PDMA Driver User Guide V1.00.01

Support Chips:

ISD9160

Support Platforms:

Nuvoton

Content

1. PDMA Driver

1.1 PDMA Introduction

The ISD91XX incorporates a Peripheral Direct Memory Access (PDMA) controller that transfers data between SRAM and APB devices. The PDMA has four channels of DMA PDMA CH0~CH3). PDMA transfers are unidirectional and can be Peripheral-to-SRAM, SRAM-to-Peripheral or SRAM-to-SRAM. The peripherals available for PDMA transfer are SPI, UART, I2S, ADC and DPWM. PDMA operation is controlled for each channel by configuring a source and destination address and specifying a number of bytes to transfer. Source and destination addresses can be fixed, automatically increment or wrap around a circular buffer. When PDMA operation is complete, controller can be configured to provide CPU with an interrupt.

1.2 PDMA Feature

- Provides access to SPI, UART, I2S, ADC and DPWM peripherals.
- AMBA AHB master/slave interface, transfers can occur concurrently with CPU access to flash memory.
- PDMA source and destination addressing modes allow fixed, incrementing, and wrap-around addressing.

1.3 Type Definition

E_DRVPDMA_OPERATION

Enumeration Identifier	Value	Description
eDRVPDMA_DISABLE	0	PDMA is disabled.
eDRVPDMA_ENABLE	1	PDMA is enabled.

E_DRVPDMA_CHANNEL_INDEX

Enumeration Identifier	Value	Description
eDRVPDMA_CHANNEL_0	0	PDMA channel 0
eDRVPDMA_CHANNEL_1	1	PDMA channel 1
eDRVPDMA_CHANNEL_2	2	PDMA channel 2
eDRVPDMA_CHANNEL_3	3	PDMA channel 3

E_DRVPDMA_TARGET

Enumeration Identifier	Value	Description
eDRVPDMA_TARGET_SOURCE	0	Specified PDMA setting is source
eDRVPDMA_TARGET_DESTINATION	1	Specified PDMA setting is destination

E_DRVPDMA_INT_FLAG

Enumeration Identifier	Value	Description
eDRVPDMA_TABORT_FLAG	1	Target abort flag
eDRVPDMA_BLKD_FLAG	2	Transferred done flag
eDRVPDMA_WAR_EMPTY_FLAG	0x100	Wrap – empty flag
eDRVPDMA_WAR_THREE_FOURTHS_FLAG	0x200	Wrap – 3/4 flag
eDRVPDMA_WAR_HALF_FLAG	0x400	Wrap – half flag
eDRVPDMA_WAR_QUARTER_FLAG	0x800	Wrap – 1/4 flag

E_DRVPDMA_WRAP_INT_ENABLE

Enumeration Identifier	Value	Description
eDRVPDMA_WRA_EMPTY_INT	0x01	Wrap – empty interrupt enable
eDRVPDMA_WRA_HALF_INT	0x04	Wrap – half interrupt enable
eDRVPDMA_WAR_NO_INT	0x00	Wrap – no interrupt enable

E_DRVPDMA_DIRECTION_SELECT

Enumeration Identifier	Value	Description
eDRVPDMA_DIRECTION_INCREMENTED	0	Source/Destination Address Direction is incremented.
eDRVPDMA_DIRECTION_DECREMENTED	1	Source/Destination Address Direction is decremented.
eDRVPDMA_DIRECTION_FIXED	2	Source/Destination Address Direction is fixed.
eDRVPDMA_DIRECTION_WRAPAROUND	3	Source/Destination Address Direction is wrapped.

E_DRVPDMA_TRANSFER_WIDTH

Enumeration Identifier	Value	Description
eDRVPDMA_WIDTH_32BITS	0	One word is transferred for every PDMA operation in IP-to-Memory/Memory-to-IP mode.
eDRVPDMA_WIDTH_8BITS	1	One byte is transferred for every PDMA operation in IP-to-Memory/Memory-to-IP mode.
eDRVPDMA_WIDTH_16BITS	2	Half word is transferred for every PDMA operation in IP-to-Memory/Memory-to-IP mode.

