Ambarella A7 Hardware Abstraction Layer

Generated by Doxygen 1.7.6.1

Mon Aug 5 2013 23:32:20

Contents

Chapter 1

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:

Chapter 2

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.

Chapter 3

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.

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

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

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

$$core\_clock\_frequency = core$$

$$dram\_clock\_frequency = dram$$

```
#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,
      216000000},
[AMB_OPERATING_MODE_PREVIEW] [AMB_PERFORMANCE_720P60]  = {0, 0, 0, 216000000,
      336000000},
[AMB_OPERATING_MODE_PREVIEW] [AMB_PERFORMANCE_1080I60] = {0, 0, 0, 144000000,
      216000000},
[AMB_OPERATING_MODE_PREVIEW] [AMB_PERFORMANCE_1080P30] = {0, 0, 0, 144000000,
      216000000},
[AMB_OPERATING_MODE_PREVIEW] [AMB_PERFORMANCE_1080P60] = {0, 0, 0, 216000000,
      216000000},
[AMB_OPERATING_MODE_PREVIEW] [AMB_PERFORMANCE_4KP30]   = {0, 0, 0, 216000000,
      336000000},

[AMB_OPERATING_MODE_CAPTURE] [AMB_PERFORMANCE_480P30]  = {0, 0, 0, 216000000,
      336000000},
[AMB_OPERATING_MODE_CAPTURE] [AMB_PERFORMANCE_720P30]  = {0, 0, 0, 144000000,
      216000000},
[AMB_OPERATING_MODE_CAPTURE] [AMB_PERFORMANCE_720P60]  = {0, 0, 0, 216000000,
      336000000},
[AMB_OPERATING_MODE_CAPTURE] [AMB_PERFORMANCE_1080I60] = {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_1080I60] = {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_1080I60] = {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]  = {504000000, 2, 1,
      216000000, 336000000},
[AMB_OPERATING_MODE_PLAYBACK] [AMB_PERFORMANCE_720P30]  = {504000000, 2,  1,
      216000000, 336000000},
[AMB_OPERATING_MODE_PLAYBACK] [AMB_PERFORMANCE_720P60]  = {504000000, 2,  1,
      216000000, 336000000},
[AMB_OPERATING_MODE_PLAYBACK] [AMB_PERFORMANCE_1080I60] = {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_1080I60] = {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, 48000000},
[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_1080I60] = {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_1080I60]    = {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 }
[AMB_VIDCAP_4096X3575]          = {504000000, 2, 1},  // { 504000000,
       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 }
[AMB_VIDCAP_1312X984]           = {504000000, 7, 1},  // { 504000000,
       72000000, 480000000 }
[AMB_VIDCAP_1536X384]           = {504000000, 7, 1},  // { 504000000,
       72000000, 432000000 }
[AMB_VIDCAP_1536X384_SMALL_VB]  = {504000000, 7, 1}   // { 504000000,
       72000000, 432000000 }
} ;
```

Chapter 4

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.

```
amb_hal_success_t success ;
success = amb_set_sensor_clock_frequency (amb_hal_base_address, 74000000) ;

// if this assertion goes off you did something wrong.
// either the requested clock frequency is invalid (AMB_HAL_FAIL)
// or the pll is not locked and it is not ready to be reprogrammed
      (AMB_HAL_RETRY)
K_ASSERT (success == AMB_HAL_SUCCESS) ;

// if you get here the pll has locked and you are good to go
```

Chapter 5

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.

Chapter 6

Module Index

6.1 Modules

Here is a list of all modules:

Chapter 7

Data Structure Index

7.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 8

Module Documentation

8.1 Initialization

Enumerations

- enum amb_dram_arbiter_priority_t { AMB_DRAM_ARBITER_DSP_VERY_LOW_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_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_-base_-address*	Virtual address where ambhal is loaded by OS.
`in`	*amb_apb_-peripherals-_base_-address*	Virtual address of peripherals (corresponding to physical address 0x70000000)
`in`	*amb_ahb_-peripherals-_base_-address*	Virtual address of peripherals (corresponding to physical address 0x60000000)

Return values

AMB_HAL_SUCCE-SS	ambhal initialization was successful
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_-base_-address*	Virtual address where ambhal is loaded by OS.
out	*amb_hal_-chip_name*	Pointer to the name of the device.

Return values

AMB_HAL_SUCCE-SS	Always returns success

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_-base_-address*	Virtual address where ambhal is loaded by OS.
out	*amb_hal_-version*	Pointer to the version of hal.

