Ambarella

SDK6 API B5

Draft Version 0.1

October 28, 2014

US
3101 Jay Street
Ste. 110
Santa Clara, CA 95054, USA
Phone: +1.408.734.8888
Fax: +1.408.734.0788

Taiwan
Suite C1, No. 1, Li-Hsin Road 1
Science-Based Industrial Park
Hsinchu 30078, Taiwan
Phone: +886.3.666.8828
Fax: +886.3.666.1282

Hong Kong
Unit A&B, 18/F, Spectrum Tower
53 Hung To Road
Kwun Tong, Kowloon
Phone: +85.2.2806.8711
Fax: +85.2.2806.8722

Japan - Yokohama
Shin-Yokohama Business Center Bldg. 5th Floor
3-2-6 Shin-Yokohama, Kohoku-ku,
Yokohama, Kanagawa, 222-0033, Japan
Phone: +81.45.548.6150
Fax: +81.45.548.6151

Korea
6 Floor, Hanwon-Bldg.
Sunae-Dong, 6-1, Bundang-Gu
SeongNam-City, Kyunggi-Do
Republic of Korea 463-825
Phone: +031.717.2780
Fax: +031.717.2782

China - Shenzhen
Unit E, 5th Floor
No. 2 Finance Base
8 Ke Fa Road
Shenzhen, 518057, China
Phone: +86.755.3301.0366
Fax: +86.755.3301.0966

China - Shanghai
9th Floor, Park Center
1088 Fangdian Road, Pudong New District
Shanghai 201204, China
Phone: +86.21.6088.0608
Fax: +86.21.6088.0366

I Contents

II Preface

This document provides technical details using a set of consistent typographical conventions to help the user differentiate key concepts at a glance.

Conventions include:

Example	Description
AmbaGuiGen, **DirectUSB** **Save**, **File > Save** **Power**, **Reset**, **Home**	Software names GUI commands and command sequences Computer / Hardware buttons
Flash_IO_control **da**, **status**, **enable**	Register names and register fields. For example, **Flash_IO_control** is the register for global control of Flash I/O, and bit 17 (**da)** is used for DMA acknowledgement.
GPIO81, **CLK_AU**	Hardware external pins
VIL, VIH, VOL, VOH	Hardware pin parameters
INT_O, RXDATA_I	Hardware pin signals
AmbaI2C_Init() **AMBA_I2C_CTRL_s AMBA_I2C_CHANNEL_e** **AMBA_GIC_ISR_fAMBA_KAL_TASK_t**	API Functions API Structures API Enumerations API Function pointers API Typedef of ThreadX kernel abstraction layer

`DSC_Platform\Tools`
`AmbaI2C.h`
`RetStatus = AmbaI2C_Init();`
	User entries into software dialogues and GUI windows
	File names and paths
	Command line scripting and Code

Table II-1. Typographical Conventions for Technical Documents.

Additional Ambarella typographical conventions include:

- Acronyms are given in UPPER CASE using the default font (e.g., AHB, ARM11 and DDRIO).
- Names of Ambarella documents and publicly available standards, specifications, and databooks appear in *italic* type.

1 Overview

1.1 Overview: Introduction

This document provides Ambarella B5 (AmbaB5) library application programming interfaces (APIs) for multi-channel digital processing products. The library includes the following modules, organized by chapter:

- Chapter 2 “B5 System”
- Chapter 3 “B5 Communication”
- Chapter 4 “B5 Pin Muxing”
- Chapter 5 “B5 PLL Clock Generator (PLL)”
- Chapter 6 “B5 Serial Synchronous / Serial Peripheral Interfaces (SPI / SSI)”
- Chapter 7 “B5 Inter-Integrated Circuit (I2C / IDC)”
- Chapter 8 “B5 Video Input (VIN)”
- Chapter 9 “B5 Prescaler”
- Chapter 10 “B5 Video Output Formatter (VOUTF)”

1.2 Overview: Scope of Document

This document focuses strictly on the B5 library APIs. Users of this document are assumed to be familiar with the B5 chip hardware, system capabilities, software architecture and reference applications. The reader is referred to the following for a background overview:

- The chip B5 datasheet provides hardware pin and package details including a feature list with descriptions of chip performance, brief interface descriptions, a complete power-on configuration table and electrical characteristics.
- “*B5 Programming Reference Manual*” lists software-programmable registers accessible from the B5 CPU core, providing comprehensive information on each field of each register, as well as step-by-step programming instructions.
- “*B5 Application Note: System Hardware*” includes system hardware details for the Ambarella B5 family of co-processors.

2 B5 System

2.1 B5 System: Overview

This chapter provides functions for B5 system initialization and configuration. Please consult the relevant chip datasheet or programmer manual for details.

2.1.1 AmbaB5_Init

API Syntax:

AmbaB5_Init (AMBA_B5_CHANNEL_s *pSysB5Chan, AMBA_B5_CHANNEL_s *pCurB5Chan, AMBA_B5_PIN_CONFIG_s *pPinConfig)

Function Description:

- This function initializes B5 system, including control interfaces to be used, configurations for inter-chip communication, and configurations for inter-chip data transmission.

Parameters:

Type	Parameter	Description
AMBA_B5_CHANNEL_s	***pSysB5Chan**	Pointer to System B5 channel configuration. (**AMBA_B5_CHANNEL_s** is defined in `AmbaB5.h`). Please refer to Section 2.1.1.1 for more details.
AMBA_B5_CHANNEL_s	***pCurB5Chan**	Pointer to Current B5 channel configuration. (**AMBA_B5_CHANNEL_s** is defined in `AmbaB5.h`). Please refer to Section 2.1.1.1 for more details.
AMBA_B5_PIN_CONFIG_s	***pPinConfig**	Pointer to B5 pin configuration. (**AMBA_B5_PIN_CONFIG_s** is defined in `AmbaB5.h`). Please refer to Section 2.1.1.2 for more details.

*Table 2-1. Parameters for System API **AmbaB5_Init()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 2-2. Returns for System API **AmbaB5_Init()**.*

Example:

```
static AMBA_B5_CHANNEL_s B5_SysChan = {
    .Active = {
        [0] = AMBA_B5_CHANNEL_FAR_END,
        [1] = AMBA_B5_CHANNEL_FAR_END,
        [2] = AMBA_B5_CHANNEL_FAR_END,
        [3] = AMBA_B5_CHANNEL_FAR_END,
    },
    .Inactive = {
        [0] = AMBA_B5_CHANNEL_INTERNAL,
        [1] = AMBA_B5_CHANNEL_INTERNAL,
        [2] = AMBA_B5_CHANNEL_INTERNAL,
        [3] = AMBA_B5_CHANNEL_INTERNAL,
    },
    .SensorID = 0xF
};
```

```
    static AMBA_B5_CHANNEL_s B5_CurChan;

    /* B5 Pin configurations */
    static AMBA_B5_PIN_CONFIG_s B5_IMX122PinConfig = {
        .B5nPinMux = AMBA_B5_PIN_B5N_CTRL_SPI,
        .B5fPinMux = AMBA_B5_PIN_B5F_CTRL_PWM_DIFFERENTIAL,
        .SensorPinMux = AMBA_B5_PIN_SENSOR_CTRL_SPI,
        .VideoSyncPinMux = AMBA_B5_PIN_VIDEO_SYNC_NONE,
    };

    /* In case only CH0 and CH2 are connected */
    B5_CurChan.SensorID = 0x5;
    B5_CurChan.Active[0] = AMBA_B5_CHANNEL_FAR_END;
    B5_CurChan.Active[2] = AMBA_B5_CHANNEL_FAR_END;

    /* Initialize B5 system */
    AmbaB5_Init(&B5_SysChan, &B5_CurChan, &B5_PinConfig);
```

See Also:

None

2.1.1.1 AmbaB5_Init > AMBA_B5_CHANNEL_s

Type	Field	Description
AMBA_B5_CHANNEL_CONFIG_e	**Active[AMBA_NUM_B5_CHANNEL]**	B5 channel config: 0: AMBA_B5_CHANNEL_DISABLED 1: AMBA_B5_CHANNEL_INTERNAL 2: AMBA_B5_CHANNEL_NEAR_END 3: AMBA_B5_CHANNEL_FAR_END
AMBA_B5_CHANNEL_CONFIG_e	**Inactive[AMBA_NUM_B5_CHANNEL]**	B5 channel config: 0: AMBA_B5_CHANNEL_DISABLED 1: AMBA_B5_CHANNEL_INTERNAL 2: AMBA_B5_CHANNEL_NEAR_END 3: AMBA_B5_CHANNEL_FAR_END
UINT32	**SensorID**	Active channel config to be selected: 0x0001: Active[0] …… 0x1111: All channels

*Table 2-3. Definition of **AMBA_B5_CHANNEL_s** for System API **AmbaB5_Init()**.*

2.1.1.2 AmbaB5_Init > AMBA_B5_PIN_CONFIG_s

Type	Field	Description
AMBA_B5_PIN_B5N_CTRL_e	**Active[AMBA_NUM_B5_CHANNEL]**	B5N Pin control: 0: AMBA_B5_PIN_B5N_CTRL_SPI 1: AMBA_B5_PIN_B5N_CTRL_I2C
AMBA_B5_PIN_B5F_CTRL_e	**Inactive[AMBA_NUM_B5_CHANNEL]**	B5F Pin control: 0: AMBA_B5_PIN_B5F_CTRL_NONE 1: AMBA_B5_PIN_B5F_CTRL_PWM_DIFFERENTIAL 2: AMBA_B5_PIN_B5F_CTRL_PWM_SINGLE_ENDED
AMBA_B5_PIN_SENSOR_CTRL_e	**SensorID**	Sensor Pin control: 0: AMBA_B5_PIN_SENSOR_CTRL_SPI 1: AMBA_B5_PIN_SENSOR_CTRL_I2C0 2: AMBA_B5_PIN_SENSOR_CTRL_I2C1 3: AMBA_B5_PIN_SENSOR_CTRL_I2C_BRDIGE
AMBA_B5_PIN_VIDEO_SYNC_CTRL_e	**VideoSyncPinMux**	Video Sync Pin control: 0: AMBA_B5_PIN_VIDEO_SYNC_NONE 1: AMBA_B5_PIN_VIDEO_SYNC_HORIZONTAL 2: AMBA_B5_PIN_VIDEO_SYNC_VERTICAL 3: AMBA_B5_PIN_VIDEO_SYNC_HORIZONTAL_VERTICAL

*Table 2-4. Definition of **AMBA_B5_PIN_CONFIG_s** for System API **AmbaB5_Init()**.*

2.1.2 AmbaB5_Enable

API Syntax:

AmbaB5_Enable (AMBA_B5_CHANNEL_s *pB5Chan, UINT32 Width, UINT32 Height, AMBA_B5_COMPRESS_RATIO_e Ratio)

Function Description:

- This function enables the B5 data transmission.

