## **Typedefs**

· typedef unsigned int amb\_ms\_delay\_t

#### **Enumerations**

enum amb\_ms\_status\_t { AMB\_MS\_DISABLE, AMB\_MS\_ENABLE }

#### **Functions**

- static INLINE amb\_hal\_success\_t amb\_set\_ms\_clock\_frequency (void \*amb\_-hal\_base\_address, amb\_clock\_frequency\_t amb\_clock\_frequency)
- static INLINE amb\_clock\_frequency\_t amb\_get\_ms\_clock\_frequency (void \*amb\_hal\_base\_address)
- static INLINE amb\_hal\_success\_t amb\_set\_ms\_sclk\_delay (void \*amb\_hal\_base\_address, amb\_ms\_delay\_t amb\_ms\_delay)
- static INLINE amb\_hal\_success\_t amb\_set\_ms\_sd\_input\_delay (void \*amb\_hal\_base\_address, amb\_ms\_delay\_t amb\_ms\_delay)
- static INLINE amb\_hal\_success\_t amb\_set\_ms\_sd\_output\_delay (void \*amb\_hal\_base\_address, amb\_ms\_delay\_t amb\_ms\_delay)
- static INLINE amb\_hal\_success\_t amb\_set\_ms\_read\_delay (void \*amb\_hal\_base\_address, amb\_ms\_delay\_t amb\_ms\_delay)
- static INLINE amb\_hal\_success\_t amb\_get\_ms\_delay\_configuration (void \*amb\_hal\_base\_address, amb\_ms\_delay\_configuration\_t \*amb\_ms\_delay\_configuration)
- static INLINE amb\_hal\_success\_t amb\_set\_ms\_status (void \*amb\_hal\_base\_-address, amb\_ms\_status\_t amb\_ms\_status)
- static INLINE amb\_ms\_status\_t amb\_get\_ms\_status (void \*amb\_hal\_base\_-address)

## 8.26.1 Detailed Description

Various API calls to change/query MS clock frequency and also to control various parameters of the MS IO interface.

## 8.26.2 Enumeration Type Documentation

8.26.2.1 enum amb\_ms\_status\_t

Memory stick controller status.

#### **Enumerator:**

AMB\_MS\_DISABLE Memory Stick controller disabled.

AMB\_MS\_ENABLE Memory Stick controller enabled.

## 8.26.3 Function Documentation

8.26.3.1 static INLINE amb\_hal\_success\_t amb\_set\_ms\_clock\_frequency ( void \* amb\_hal\_base\_address, amb\_clock\_frequency\_t amb\_clock\_frequency )
[static]

Set the clock frequency of the memory stick controller.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_clock	New memory stick frequency
	frequency	

#### Return values

AMB_HAL_SUCCE-	The new frequency has been set
SS	
AMB_HAL_FAIL	The new requested frequency is not valid

8.26.3.2 static INLINE amb\_clock\_frequency\_t amb\_get\_ms\_clock\_frequency ( void \* amb\_hal\_base\_address ) [static]

Get the frequency the memory stick controller clock.

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

amb_clock	Requested clock frequency
frequency_t	

8.26.3.3 static INLINE amb\_hal\_success\_t amb\_set\_ms\_sclk\_delay ( void \* amb\_hal\_base\_address, amb\_ms\_delay\_t amb\_ms\_delay ) [static]

Set the memory stick sclk delay.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_ms	Requested delay.
	delay	

#### Return values

```
AMB_HAL_SUCCE- always returns success.

SS
```

8.26.3.4 static INLINE amb\_hal\_success\_t amb\_set\_ms\_sd\_input\_delay ( void \* amb\_hal\_base\_address, amb\_ms\_delay\_t amb\_ms\_delay ) [static]

Set the memory stick sd input delay.

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_ms	Requested delay.
	delay	

## Return values

AMB_HAL_SUCCE-	always returns success.
SS	

8.26.3.5 static INLINE amb\_hal\_success\_t amb\_set\_ms\_sd\_output\_delay ( void \* amb\_hal\_base\_address, amb\_ms\_delay\_t amb\_ms\_delay ) [static]

Set the memory stick sd output delay.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_ms	Requested delay.
	delay	

## Return values

AMB_HAL_SUCCE-	always returns success.
SS	

8.26.3.6 static INLINE amb\_hal\_success\_t amb\_set\_ms\_read\_delay ( void \* amb\_hal\_base\_address, amb\_ms\_delay\_t amb\_ms\_delay ) [static]

Set the memory stick read delay.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_ms	Requested delay.
	delay	

#### **Return values**

8.26.3.7 static INLINE amb\_hal\_success\_t amb\_get\_ms\_delay\_configuration ( void \* amb\_hal\_base\_address, amb\_ms\_delay\_configuration ) [static]

Get the current memory stick delay configuration.

#### **Parameters**

_			
	in	amb_hal	Virtual address where ambhal is loaded by OS.
		base	
		address	
	out	amb_ms	Memory stick delays read from the delay configuration reg-
		delay	
		configuration	

AMB_HAL_SUCCE-	always returns success.
SS	

8.26.3.8 static INLINE amb\_hal\_success\_t amb\_set\_ms\_status ( void \* amb\_hal\_base\_address, amb\_ms\_status\_t amb\_ms\_status\_) [static]

Enable/Disable the memory stick controller.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_ms	Status of the memory stick controller
	status	

#### Return values

```
AMB_HAL_SUCCE- always returns success.

SS
```

8.26.3.9 static INLINE amb\_ms\_status\_t amb\_get\_ms\_status ( void \*  $amb\_hal\_base\_address$  ) [static]

Get the status of the memory stick controller.