Return values

AMB_HAL_SUCCE-SS	Always returns success

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_base_address*	Virtual address where ambhal is loaded by OS.
`in`	*amb_apb_peripherals_base_address*	Virtual address of peripherals (corresponding to physical address 0x70000000)
`in`	*amb_ahb_peripherals_base_address*	Virtual address of peripherals (corresponding to physical address 0x60000000)

Return values

AMB_HAL_SUCCESS	ambhal initialization was successful
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_base_address*	Virtual address where ambhal is loaded by OS.
`in`	*amb_dram_arbiter_priority*	Priority given to dsp clients by DRAM arbiter.

Return values

AMB_HAL_SUCCESS	ambhal initialization was successful
AMB_HAL_FAIL	The amb_dram_arbiter_priority is not defined

8.2 Command Status

Enumerations

- enum amb_hal_success_t { AMB_HAL_SUCCESS = 0x00000000UL, AMB_H-AL_FAIL = 0xffffffffUL, AMB_HAL_RETRY = 0xfffffffeUL }

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_PERFOR-MANCE_720P30, AMB_PERFORMANCE_720P60, AMB_PERFORMANCE_-1080I60, AMB_PERFORMANCE_1080P30, AMB_PERFORMANCE_1080P60, AMB_PERFORMANCE_4KP30 }
- enum amb_mode_t { AMB_OPERATING_MODE_PREVIEW, AMB_OPERATI-NG_MODE_STILL_CAPTURE, AMB_OPERATING_MODE_CAPTURE, AMB-_OPERATING_MODE_PLAYBACK, AMB_OPERATING_MODE_DISPLAY_A-ND_ARM, AMB_OPERATING_MODE_STANDBY, AMB_OPERATING_MOD-E_LCD_BYPASS, AMB_OPERATING_MODE_STILL_PREVIEW, AMB_OPE-RATING_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_t amb_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_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_-operating_-mode*	New operating mode.

Return values

AMB_HAL_SUCCE-SS	The new operating mode has been programmed.

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]

Get the current operating mode for the system.

Parameters

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.
out	*amb_-operating_-mode*	Current operating mode.

Return values

AMB_HAL_SUCCE-SS	Always returns success.

8.3.3.3 static INLINE amb_hal_success_t amb_get_operating_mode_status (void ∗ *amb_hal_base_address*) [static]

Check whether a previous amb_set_operating_mode() call has completed.

Parameters

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

AMB_HAL_SUCCE-SS	The new operating mode has been set.
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:

```
 {
[AMB_VIDCAP_4KP30]                = {672000000, 2, 1},
[AMB_VIDCAP_4096X2176_60FPS]      = {540000000, 2, 1},
[AMB_VIDCAP_4096X3575]            = {504000000, 2, 1},
[AMB_VIDCAP_4000X2250]            = {432000000, 2, 1},
[AMB_VIDCAP_2304X1296]            = {408000000, 2, 1},
[AMB_VIDCAP_1984X1116]            = {456000000, 3, 1},
[AMB_VIDCAP_2048X1536]            = {432000000, 4, 1},
[AMB_VIDCAP_1312X984]             = {504000000, 7, 1},
[AMB_VIDCAP_1536X384]             = {504000000, 7, 1},
[AMB_VIDCAP_1536X384_SMALL_VB]    = {504000000, 7, 1}
}
```

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 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 = 0x00800000UL, AMB_SYSTEM_CONFIGURATION_RMII_SELECTED = 0x8000UL, AMB_SYSTEM_CONFIGURATION_HIF_SECURE_MODE = 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_ACTIVE_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_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

amb_clock_-frequency_t	The reference clock frequency from the system configuration pins.

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_-base_-address*	Virtual address where ambhal is loaded by OS.

Note

Use the flags defined in amb_system_configuration_t to determine what system configuration was set.

Return values

amb_system_-configuration_t	The system configuration.

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_-base_-address*	Virtual address where ambhal is loaded by OS.

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_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

amb_hif_type_t	The host interface type selected.