Parameters:

Type	Parameter	Description
AMBA_B5_ CHANNEL_s	***pB5Chan**	Pointer to B5 channel configuration. (**AMBA_B5_CHANNEL_s** is defined in `AmbaB5.h`). Please refer to Section 2.1.1.1 for more details.
UINT32	**Width**	Prescaler output width (in pixels)
UINT32	**Height**	Prescaler output height (in lines)
AMBA_B5_ COMPRESS_ RATIO_e	**Ratio**	Compress ratio: 0: AMBA_B5_COMPRESS_NONE 1: AMBA_B5_COMPRESS_5P75 /* 5.75 bits per pixel */ 2: AMBA_B5_COMPRESS_6P5 /* 6.5 bits per pixel */ 3: AMBA_B5_COMPRESS_6P75 /* 6.75 bits per pixel */ 4: AMBA_B5_COMPRESS_7P5 /* 7.5 bits per pixel */ 5: AMBA_B5_COMPRESS_7P75 /* 7.75 bits per pixel */ 6: AMBA_B5_COMPRESS_8P5 /* 8.5 bits per pixel */ 7: AMBA_B5_COMPRESS_8P75 /* 8.75 bits per pixel */ 8: AMBA_B5_COMPRESS_9P5: /* 9.5 bits per pixel */ 9: AMBA_B5_COMPRESS_9P75: /* 9.75 bits per pixel */ 10: AMBA_B5_COMPRESS_10P5: /* 10.5 bits per pixel */

*Table 2-5. Parameters for System API **AmbaB5_Enable()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 2-6. Returns for System API **AmbaB5_Enable()**.*

Example:

```
/* Initialize B5 system */
AmbaB5_Init(&B5_SysChan, &B5_CurChan, &B5_PinConfig);
.
.
.
/* Enable B5 data transmission */
AmbaB5_Enable(&B5_Chan, Width, Height, AMBA_B5_COMPRESS_8P5);
```

See Also:

AmbaB5_Init()

2.1.3 AmbaB5_GetNumActiveChannel

API Syntax:

AmbaB5_GetNumActiveChannel (AMBA_B5_CHANNEL_s *pB5Chan)

Function Description:

- This function gets the B5 active channel count.

Parameters:

Type	Parameter	Description
AMBA_B5_ CHANNEL_s	*** pB5Chan**	Pointer to B5 channel configuration. (**AMBA_B5_CHANNEL_s** is defined in `AmbaB5.h`). Please refer to Section 2.1.1.1 for more details.

*Table 2-7. Parameters for System API **AmbaB5_GetNumActiveChannel()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 2-8. Returns for System API **AmbaB5_GetNumActiveChannel()**.*

Example:

```
/* Get B5 active channel count */
SensorCount = AmbaB5_GetNumActiveChannel(&B5_CurChan);
```

See Also:

None

2.1.4 AmbaB5_PwmReset

API Syntax:

AmbaB5_PwmReset (AMBA_B5_CHIP_ID_u ChipID)

Function Description:

- This function resets the error status of the PWM encoder of B5N for specific channels.

Parameters:

Type	Parameter	Description
AMBA_B5_ CHIP_ID_u	**ChipID**	ChipID.Data: 0x0000: B5N (not allowed here) 0x0001: B5F0 …… 0x1111: All B5F (AMBA_B5_CHIP_ID_u is defined in AmbaB5.h). Please refer to Section 2.1.4.1 for more details.

*Table 2-9. Parameters for System API **AmbaB5_PwmReset()**.*

Returns:

None

Example:

```
AMBA_B5_CHIP_ID_u ChipID;

/* Reset the error status of PWM encoder for CH2 */
ChipID = 0x4;
AmbaB5_PwmReset(ChipID);
```

See Also:

None

2.1.4.1 AmbaB5_PwmReset > AMBA_B5_CHIP_ID_u

Type	Parameter	Description
UINT8	**Data**	Chip ID data
UINT8	**Chan0B5F: 1**	[0]: Channel B5F0
UINT8	**Chan1B5F: 1**	[1]: Channel B5F1
UINT8	**Chan2B5F: 1**	[2]: Channel B5F2
UINT8	**Chan3B5F: 1**	[3]: Channel B5F3
UINT8	**Reserved: 4**	[7:4]: Reserved

*Table 2-10. Definition of **AMBA_B5_CHIP_ID_u** for System API **AmbaB5_PwmReset()**.*

3 B5 Communication

3.1 B5 Communication: Overview

This chapter provides APIs related to B5 communication.

3.1.1 AmbaB5_CommInit

API Syntax:

AmbaB5_CommInit (void)

Function Description:

- This function is used to initialize the B5 Communication, which is used in **AmbaB5_Init()** for Ambarella chips to access B5N and B5F.

Parameters:

None

Returns:

Return	Description
0	Success
- 1	Failure

*Table 3-1. Returns for Communication API **AmbaB5_CommInit()**.*

Example:

None

See Also:

AmbaB5_Init()

3.1.2 AmbaB5_RegWrite

API Syntax:

AmbaB5_RegWrite (UINT16 ChipIdData, UINT32 RegAddr, UINT8 IsIncAddr, AMBA_B5_DATA_WIDTH_e DataWidth, int DataSize, void *pTxDataBuf)

Function Description:

- This function is used to write B5N/B5F registers.

Parameters:

Type	Parameter	Description
UINT16	**ChipIdData**	0: B5N 1/2/4/8: B5F0/B5F1/B5F2/B5F3
UINT32	**RegAddr**	Register address
UINT8	**IsIncAddr**	Increment address for continuous register read/write 0: Fixed address 1: Increment
AMBA_B5_DATA_WIDTH_e	**DataWidth**	Data unit is 8/16/32-bit 0: AMBA_B5_DATA_WIDTH_8BIT 1: AMBA_B5_DATA_WIDTH_16BIT 2: AMBA_B5_DATA_WIDTH_32BIT
int	**DataSize**	Size of data to be transmitted
void	***pTxDataBuf**	Pointer to Tx data buffer

*Table 3-2. Parameters for Communication API **AmbaB5_RegWrite()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 3-3. Returns for Communication API **AmbaB5_RegWrite()**.*

Example:

```
UINT32 TxData[1];

/* Write 0x1 to Enable register of B5N SPI module */
TxData[0] = 0x1;
AmbaB5_RegWrite(0, (UINT32)&(pAmbaB5_SpiReg->Enable), 0, AMBA_B5_DATA_
WIDTH_32BIT, 1, TxData);

/* Write 0x1 to Enable register of B5F2 SPI module */
TxData[0] = 0x1;
AmbaB5_RegWrite(4, (UINT32)&(pAmbaB5_SpiReg->Enable), 0, AMBA_B5_DATA_
WIDTH_32BIT, 1, TxData);
```

See Also:

AmbaB5N_RegWrite(),
AmbaB5F_RegWrite()

3.1.3 AmbaB5_RegRead

API Syntax:

AmbaB5_RegRead (UINT16 ChipIdData, UINT32 RegAddr, UINT8 IsIncAddr, AMBA_B5_DATA_WIDTH_e DataWidth, int DataSize, void *pRxDataBuf)

Function Description:

- This function is used to read B5N/B5F registers.

Parameters:

Type	Parameter	Description
UINT16	**ChipIdData**	0: B5N 1/2/4/8: B5F0/B5F1/B5F2/B5F3
UINT32	**RegAddr**	Register address
UINT8	**IsIncAddr**	Increment address for continuous register read/write 0: Fixed address 1: Increment
AMBA_B5_ DATA_WIDTH_e	**DataWidth**	Data unit is 8/16/32-bit 0: AMBA_B5_DATA_WIDTH_8BIT 1: AMBA_B5_DATA_WIDTH_16BIT 2: AMBA_B5_DATA_WIDTH_32BIT
int	**DataSize**	Size of data to be transmitted
void	***pRxDataBuf**	Pointer to Rx data buffer

*Table 3-4. Parameters for Communication API **AmbaB5_RegRead()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 3-5. Returns for Communication API **AmbaB5_RegRead()**.*

Example:

```
UINT32 RxData[1];

/* Read setting from Enable register of B5N SPI module */
AmbaB5_RegRead(0, (UINT32)&(pAmbaB5_SpiReg->Enable), 0, AMBA_B5_DATA_
WIDTH_32BIT, 1, RxData);
.
.
.
/* Read setting from Enable register of B5F2 SPI module */
AmbaB5_RegRead(4, (UINT32)&(pAmbaB5_SpiReg->Enable), 0,
AMBA_B5_DATA_WIDTH_32BIT, 1, RxData);
```

See Also:

AmbaB5N_RegRead()
AmbaB5F_RegRead()

3.1.4 AmbaB5N_RegWrite

API Syntax:

AmbaB5N_RegWrite (UINT32 RegAddr, UINT8 IsIncAddr, AMBA_B5_DATA_WIDTH_e DataWidth, int DataSize, void *pTxDataBuf)

Function Description:

- This function is used to write B5N registers.

Parameters:

Type	Parameter	Description
UINT32	**RegAddr**	Register address
UINT8	**IsIncAddr**	Increment address for continuous register read/write 0: Fixed address 1: Increment
AMBA_B5_ DATA_WIDTH_e	**DataWidth**	Data unit is 8/16/32-bit 0: AMBA_B5_DATA_WIDTH_8BIT 1: AMBA_B5_DATA_WIDTH_16BIT 2: AMBA_B5_DATA_WIDTH_32BIT
int	**DataSize**	Size of data to be transmitted
void	***pTxDataBuf**	Pointer to Tx data buffer

*Table 3-6. Parameters for Communication API **AmbaB5N_RegWrite()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 3-7. Returns for Communication API **AmbaB5N_RegWrite()**.*

Example:

```
UINT32 TxData[1];

/* Write 0x1 to Enable register of B5N SPI module */
TxData[0] = 0x1;
AmbaB5N_Write((UINT32)&(pAmbaB5_SpiReg->Enable), 0,
AMBA_B5_DATA_WIDTH_32BIT, 1, TxData);
```

See Also:

AmbaB5_RegWrite()

3.1.5 AmbaB5N_RegRead

API Syntax:

AmbaB5N_RegRead (UINT32 RegAddr, UINT8 IsIncAddr, AMBA_B5_DATA_WIDTH_e DataWidth, int DataSize, void *pRxDataBuf)

Function Description:

- This function is used to read B5N registers.