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

## Return values

amb\_ms\_status\_t | Status of the memory stick controller

8.27 PWM 101

## 8.27 PWM

#### **Functions**

static INLINE amb\_hal\_success\_t amb\_set\_pwm\_clock\_frequency (void \*amb\_hal\_base\_address, amb\_clock\_frequency)

- static INLINE amb\_clock\_frequency\_t amb\_get\_pwm\_clock\_frequency (void \*amb\_hal\_base\_address)
- static INLINE amb\_clock\_frequency\_t amb\_get\_pwm\_clock\_configuration (void \*amb\_hal\_base\_address, amb\_divider\_configuration\_t \*amb\_pwm\_divider\_configuration)
- static INLINE amb\_hal\_success\_t amb\_set\_pwm\_clock\_source (void \*amb\_hal\_base\_address, amb\_clock\_source\_t amb\_clock\_source, amb\_clock\_frequency-t amb\_clock\_frequency)

## 8.27.1 Detailed Description

API calls to change/query the frequency of the PWM controller.

## 8.27.2 Function Documentation

```
8.27.2.1 static INLINE amb_hal_success_t amb_set_pwm_clock_frequency ( void * amb_hal_base_address, amb_clock_frequency_t amb_clock_frequency )
[static]
```

Set the clock frequency of the pwm.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_clock	New pwm frequency
	frequency	

#### Return values

AMB_HAL_SUCCE-	the new frequency has been set
SS	
AMB_HAL_FAIL	the new requested frequency is not valid

8.27.2.2 static INLINE amb\_clock\_frequency\_t amb\_get\_pwm\_clock\_frequency(
 void \* amb\_hal\_base\_address ) [static]

Get the frequency of the pwm clock.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

## Return values

amb_clock	Requested clock frequency
frequency_t	

8.27.2.3 static INLINE amb\_clock\_frequency\_t amb\_get\_pwm\_clock\_configuration ( void \* amb\_hal\_base\_address, amb\_divider\_configuration\_t \* amb\_pwm\_divider\_configuration ) [static]

Get the configuration of the pwm clock divider.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
out	amb_pwm	Clock configuration information read from divider.
	divider	
	configuration	

## Return values

Set the clock source of the pwm clock divider.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_clock	Clock source for the pwm divider.
	source	
in	amb_clock	Clock frequency of the selected source.
	frequency	

8.27 PWM 103

## Note

: Valid clock sources are: AMB\_REFERENCE\_CLOCK\_SOURCE\_APB or AMB\_REFERENCE\_CLOCK\_SOURCE\_LVDS\_IDSP\_SCLK. The amb\_clock\_frequency must be specified for AMB\_REFERENCE\_CLOCK\_SOURCE\_LVDS\_IDSP\_SCL-K. Otherwise it should be set to 0.

AMB_HAL_SUCCE-	The function always returns success.
SS	

## 8.28 PWM

#### **Functions**

- static INLINE amb\_hal\_success\_t amb\_set\_pwmois\_clock\_frequency (void \*amb\_hal\_base\_address, amb\_clock\_frequency\_t amb\_clock\_frequency)
- static INLINE amb\_clock\_frequency\_t amb\_get\_pwmois\_clock\_frequency (void \*amb\_hal\_base\_address)
- static INLINE amb\_clock\_frequency\_t amb\_get\_pwmois\_clock\_configuration (void \*amb\_hal\_base\_address, amb\_divider\_configuration\_t \*amb\_pwmois\_divider configuration)
- static INLINE amb\_hal\_success\_t amb\_set\_pwmois\_clock\_source (void \*amb\_hal\_base\_address, amb\_clock\_source\_t amb\_clock\_source, amb\_clock\_frequency)

## 8.28.1 Detailed Description

API calls to change/query the frequency of the PWM OIS block.

#### 8.28.2 Function Documentation

Set the clock frequency of the pwmois.

#### **Parameters**

	in	amb_hal	Virtual address where ambhal is loaded by OS.
		base	
		address	
Ī	in	amb_clock	New pwmois frequency
		frequency	

#### Return values

AMB_HAL_SUCCE-	the new frequency has been set
SS	
AMB_HAL_FAIL	the new requested frequency is not valid

8.28.2.2 static INLINE amb\_clock\_frequency\_t amb\_get\_pwmois\_clock\_frequency(
 void \* amb\_hal\_base\_address ) [static]

Get the frequency of the pwmois clock.

8.28 PWM 105

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

## Return values

amb_clock	Requested clock frequency
frequency_t	

8.28.2.3 static INLINE amb\_clock\_frequency\_t amb\_get\_pwmois\_clock\_configuration ( void \* amb\_hal\_base\_address, amb\_divider\_configuration\_t \*
amb\_pwmois\_divider\_configuration ) [static]

Get the configuration of the pwmois clock divider.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
out	amb	Clock configuration information read from divider.
	pwmois	
	divider	
	configuration	

#### **Return values**

I	AMB_HAL_SUCCE-	Always returns success.
	SS	

8.28.2.4 static INLINE amb\_hal\_success\_t amb\_set\_pwmois\_clock\_source (
 void \* amb\_hal\_base\_address, amb\_clock\_source\_t amb\_clock\_source,
 amb\_clock\_frequency\_t amb\_clock\_frequency) [static]

Set the clock source of the pwmois clock divider.