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_R-EF , 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_COR-E_PLL_VCO, AMB_REFERENCE_CLOCK_SOURCE_CORE_PLL, AMB_RE-FERENCE_CLOCK_SOURCE_AUDIO_PLL_VCO, AMB_REFERENCE_CLO-CK_SOURCE_AUDIO_PLL, AMB_REFERENCE_CLOCK_SOURCE_IDSP_P-LL_VCO, AMB_REFERENCE_CLOCK_SOURCE_CORTEX_PLL_VCO, AMB-_REFERENCE_CLOCK_SOURCE_APB, AMB_REFERENCE_CLOCK_SOUR-CE_CORE, AMB_REFERENCE_CLOCK_SOURCE_ARM, AMB_REFERENC-E_CLOCK_SOURCE_IDSP }

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 ∼50 KHz may be requested.

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_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

AMB_HAL_SUCCE-SS	Always returns success.

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_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_clock_-frequency*	New adc clock frequency

Return values

AMB_HAL_SUCCE-SS	The new frequency has been set
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_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

amb_clock_-frequency_t	Requested clock frequency

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_-base_-address*	Virtual address where ambhal is loaded by OS.
out	*amb_adc_-divider_-configuration*	Clock configuration information read from divider.

Return values

AMB_HAL_SUCCE-SS	Always returns success.

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_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

amb_clock_-frequency_t	The arm clock frequency.

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_base_address*	Virtual address where ambhal is loaded by OS.

Return values

amb_clock_frequency_t	The AHB clock frequency.

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_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

amb_clock_-frequency_t	The APB clock frequency.

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

8.11.2.1 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]`

Set the clock source for audio.

Parameters

`in`	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.
`in`	*amb_clock_-source*	the new clock source.
`in`	*amb_clock_-frequency*	the clock frequency of the new source.

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-SS	a new clock source has been set.
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_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_audio-_clock_-frequency*	The requested frequency.

Return values

AMB_HAL_SUCCE-SS	the new requested pll frequency is valid and it has been programmed.
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_-base_-address*	Virtual address where ambhal is loaded by OS.
out	*amb_audio-_pll_-configuration*	Sensor pll configuration information read from pll registers.

Return values

AMB_HAL_SUCCE-SS	always returns success.

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_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

amb_clock_-frequency_t	Requested clock frequency.

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_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

AMB_HAL_SUCCE-SS	the pll has locked to the new frequency.
AMB_HAL_FAIL	the pll lock has failed to lock in a reasonable amount of time. something 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.

Note

A divided by 16 version of the clock may be observed on the xx_clk_si pin.

Parameters

`in`	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

AMB_HAL_SUCCE-SS	Always returns success

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_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_clock_-frequency*	New au1ch frequency

Return values

AMB_HAL_SUCCE-SS	the new frequency has been set
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.

Parameters

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

amb_clock_-frequency_t	Requested clock frequency

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*) [static]

Get the configuration of the au1ch clock divider.

Parameters

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.
out	*amb_au1ch-_divider_-configuration*	Clock configuration information read from divider.

Return values

AMB_HAL_SUCCE-SS	Always returns success.

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_-base_-address*	Virtual address where ambhal is loaded by OS.
`out`	*amb_core_-pll_-configuration*	pll configuration information read from pll registers.

Return values

AMB_HAL_SUCCE-SS	always returns success.

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.

Parameters

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

amb_clock_-frequency_t	Requested clock frequency.

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_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

AMB_HAL_SUCCE-SS	the pll has locked to the new frequency.
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.

8.13.2.4 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_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

AMB_HAL_SUCCE-SS	Always returns success

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_-base_-address*	Virtual address where ambhal is loaded by OS.
out	*amb_ddr_-pll_-configuration*	pll configuration information read from pll registers.

Return values

AMB_HAL_SUCCE-SS	always returns success.

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_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

amb_clock_-frequency_t	Requested clock frequency.

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_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

AMB_HAL_SUCCE-SS	the pll has locked to the new frequency.
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_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

AMB_HAL_SUCCESS	Always returns success

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_-base_-address*	Virtual address where ambhal is loaded by OS.
`in`	*amb_cortex-_clock_-frequency*	The requested frequency.

Return values

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

8.15.2.2 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_-base_-address*	Virtual address where ambhal is loaded by OS.
out	*amb_cortex-_pll_-configuration*	pll configuration information read from pll registers.

Return values

AMB_HAL_SUCCE-SS	Always returns success.

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_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

amb_clock_-frequency_t	Requested clock frequency.

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_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

AMB_HAL_SUCCE-SS	the pll has locked to the new frequency.
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_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

AMB_HAL_SUCCE-SS	Always returns success

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_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

amb_clock_-frequency_t	The AXI clock frequency.