Parameters:

Type	Parameter	Description
UINT32	**RegAddr**	Register address
UINT8	**IsIncAddr**	Increment address for continuous register read/write 0: Fixed address 1: Increment
AMBA_B5_ DATA_WIDTH_e	**DataWidth**	Data unit is 8/16/32-bit 0: AMBA_B5_DATA_WIDTH_8BIT 1: AMBA_B5_DATA_WIDTH_16BIT 2: AMBA_B5_DATA_WIDTH_32BIT
int	**DataSize**	Size of data to be transmitted
void	***pRxDataBuf**	Pointer to data buffer of each channel

*Table 3-8. Parameters for Communication API **AmbaB5N_RegRead()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 3-9. Returns for Communication API **AmbaB5N_RegRead()**.*

Example:

```
UINT32 RxData[1];

/* Read setting from Enable register of B5N SPI module */
AmbaB5N_Read((UINT32)&(pAmbaB5_SpiReg->Enable), 0,
AMBA_B5_DATA_WIDTH_32BIT, 1, RxData);
```

See Also:

AmbaB5_RegRead()

3.1.6 AmbaB5F_RegWrite

API Syntax:

AmbaB5F_RegWrite (AMBA_B5_CHIP_ID_u ChipID, UINT32 RegAddr, UINT8 IsIncAddr, AMBA_B5_DATA_WIDTH_e DataWidth, int DataSize, void *pTxDataBuf)

Function Description:

- This function is used to write B5F registers.

Parameters:

Type	Parameter	Description
AMBA_B5_ CHIP_ID_u	**ChipID**	ChipID.Data: 0x0000: B5N (not allowed here) 0x0001: B5F0 …… 0x1111: all B5F (**AMBA_B5_CHIP_ID_u** is defined in `AmbaB5.h`). Please refer to Section 2.1.4.1 for more details.
UINT32	**RegAddr**	Register address
UINT8	**IsIncAddr**	Increment address for continuous register read/write 0: Fixed address 1: Increment
AMBA_B5_ DATA_WIDTH_e	**DataWidth**	Data unit is 8/16/32-bit 0: AMBA_B5_DATA_WIDTH_8BIT 1: AMBA_B5_DATA_WIDTH_16BIT 2: AMBA_B5_DATA_WIDTH_32BIT
int	**DataSize**	Size of data to be transmitted
void	***pTxDataBuf**	Pointer to data buffer

*Table 3-10. Parameters for Communication API **AmbaB5F_RegWrite()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 3-11. Returns for Communication API **AmbaB5F_RegWrite()**.*

Example:

```
UINT32 TxData[1];
AMBA_B5_CHIP_ID_u ChipID = {0};

/* Write 0x1 to Enable register of B5F2 SPI module */
ChipID.Data = 0x4;
TxData[0] = 0x1;
AmbaB5F_Write(ChipID, (UINT32)&(pAmbaB5_SpiReg->Enable), 0,
AMBA_B5_DATA_WIDTH_32BIT, 1, TxData);
```

See Also:

AmbaB5_RegWrite()

3.1.7 AmbaB5F_RegRead

API Syntax:

AmbaB5F_RegRead (AMBA_B5_CHIP_ID_u ChipID, UINT32 RegAddr, UINT8 IsIncAddr, AMBA_B5_DATA_WIDTH_e DataWidth, int DataSize, void *pRxDataBuf[4])

Function Description:

- This function is used to write B5F registers.

Parameters:

Type	Parameter	Description
AMBA_B5_CHIP_ID_u	**ChipID**	ChipID.Data: 0x0000: B5N (not allowed here) 0x0001: B5F0 …… 0x1111: all B5F (**AMBA_B5_CHIP_ID_u** is defined in `AmbaB5.h`). Please refer to Section 2.1.4.1 for more details.
UINT32	**RegAddr**	Register address
UINT8	**IsIncAddr**	Increment address for continuous register read/write 0: Fixed address 1: Increment
AMBA_B5_DATA_WIDTH_e	**DataWidth**	Data unit is 8/16/32-bit 0: AMBA_B5_DATA_WIDTH_8BIT 1: AMBA_B5_DATA_WIDTH_16BIT 2: AMBA_B5_DATA_WIDTH_32BIT
int	**DataSize**	Size of data to be transmitted
void	***pRxDataBuf[4]**	Pointer to data buffer of each channel

*Table 3-12. Parameters for Communication API **AmbaB5F_RegRead()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 3-13. Returns for Communication API **AmbaB5F_RegRead()**.*

Example:

```
static UINT8 R_Buf[AMBA_NUM_B5_CHANNEL][128 << 2];
AMBA_B5_CHIP_ID_u ChipID = {0};

UINT32 *DataBuf[AMBA_NUM_B5_CHANNEL] = {(UINT32 *)R_Buf[0], (UINT32 *)
R_Buf[1], (UINT32 *) R_Buf[2], (UINT32 *) R_Buf[3]};

/* Read settings from Enable register of B5F0 and B5F2 SPI module */
ChipID.Data = 0x5;
AmbaB5F_RegRead(ChipID, (UINT32)&(pAmbaB5_SpiReg->Enable), 0,
AMBA_B5_DATA_WIDTH_32BIT, 1, (void **)DataBuf);
```

See Also:

AmbaB5_RegWrite()

3.1.8 AmbaB5_SetPwmBrokenFlag

API Syntax:

AmbaB5_SetPwmBrokenFlag (UINT8 Value)

Function Description:

- This function is used to set or reset PWM broken flag. In case of PWM broken flag raised, B5 Read/ Write functions to B5F channels will be skipped until the corresponding flags are clear.

Parameters:

Type	Parameter	Description
UINT8	**Value**	Broken Flag: 0x0000: Reset 0x0001: Set for B5F0 …… 0x1111: Set for all B5F

Table 3-14. Parameters for Communication API ***AmbaB5_SetPwmBrokenFlag()***.

Returns:

None

Example:

```
/* Reset Broken Flag */
AmbaB5_SetPwmBrokenFlag(0);
```

See Also:

AmbaB5_GetPwmBrokenFlag()

3.1.9 AmbaB5_GetPwmBrokenFlag

API Syntax:

AmbaB5_GetPwmBrokenFlag (void)

Function Description:

- This function is used to get current PWM broken flag. The flag indicates that somehow the PWM encoder of B5N did not get ACK from some of B5F channels until timeout reached.

Parameters:

None

Returns:

Return	Description
UINT8	Broken Flag

Table 3-15. Returns for API ***AmbaB5_GetPwmBrokenFlag()***.

Example

```
UINT8 BrokenFlag;

/* Get Broken Flag */
BrokenFlag = AmbaB5_SetPwmBrokenFlag();
```

See Also:

AmbaB5_SetPwmBrokenFlag()

4 B5 Pin Muxing

4.1 B5 Pin Muxing: Overview

This chapter provides APIs related to the B5N/B5F pin muxing configuration.

4.1.1 AmbaB5N_PinMuxConfig

API Syntax:

AmbaB5N_PinMuxConfig (AMBA_B5_PIN_CONFIG_s *pPinConfig)

Function Description:

- This function is used to configure B5N pin multiplexer, which is used in **AmbaB5_Init()**.

Parameters:

Type	Parameter	Description
AMBA_B5_ PIN_CONFIG_s	***pPinConfig**	Pointer to B5N pin configuration. (**AMBA_B5_PIN_CONFIG_s** is defined in `AmbaB5.h`). Please refer to Section 2.1.1.2 below for more details.

*Table 4-1. Parameters for Pin Muxing API **AmbaB5N_PinMuxConfig()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 4-2. Returns for Pin Muxing API **AmbaB5N_PinMuxConfig()**.*

Example:

None

See Also:

AmbaB5_Init()

4.1.2 AmbaB5F_PinMuxConfig

API Syntax:

AmbaB5F_PinMuxConfig (AMBA_B5_CHIP_ID_u ChipID, AMBA_B5_PIN_CONFIG_s *pPinConfig)

Function Description:

- This function is used to configure B5F pin multiplexer, which is used in **AmbaB5_Init()**.

Parameters:

Type	Parameter	Description
AMBA_B5_CHIP_ID_u	**ChipID**	ChipID.Data: 0x0000: B5N (not allowed here) 0x0001: B5F0 …… 0x1111: All B5F (**AMBA_B5_CHIP_ID_u** is defined in `AmbaB5.h`). Please refer to Section 2.1.4.1 for more details.
AMBA_B5_PIN_CONFIG_s	***pPinConfig**	Pointer to B5F pin configuration. (**AMBA_B5_PIN_CONFIG_s** is defined in `AmbaB5.h`). Please refer to Section 2.1.1.1 for more details.

*Table 4-3. Parameters for Pin Muxing API **AmbaB5F_PinMuxConfig()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 4-4. Returns for Pin Muxing API **AmbaB5F_PinMuxConfig()**.*

Example:

None

See Also:

AmbaB5_Init()

4.1.3 AmbaB5N_PinMuxSetFunc

API Syntax:

AmbaB5N_PinMuxSetFunc (AMBA_B5_PIN_FUNC_u PinFunc)

Function Description:

- This function is used to set pin function of B5N, which is used in **AmbaB5_Init()**.

Parameters:

Type	Parameter	Description
AMBA_B5_ PIN_FUNC_u	**PinFunc**	Pin function. (**AMBA_B5_PIN_FUNC_u** is defined in AmbaB5_PinMux.h). Please refer to Section 4.1.3.1 for more details.