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_clock	Clock source for the pwmois divider.
	source	
in	amb_clock	Clock frequency of the selected source.
	frequency	

#### Note

: Valid clock sources are: AMB\_REFERENCE\_CLOCK\_SOURCE\_IDSP\_PLL\_VCO, AMB\_REFERENCE\_CLOCK\_SOURCE\_AUDIO\_PLL\_VCO, AMB\_REFERENCE\_CLOCK\_SOURCE\_CLOCK\_SOURCE\_LVDS\_IDSP\_SCLK. The amb\_clock\_frequency must be specified for AMB\_REFERENCE\_CLOCK\_SOURCE\_LVDS\_IDSP\_SCLK. Otherwise it should be set to 0.

AMB_HAL_SUCCE-	The function always returns success.
SS	

8.29 Reset 107

## 8.29 **Reset**

## **Functions**

static INLINE amb\_hal\_success\_t amb\_reset\_chip (void \*amb\_hal\_base\_-address)

## 8.29.1 Function Documentation

```
8.29.1.1 static INLINE amb_hal_success_t amb_reset_chip ( void * amb_hal_base_address ) [static]
```

Reset the chip.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

#### Note

This api call is implemented so that it returns AMB\_HAL\_SUCCESS but a warm reset will restart the entire system and so do not expect to do anything else after this call is made.

AMB_HAL_SUCCE-	always returns success.
SS	

## 8.30 SD

## **Functions**

- static INLINE amb\_hal\_success\_t amb\_set\_sd\_clock\_frequency (void \*amb\_hal\_base\_address, amb\_clock\_frequency\_t amb\_clock\_frequency)
- static INLINE amb\_clock\_frequency\_t amb\_get\_sd\_clock\_frequency (void \*amb-hal\_base\_address)
- static INLINE amb\_clock\_frequency\_t amb\_get\_sd\_clock\_configuration (void \*amb\_hal\_base\_address, amb\_divider\_configuration\_t \*amb\_sd\_divider\_configuration)

## 8.30.1 Detailed Description

API calls to change/query the frequency of the SD controller.

## 8.30.2 Function Documentation

8.30.2.1 static INLINE amb\_hal\_success\_t amb\_set\_sd\_clock\_frequency ( void \* amb\_hal\_base\_address, amb\_clock\_frequency\_t amb\_clock\_frequency )
[static]

Set the clock frequency of the sd controller.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_clock	New smiod frequency
	frequency	

#### **Return values**

AMB_HAL_SUCCE-	The new frequency has been set
SS	
AMB_HAL_FAIL	The new requested frequency is not valid

8.30.2.2 static INLINE amb\_clock\_frequency\_t amb\_get\_sd\_clock\_frequency ( void \* amb\_hal\_base\_address ) [static]

Get the frequency of the sd clock.

8.30 SD 109

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

## Return values

amb_clock	Requested clock frequency
frequency_t	

8.30.2.3 static INLINE amb\_clock\_frequency\_t amb\_get\_sd\_clock\_configuration ( void \* amb\_hal\_base\_address, amb\_divider\_configuration\_t \*  $amb\_sd\_divider\_configuration$ ) [static]

Get the configuration of the sd clock divider.

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
out	amb_sd	Clock configuration information read from divider.
	divider	
	configuration	

AMB_HAL_SUCCE-	Always returns success.
SS	

## 8.31 SDIO

## **Functions**

- static INLINE amb\_hal\_success\_t amb\_set\_sdio\_clock\_frequency (void \*amb\_hal\_base\_address, amb\_clock\_frequency\_t amb\_clock\_frequency)
- static INLINE amb\_clock\_frequency\_t amb\_get\_sdio\_clock\_frequency (void \*amb\_hal\_base\_address)
- static INLINE amb\_clock\_frequency\_t amb\_get\_sdio\_clock\_configuration (void \*amb\_hal\_base\_address, amb\_divider\_configuration\_t \*amb\_sdio\_divider\_configuration)

## 8.31.1 Detailed Description

API calls to change/query the frequency of the SDIO block.

## 8.31.2 Function Documentation

8.31.2.1 static INLINE amb\_hal\_success\_t amb\_set\_sdio\_clock\_frequency ( void \* amb\_hal\_base\_address, amb\_clock\_frequency\_t amb\_clock\_frequency )

[static]

Set the clock frequency of the sdio clock.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_clock	New sdio frequency
	frequency	

#### **Return values**

AMB_HAL_SUCCE-	The new frequency has been set
SS	
AMB_HAL_FAIL	The new requested frequency is not valid

8.31.2.2 static INLINE amb\_clock\_frequency\_t amb\_get\_sdio\_clock\_frequency( void \* amb\_hal\_base\_address ) [static]

Get the frequency of the sdio clock.

8.31 SDIO 111

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

## Return values

amb_clock	Requested clock frequency
frequency_t	

8.31.2.3 static INLINE amb\_clock\_frequency\_t amb\_get\_sdio\_clock\_configuration ( void \* amb\_hal\_base\_address, amb\_divider\_configuration\_t \*  $amb\_sdio\_divider\_configuration \ ) \quad [static]$ 

Get the configuration of the sdio clock divider.