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_-base_-address*	Virtual address where ambhal is loaded by OS.
`in`	*amb_clock_-frequency*	New fdet frequency

Return values

AMB_HAL_SUCCE-SS	the new frequency has been set
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_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

amb_clock_-frequency_t	Requested clock frequency

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*) [static]

Get the configuration of the fdet clock divider.

Parameters

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.
out	*amb_fdet_-divider_-configuration*	Clock configuration information read from divider.

Return values

AMB_HAL_SUCCE-SS	Always returns success.

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_base_address*	Virtual address where ambhal is loaded by OS.

Return values

AMB_HAL_SUCCESS	always returns success.

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_base_address*	Virtual address where ambhal is loaded by OS.

Return values

AMB_HAL_SUCCESS	always returns success.

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_base_address*	Virtual address where ambhal is loaded by OS.

Return values

AMB_HAL_SUCCESS	always returns success.

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_base_address*	Virtual address where ambhal is loaded by OS.

Return values

AMB_HAL_SUCCESS	always returns success.

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.

Parameters

in	*amb_hal_base_address*	Virtual address where ambhal is loaded by OS.

Return values

AMB_HAL_SUCCE-SS	always returns success.

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_base_address*	Virtual address where ambhal is loaded by OS.
`in`	*amb_clock_frequency*	New gtx frequency

Return values

AMB_HAL_SUCCESS	the new frequency has been set
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_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

amb_clock_-frequency_t	Requested clock frequency

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_-base_-address*	Virtual address where ambhal is loaded by OS.
out	*amb_gtx_-divider_-configuration*	Clock configuration information read from divider.

Return values

AMB_HAL_SUCCE-SS	Always returns success.

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.

Parameters

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_gtx_-clock_-source*	The gtx clock source.

Note

: Valid clock sources are: AMB_REFERENCE_CLOCK_SOURCE_CORTEX_PLL_VCO and AMB_EXTERNAL_CLOCK_SOURCE.

Return values

AMB_HAL_SUCCESS	The gtx clock source was set properly.
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_base_address*	Virtual address where ambhal is loaded by OS.

Return values

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.

Parameters

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

amb_clock_-frequency_t	The hdmi clock frequency.

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_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

AMB_HAL_SUCCE-SS	Always returns success

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_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

amb_clock_-frequency_t	The hdmi4k clock frequency.

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_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

AMB_HAL_SUCCE-SS	Always returns success

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_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

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

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_-base_-address*	Virtual address where ambhal is loaded by OS.
`out`	*amb_idsp_-pll_-configuration*	pll configuration information read from pll registers.

Return values

AMB_HAL_SUCCE-SS	Always returns success.

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_base_address*	Virtual address where ambhal is loaded by OS.

Return values

amb_clock_frequency_t	Requested clock frequency.

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_base_address*	Virtual address where ambhal is loaded by OS.

Return values

AMB_HAL_SUCCESS	the pll has locked to the new frequency.
AMB_HAL_FAIL	the pll lock has failed to lock in a reasonable amount of time. something 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.

Note

A divided by 16 version of the clock may be observed on the xx_clk_si pin.

Parameters

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

AMB_HAL_SUCCE-SS	Always returns success

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_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

amb_clock_-frequency_t	Requested clock frequency.

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_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_clock_-frequency*	New infrared clock frequency

Return values

AMB_HAL_SUCCE-SS	The new frequency has been set
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.

Parameters

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

amb_clock_-frequency_t	Requested clock frequency.

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_-base_-address*	Virtual address where ambhal is loaded by OS.
out	*amb_ir_-divider_-configuration*	Clock configuration information read from divider.

Return values

AMB_HAL_SUCCE-SS	Always returns success.

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_STRENGTH_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_IOCTRL_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

8.22.3 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_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_ioctrl_-drive_-strength*	The drive strength of the io pad

Return values

AMB_HAL_SUCCE-SS	The new drive strength was set.

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_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_ioctrl_-pullupdown*	The pullup/pulldown of the io pad

Return values

AMB_HAL_SUCCE-SS	The new pullup/pulldown was set.

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_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_ioctrl_-input_type*	The input type of the io pad

Return values

AMB_HAL_SUCCE-SS	The new input type was set.

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_-base_-address*	Virtual address where ambhal is loaded by OS.
out	*amb_ioctrl_-configuration*	The current configuration of the io pad

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_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_ioctrl_-drive_-strength*	The drive strength of the io pad

Return values

AMB_HAL_SUCCE-SS	The new drive strength was set.