*Table 4-5. Parameters for Pin Muxing API **AmbaB5N_PinMuxSetFunc()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 4-6. Returns for Pin Muxing API **AmbaB5N_PinMuxSetFunc()**.*

Example:

None

See Also:

AmbaB5_Init()

4.1.3.1 AmbaB5_PinMuxSetFunc > AMBA_B5_PIN_FUNC_u

Type	Field	Description
AMBA_B5_PIN_ ID_e	**Data**	Pin ID data. (**AMBA_B5_PIN_ID_e** is defined in `AmbaB5_ PinMux.h`).
UINT8	**PinID**	Pin number
UINT8	**AltFunc**	Alternative function

*Table 4-7. Definition of **AMBA_B5_PIN_FUNC_u** for Pin Muxing API **AmbaB5N_PinMuxSetFunc()**.*

4.1.4 AmbaB5F_PinMuxSetFunc

API Syntax:

AmbaB5F_PinMuxSetFunc (AMBA_B5_CHIP_ID_u ChipID, AMBA_B5_PIN_FUNC_u PinFunc)

Function Description:

- This function is used to set pin function of B5F, which is used in **AmbaB5_Init()**.

Parameters:

Type	Parameter	Description
AMBA_B5_ CHIP_ID_u	**ChipID**	ChipID.Data: 0x0000: B5N (not allowed here) 0x0001: B5F0 …… 0x1111: all B5F (**AMBA_B5_CHIP_ID_u** is defined in `AmbaB5.h`). Please refer to Section 2.1.4.1 for more details.
AMBA_B5_ PIN_FUNC_u	**PinFunc**	Pin function. (**AMBA_B5_PIN_FUNC_u** is defined in `AmbaB5_PinMux.h`). Please refer to Section 4.1.3.1 for more details.

*Table 4-8. Parameters for Pin Muxing API **AmbaB5F_PinMuxSetFunc()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 4-9. Returns for Pin Muxing API **AmbaB5F_PinMuxSetFunc()**.*

Example:

None

See Also:

AmbaB5_Init()

5 B5 PLL Clock Generator (PLL)

5.1 B5 PLL: Overview

This chapter provides functions for B5 PLL Clock Generator (PLL) operations.

5.1.1 AmbaB5_PllSetSensorClk

API Syntax:

AmbaB5_PllSetSensorClk (AMBA_B5_CHANNEL_s *pB5Chan, UINT32 Frequency)

Function Description:

- This function is used to set sensor clock generated by B5.

Parameters:

Type	Parameter	Description
AMBA_B5_ CHANNEL_s	***pB5Chan**	Pointer to B5 channel configuration. (**AMBA_B5_CHANNEL_s** is defined in `AmbaB5.h`). Please refer to Section 2.1.1.1 for more details.
UINT32	**Frequency**	Clock frequency (unit: Hz)

Table 5-1. Parameters for PLL API ***AmbaB5_PllSetSensorClk()****.*

Returns:

Return	Description
0	Success
- 1	Failure

Table 5-2. Returns for PLL API ***AmbaB5_PllSetSensorClk()****.*

Example:

```
/* Set sensor clock to be 37.125 MHz */
AmbaB5_PllSetSensorClk(&B5_Chan, 37125000);
```

See Also:

None

5.1.2 AmbaB5_PllSwPllConfig

API Syntax:

AmbaB5_PllSwPllConfig (UINT32 FrameTimeInMs)

Function Description:

- This function configures necessary parameters for B5 software PLL. This must be done before the calling of **AmbaB5_PllSwPllEnable()** each time when frame time is changed.

Parameters:

Type	Parameter	Description
UINT32	**FrameTimeInMs**	Frame time (unit: ms)

*Table 5-3. Parameters for API **AmbaB5_PllSwPllConfig()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 5-4. Returns for API **AmbaB5_PllSwPllConfig()**.*

Example:

```
/* Disable B5 software PLL */ AmbaB5_PllSwPllDisable ();
.
/* Perform sensor mode change flow here */
.
.
.
/* Set frame time as 33 ms */ AmbaB5_PllSwPllConfig (33);

/* Enable B5 software PLL */
AmbaB5_PllSwPllEnable ();
```

See Also:

AmbaB5_PllSwPllEnable()
AmbaB5_PllSwPllDisable()

5.1.3 AmbaB5_PllSwPllEnable

API Syntax:

AmbaB5_PllSwPllEnable (void)

Function Description:

- This function is used to disable B5 software PLL. B5 software PLL must be disabled at beginning of sensor mode change flow.

Parameters:

None

Returns:

Return	Description
0	Success
- 1	Failure

Table 5-5. Returns for PLL API ***AmbaB5_PllSwPllEnable()****.*

Example:

```
/* Disable B5 software PLL */
AmbaB5_PllSwPllDisable ();
.
/* Perform sensor mode change flow here */
.
.
.
/* Set frame time as 33 ms */
AmbaB5_PllSwPllConfig (33);

/* Enable B5 software PLL */
AmbaB5_PllSwPllEnable ()
```

See Also:

AmbaB5_PllSwPllConfig()
AmbaB5_PllSwPllDisable()

5.1.4 AmbaB5_PllSwPllDisable

API Syntax:

AmbaB5_PllSwPllDisable (void)

Function Description:

- This function is used to disable B5 software PLL.

Parameters:

None

Returns:

Return	Description
0	Success
- 1	Failure

Table 5-6. Returns for PLL API ***AmbaB5_PllSwPllDisable()***.

Example:

```
/* Disable B5 software PLL */
AmbaB5_PllSwPllDisable ();
.
/* Perform sensor mode change flow here */
.
.
.
/* Set frame time as 33 ms */
AmbaB5_PllSwPllConfig (33);

/* Enable B5 software PLL */
AmbaB5_PllSwPllEnable ();
```

See Also:

AmbaB5_PllSwPllConfig()
AmbaB5_PllSwPllEnable()

5.1.5 AmbaB5_PllSwPllShowMsg

API Syntax:

AmbaB5_PllSwPllShowMsg (void)

Function Description:

- This function is used to show the debug message of the B5 software PLL.

Parameters:

Type	Parameter	Description
UINT32	**Flag**	0: Disable 1: Enable

*Table 5-7. Parameters for PLL API **AmbaB5_PllSwPllShowMsg()**.*

Returns:

None

Example:

```
/* Enable B5 software PLL debug message */
AmbaB5_PllSwPllShowMsg(1);
```

See Also:

None

5.1.6 AmbaB5_PllSwPllVinHookFunc

API Syntax:

AmbaB5_PllSwPllVinHookFunc (UINT32 EntryArg)

Function Description:

- This function is an interrupt service routine. Users must hook this routine as vin handler after the calling of **AmbaB5_Init()**.

Parameters:

Type	Parameter	Description
UINT32	**EntryArg**	Entry argument

Table 5-8. Parameters for PLL API ***AmbaB5_PllSwPllVinHookFunc()***.

Returns:

None

Example:

None

See Also:

AmbaB5_Init()

6 B5 Serial Synchronous / Serial Peripheral Interfaces (SPI / SSI)

6.1 B5 SPI / SSI: Overview

This chapter provides the APIs for B5 SPI / SSI module, which is used to control the image sensors connected to B5N and B5Fs.

6.1.1 AmbaB5_SpiInit

API Syntax:

AmbaB5_SpiInit (void)

Function Description:

- This function initializes the B5 SPI / SSI module, which is used in **AmbaB5_Init()**.

Parameters:

None

Returns:

Return	Description
0	Success
- 1	Failure

*Table 6-1. Returns for SPI/SSI API **AmbaB5_SpiInit()**.*

Example:

None

See Also:

AmbaB5_Init()

6.1.2 AmbaB5_SpiTransfer

API Syntax:

AmbaB5_SpiTransfer (AMBA_B5_CHANNEL_s *pB5Chan, AMBA_B5_CHANNEL_e RxChan, AMBA_B5_SPI_CONFIG_s *pSpiConfig, UINT32 DataSize, UINT32 *pTxDataBuf, UINT32 *pRxDataBuf)

Function Description:

- This function starts SPI / SSI transfer.

Parameters:

Type	Parameter	Description
AMBA_B5_CHANNEL_s	***pB5Chan**	Pointer to B5 channel setting for Tx. (**AMBA_B5_CHANNEL_s** is defined in `AmbaB5.h`). Please refer to Section 2.2.1.1 for the details.
AMBA_B5_CHANNEL_e	**RxChan**	B5 channel for Rx 0: AMBA_B5_CHANNEL0 /* B5 Channel 0 */ 1: AMBA_B5_CHANNEL1 /* B5 Channel 1 */ 2: AMBA_B5_CHANNEL2 /* B5 Channel 2 */ 3: AMBA_B5_CHANNEL3 /* B5 Channel 3 */
AMBA_B5_ SPI_CONFIG_s	***pSpiConfig**	Pointer to configuration of SPI transaction. (**AMBA_B5_SPI_CONFIG_s** is defined in `AmbaB5_SPI.h`). Please refer to Section 6.1.2.1 for more details.
UINT32	**DataSize**	Data buffer size (in frame count) for either Tx or/and Rx data buffer
UINT32	***pTxDataBuf**	Pointer to the Tx data buffer (NULL - do not send data)
UINT32	***pRxDataBuf**	Pointer to the Rx data buffer (NULL - do not receive data)

*Table 6-2. Parameters for SPI/SSI API **AmbaB5_SpiTransfer()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 6-3. Returns for SPI/SSI API **AmbaB5_SpiTransfer()**.*