E_DRVPDMA_INT_ENABLE

Enumeration Identifier	Value	Description
eDRVPDMA_TABORT	1	Target abort interrupt/flag
eDRVPDMA_BLKD	2	Transferred done interrupt/flag
eDRVPDMA_WAR	4	Wrap interrupt

E_DRVPDMA_APB_DEVICE

Enumeration Identifier	Value	Description
eDRVPDMA_SPI0	0	PDMA source/destination APB device is SPI0
eDRVPDMA_DPWM	1	PDMA destination APB device is DPWM
eDRVPDMA_UART0	2	PDMA source/destination APB device is UART0
eDRVPDMA_I2S	3	PDMA source/destination APB device is I2S
eDRVPDMA_ADC	4	PDMA source APB device is ADC

E_DRVPDMA_APB_RW

Enumeration Identifier	Value	Description
eDRVPDMA_READ_APB	0	Read data from APB device to memory

eDRVPDMA_WRITE_APB	1	Write data from memory to APB device

E_DRVPDMA_MODE

Enumeration Identifier	Value	Description
eDRVPDMA_MODE_MEM2MEM	0	PDMA mode is Memory-to-Memory
eDRVPDMA_MODE_APB2MEM	1	PDMA mode is APB device-to-Memory
eDRVPDMA_MODE_MEM2APB	2	PDMA mode is Memory-to-APB device

Constant Identifier	Value	Description
CHANNEL_OFFSET	0x100	PDMA channel register offset

Error Code Identifier	Value	Description
E_DRVPDMA_FALSE_INPUT	1	Non-support specified parameter
E_DRVPDMA_ERR_PORT_INVALID	2	Invalid port parameter

1.4 Functions

DrvPDMA_Init

Prototype

void

DrvPDMA_Init(void);

Description

The function is used to enable AHB PDMA engine clock.

Parameters

None

Include

Driver\DrvPDMA.h

Return Value

None

Example

/* Enable AHB PDMA engine clock */

DrvPDMA_Init();

DrvPDMA_Close

Prototype

void

DrvPDMA_Close(void);

Description

The function is used to disable all PDMA channel clock and AHB PDMA clock.

Parameters

None

Include

Driver\DrvPDMA.h

Return Value

None

Example

/* Disable all PDMA channel clock and AHB PDMA clock */

DrvPDMA_Close();

DrvPDMA_DisableInt

Prototype

void

DrvPDMA_DisableInt(

E_DRVPDMA_CHANNEL_INDEX eChannel,

E_DRVPDMA_INT_ENABLE eIntSource

);

Description

The function is used to disable interrupt for specified channel.

Parameters

eChannel [in]

Specify eDRVPDMA_CHANNEL_0~3

eIntSource [in] : Interrupt source

eDRVPDMA_TABORT: Read/Write Target Abort.

eDRVPDMA_BLKD: Block Transfer done .

eDRVPDMA_WRA: Wraparound interrupt enable.

Include

Driver\DrvPDMA.h

Return Value

E_SUCCESS: Success

E_DRVPDMA_ERR_PORT_INVALID: invalid port number

Example

/* Disable channel 3 read/write target abort interrupt */

DrvPDMA_DisableInt(eDRVPDMA_CHANNEL_3,   eDRVPDMA_TABORT);

DrvPDMA_ClearInt

Prototype

void

DrvPDMA_ClearInt(

E_DRVPDMA_CHANNEL_INDEX eChannel,

E_DRVPDMA_INT_FLAG eIntFlag

);

Description

The function is used to clear interrupt status for specified channel.

Parameters

eChannel [in]

Specify eDRVPDMA_CHANNEL_0~3

eIntFlag [in] : Interrupt source

eDRVPDMA_TABORT_FLAG: Read/Write target abort flag

eDRVPDMA_BLKD_FLAG: Block transfer done flag

eDRVPDMA_WRA_EMPTY_FLAG: Current transfer finished flag

eDRVPDMA_WRA_HALF_FLAG: Current transfer half complete flag

Include

Driver\DrvPDMA.h

Return Value

None

Example

/* Clear channel 0 block transfer done interrupt flag. */

DrvPDMA_ClearInt(eDRVPDMA_CHANNEL_0, eDRVPDMA_BLKD_FLAG);

DrvPDMA_EnableCH

Prototype

void

DrvPDMA_EnableCH(

E_DRVPDMA_CHANNEL_INDEX eChannel,

E_DRVPDMA_OPERATION eOP

);

Description

The function is used to enable channel.