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

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_ioctrl_-pullupdown*	The pullup/pulldown of the io pad

Return values

AMB_HAL_SUCCE-SS	The new pullup/pulldown was set.

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.

Parameters

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_ioctrl_-input_type*	The input type of the io pad

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_-base_-address*	Virtual address where ambhal is loaded by OS.
out	*amb_ioctrl_-configuration*	The current configuration of the io pad

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_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_ioctrl_-drive_-strength*	The drive strength of the io pad

Return values

AMB_HAL_SUCCE-SS	The new drive strength was set.

8.22.3.10 static INLINE amb_hal_success_t amb_set_smioa_ioctrl_pullupdown (void * *amb_hal_base_address,* amb_ioctrl_pullupdown_t *amb_ioctrl_pullupdown*) [static]

Set the smioa io pad pullup or pulldown.

Parameters

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_ioctrl_-pullupdown*	The pullup/pulldown of the io pad

Return values

AMB_HAL_SUCCE-SS	The new pullup/pulldown was set.

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_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_ioctrl_-input_type*	The input type of the io pad

Return values

AMB_HAL_SUCCE-SS	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.

Parameters

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.
out	*amb_ioctrl_-configuration*	The current configuration of the io pad

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

in	*amb_hal_base_address*	Virtual address where ambhal is loaded by OS.
in	*amb_ioctrl_drive_strength*	The drive strength of the io pad

Return values

AMB_HAL_SUCCESS	The new drive strength was set.

8.22.3.14 static INLINE amb_hal_success_t amb_set_smiob_ioctrl_pullupdown (void ∗ *amb_hal_base_address*, amb_ioctrl_pullupdown_t *amb_ioctrl_pullupdown*) [static]

Set the smiob io pad pullup or pulldown.

Parameters

in	*amb_hal_base_address*	Virtual address where ambhal is loaded by OS.
in	*amb_ioctrl_pullupdown*	The pullup/pulldown of the io pad

Return values

AMB_HAL_SUCCESS	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_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_ioctrl_-input_type*	The input type of the io pad

Return values

AMB_HAL_SUCCE-SS	The new input type was set.

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_-base_-address*	Virtual address where ambhal is loaded by OS.
out	*amb_ioctrl_-configuration*	The current configuration of the io pad

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_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_ioctrl_-drive_-strength*	The drive strength of the io pad

Return values

AMB_HAL_SUCCE-SS	The new drive strength was set.

8.22.3.18 static INLINE amb_hal_success_t amb_set_smioc_ioctrl_pullupdown (void ∗ *amb_hal_base_address,* amb_ioctrl_pullupdown_t *amb_ioctrl_pullupdown*) [static]

Set the smioc io pad pullup or pulldown.

Parameters

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_ioctrl_-pullupdown*	The pullup/pulldown of the io pad

Return values

AMB_HAL_SUCCE-SS	The new pullup/pulldown was set.

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	*amb_ioctrl_-input_type*	The input type of the io pad

Return values

AMB_HAL_SUCCE-SS	The new input type was set.

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_-base_-address*	Virtual address where ambhal is loaded by OS.
out	*amb_ioctrl_-configuration*	The current configuration of the io pad

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_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_ioctrl_-drive_-strength*	The drive strength of the io pad

Return values

AMB_HAL_SUCCE-SS	The new drive strength was set.

8.22.3.22 static INLINE amb_hal_success_t amb_set_smiod_ioctrl_pullupdown (void * *amb_hal_base_address*, amb_ioctrl_pullupdown_t *amb_ioctrl_pullupdown*) [static]

Set the smiod io pad pullup or pulldown.

Parameters

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_ioctrl_-pullupdown*	The pullup/pulldown of the io pad

Return values

AMB_HAL_SUCCE-SS	The new pullup/pulldown was set.

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_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_ioctrl_-input_type*	The input type of the io pad

Return values

AMB_HAL_SUCCE-SS	The new input type was set.

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	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.
out	*amb_ioctrl_-configuration*	The current configuration of the io pad

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.

Parameters

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_ioctrl_-drive_-strength*	The drive strength of the io pad

Return values

AMB_HAL_SUCCESS	The new drive strength was set.

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_pullupdown*	The pullup/pulldown of the io pad

Return values

AMB_HAL_SUCCESS	The new pullup/pulldown was set.