Example:

```
static AMBA_B5_SPI_CONFIG_s SpiConfig;
static AMBA_B5_CHANNEL_s B5_Chan = {
    .Active = {
        [0] = AMBA_B5_CHANNEL_FAR_END,
        [1] = AMBA_B5_CHANNEL_FAR_END,
        [2] = AMBA_B5_CHANNEL_FAR_END,
        [3] = AMBA_B5_CHANNEL_FAR_END,
    },
    .Inactive = {
        [0] = AMBA_B5_CHANNEL_INTERNAL,
        [1] = AMBA_B5_CHANNEL_INTERNAL,
        [2] = AMBA_B5_CHANNEL_INTERNAL,
        [3] = AMBA_B5_CHANNEL_INTERNAL,
    },
    .SensorID = 0xF
};
AMBA_B5_CHANNEL_s B5_Chan;
AMBA_B5_CHANNEL_e RxChan;
UINT32 *pDataBuf;

UINT32 SpiBuf[2][AMBA_B5_SPI_MASTER_MAX_FIFO_ENTRY] = {0};

/* SPI configuration */
SpiConfig.ProtocolMode = AMBA_B5_SPI_CPOL_HIGH_CPHA_HIGH;
SpiConfig.CsPolarity = AMBA_B5_SPI_CHIP_SELECT_ACTIVE_LOW;
SpiConfig.DataFrameSize = 8;    /* Data Frame Size in Bit */
SpiConfig.BaudRate = 22000000; /* Transfer BaudRate in Hz */

/* B5 channel for Rx */
RxChan = AMBA_B5_CHANNEL2;

/* Fill in SPI Buffer */
pDataBuf = &SpiBuf[0][0];
*pDataBuf++ = 0x02;
*pDataBuf++ = 0x02;
*pDataBuf++ = 0x0f;

/* Start SPI transfer */
AmbaB5_SpiTransfer(&B5_Chan, RxChan, &SpiConfig, 3, SpiBuf[0], SpiBuf[1]);
```

See Also:

None

6.1.2.1 AmbaB5_SpiTransfer > AMBA_B5_SPI_CONFIG_s

Type	Field	Description
AMBA_B5_SPI_PROTOCOL_MODE_e	**ProtocolMode**	SPI Protocol mode: 0: AMBA_B5_SPI_CPOL_LOW_CPHA_LOW 1: AMBA_B5_SPI_CPOL_LOW_CPHA_HIGH 2: AMBA_B5_SPI_CPOL_HIGH_CPHA_LOW 3: AMBA_B5_SPI_CPOL_HIGH_CPHA_HIGH
AMBA_B5_SPI_CHIP_SELECT_POL_e	**CsPolarity**	Slave select polarity 0: AMBA_B5_SPI_CHIP_SELECT_ACTIVE_LOW 1: AMBA_B5_SPI_CHIP_SELECT_ACTIVE_HIGH
UINT8	**DataFrameSize**	Data frame size in number of bits
UINT32	**BaudRate**	Transfer Baud Rate in Hz
UINT32	**NumDataFrames**	Number of data frames for read-only operation mode

Table 6-4. Parameters for SPI/SSI API ***AmbaB5_SpiTransfer()****.*

7 B5 Inter-Integrated Circuit (I2C / IDC)

7.1 B5 I2C / IDC: Overview

This chapter provides the APIs for B5 I2C / IDC interface.

7.1.1 AmbaB5_I2cInit

API Syntax:

AmbaB5_I2cInit (void)

Function Description:

- This function initializes B5 I2C device driver, which is used in **AmbaB5_Init()**.

Parameters:

None

Returns:

Return	Description
0	Success
- 1	Failure

*Table 7-1. Returns for API **AmbaB5_I2cInit()**.*

Example:

```
/* B5 I2C device driver initialization */
AmbaB5_I2C_Init();
```

See Also:

AmbaB5_Init()

7.1.2 AmbaB5_I2cWrite

API Syntax:

AmbaB5_I2cWrite (AMBA_B5_CHANNEL_s *pB5Chan, AMBA_B5_I2C_CHANNEL_e I2cChanNo, AMBA_B5_I2C_SPEED_e I2cSpeed, UINT8 SlaveAddr, int TxDataSize, UINT8 *pTxDataBuf)

Function Description:

- This function configures B5 I2C Master write-data operation.

Parameters:

Type	Parameter	Description
AMBA_B5_ CHANNEL_s	***pB5Chan**	Pointer to B5 channel configuration. (**AMBA_B5_CHANNEL_s** is defined in `AmbaB5.h`). Please refer to Section 2.1.1.1 for more details.
AMBA_B5_I2C_ CHANNEL_e	**I2cChanNo**	I2C channel number: 0: AMBA_B5_I2C_CHANNEL0 1: AMBA_B5_I2C_CHANNEL1
AMBA_B5_I2C_ SPEED_e	**I2cSpeed**	2C communication speed: 0: AMBA_B5_I2C_SPEED_STANDARD /* 100Kbps */ 1: AMBA_B5_I2C_SPEED_FAST /* 400Kbps */ 2: AMBA_B5_I2C_SPEED_FAST_PLUS /* 1Mbps */ 3: AMBA_B5_I2C_SPEED_HIGH /* 3.4Mbps */
UINT8	**SlaveAddr**	Slave address
int	**TxDataSize**	Data size in Byte
UINT8	**pTxDataBuf**	Pointer to the Tx data buffer (the first Byte can be the I2C Subaddress)

Table 7-2. Parameters for I2C/IDC API ***AmbaB5_I2cWrite()****.*

Returns:

Return	Description
0	Success
- 1	Failure

Table 7-3. Returns for I2C/IDC API ***AmbaB5_I2cWrite()****.*

Example:

```
#define SENSOR_SLAVE_ADDR      0x6C
UINT8 TxDataBuf[3];

TxDataBuf[0]=Data0;
TxDataBuf[1]=Data1;
TxDataBuf[2]=Data2;

/* B5 I2C device driver initialization */
AmbaB5_I2C_Init();


/* Write data operation */
AmbaB5_I2cWrite(pB5Chan, AMBA_B5_I2C_CHANNEL0, AMBA_B5_I2C_SPEED_STANDARD,
SENSOR_SLAVE_ADDR, 3, &TxDataBuf);
```

See Also:

AmbaB5_I2cBurstWrite()

7.1.3 AmbaB5_I2cBurstWrite

API Syntax:

AmbaB5_I2cBurstWrite (AMBA_B5_CHANNEL_s *pB5Chan, AMBA_B5_I2C_CHANNEL_e I2cChanNo, AMBA_B5_I2C_SPEED_e I2cSpeed, UINT8 SlaveAddr, int TxDataSize, UINT8 *pTxDataBu)

Function Description:

- This function configures the B5 I2C Master burst-write-data operation.

Parameters:

Type	Parameter	Description
AMBA_B5_ CHANNEL_s	***pB5Chan**	Pointer to B5 channel configuration. (**AMBA_B5_CHANNEL_s** is defined in `AmbaB5.h`). Please refer to Section 2.1.1.1 for more details.
AMBA_B5_I2C_ CHANNEL_e	**I2cChanNo**	I2C channel number: 0: AMBA_B5_I2C_CHANNEL0 1: AMBA_B5_I2C_CHANNEL1
AMBA_B5_I2C_ SPEED_e	**I2cSpeed**	2C communication speed: 0: AMBA_B5_I2C_SPEED_STANDARD /* 100Kbps */ 1: AMBA_B5_I2C_SPEED_FAST /* 400Kbps */ 2: AMBA_B5_I2C_SPEED_FAST_PLUS /* 1Mbps */ 3: AMBA_B5_I2C_SPEED_HIGH /* 3.4Mbps */
UINT8	**SlaveAddr**	Slave address
int	**TxDataSize**	Data size in Byte
UINT8	**pTxDataBuf**	Pointer to the Tx data buffer (the first Byte can be the I2C Subaddress)

*Table 7-4. Parameters for I2C/IDC API **AmbaB5_I2cBurstWrite()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 7-5. Returns for I2C/IDC API **AmbaB5_I2cBurstWrite()**.*

Example:

None

See Also:

AmbaB5_I2cWrite()

7.1.4 AmbaB5_I2cRead

API Syntax:

AmbaB5_I2cRead (AMBA_B5_I2C_CHANNEL_e I2cChanNo, AMBA_B5_I2C_SPEED_e I2cSpeed, UINT8 SlaveAddr, int RxDataSize, UINT8 *pRxDataBuf, AMBA_B5_CHANNEL_e RxChan)

Function Description:

- This function configures the B5 I2C Master read-data operation.

Parameters:

Type	Parameter	Description
AMBA_B5_ CHANNEL_s	***pB5Chan**	Pointer to B5 channel configuration. (**AMBA_B5_CHANNEL_s** is defined in `AmbaB5.h`). Please refer to Section 2.1.1.1 for more details.
AMBA_B5_I2C_ CHANNEL_e	**I2cChanNo**	I2C channel number 0: AMBA_B5_I2C_CHANNEL0 1: AMBA_B5_I2C_CHANNEL1
AMBA_B5_I2C_ SPEED_e	**I2cSpeed**	I2C communication speed: 0: AMBA_B5_I2C_SPEED_STANDARD /* 100Kbps */ 1: AMBA_B5_I2C_SPEED_FAST /* 400Kbps */ 2: AMBA_B5_I2C_SPEED_FAST_PLUS /* 1Mbps */ 3: AMBA_B5_I2C_SPEED_HIGH /* 3.4Mbps */
UINT8	**SlaveAddr**	Slave address
int	**RxDataSize**	Data size in Byte
UINT8	**pRxDataBuf**	Pointer to the Rx data buffer
AMBA_B5_ CHANNEL_e	**RxChan**	B5 channel for RX: 0: AMBA_B5_CHANNEL0 /* B5 Channel 0 */ 1: AMBA_B5_CHANNEL1 /* B5 Channel 1 */ 2: AMBA_B5_CHANNEL2 /* B5 Channel 2 */ 3: AMBA_B5_CHANNEL3 /* B5 Channel 3 */

*Table 7-6. Parameters for I2C/IDC API **AmbaB5_I2cRead()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 7-7. Returns for I2C/IDC API **AmbaB5_I2cRead()**.*

Example:

```
#define SENSOR_SLAVE_ADDR      0x6C
UINT8 RxDataBuf[3];

/* B5 I2C device driver initialization */
AmbaB5_I2C_Init();

/* Read data operation */
AmbaB5_I2cRead(pB5Chan, AMBA_B5_I2C_CHANNEL0, AMBA_B5_I2C_SPEED_STANDARD,
SENSOR_SLAVE_ADDR, 3, &RxDataBuf, AMBA_B5_CHANNEL0)
```

See Also:

None

7.1.5 AmbaB5_I2cReadAfterWrite

API Syntax:

AmbaB5_I2cReadAfterWrite (AMBA_B5_CHANNEL_s *pB5Chan, AMBA_B5_I2C_CHANNEL_e I2cChanNo, AMBA_B5_I2C_SPEED_e I2cSpeed, int NumTxTransaction AMBA_B5_I2C_TRANSACTION_s *pTxTransaction, AMBA_B5_I2C_TRANSACTION_s *pRxTransaction, AMBA_B5_CHANNEL_e RxChan)

Function Description:

- This function configures the B5 I2C Master write-then-read data operation.