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
out	amb_sdio	Clock configuration information read from divider.
	divider	
	configuration	

AMB_HAL_SUCCE-	Always returns success.
SS	

## 8.32 Sensor

#### **Enumerations**

 enum amb\_sensor\_clock\_pad\_mode\_t { AMB\_SENSOR\_CLOCK\_PAD\_OUTP-UT\_MODE, AMB\_SENSOR\_CLOCK\_PAD\_INPUT\_MODE }

#### **Functions**

- static INLINE amb\_hal\_success\_t amb\_set\_sensor\_clock\_frequency (void \*amb\_hal\_base\_address, amb\_clock\_frequency\_t amb\_sensor\_clock\_frequency)
- static INLINE amb\_hal\_success\_t amb\_get\_sensor\_pll\_configuration (void \*amb\_hal\_base\_address, amb\_pll\_configuration\_t \*amb\_sensor\_pll\_configuration)
- static INLINE amb\_clock\_frequency\_t amb\_get\_sensor\_clock\_frequency (void \*amb hal base address)
- static INLINE amb\_hal\_success\_t amb\_get\_sensor\_pll\_lock\_status (void \*amb-hal\_base\_address)
- static INLINE amb\_hal\_success\_t amb\_enable\_sensor\_clock\_observation (void \*amb\_hal\_base\_address)
- static INLINE amb\_hal\_success\_t amb\_set\_sensor\_clock\_pad\_mode (void \*amb\_hal\_base\_address, amb\_sensor\_clock\_pad\_mode\_t amb\_sensor\_clock-\_pad\_mode)
- static INLINE amb\_sensor\_clock\_pad\_mode\_t amb\_get\_sensor\_clock\_pad\_mode (void \*amb\_hal\_base\_address)

# 8.32.1 Detailed Description

API calls to change/query frequency of the Sensor pll. See also Changing PLL - Frequency and Changing Clock Sources.

## 8.32.2 Enumeration Type Documentation

8.32.2.1 enum amb\_sensor\_clock\_pad\_mode\_t

Sensor clock pad mode.

#### **Enumerator:**

AMB\_SENSOR\_CLOCK\_PAD\_OUTPUT\_MODE Sensor clock pad is an output.

AMB\_SENSOR\_CLOCK\_PAD\_INPUT\_MODE Sensor clock pad is an input.

8.32 Sensor 113

## 8.32.3 Function Documentation

8.32.3.1 static INLINE amb\_hal\_success\_t amb\_set\_sensor\_clock\_frequency ( void \* amb\_hal\_base\_address, amb\_clock\_frequency\_t amb\_sensor\_clock\_frequency ) [static]

Set the sensor pll frequency.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb	The requested frequency.
	sensor	
	clock	
	frequency	

#### **Return values**

AMB_HAL_SUCCE-	the new requested pll frequency is valid and it has been pro-
SS	grammed.
AMB_HAL_FAIL	the new pll frequency requested is not supported.
AMB_HAL_RETRY	a previous pll frequency change request is still outstanding.

8.32.3.2 static INLINE amb\_hal\_success\_t amb\_get\_sensor\_pll\_configuration ( void \* amb\_hal\_base\_address, amb\_pll\_configuration\_t \* amb\_sensor\_pll\_configuration ) [static]

Get the current sensor pll configuration.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
out	amb	Sensor pll configuration information read from pll registers.
	sensor_pll	
	configuration	

AMB_HAL_SUCCE-	always returns success.
SS	

8.32.3.3 static INLINE amb\_clock\_frequency\_t amb\_get\_sensor\_clock\_frequency(
 void \* amb\_hal\_base\_address) [static]

Get the current sensor pll frequency.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

#### Return values

amb_clock	Requested clock frequency.
frequency_t	

8.32.3.4 static INLINE amb\_hal\_success\_t amb\_get\_sensor\_pll\_lock\_status ( void \* amb\_hal\_base\_address ) [static]

Get the status of the previous requested sensor pll frequency change.

#### Note

A new sensor pll frequency change may be requested after this function returns AMB\_HAL\_SUCCESS.

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

#### Return values

AMB_HAL_SUCCE-	the pll has locked to the new frequency.
SS	
AMB_HAL_FAIL	the pll lock has failed to lock in a reasonable amount of time. some-
	thing is wrong.
AMB_HAL_RETRY	the pll has not locked yet. try again.

8.32.3.5 static INLINE amb\_hal\_success\_t amb\_enable\_sensor\_clock\_observation( void \* amb\_hal\_base\_address ) [static]

Enable observation of sensor clock.

8.32 Sensor 115

#### Note

A divided by 16 version of the clock may be observed on the xx\_clk\_si pin.