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_input_type*	The input type of the io pad

Return values

AMB_HAL_SUCCESS	The new input type was set.

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

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.
out	*amb_ioctrl_-configuration*	The current configuration of the io pad

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_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_ioctrl_-drive_-strength*	The drive strength of the io pad

Return values

AMB_HAL_SUCCE-SS	The new drive strength was set.

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.

Parameters

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_ioctrl_-pullupdown*	The pullup/pulldown of the io pad

Return values

AMB_HAL_SUCCE-SS	The new pullup/pulldown was set.

8.22.3.31 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]

Set the sensor io pad input type.

Parameters

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_ioctrl_-input_type*	The input type of the io pad

Return values

AMB_HAL_SUCCE-SS	The new input type was set.

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.

Parameters

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.
out	*amb_ioctrl_-configuration*	The current configuration of the io pad

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

8.23.2.1 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]`

Set the clock source for lcd.

Parameters

`in`	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.
`in`	*amb_clock_-source*	the new clock source.
`in`	*amb_clock_-frequency*	the clock frequency of the new source.

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_SUCCESS	a new clock source has been set.
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_base_address*	Virtual address where ambhal is loaded by OS.
in	*amb_lcd_clock_frequency*	The requested frequency.

Return values

AMB_HAL_SUCCESS	the new requested pll frequency is valid and it has been programmed.
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_base_address*	Virtual address where ambhal is loaded by OS.
out	*amb_lcd_pll_configuration*	Sensor pll configuration information read from pll registers.

Return values

AMB_HAL_SUCCESS	always returns success.

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_base_address*	Virtual address where ambhal is loaded by OS.

Return values

amb_clock_frequency_t	Requested clock frequency.

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_base_address*	Virtual address where ambhal is loaded by OS.

Return values

AMB_HAL_SUCCESS	the pll has locked to the new frequency.
AMB_HAL_FAIL	the pll lock has failed to lock in a reasonable amount of time. something 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.

Note

A divided by 16 version of the clock may be observed on the xx_clk_si pin.

Parameters

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

AMB_HAL_SUCCE-SS	Always returns success

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_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_lvds_-pad_mode*	The mode for lvds pads

Return values

AMB_HAL_SUCCE-SS	The new pad mode was set.
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.

Parameters

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

amb_lvds_pad_-mode_t	The current pad mode setting.

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_-base_-address*	Virtual address where ambhal is loaded by OS.
`in`	*amb_clock_-frequency*	New motor clock frequency

Return values

AMB_HAL_SUCCE-SS	The new frequency has been set
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.

Parameters

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

amb_clock_-frequency_t	Requested clock frequency.

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*) [static]

Get the configuration of the motor clock divider.

Parameters

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.
out	*amb_motor-_divider_-configuration*	Clock configuration information read from divider.

Return values

AMB_HAL_SUCCE-SS	Always returns success.

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_-base_-address*	Virtual address where ambhal is loaded by OS.
`in`	*amb_clock_-frequency*	New memory stick frequency

Return values

AMB_HAL_SUCCE-SS	The new frequency has been set
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_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

amb_clock_-frequency_t	Requested clock frequency

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_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_ms_-delay*	Requested delay.

Return values

AMB_HAL_SUCCE-SS	always returns success.

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_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_ms_-delay*	Requested delay.

Return values

AMB_HAL_SUCCE-SS	always returns success.

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_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_ms_-delay*	Requested delay.

Return values

AMB_HAL_SUCCESS	always returns success.

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_base_address*	Virtual address where ambhal is loaded by OS.
in	*amb_ms_delay*	Requested delay.

Return values

AMB_HAL_SUCCESS	always returns success.

8.26.3.7 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]

Get the current memory stick delay configuration.

Parameters

in	*amb_hal_base_address*	Virtual address where ambhal is loaded by OS.
out	*amb_ms_delay_configuration*	Memory stick delays read from the delay configuration register.

Return values

AMB_HAL_SUCCESS	always returns success.

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_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_ms_-status*	Status of the memory stick controller

Return values

AMB_HAL_SUCCE-SS	always returns success.

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_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

amb_ms_status_t	Status of the memory stick controller

8.27 PWM

Functions

- static INLINE amb_hal_success_t amb_set_pwm_clock_frequency (void ∗amb_-hal_base_address, amb_clock_frequency_t 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_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_clock_-frequency*	New pwm frequency

Return values

AMB_HAL_SUCCE-SS	the new frequency has been set
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_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

amb_clock_-frequency_t	Requested clock frequency

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_-base_-address*	Virtual address where ambhal is loaded by OS.
out	*amb_pwm_-divider_-configuration*	Clock configuration information read from divider.