Parameters

eChannel [in]

Specify eDRVPDMA_CHANNEL_0~3

eOP [in]

eDRVPDMA_DISABLE: PDMA disable

eDRVPDMA_ENABLE: PDMA enable

Include

Driver\DrvPDMA.h

Return Value

None

Example

```
/* Enable PDMA channel. 0*/
DrvPDMA_EnableCH(eDRVPDMA_CHANNEL_0, eDRVPDMA_ENABLE);
```

DrvPDMA_IsCHBusy

Prototype

```
int32_t
DrvPDMA_IsCHBusy(
E_DRVPDMA_CHANNEL_INDEX eChannel
);
```

Description

The function is used to get channel enable/disable status.

Parameters

eChannel [in]

Specify eDRVPDMA_CHANNEL_0~3

Include

Driver\DrvPDMA.h

Return Value

TRUE: The channel is busy.

FALSE: The channel is unused.

E_DRVPDMA_ERR_PORT_INVALID: invalid port number

Example

```
/* Get channel 0 bus status */
int32_t i32Channel0BusStatus;
i32Channel0BusStatus = DrvPDMA_IsCHBusy(eDRVPDMA_CHANNEL_0);
```

DrvPDMA_Open

Prototype

```
int32_t
DrvPDMA_Open(
E_DRVPDMA_CHANNEL_INDEX sChannel,
STR_PDMA_T *sParam
);
```

Description

The function configures PDMA setting.

Parameters

eChannel [in]

Specify eDRVPDMA_CHANNEL_0~3

sParam [in]

The struct parameter to configure PDMA.

It includes

sSrcAddr.u32Addr : Set Source Address

sSrcAddr.eAddrDirection : Set Source Address Direction. It could be

- eDRVPDMA_DIRECTION_INCREMENTED
- eDRVPDMA_DIRECTION_DECREMENTED
- eDRVPDMA_DIRECTION_FIXED
- eDRVPDMA_DIRECTION_WRAPAROUND

sDestAddr.u32Addr : Set Destination Address

sDestAddr.eAddrDirection : Set Destination Address Direction. It could be

- eDRVPDMA_DIRECTION_INCREMENTED
- eDRVPDMA_DIRECTION_DECREMENTED
- eDRVPDMA_DIRECTION_FIXED
- eDRVPDMA_DIRECTION_WRAPAROUND

u8TransWidth : Peripheral Transfer Width. This field is meaningful only when the operation mode setting are APB to memory or memory to APB. It could be

- eDRVPDMA_WIDTH_8BITS
- eDRVPDMA_WIDTH_16BITS
- eDRVPDMA_WIDTH_32BITS

u8Mode : Operation Mode

- eDRVPDMA_MODE_MEM2MEM
- eDRVPDMA_MODE_APM2MEM
- eDRVPDMA_MODE_MEM2APB

i32ByteCnt : PDMA Transfer Byte Count

Include

Driver\DrvPDMA.h

Return Value

E_SUCCESS: Success

E_DRVPDMA_ERR_PORT_INVALID: Invalid port number

Example

```
/* CH1 TX Setting */
sPDMA.sSrcAddr.u32Addr              = (uint32_t)SrcArray;
sPDMA.sDestAddr.u32Addr             = UARTPort;
sPDMA.u8TransWidth                  = eDRVPDMA_WIDTH_8BITS;
sPDMA.u8Mode                        = eDRVPDMA_MODE_MEM2APB;
sPDMA.sSrcAddr.eAddrDirection        = eDRVPDMA_DIRECTION_INCREMENTED;
sPDMA.sDestAddr.eAddrDirection      = eDRVPDMA_DIRECTION_FIXED;
sPDMA.i32ByteCnt                   = UART_TEST_LENGTH;
DrvPDMA_Open(eDRVPDMA_CHANNEL_1,&sPDMA);
```

DrvPDMA_IsEnabledCH

Prototype

```
int32_t
DrvPDMA_IsEnabledCH(
E_DRVPDMA_CHANNEL_INDEX eChannel
);
```

Description

The function is used to check channel enable status..

Parameters

eChannel [in]

Specify eDRVPDMA_CHANNEL_0~3

Include

Driver\DrvPDMA.h

Return Value

1: PDMA channel is enabled.

0: PDMA channel is not enabled.