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

#### Return values

AMB_HAL_SUCCE-	Always returns success
SS	

8.32.3.6 static INLINE amb\_hal\_success\_t amb\_set\_sensor\_clock\_pad\_mode ( void \* amb\_hal\_base\_address, amb\_sensor\_clock\_pad\_mode\_t amb\_sensor\_clock\_pad\_mode ) [static]

Set the direction of the sensor clock pad.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb	The sensor clock pad mode.
	sensor	
	clock_pad	
	mode	

8.32.3.7 static INLINE amb\_sensor\_clock\_pad\_mode\_t amb\_get\_sensor\_clock\_pad\_mode ( void \* amb\_hal\_base\_address ) [static]

Get the direction of the sensor clock pad.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

The sensor clock pad direction (input or output).	amb sensor clock -	
	pad mode t	
	pad_mode_t	

## 8.33 SSI

## **Functions**

- static INLINE amb\_hal\_success\_t amb\_set\_ssi\_clock\_frequency (void \*amb\_hal\_base\_address, amb\_clock\_frequency\_t amb\_clock\_frequency)
- static INLINE amb\_clock\_frequency\_t amb\_get\_ssi\_clock\_frequency (void \*amb\_hal\_base\_address)
- static INLINE amb\_clock\_frequency\_t amb\_get\_ssi\_clock\_configuration (void \*amb\_hal\_base\_address, amb\_divider\_configuration\_t \*amb\_ssi\_divider\_configuration)
- static INLINE amb\_hal\_success\_t amb\_set\_ssi\_clock\_source (void \*amb\_hal\_base\_address, amb\_clock\_source\_t amb\_clock\_source, amb\_clock\_frequency\_t amb\_clock\_frequency)

## 8.33.1 Detailed Description

API calls to change/query the frequency of the SSI controller.

#### 8.33.2 Function Documentation

8.33.2.1 static INLINE amb\_hal\_success\_t amb\_set\_ssi\_clock\_frequency ( void \* amb\_hal\_base\_address, amb\_clock\_frequency\_t amb\_clock\_frequency ) [static]

Set the clock frequency of the ssi.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_clock	New ssi frequency
	frequency	

#### Return values

AMB_HAL_SUCCE-	the new frequency has been set
SS	
AMB_HAL_FAIL	the new requested frequency is not valid

8.33.2.2 static INLINE amb\_clock\_frequency\_t amb\_get\_ssi\_clock\_frequency ( void \* amb\_hal\_base\_address ) [static]

Get the frequency of the ssi clock.

8.33 SSI 117

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

## Return values

amb_clock	Requested clock frequency
frequency_t	

8.33.2.3 static INLINE amb\_clock\_frequency\_t amb\_get\_ssi\_clock\_configuration ( void \* amb\_hal\_base\_address, amb\_divider\_configuration\_t \* amb\_ssi\_divider\_configuration ) [static]

Get the configuration of the ssi clock divider.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
out	amb_ssi	Clock configuration information read from divider.
	divider	
	configuration	

#### Return values

AMB_HAL_SUCCE-	Always returns success.
SS	

8.33.2.4 static INLINE amb\_hal\_success\_t amb\_set\_ssi\_clock\_source (
 void \* amb\_hal\_base\_address, amb\_clock\_source\_t amb\_clock\_source,
 amb\_clock\_frequency\_t amb\_clock\_frequency) [static]

Set the clock source of the ssi AND ssi2 clock divider.

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_clock	Clock source for the ssi divider.
	source	
in	amb_clock	Clock frequency of the selected source.
	frequency	

## Note

: Valid clock sources are: AMB\_REFERENCE\_CLOCK\_SOURCE\_APB or AMB\_REFERENCE\_CLOCK\_SOURCE\_SPCLK\_C. The amb\_clock\_frequency must be specified for AMB\_REFERENCE\_CLOCK\_SOURCE\_SPCLK\_C. Otherwise it should be set to 0.

AMB_HAL_SUCCE-	The function always returns success.
SS	

8.34 SSI2 119

## 8.34 SSI2

#### **Functions**

static INLINE amb\_hal\_success\_t amb\_set\_ssi2\_clock\_frequency (void \*amb\_hal\_base\_address, amb\_clock\_frequency\_t amb\_clock\_frequency)

- static INLINE amb\_clock\_frequency\_t amb\_get\_ssi2\_clock\_frequency (void \*amb\_hal\_base\_address)
- static INLINE amb\_hal\_success\_t amb\_set\_ssi2\_clock\_source (void \*amb\_hal\_base\_address, amb\_clock\_source\_t amb\_clock\_source, amb\_clock\_frequency\_t amb\_clock\_frequency)

# 8.34.1 Detailed Description

API calls to change/query the frequency of the SSI2 controller.

## 8.34.2 Function Documentation

8.34.2.1 static INLINE amb\_hal\_success\_t amb\_set\_ssi2\_clock\_frequency ( void \* amb\_hal\_base\_address, amb\_clock\_frequency\_t amb\_clock\_frequency )
[static]

Set the clock frequency of the ssi2.

#### **Parameters**

_			
	in	amb_hal	Virtual address where ambhal is loaded by OS.
		base	
		address	
Γ	in	amb_clock	New ssi2 frequency
		frequency	

#### **Return values**

	AMB_HAL_SUCCE-	the new frequency has been set
	SS	
İ	AMB_HAL_FAIL	the new requested frequency is not valid

8.34.2.2 static INLINE amb\_clock\_frequency\_t amb\_get\_ssi2\_clock\_frequency( void \* amb\_hal\_base\_address ) [static]

Get the frequency of the ssi2 clock.

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

## Return values

amb_clock	Requested clock frequency
frequency_t	

Set the clock source of the ssi AND ssi2 clock divider.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_clock	Clock source for the ssi divider.
	source	
in	amb_clock	Clock frequency of the selected source.
	frequency	

#### Note

: Valid clock sources are: AMB\_REFERENCE\_CLOCK\_SOURCE\_APB or AMB\_REFERENCE\_CLOCK\_SOURCE\_SPCLK\_C. The amb\_clock\_frequency must be specified for AMB\_REFERENCE\_CLOCK\_SOURCE\_SPCLK\_C. Otherwise it should be set to 0.