Return values

AMB_HAL_SUCCE-SS	Always returns success.

8.27.2.4 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*) `[static]`

Set the clock source of the pwm clock divider.

Parameters

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_clock_-source*	Clock source for the pwm divider.
in	*amb_clock_-frequency*	Clock frequency of the selected source.

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.

Return values

AMB_HAL_SUCCE-SS	The function always returns success.

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_t 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

8.28.2.1 static INLINE amb_hal_success_t amb_set_pwmois_clock_frequency (void ∗ *amb_hal_base_address,* amb_clock_frequency_t *amb_clock_frequency*) [static]

Set the clock frequency of the pwmois.

Parameters

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_clock_-frequency*	New pwmois frequency

Return values

AMB_HAL_SUCCE-SS	the new frequency has been set
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.

Parameters

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

amb_clock_-frequency_t	Requested clock frequency

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_-base_-address*	Virtual address where ambhal is loaded by OS.
out	*amb_-pwmois_-divider_-configuration*	Clock configuration information read from divider.

Return values

AMB_HAL_SUCCE-SS	Always returns success.

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_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_clock_-source*	Clock source for the pwmois divider.
in	*amb_clock_-frequency*	Clock frequency of the selected source.

Note

: Valid clock sources are: AMB_REFERENCE_CLOCK_SOURCE_IDSP_PLL_V-CO, AMB_REFERENCE_CLOCK_SOURCE_AUDIO_PLL_VCO, AMB_REFERE-NCE_CLOCK_SOURCE_CORE_PLL_VCO or AMB_REFERENCE_CLOCK_SO-URCE_LVDS_IDSP_SCLK. The amb_clock_frequency must be specified for AM-B_REFERENCE_CLOCK_SOURCE_LVDS_IDSP_SCLK. Otherwise it should be set to 0.

Return values

AMB_HAL_SUCCE-SS	The function always returns success.

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_-base_-address*	Virtual address where ambhal is loaded by OS.

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.

Return values

AMB_HAL_SUCCE-SS	always returns success.

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_-base_-address*	Virtual address where ambhal is loaded by OS.
`in`	*amb_clock_-frequency*	New smiod frequency

Return values

AMB_HAL_SUCCE-SS	The new frequency has been set
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.

Parameters

`in`	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

amb_clock_-frequency_t	Requested clock frequency

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_-base_-address*	Virtual address where ambhal is loaded by OS.
`out`	*amb_sd_-divider_-configuration*	Clock configuration information read from divider.

Return values

AMB_HAL_SUCCE-SS	Always returns success.

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_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_clock_-frequency*	New sdio frequency

Return values

AMB_HAL_SUCCE-SS	The new frequency has been set
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.

Parameters

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

amb_clock_-frequency_t	Requested clock frequency

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*) [static]

Get the configuration of the sdio clock divider.

Parameters

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.
out	*amb_sdio_-divider_-configuration*	Clock configuration information read from divider.

Return values

AMB_HAL_SUCCE-SS	Always returns success.

8.32 Sensor

Enumerations

- enum amb_sensor_clock_pad_mode_t { AMB_SENSOR_CLOCK_PAD_OUTPUT_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.

8.32.3 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_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_-sensor_-clock_-frequency*	The requested frequency.

Return values

AMB_HAL_SUCCE-SS	the new requested pll frequency is valid and it has been pro-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_-base_-address*	Virtual address where ambhal is loaded by OS.
out	*amb_-sensor_pll_-configuration*	Sensor pll configuration information read from pll registers.

Return values

AMB_HAL_SUCCE-SS	always returns success.

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_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

amb_clock_-frequency_t	Requested clock frequency.

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_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

AMB_HAL_SUCCE-SS	the pll has locked to the new frequency.
AMB_HAL_FAIL	the pll lock has failed to lock in a reasonable amount of time. something 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.

Note

A divided by 16 version of the clock may be observed on the xx_clk_si pin.

Parameters

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

AMB_HAL_SUCCE-SS	Always returns success

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_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_-sensor_-clock_pad_-mode*	The 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_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

amb_sensor_clock_-pad_mode_t	The sensor clock pad direction (input or output).