Parameters:

Type	Parameter	Description
AMBA_B5_CHANNEL_s	***pB5Chan**	Pointer to B5 channel configuration. (**AMBA_B5_CHANNEL_s** is defined in `AmbaB5.h`). Please refer to Section 2.1.1.1 for more details.
AMBA_B5_I2C_CHANNEL_e	**I2cChanNo**	I2C channel number 0: AMBA_B5_I2C_CHANNEL0 1: AMBA_B5_I2C_CHANNEL1
AMBA_B5_I2C_SPEED_e	**I2cSpeed**	I2C communication speed 0: AMBA_B5_I2C_SPEED_STANDARD /* 100Kbps */ 1: AMBA_B5_I2C_SPEED_FAST /* 400Kbps */ 2: AMBA_B5_I2C_SPEED_FAST_PLUS /* 1Mbps */ 3: AMBA_B5_I2C_SPEED_HIGH /* 3.4Mbps */
int	**NumTxTransaction**	Number of TX transaction
AMBA_B5_I2C_TRANSACTION_s	***pTxTransaction**	TX transactions. (**AMBA_B5_I2C_TRANSACTION_s** is defined in `AmbaB5_I2C.h`). Please refer to Section 7.1.5.1 for more details.
AMBA_B5_I2C_TRANSACTION_s	***pRxTransaction**	RX transaction. (**AMBA_B5_I2C_TRANSACTION_s** is defined in `AmbaB5_I2C.h`). Please refer to Section 7.1.5.1 for more details.
AMBA_B5_CHANNEL_e	**RxChan**	B5 channel for RX 0: AMBA_B5_CHANNEL0 /* B5 Channel 0 */ 1: AMBA_B5_CHANNEL1 /* B5 Channel 1 */ 2: AMBA_B5_CHANNEL2 /* B5 Channel 2 */ 3: AMBA_B5_CHANNEL3 /* B5 Channel 3 */

*Table 7-8. Parameters for I2C/IDC API **AmbaB5_I2cReadAfterWrite()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 7-9. Returns for API **AmbaB5_I2cReadAfterWrite()**.*

Example:

```
#define SENSOR_SLAVE_ADDR      0x6C

AMBA_B5_I2C_TRANSACTION_s I2cTransaction[2];
UINT8 TxDataBuf[3];
UINT8 RxDataBuf[3];
UINT8 DataSize; = 3;

TxDataBuf[0] = RegAddr >> 8;;
TxDataBuf[1] = RegAddr & 0xFF;
DataSize = 3;

I2cTransaction[0].SlaveAddr = SENSOR_SLAVE_ADDR;
I2cTransaction[0].DataSize  = 2;
I2cTransaction[0].pDataBuf  = TxDataBuf;
I2cTransaction[1].SlaveAddr = SENSOR_SLAVE_ADDR | 0x1;
I2cTransaction[1].DataSize  = DataSize - 2;
I2cTransaction[1].pDataBuf  = RxDataBuf;

/* B5 I2C device driver initialization */
AmbaB5_I2C_Init();

/* B5 I2C Master write and then read data operation */
AmbaB5_I2cReadAfterWrite(pB5Chan, AMBA_B5_I2C_CHANNEL0,
AMBA_B5_I2C_SPEED_STANDARD, 1, &I2cTransaction[0], &I2cTransaction[1],
AMBA_B5_CHANNEL0)
```

See Also:

None

7.1.5.1 AmbaB5_I2cReadAfterWrite > AMBA_B5_I2C_TRANSACTION_s

Type	Field	Description
UINT8	**SlaveAddr**	I2C Slave address
int	**DataSize**	Data buffer size
UINT8	***pDataBuf**	Data buffer base address

*Table 7-10. Definition of **AMBA_B5_I2C_TRANSACTION_s** for I2C/IDC API **AmbaB5_I2cReadAfterWrite()**.*

8 B5 Video Input (VIN)

8.1 B5 VIN: Overview

This chapter provides the API for B5 VIN module.

8.1.1 AmbaB5_VinReset

API Syntax:

AmbaB5_VinReset (AMBA_B5_CHANNEL_s *pB5Chan)

Function Description:

- This function is used to reset VIN PHY.

Parameters:

Type	Parameter	Description
AMBA_B5_ CHANNEL_s	***pB5Chan**	Pointer to B5 channel configuration. (**AMBA_B5_CHANNEL_s** is defined in `AmbaB5.h`). Please refer to Section 2.1.1.1 for more details.

Table 8-1. Parameters for VIN API ***AmbaB5_VinReset()***.

Returns:

Return	Description
0	Success
- 1	Failure

Table 8-2. Returns for VIN API ***AmbaB5_VinReset()***.

Example:

```
/* Reset B5 VIN PHY */
AmbaB5_VinReset(&B5_Chan);
```

See Also:

None

8.1.2 AmbaB5_VinPhySetDVP

API Syntax:

AmbaB5_VinPhySetDVP (AMBA_B5_CHANNEL_s *pB5Chan)

Function Description:

- This function is used to set PHY for the DVP sensor.

Parameters:

Type	Parameter	Description
AMBA_B5_ CHANNEL_s	***pB5Chan**	Pointer to B5 channel configuration. (**AMBA_B5_CHANNEL_s** is defined in `AmbaB5.h`). Please refer to Section 2.1.1.1 for more details.

Table 8-3. Parameters for VIN API ***AmbaB5_VinPhySetDVP()****.*

Returns:

Return	Description
0	Success
- 1	Failure

Table 8-4. Returns for VIN API ***AmbaB5_VinPhySetDVP()****.*

Example:

```
/* Set PHY for DVP sensor */
AmbaB5_VinPhySetDVP(&B5_Chan);
```

See Also:

None

8.1.3 AmbaB5_VinPhySetSLVS

API Syntax:

AmbaB5_VinPhySetSLVS (AMBA_B5_CHANNEL_s *pB5Chan)

Function Description:

- This function is used to set PHY for the SLVS sensor.

Parameters:

Type	Parameter	Description
AMBA_B5_ CHANNEL_s	***pB5Chan**	Pointer to B5 channel configuration. (**AMBA_B5_CHANNEL_s** is defined in `AmbaB5.h`). Please refer to Section 2.1.1.1 for more details.

*Table 8-5. Parameters for VIN API **AmbaB5_VinPhySetSLVS()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 8-6. Returns for VIN API **AmbaB5_VinPhySetSLVS()**.*

Example:

```
/* Set PHY for SLVS sensor */
AmbaB5_VinPhySetSLVS(&B5_Chan);
```

See Also:

None

8.1.4 AmbaB5_VinPhySetMIPI

API Syntax:

AmbaB5_VinPhySetMIPI (AMBA_B5_CHANNEL_s *pB5Chan, AMBA_B5_VIN_MIPI_DPHY_CONFIG_s *pDphyConfig)

Function Description:

- This function is used to set PHY for the MIPI sensor.

Parameters:

Type	Parameter	Description
AMBA_B5_CHANNEL_s	***pB5Chan**	Pointer to B5 channel config. (**AMBA_B5_CHANNEL_s** is defined in `AmbaB5.h`). Please refer to Section 2.1.1.1 for more details.
AMBA_B5_VIN_MIPI_DPHY_CONFIG_s	***pDphyConfig**	Pointer to D-PHY config. (**AMBA_B5_VIN_MIPI_DPHY_CONFIG_s** is defined in `AmbaB5_VIN.h`). Please refer to Section 8.1.4.1 for more details.

*Table 8-7. Parameters for VIN API **AmbaB5_VinPhySetMIPI()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 8-8. Returns for VIN API **AmbaB5_VinPhySetMIPI()**.*

Example:

```
/* Set PHY for MIPI sensor */
AmbaB5_VinPhySetMIPI(&B5_Chan, &DphyConfig);
```

See Also:

None

8.1.4.1 AmbaB5_VinPhySetMIPI > AMBA_B5_VIN_MIPI_DPHY_CONFIG_s

Type	Field	Description
UINT8	**HsSettleTime**	D-PHY HS-SETTLE time
UINT8	**HsTermTime**	D-PHY HS-TERM time
UINT8	**ClkSettleTime**	D-PHY CLK-SETTLE time
UINT8	**ClkTermTime**	D-PHY CLK-TERM time
UINT8	**ClkMissTime**	D-PHY CLK-MISS time
UINT8	**RxInitTime**	D-PHY RX-INIT time

*Table 8-9. Definition of **AMBA_B5_VIN_MIPI_DPHY_CONFIG_s** for VIN API **AmbaB5_VinPhySetMIPI()**.*

8.1.5 AmbaB5_VinConfigDVP

API Syntax:

AmbaB5_VinConfigDVP (AMBA_B5_CHANNEL_s *pB5Chan, AMBA_B5_VIN_DVP_CONFIG_s *pVinD-vpConfig)

Function Description:

- This function is used to transfer the user DVP VIN config to thB5 VIN register.

Parameters:

Type	Parameter	Description
AMBA_B5_ CHANNEL_s	***pB5Chan**	Pointer to B5 channel config. (**AMBA_B5_CHANNEL_s** is defined in `AmbaB5.h`). Please refer to Section 2.1.1.1 for more details.
AMBA_B5_VIN_ DVP_CONFIG_s	***pVinDvpConfig**	Pointer to user Vin config. (**AMBA_B5_VIN_DVP_CONFIG_s** is defined in `AmbaB5_VIN.h`). Please refer to Section 8.1.5.1 for more details.