AMB_HAL_SUCCE-	The function always returns success.
SS	

8.35 UART 121

## 8.35 **UART**

## **Functions**

static INLINE amb\_hal\_success\_t amb\_set\_uart\_clock\_frequency (void \*amb\_hal\_base\_address, amb\_clock\_frequency\_t amb\_clock\_frequency)

- static INLINE amb\_clock\_frequency\_t amb\_get\_uart\_clock\_frequency (void \*amb\_hal\_base\_address)
- static INLINE amb\_hal\_success\_t amb\_uart\_init (void \*amb\_hal\_base\_address)
- static INLINE amb\_clock\_frequency\_t amb\_get\_uart\_clock\_configuration (void \*amb\_hal\_base\_address, amb\_divider\_configuration\_t \*amb\_uart\_divider\_configuration)
- static INLINE amb\_hal\_success\_t amb\_set\_uart\_clock\_source (void \*amb\_hal\_base\_address, amb\_clock\_source\_t amb\_clock\_source)

#### 8.35.1 Detailed Description

API calls to change/query the frequency of the UART controller.

#### 8.35.2 Function Documentation

8.35.2.1 static INLINE amb\_hal\_success\_t amb\_set\_uart\_clock\_frequency ( void \* amb\_hal\_base\_address, amb\_clock\_frequency\_t amb\_clock\_frequency )
[static]

Set the uart clock frequency.

## Note

This is not the baud rate.

#### **Parameters**

_			
	in	amb_hal	Virtual address where ambhal is loaded by OS.
		base	
		address	
	in	amb_clock	New uart clock frequency
		frequency	

ĺ	AMB_HAL_SUCCE-	The frequency has been set
	SS	
	AMB_HAL_FAIL	The new frequency is invalid and has not been set

8.35.2.2 static INLINE amb\_clock\_frequency\_t amb\_get\_uart\_clock\_frequency( void \* amb\_hal\_base\_address ) [static]

Get the frequency of the uart clock.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

## Return values

amb_clock	Requested clock frequency.
frequency_t	

8.35.2.3 static INLINE amb\_hal\_success\_t amb\_uart\_init ( void \* amb\_hal\_base\_address ) [static]

Initialize the uart.

This api call sets up the uart for 115200 bauds, 8 bit data, no parity, 1 stop bit.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

#### Return values

```
AMB_HAL_SUCCE- SS Always returns success
```

8.35.2.4 static INLINE amb\_clock\_frequency\_t amb\_get\_uart\_clock\_configuration ( void \* amb\_hal\_base\_address, amb\_divider\_configuration\_t \* amb\_uart\_divider\_configuration ) [static]

Get the configuration of the uart clock divider.

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
out	amb_uart	Clock configuration information read from divider.
	divider	
	configuration	

8.35 UART 123

## **Return values**

AMB_HAL_SUCCE-	Always returns success.
SS	

8.35.2.5 static INLINE amb\_hal\_success\_t amb\_set\_uart\_clock\_source ( void \* amb\_hal\_base\_address, amb\_clock\_source\_t amb\_clock\_source ) [static]

Set the clock source of the uart clock divider.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_clock	Clock source for the uart divider.
	source	

#### Note

: Valid clock sources are: AMB\_REFERENCE\_CLOCK\_SOURCE\_IDSP, AMB\_REFERENCE\_CLOCK\_SOURCE\_ARM, AMB\_REFERENCE\_CLOCK\_SOURCE\_CLK\_REF.

AMB HAL SUCCE-	The function always returns success.
/IIVID_II/IL_OOOOL	The function always returns success.
ee.	
33	

## 8.36 VIN

## **Functions**

- static INLINE amb\_hal\_success\_t amb\_set\_vin\_clock\_frequency (void \*amb\_hal\_base\_address, amb\_clock\_frequency\_t amb\_clock\_frequency)
- static INLINE amb\_clock\_frequency\_t amb\_get\_vin\_clock\_frequency (void \*amb\_hal\_base\_address)
- static INLINE amb\_clock\_frequency\_t amb\_get\_vin\_clock\_configuration (void \*amb\_hal\_base\_address, amb\_divider\_configuration\_t \*amb\_vin\_divider\_configuration)

# 8.36.1 Detailed Description

API calls to change/query the frequency of the VIN controller.

## 8.36.2 Function Documentation

8.36.2.1 static INLINE amb\_hal\_success\_t amb\_set\_vin\_clock\_frequency ( void \* amb\_hal\_base\_address, amb\_clock\_frequency\_t amb\_clock\_frequency )
[static]

Set the vin clock frequency.

#### **Parameters**

	in	amb_hal	Virtual address where ambhal is loaded by OS.
		base	
		address	
ľ	in	amb_clock	New vin clock frequency
		frequency	

#### **Return values**

AMB_HAL_SUCCE-	The frequency has been set
SS	
AMB_HAL_FAIL	The new frequency is invalid and has not been set

8.36.2.2 static INLINE amb\_clock\_frequency\_t amb\_get\_vin\_clock\_frequency ( void \* amb\_hal\_base\_address ) [static]

Get the frequency of the vin clock.