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_base_address*	Virtual address where ambhal is loaded by OS.
`in`	*amb_clock_frequency*	New ssi frequency

Return values

AMB_HAL_SUCCESS	the new frequency has been set
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.

Parameters

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

amb_clock_-frequency_t	Requested clock frequency

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_-base_-address*	Virtual address where ambhal is loaded by OS.
out	*amb_ssi_-divider_-configuration*	Clock configuration information read from divider.

Return values

AMB_HAL_SUCCE-SS	Always returns success.

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_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_clock_-source*	Clock source for the ssi divider.
in	*amb_clock_-frequency*	Clock frequency of the selected source.

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.

Return values

AMB_HAL_SUCCESS	The function always returns success.

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_base_address*	Virtual address where ambhal is loaded by OS.
`in`	*amb_clock_frequency*	New ssi2 frequency

Return values

AMB_HAL_SUCCESS	the new frequency has been set
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_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

amb_clock_-frequency_t	Requested clock frequency

8.34.2.3 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*) [static]

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

Parameters

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_clock_-source*	Clock source for the ssi divider.
in	*amb_clock_-frequency*	Clock frequency of the selected source.

Note

: Valid clock sources are: AMB_REFERENCE_CLOCK_SOURCE_APB or AM-B_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.

Return values

AMB_HAL_SUCCE-SS	The function always returns success.

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_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_clock_-frequency*	New uart clock frequency

Return values

AMB_HAL_SUCCE-SS	The frequency has been set
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_base_address*	Virtual address where ambhal is loaded by OS.

Return values

amb_clock_frequency_t	Requested clock frequency.

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_base_address*	Virtual address where ambhal is loaded by OS.

Return values

AMB_HAL_SUCCESS	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_base_address*	Virtual address where ambhal is loaded by OS.
out	*amb_uart_divider_configuration*	Clock configuration information read from divider.

Return values

AMB_HAL_SUCCESS	Always returns success.

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_base_address*	Virtual address where ambhal is loaded by OS.
in	*amb_clock_source*	Clock source for the uart divider.

Note

: Valid clock sources are: AMB_REFERENCE_CLOCK_SOURCE_IDSP, AMB_REFERENCE_CLOCK_SOURCE_ARM, AMB_REFERENCE_CLOCK_SOURCE_CORE or AMB_REFERENCE_CLOCK_SOURCE_CLK_REF.

Return values

AMB_HAL_SUCCESS	The function always returns success.

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_base_address*	Virtual address where ambhal is loaded by OS.
`in`	*amb_clock_frequency*	New vin clock frequency

Return values

AMB_HAL_SUCCESS	The frequency has been set
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.

Parameters

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

amb_clock_-frequency_t	Requested clock frequency.

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_-base_-address*	Virtual address where ambhal is loaded by OS.
out	*amb_vin_-divider_-configuration*	Clock configuration information read from divider.

Return values

AMB_HAL_SUCCE-SS	Always returns success.

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

Parameters

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

AMB_HAL_SUCCE-SS	reset sequence has completed

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_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

AMB_HAL_SUCCE-SS	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_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*usb_port_-state*	Requested State of the USB Device

Return values

AMB_HAL_SUCCESS	Port state has been set
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_base_address*	Virtual address where ambhal is loaded by OS.

Return values

AMB_USB_ON	if USB Device is enabled
AMB_USB_SUSPEND	if USB Device is suspended

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

8.38.2.1 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]

Set the clock source for vout.

Parameters

in	*amb_hal_-base_-address*	Virtual address where ambhal is loaded by OS.
in	*amb_clock_-source*	the new clock source.
in	*amb_clock_-frequency*	the clock frequency of the new source.

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_SUCCESS	a new clock source has been set.
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_base_address*	Virtual address where ambhal is loaded by OS.
in	*amb_vout_clock_frequency*	The requested frequency.

Return values

AMB_HAL_SUCCESS	the new requested pll frequency is valid and it has been programmed.
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_base_address*	Virtual address where ambhal is loaded by OS.
out	*amb_vout_pll_configuration*	Sensor pll configuration information read from pll registers.

Return values

AMB_HAL_SUCCESS	always returns success.

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_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

AMB_HAL_SUCCE-SS	the pll has locked to the new frequency.
AMB_HAL_FAIL	the pll lock has failed to lock in a reasonable amount of time. something 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_-base_-address*	Virtual address where ambhal is loaded by OS.

Return values

AMB_HAL_SUCCE-SS	Always returns success

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.

ALl 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:

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

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