*Table 8-10. Parameters for VIN API **AmbaB5_VinConfigDVP()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 8-11. Returns for VIN API **AmbaB5_VinConfigDVP()**.*

Example:

```
/* Transfer user DVP Vin config to B5 Vin register */
AmbaB5_AmbaB5_VinConfigDVP(&B5_Chan, &VinDvpConfig);
```

See Also:

None

8.1.5.1 AmbaB5_VinConfigDVP > AMBA_B5_VIN_DVP_CONFIG_s

Type	Field	Description
UINT8	**NumDataBits**	Bit depth of pixel data
AMBA_B5_VIN_ DVP_TYPE_e	**DvpType**	Type of parallel transmission (DVP) 0: AMBA_B5_VIN_DVP_SINGLE_PEL_SDR 1: AMBA_B5_VIN_DVP_SINGLE_PEL_DDR 2: AMBA_B5_VIN_DVP_DOUBLE_PEL_SDR 3: AMBA_B5_VIN_DVP_DOUBLE_PEL_DDR
AMBA_B5_VIN_ SYNC_TYPE_e	**SyncType**	BT.601/BT.656 line sync and frame sync 0: AMBA_B5_VIN_SYNC_BT601 1: AMBA_B5_VIN_SYNC_BT656_LOWER_PEL 2: AMBA_B5_VIN_SYNC_BT656_UPPER_PEL 3: AMBA_B5_VIN_SYNC_BT656_BOTH_PEL
AMBA_B5_VIN_ SIGNAL_EDGE_ TYPE_e	**DataClockEdge**	Data are valid on rising/falling clock edge 0: AMBA_B5_VIN_SIGNAL_RISING_EDGE 1: AMBA_B5_VIN_SIGNAL_FALLING_EDGE
AMBA_B5_VIN_ SIGNAL_EDGE_ TYPE_e	**HsyncPolarity**	Leading edge of H-sync/line-sync pulse 0: AMBA_B5_VIN_SIGNAL_RISING_EDGE 1: AMBA_B5_VIN_SIGNAL_FALLING_EDGE
AMBA_B5_VIN_ SIGNAL_EDGE_ TYPE_e	**VsyncPolarity**	Leading edge of V-sync/frame-sync pulse 0: AMBA_B5_VIN_SIGNAL_RISING_EDGE 1: AMBA_B5_VIN_SIGNAL_FALLING_EDGE
AMBA_B5_VIN_ SIGNAL_EDGE_ TYPE_e	**FieldPolarity**	Leading edge of field pulse 0: AMBA_B5_VIN_SIGNAL_RISING_EDGE 1: AMBA_B5_VIN_SIGNAL_FALLING_EDGE
AMBA_B5_VIN_ RX_HV_SYNC_s	**RxHvSyncCtrl**	Input H/V sync signal format (from sensor) (**AMBA_B5_VIN_RX_HV_SYNC_s** is defined in `AmbaB5_VIN.h`)
AMBA_B5_VIN_ TRIGGER_ PULSE_s	**VinTrigPulse[AMBA_B5_ NUM_ VIN_TRIGGER_PULSE];**	Vin trigger pulse (**AMBA_B5_VIN_TRIGGER_PULSE_s** is defined in `AmbaB5_VIN.h`)
AMBA_B5_VIN_ VOUT_SYNC_s	**VinVoutSync[AMBA_B5_ NUM_ VIN_VOUT_SYNC];**	Vin-Vout sync (**AMBA_B5_VIN_VOUT_SYNC_s** is defined in `AmbaB5_VIN.h`)

*Table 8-12. Definition of **AMBA_B5_VIN_DVP_CONFIG_s** for VIN API **AmbaB5_VinConfigDVP()**.*

8.1.6 AmbaB5_VinConfigSLVS

API Syntax:

AmbaB5_VinConfigSLVS (AMBA_B5_CHANNEL_s *pB5Chan, AMBA_B5_VIN_SLVS_CONFIG_s *pVin-SlvsConfig)

Function Description:

- This function is used to transfer the user SLVS VIN config to theB5 VIN register.

Parameters:

Type	Parameter	Description
AMBA_B5_ CHANNEL_s	***pB5Chan**	Pointer to B5 channel config. (**AMBA_B5_CHANNEL_s** is defined in `AmbaB5.h`). Please refer to Section 2.1.1.1 for more details.
AMBA_B5_ VIN_ SLVS_ CONFIG_s	***pVinSlvsConfig**	Pointer to user Vin config. (**AMBA_B5_VIN_SLVS_CONFIG_s** is defined in `AmbaB5_VIN.h`). Please refer to Section 8.1.6.1 for more details.

*Table 8-13. Parameters for VIN API **AmbaB5_VinConfigSLVS()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 8-14. Returns for VIN API **AmbaB5_VinConfigSLVS()**.*

Example:

```
/* Transfer user SLVS Vin config to B5 Vin register */
AmbaB5_AmbaB5_VinConfigSLVS(&B5_Chan, &VinSlvsConfig);
```

See Also:

None

8.1.6.1 AmbaB5_VinConfigDVP > AMBA_B5_VIN_DVP_CONFIG_s

Type	Field	Description
UINT8	**NumDataBits**	Bit depth of pixel data
UINT8	**NumDataLane**	Number of active data lanes
UINT8	**DataLaneSelect[4]**	Logical to Physical Lane Mapping
AMBA_B5_VIN_SLVS_CUS-TOM_ SYNC_CODE_s	**SyncDetectCtrl**	Sync code detection control (**AMBA_B5_VIN_SLVS_CUSTOM_SYNC_CODE_s** is defined in `AmbaB5_VIN.h`)
AMBA_B5_VIN_RX_HV_SYNC_s	**RxHvSyncCtrl**	Input H/V sync signal format (from sensor) (**AMBA_B5_VIN_RX_HV_SYNC_s** is defined in `AmbaB5_VIN.h`)
AMBA_B5_VIN_TX_HV_SYNC_s	**TxHvSyncCtrl**	Output H/V sync signal format (to sensor) (**AMBA_B5_VIN_TX_HV_SYNC_s** is defined in `AmbaB5_VIN.h`)
AMBA_B5_VIN_TRIGGER_PULSE_s	**VinTrigPulse[AMBA_B5_NUM_VIN_TRIGGER_PULSE];**	VIN trigger pulse (**AMBA_B5_VIN_TRIGGER_PULSE_s** is defined in `AmbaB5_VIN.h`)
AMBA_B5_VIN_VOUT_SYNC_s	**VinVoutSync[AMBA_B5_ NUM_VIN_VOUT_SYNC];**	Vin-Vout sync (**AMBA_B5_VIN_VOUT_SYNC_s** is defined in `AmbaB5_VIN.h`)

*Table 8-15. Definition of **AMBA_B5_VIN_SLVS_CONFIG_s** for VIN API **AmbaB5_VinConfigDVP()**.*

8.1.7 AmbaB5_VinConfigMIPI

API Syntax:

AmbaB5_VinConfigMIPI (AMBA_B5_CHANNEL_s *pB5Chan, AMBA_B5_VIN_MIPI_CONFIG_s *pVin-MipiConfig)

Function Description:

- This function is used to transfer the user MIPI VIN config to thB5 VIN register.

Parameters:

Type	Parameter	Description
AMBA_B5_ CHANNEL_s	***pB5Chan**	Pointer to B5 channel config. (**AMBA_B5_CHANNEL_s** is defined in `AmbaB5.h`). Please refer to Section 2.1.1.1 for more details.
AMBA_B5_VIN_ MIPI_CONFIG_s	*** pVinMipiConfig**	Pointer to user Vin config. (**AMBA_B5_VIN_MIPI_CONFIG_s** is defined in `AmbaB5_VIN.h`). Please refer to Section 8.1.7.1 for more details.

*Table 8-16. Parameters for VIN API **AmbaB5_VinConfigMIPI()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 8-17. Returns for VIN API **AmbaB5_VinConfigMIPI()**.*

Example

```
/* Transfer user MIPI Vin config to B5 Vin register */
AmbaB5_VinConfigMIPI(&B5_Chan, &VinMipiConfig);
```

See Also:

None

8.1.7.1 AmbaB5_VinConfigMIPI > AMBA_B5_VIN_MIPI_CONFIG_s

Type	Field	Description
UINT8	**NumDataBits**	Bit depth of pixel data
UINT8	**NumDataLane**	Number of active data lanes
AMBA_B5_VIN_RX_HV_SYNC_s	**RxHvSyncCtrl**	Input H/V sync signal format (from sensor) (**AMBA_B5_VIN_RX_HV_SYNC_s** is defined in `AmbaB5_VIN.h`)
AMBA_B5_VIN_TX_HV_SYNC_s	**TxHvSyncCtrl**	Output H/V sync signal format (to sensor) (**AMBA_B5_VIN_TX_HV_SYNC_s** is defined in `AmbaB5_VIN.h`)
AMBA_B5_VIN_TRIGGER_PULSE_s	**VinTrigPulse[AMBA_B5_NUM_VIN_TRIGGER_PULSE];**	VIN trigger pulse (**AMBA_B5_VIN_TRIGGER_PULSE_s** is defined in `AmbaB5_VIN.h`)
AMBA_B5_VIN_VOUT_SYNC_s	**VinTrigPulse[AMBA_B5_NUM_VIN_TRIGGER_PULSE];**	Vin-Vout sync (**AMBA_B5_VIN_VOUT_SYNC_s** is defined in `AmbaB5_VIN.h`)

*Table 8-18. Definition of **AMBA_B5_VIN_MIPI_CONFIG_s** for VIN API **AmbaB5_VinConfigMIPI**().*

8.1.8 AmbaB5_VinCaptureConfig

API Syntax:

AmbaB5_VinCaptureConfig (AMBA_B5_CHANNEL_s *pB5Chan, AMBA_B5_VIN_WINDOW_s *pCaptureWindow)

Function Description:

- This function is used to set the VIN capture configuration.

Parameters:

Type	Parameter	Description
AMBA_B5_CHANNEL_s	***pB5Chan**	Pointer to B5 channel config. (**AMBA_B5_CHANNEL_s** is defined in `AmbaB5.h`). Please refer to Section 2.1.1.1 for more details.
AMBA_B5_VIN_WINDOW_s	***pCaptureWindow**	Pointer to Vin capture window configuration. (**AMBA_B5_VIN_WINDOW_s** is defined in `AmbaB5_VIN.h`). Please refer to Section 8.1.8.1 for more details.

*Table 8-19. Parameters for VIN API **AmbaB5_VinCaptureConfig()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 8-20. Returns for VIN API **AmbaB5_VinCaptureConfig()**.*

Example:

```
/* Vin capture configuration */
AmbaB5_VinCaptureConfig(&B5_Chan, &CaptureWindow);
```

See Also:

None

8.1.8.1 AmbaB5_VinCaptureConfig > AMBA_B5_VIN_WINDOW_s

Type	Field	Description
UINT16	**StartX**	Crop Start Column
UINT16	**StartY**	Crop Start Row
UINT16	**EndX**	Crop End Column
UINT16	**EndY**	Crop End Row

*Table 8-21. Definition of **AMBA_B5_VIN_WINDOW_s** for VIN API **AmbaB5_VinCaptureConfig**().*

9 B5 Prescaler

9.1 B5 Prescaler: Overview

This chapter provides B5 Prescaler module.