8.36 VIN 125

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

## Return values

amb_clock	Requested clock frequency.
frequency_t	

8.36.2.3 static INLINE amb\_clock\_frequency\_t amb\_get\_vin\_clock\_configuration ( void \* amb\_hal\_base\_address, amb\_divider\_configuration\_t \*  $amb\_vin\_divider\_configuration$ ) [static]

Get the configuration of the vin clock divider.

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
out	amb_vin	Clock configuration information read from divider.
	divider	
	configuration	

AMB_HAL_SUCCE-	Always returns success.
SS	

## 8.37 USB

#### **Enumerations**

 enum amb\_usb\_port\_state\_t { AMB\_USB\_OFF, AMB\_USB\_ON, AMB\_USB\_-SUSPEND, AMB\_USB\_ALWAYS\_ON }

#### **Functions**

- static INLINE amb\_hal\_success\_t amb\_usb\_device\_soft\_reset (void \*amb\_hal\_base\_address)
- static INLINE amb\_hal\_success\_t amb\_usb\_host\_soft\_reset (void \*amb\_hal\_base\_address)
- static INLINE amb\_hal\_success\_t amb\_set\_usb\_port\_state (void \*amb\_hal\_base\_address, amb\_usb\_port\_state\_t usb\_port\_state)
- static INLINE amb\_usb\_port\_state\_t amb\_get\_usb\_port\_state (void \*amb\_hal\_base\_address)

## 8.37.1 Detailed Description

Set of api calls to change the state of the USB PHYs and subsystem.

## 8.37.2 Enumeration Type Documentation

8.37.2.1 enum amb\_usb\_port\_state\_t

USB Port State Settings.

#### **Enumerator:**

```
AMB_USB_OFF Disable USB port.
```

AMB\_USB\_ON Enable USB port.

AMB\_USB\_SUSPEND Force USB port into suspend state.

AMB\_USB\_ALWAYS\_ON Enable USB port & force USB to never suspend.

## 8.37.3 Function Documentation

```
8.37.3.1 static INLINE amb_hal_success_t amb_usb_device_soft_reset ( void * amb_hal_base_address ) [static]
```

Apply the usb device controller soft reset.

## Note

This function triggers a soft reset for the usb device controller

8.37 USB 127

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

## Return values

AMB_HAL_SUCCE-	reset sequence has completed
SS	

8.37.3.2 static INLINE amb\_hal\_success\_t amb\_usb\_host\_soft\_reset ( void \* amb\_hal\_base\_address ) [static]

Apply the usb host controller soft reset.

Note

This function triggers a soft reset for the usb host controller

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

## Return values

```
AMB_HAL_SUCCE- reset sequence has completed
```

8.37.3.3 static INLINE amb\_hal\_success\_t amb\_set\_usb\_port\_state ( void \* amb\_hal\_base\_address, amb\_usb\_port\_state\_t usb\_port\_state ) [static]

Suspend/un-suspend USB Port.

Note

This function suspends the USB Port if AMB\_USB\_SUSPEND is specified.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	usb_port	Requested State of the USB Device
	state	

# Return values

	AMB_HAL_SUCCE-	Port state has been set
	SS	
Ì	AMB_HAL_FAIL	Port state is not valid

8.37.3.4 static INLINE amb\_usb\_port\_state\_t amb\_get\_usb\_port\_state ( void \* amb\_hal\_base\_address ) [static]

Get the state of the USB Port.

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

AMB_USB_ON	if USB Device is enabled
AMB_USB_SUSPE-	if USB Device is suspended
ND	

8.38 Video Out 129

## 8.38 Video Out

## **Functions**

static INLINE amb\_hal\_success\_t amb\_set\_vout\_clock\_source (void \*amb\_hal\_base\_address, amb\_clock\_source\_t amb\_clock\_source, amb\_clock\_frequency\_t amb\_clock\_frequency)

- static INLINE amb\_hal\_success\_t amb\_set\_vout\_clock\_frequency (void \*amb\_-hal\_base\_address, amb\_clock\_frequency\_t amb\_vout\_clock\_frequency)
- static INLINE amb\_hal\_success\_t amb\_get\_vout\_pll\_configuration (void \*amb\_hal\_base\_address, amb\_pll\_configuration\_t \*amb\_vout\_pll\_configuration)
- static INLINE amb\_hal\_success\_t amb\_get\_vout\_pll\_lock\_status (void \*amb\_-hal base address)
- static INLINE amb\_hal\_success\_t amb\_enable\_vout\_clock\_observation (void \*amb\_hal\_base\_address)

## 8.38.1 Detailed Description

API calls to change/query frequency of the Vout pll. See also Changing PLL Frequency and Changing Clock Sources.

#### 8.38.2 Function Documentation

Set the clock source for vout.

#### **Parameters**

i	in	amb_hal	Virtual address where ambhal is loaded by OS.
		base	
		address	
i	Ln	amb_clock	the new clock source.
		source	
i	in	amb_clock	the clock frequency of the new source.
		frequency	

#### Note

The amb\_clock\_frequency only needs to be specified for the clock sources AMB\_-PLL\_REFERENCE\_CLOCK\_SOURCE\_CLK\_SI and AMB\_PLL\_REFERENCE\_C-LOCK\_SOURCE\_LVDS\_IDSP\_SCLK. Specify an amb\_clock\_frequency of 0 for all other clock sources. The topic Changing Clock Sources covers this in more details.