Example

```
/* check channel 0 enable status */
DrvPDMA_IsEnabledCH(eDRVPDMA_CHANNEL_0);
```

DrvPDMA_GetTransferLength

Prototype

```
int32_t
```

DrvPDMA_GetTransferLength(

E_DRVPDMA_CHANNEL_INDEX eChannel,

uint32_t* pu32TransferLength

);

Description

The function is used to get channel transfer length setting. The unit of *pu32TransferLength is byte.

Parameters

eChannel [in]

Specify eDRVPDMA_CHANNEL_0~3

pu32TransferLength [in]

The data pointer to save transfer length

Include

Driver\DrvPDMA.h

Return Value

E_SUCCESS: Success

Example

/* Get the transfer byte count setting of channels. */

uint32_t u32GetTransferByteCountSetting;

DrvPDMA_GetTransferLength(eDRVPDMA_CHANNEL_0, & u32GetTransferByteCountSetting);

DrvPDMA_SetAPBTransferWidth

Prototype

int32_t

DrvPDMA_SetAPBTransferWidth(

E_DRVPDMA_CHANNEL_INDEX eChannel,

E_DRVPDMA_TRANSFER_WIDTH eTransferWidth

);

Description

The function is used to set APB transfer width for specified channel.

Parameters

eChannel [in]

Specify eDRVPDMA_CHANNEL_0~3

eTransferWidth [in]

eDRVPDMA_WIDTH_8BITS

eDRVPDMA_WIDTH_16BITS

eDRVPDMA_WIDTH_32BITS

Include

Driver\DrvPDMA.h

Return Value

E_SUCCESS: Success

E_DRVPDMA_ERR_PORT_INVALID: invalid port number

Example

/* Set channel0 peripheral bus width to 8 bits */

DrvPDMA_SetAPBTransferWidth(eDRVPDMA_CHANNEL_0, eDRVPDMA_WIDTH_8BITS)

DrvPDMA_GetAPBTransferWidth

Prototype

int32_t

DrvPDMA_GetAPBTransferWidth(

E_DRVPDMA_CHANNEL_INDEX eChannel

);

Description

The function is used to get peripheral transfer width from specified channel.

Parameters

eChannel [in]

Specify eDRVPDMA_CHANNEL_0~3

Include

Driver\DrvPDMA.h

Return Value

0: One word (32 bits) is transferred for every PDMA operation.

1: One byte (8 bits) is transferred for every PDMA operation.

2: One half-word (16 bits) is transferred for every PDMA operation.

E_DRVPDMA_ERR_PORT_INVALID: invalid port number

Example

/* get peripheral transfer width of channel 0 */

int32_t i32ChTransferWidth;

i32ChTransferWidth =DrvPDMA_GetAPBTransferWidth(eDRVPDMA_CHANNEL_0);

DrvPDMA_GetCHForAPBDevice

Prototype

int32_t

DrvPDMA_GetCHForAPBDevice(

E_DRVPDMA_APB_DEVICE eDevice,

E_DRVPDMA_APB_RW eRWAPB

);

Description

The function is used to get PDMA channel for specified APB device.

Parameters

eDevice [in]

Channel for APB device. It includes of

eDRVPDMA_SPI0, eDRVPDMA_UART0, eDRVPDMA_ADC, eDRVPDMA_DPWM,

eDRVPDMA_I2S

eRWAPB [in]

Specify APB direction

eDRVPDMA_READ_APB: APB to memory

eDRVPDMA_WRITE_APB: memory to APB

Include

Driver\DrvPDMA.h

Return Value

0: channel 0

1: channel 1

2: channel 2

3: channel 3

E_DRVPDMA_FALSE_INPUT: Wrong parameter

Example

/* Get UART0 RX PDMA channel */

int32_t i32GetChannel4APBDevice;

i32GetChannel4APBDevice=DrvPDMA_GetCHForAPBDevice(eDRVPDMA_UART0,

eDRVPDMA_READ_APB);

DrvPDMA_GetWrapIntType

Prototype

int32_t

DrvPDMA_GetWrapIntType(

E_DRVPDMA_CHANNEL_INDEX eChannel

);

Description

The function is used to get wrap int type of channel.

Parameters

eChannel [in]

Specify eDRVPDMA_CHANNEL_0~3

Include

Driver\DrvPDMA.h

Return Value

x1xx: a wrap interrupt can be generated when half each PDMA transfer is completed.

xxx1: a wrap interrupt can be generated when each PDMA transfer is wrapped.

x1x1: both half and wrap interrupts generated.