9.1.1 AmbaB5_PrescalerInit

API Syntax:

AmbaB5_PrescalerInit (AMBA_B5_CHANNEL_s *pB5Chan, UINT16 InputWidth, UINT16 OutputWidth, UINT16 ReadoutMode)

Function Description:

- This function is used to initialize the B5 prescaler.

Parameters:

Type	Parameter	Description
AMBA_B5_CHANNEL_s	***pB5Chan**	Pointer to System B5 channel configuration. (**AMBA_B5_CHANNEL_s** is defined in `AmbaB5.h`). Please refer to Section 2.1.1.1 for more details.
UINT16	**InputWidth**	Source frame width
UINT16	**OutputWidth**	Output frame width
UINT16	**ReadoutMode**	Sensor readout mode 0: None downsample 1: IDSP_RGB_SAMPLE_MODE_BIN2 2: IDSP_RGB_SAMPLE_MODE_SKIP2 3: IDSP_RGB_SAMPLE_MODE_BIN2_SKIP2 4: IDSP_RGB_SAMPLE_MODE_SUM2_SKIP2 5: IDSP_RGB_SAMPLE_MODE_BIN4

*Table 9-1. Parameters for Prescaler API **AmbaB5_PrescalerInit()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 9-2. Returns for Prescaler API **AmbaB5_PrescalerInit()**.*

Example:

```
/* B5 Prescaler initialization */
AmbaB5_PrescalerInit(&B5_Chan, InputWidth, OutputWidth, 0);
```

See Also:

None

9.1.2 AmbaB5_PrescalerSetCoefficients

API Syntax:

AmbaB5_PrescalerSetCoefficients (AMBA_B5_CHIP_ID_u ChipID, UINT16 *CoefAddr)

Function Description:

- This function is used to set the LPF coefficients for the prescaler.

Parameters:

Type	Parameter	Description
AMBA_B5_ CHIP_ID_u	**ChipID**	B5N/B5F[0..3] (**AMBA_B5_CHIP_ID_u** is defined in `AmbaB5.h`). Please refer to Section 2.1.4.1 for more details.
UINT16	***CoefAddr**	Coef buffer address

*Table 9-3. Parameters for Prescaler API **AmbaB5_PrescalerSetCoefficients()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 9-4. Returns for Prescaler API **AmbaB5_PrescalerSetCoefficients()**.*

Example:

```
/* B5 Prescaler initialization */
AmbaB5_PrescalerInit(&B5_Chan, InputWidth, OutputWidth, 0);

/* Set B5 Prescaler coefs */
AmbaB5_PrescalerSetCoefficients(ChipID, CoefAddr);
```

See Also:

None

10 B5 Video Output Formatter (VOUTF)

10.1 VOUTF: Overview

This chapter provides the API for B5 VOUT module.

10.1.1 AmbaB5_VoutReset

API Syntax:

AmbaB5_VoutReset (void)

Function Description:

- This function is used to reset the VOUT formatter.

Parameters:

None

Returns:

Return	Description
0	Success
- 1	Failure

*Table 10-1. Returns for VOUTF API **AmbaB5_VoutReset()**.*

Example:

```
/* Reset Vout formatter */
AmbaB5_VoutReset();
```

See Also:

None

10.1.2 AmbaB5_VoutClear

API Syntax:

AmbaB5_VoutClear (void)

Function Description:

- This function is used to clear the VOUT formatter.

Parameters:

None

Returns:

Return	Description
0	Success
- 1	Failure

*Table 10-2. Returns for VOUTF API **AmbaB5_VoutClear()**.*

Example:

```
/* Clear Vout formatter */
AmbaB5_VoutClear();
```

See Also:

None

10.1.3 AmbaB5_VoutConfig

API Syntax:

AmbaB5_VoutConfig (AMBA_B5_VOUT_CONFIG_s *pVoutConfig)

Function Description:

- This function is used to config the VOUT module.

Parameters:

Type	Parameter	Description
AMBA_B5_ VOUT_ CONFIG_s	***pVoutConfig**	Pointer to Vout configuration. (**AMBA_B5_VOUT_CONFIG_s** is defined in `AmbaB5_VOUT.h`). Please refer to Section 10.1.3.1 for more details.

*Table 10-3. Parameters for VOUTF API **AmbaB5_VoutConfig()**.*

Returns:

Return	Description
0	Success
- 1	Failure

*Table 10-4. Returns for VOUTF API **AmbaB5_VoutConfig()**.*

Example:

```
/* Vout configuration */
AmbaB5_VoutConfig(&VoutConfig);
```

See Also:

None

10.1.3.1 AmbaB5_VoutConfig > AMBA_B5_VOUT_CONFIG_s

Type	Field	Description
AMBA_B5_CHANNEL_CONFIG_e	**Channel[AMBA_NUM_B5_CHANNEL]**	B5 channel config: 0: AMBA_B5_CHANNEL_DISABLED 1: AMBA_B5_CHANNEL_INTERNAL 2: AMBA_B5_CHANNEL_NEAR_END 3: AMBA_B5_CHANNEL_FAR_END
UINT16	**InputLinePixel**	Number of input pixels per line
UINT16	**OutputFrameLine**	Number of output frame lines
UINT8	**NumDataLane**	Number of data lanes
UINT8	**PixelWidth**	Pixel width
UINT16	**MaxHBlank**	Maximum Horizontal blank
UINT16	**MinHBlank**	Minimum Horizontal blank
UINT8	**MinVBlank**	Minimum Vertical blank
UINT8	**Reserved**	Reserved

*Table 10-5. Definition of **AMBA_B5_VOUT_CONFIG_s** for VOUTF API **AmbaB5_VoutConfig()**.*

10.1.4 AmbaB5_VoutEnable

API Syntax:

AmbaB5_VoutEnable (void)

Function Description:

- This function is used to enable the VOUT formatter.

Parameters:

None

Returns:

Return	Description
0	Success
- 1	Failure

*Table 10-6. Returns for VOUTF API **AmbaB5_VoutEnable()**.*

Example:

```
/* Enable Vout formatter */
AmbaB5_VoutEnable();
```

See Also:

None

10.1.5 AmbaB5_VoutDisable

API Syntax:

AmbaB5_VoutDisable (void)

Function Description:

- This function is used to disable the VOUT formatter.

Parameters:

None

Returns:

Return	Description
0	Success
- 1	Failure

Table 10-7. Returns for VOUTF API ***AmbaB5_VoutDisable()****.*

Example:

```
/* Disable Vout formatter */
AmbaB5_VoutDisable();
```

See Also:

None

10.1.6 AmbaB5_VoutGetLastFrameTimeStamp

API Syntax:

AmbaB5_VoutGetLastFrameTimeStamp (UINT32 *pTimeStamp0, UINT32 *pTimeStamp1, UINT32 *pTimeStamp2, UINT32 *pTimeStamp3)

Function Description:

- This function is used to get the last frame time stamps of each channel.

Parameters:

Type	Parameter	Description
UINT32	***pTimeStamp0**	Pointer to Time Stamp for channel 0
UINT32	***pTimeStamp1**	Pointer to Time Stamp for channel 1
UINT32	***pTimeStamp2**	Pointer to Time Stamp for channel 2
UINT32	***pTimeStamp3**	Pointer to Time Stamp for channel 3

Table 10-8. Parameters for VOUTF API ***AmbaB5_VoutGetLastFrameTimeStamp()***.

Returns:

Return	Description
0	Success
- 1	Failure

Table 10-9. Returns for VOUTF API ***AmbaB5_VoutGetLastFrameTimeStamp()***.

Example:

```
UINT32 TimeStamp[4];

/* Get the last frame time stamps of each channel */
AmbaB5_VoutGetLastFrameTimeStamp(&TimeStamp[0], &TimeStamp[1],
&TimeStamp[2], &TimeStamp[3]);
```

See Also:

None

Appendix 1 Additional Resources

Related resources include:

- *B5 Programming Reference Manual*
- *B5 Application Note: System Hardware*
- *B5F Serializer Chip Datasheet*
- *B5Nd De-Serializer Chip Datasheet*
- *B5Nq De-Serializer Chip Datasheet*

Please contact an Ambarella representative for digital copies.

Appendix 2 Important Notice

All Ambarella design specifications, datasheets, drawings, files, and other documents (together and separately, “materials”) are provided on an “*as is*” basis, and Ambarella makes no warranties, expressed, implied, statutory, or otherwise with respect to the materials, and expressly disclaims all implied warranties of noninfringement, merchantability, and fitness for a particular purpose. The information contained herein is believed to be accurate and reliable. However, Ambarella assumes no responsibility for the consequences of use of such information.

Ambarella Incorporated reserves the right to correct, modify, enhance, improve, and otherwise change its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

All products are sold subject to Ambarella’s terms and conditions of sale supplied at the time of order acknowledgment. Ambarella warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with its standard warranty. Testing and other quality control techniques are used to the extent Ambarella deems necessary to support this warranty.

Ambarella assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using Ambarella components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

Ambarella does not warrant or represent that any license, either expressed or implied, is granted under any Ambarella patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which Ambarella products or services are used. Information published by Ambarella regarding third-party products or services does not constitute a license from Ambarella to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from Ambarella under the patents or other intellectual property of Ambarella.

Reproduction of information from Ambarella documents is not permissible without prior approval from Ambarella.

Ambarella products are not authorized for use in safety-critical applications (such as life support) where a failure of the product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Customers acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of Ambarella products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by Ambarella. Further, Customers must fully indemnify Ambarella and its representatives against any damages arising out of the use of Ambarella products in such safety-critical applications.

Ambarella products are neither designed nor intended for use in automotive and military/aerospace applications or environments. Customers acknowledge and agree that any such use of Ambarella products is solely at the Customer’s risk, and they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

Appendix 3 Revision History

NOTE: Page numbers for previous drafts may differ from page numbers in the current version.

Version	Date	Comments
0.1	28 October 2014	Formatting.

Table A3-1. Revision History.