## Return values

AMB_HAL_SUCCE-	a new clock source has been set.
SS	
AMB_HAL_FAIL	the new clock source is not supported.

8.38.2.2 static INLINE amb\_hal\_success\_t amb\_set\_vout\_clock\_frequency ( void \* amb\_hal\_base\_address, amb\_clock\_frequency\_t amb\_vout\_clock\_frequency )
[static]

Set the vout pll frequency.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
in	amb_vout	The requested frequency.
	clock	
	frequency	

## **Return values**

AMB_HAL_SUCCE-	the new requested pll frequency is valid and it has been pro-
SS	grammed.
AMB_HAL_FAIL	the new pll frequency requested is not supported.
AMB_HAL_RETRY	a previous pll frequency change request is still outstanding.

8.38.2.3 static INLINE amb\_hal\_success\_t amb\_get\_vout\_pll\_configuration ( void \*  $amb\_hal\_base\_address, \ amb\_pll\_configuration\_t * amb\_vout\_pll\_configuration \ )$  [static]

Get the current vout pll configuration.

## **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	
out	amb_vout	Sensor pll configuration information read from pll registers.
	pII	
	configuration	

AMB HAL SUCCE-	always returns success.
71111B_11111L_0000L	always rotario success.
CC	
33	

8.38 Video Out 131

8.38.2.4 static INLINE amb\_hal\_success\_t amb\_get\_vout\_pll\_lock\_status ( void \* amb\_hal\_base\_address ) [static]

Get the status of the previous requested vout pll frequency change.

#### Note

A new vout pll frequency change may be requested after this function returns AM-B\_HAL\_SUCCESS.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

#### Return values

AMB_HAL_SUCCE-	the pll has locked to the new frequency.
SS	
AMB_HAL_FAIL	the pll lock has failed to lock in a reasonable amount of time. some-
	thing is wrong.
AMB_HAL_RETRY	the pll has not locked yet. try again.

8.38.2.5 static INLINE amb\_hal\_success\_t amb\_enable\_vout\_clock\_observation (  $void*amb\_hal\_base\_address$  ) [static]

Enable observation of vout clock.

#### Note

A divided by 16 version of the clock may be observed on the xx\_clk\_si pin.

#### **Parameters**

in	amb_hal	Virtual address where ambhal is loaded by OS.
	base	
	address	

AMB_HAL_SUCCE-	Always returns success
SS	

# **Chapter 9**

# **Data Structure Documentation**

# 9.1 amb\_divider\_configuration\_t Struct Reference

```
#include <ambhal.h>
```

#### **Data Fields**

- amb\_clock\_frequency\_t clock\_source\_frequency
- amb\_clock\_source\_t clock\_source
- unsigned int divider

# 9.1.1 Detailed Description

Clock Divider Configuration.

ALI the fields that make up the clock divider programming.

# 9.2 amb\_ioctrl\_configuration\_t Struct Reference

```
#include <ambhal.h>
```

# **Data Fields**

- amb\_ioctrl\_drive\_strength\_t drive\_strength
- amb\_ioctrl\_pullupdown\_t pullupdown
- amb\_ioctrl\_input\_type\_t input\_type
- amb\_ioctrl\_slew\_rate\_t slew\_rate

## 9.2.1 Detailed Description

IO pad configuration.

# 9.3 amb\_ms\_delay\_configuration\_t Struct Reference

```
#include <ambhal.h>
```

## **Data Fields**

- · unsigned int sclk delay
- · unsigned int sd\_input\_delay
- unsigned int sd\_output\_delay
- · unsigned int read\_delay

## 9.3.1 Detailed Description

Memory stick delay configuration.

# 9.4 amb\_operating\_mode\_t Struct Reference

```
#include <ambhal.h>
```

## **Data Fields**

- · amb vidcap window size t vidcap size
- amb\_performance\_t performance
- amb\_mode\_t mode
- · amb\_usb\_port\_state\_t usb\_state
- amb\_hdmi\_interface\_state\_t hdmi\_state
- amb\_dual\_stream\_state\_t dual\_stream\_state
- amb\_digital\_gamma\_mode\_t amb\_digital\_gamma\_mode

## 9.4.1 Detailed Description

Operating mode.

# 9.5 amb\_pll\_configuration\_t Struct Reference

#include <ambhal.h>

## **Data Fields**

- · unsigned int fractional mode
- unsigned int intprog
- unsigned int sdiv
- · unsigned int sout
- · int frac
- · unsigned int prescaler
- · unsigned int postscaler
- unsigned int clock\_source

## 9.5.1 Detailed Description

All the fields that make up the pll frequency programming.

The effective pll frequency may be calculated as follows:

$$f_{out} = (\frac{reference}{prescaler}) * (intprog + 1 + fraction) * (\frac{sdiv + 1}{sout + 1}) * (\frac{1}{postscaler})$$

$$fraction = \begin{cases} 0 & \text{if fractional\_mode is 0} \\ (-0.5*frac[31]) + (\frac{frac[30:0]}{2^{32}}) & \text{if fractional\_mode is 1} \end{cases}$$

Guidelines:

$$f_{jdiv} = \frac{f_{vco}}{sdiv + 1} <= 800MHz$$

$$f_{vco} = f_{out} * (sout + 1) * postscaler <= 2.2GHz$$