Example

/* Get wrap int type of channel 0.*/

DrvPDMA_GetWrapIntType(eDRVPDMA_CHANNEL_0);

DrvPDMA_CHSoftwareReset

Prototype

int32_t

DrvPDMA_CHSoftwareReset(

E_DRVPDMA_CHANNEL_INDEX eChannel

);

Description

The function is used to do software reset specified channel.

Parameters

eChannel [in]

Specify eDRVPDMA_CHANNEL_0~3

Include

Driver\DrvPDMA.h

Return Value

E_SUCCESS: Success.

E_DRVPDMA_ERR_PORT_INVALID: Invalid port number

Note

The function will reset the specified channel internal state machine and pointers. The contents of control register will not be cleared.

Example

/* Software reset PDMA channel0 and get returned value */

int32_t i32RetVal_CH0SoftwareReset;

i32RetVal_CH0SoftwareReset = DrvPDMA_CHSoftwareReset(eDRVPDMA_CHANNEL_0);

DrvPDMA_CHEnablelTransfer

Prototype

int32_t

DrvPDMA_CHEnablelTransfer(

E_DRVPDMA_CHANNEL_INDEX eChannel

);

Description

The function is used to enable PDMA specified channel and enable specified channel data read or write transfer.

Parameters

eChannel [in]

Specify eDRVPDMA_CHANNEL_0~3

Include

Driver\DrvPDMA.h

Return Value

E_SUCCESS: Success.

E_DRVPDMA_ERR_PORT_INVALID: Invalid port number

Example

/* Enable PDMA channel0 and enable channel0 data read/write transfer */

DrvPDMA_CHEnablelTransfer(eDRVPDMA_CHANNEL_0);

DrvPDMA_EnableInt

Prototype

int32_t

DrvPDMA_EnableInt(

E_DRVPDMA_CHANNEL_INDEX eChannel,

E_DRVPDMA_INT_ENABLE eIntSource

);

Description

The function is used to enable Interrupt for specified channel.

Parameters

eChannel [in]

Specify eDRVPDMA_CHANNEL_0~3

eIntSource [in] : Interrupt source

eDRVPDMA_TABORT: Read/Write Target Abort.

eDRVPDMA_BLKD: Block Transfer done .

eDRVPDMA_WRA: Wraparound interrupt enable.

Include

Driver\DrvPDMA.h

Return Value

E_SUCCESS: Success.

E_DRVPDMA_ERR_PORT_INVALID: Invalid port number

Example

/* Enable channel 0 block transfer done interrupt. */

DrvPDMA_EnableInt(eDRVPDMA_CHANNEL_0, eDRVPDMA_BLKD);

DrvPDMA_IsIntEnabled

Prototype

int32_t

DrvPDMA_IsIntEnabled(

E_DRVPDMA_CHANNEL_INDEX eChannel,

E_DRVPDMA_INT_ENABLE eIntSource

);

Description

The function is used to check if the specified interrupt source is enabled in specified channel.

Parameters

eChannel [in]

Specify eDRVPDMA_CHANNEL_0~3

eIntSource [in] : Interrupt source

eDRVPDMA_TABORT: Read/Write Target Abort.

eDRVPDMA_BLKD: Block Transfer done .

eDRVPDMA_WRA: Wraparound interrupt enable.

Include

Driver\DrvPDMA.h

Return Value

TRUE: The specified interrupt source of specified channel is enable.

FALSE: The specified interrupt source of specified channel is disable.

Example

Int32_t i32IsIntEnable;

i32IsIntEnable= DrvPDMA_IsIntEnabled (eDRVPDMA_CHANNEL_0, eDRVPDMA_BLKD);

if(i32IsIntEnable==TRUE)

printf("Channel0 Block transfer Done interrupt is enable!\n");

else if(i32IsIntEnable==FALSE)

printf("Channel0 Block transfer Done interrupt is disable!\n");

DrvPDMA_PollInt

Prototype

int32_t

DrvPDMA_PollInt(

E_DRVPDMA_CHANNEL_INDEX eChannel,

E_DRVPDMA_INT_FLAG eIntFlag

);

Description

The function is used to polling channel interrupt status.

Parameters

eChannel [in]

Specify eDRVPDMA_CHANNEL_0~3

eIntFlag [in] : Interrupt source

eDRVPDMA_TABORT_FLAG: Read/Write target abort flag

eDRVPDMA_BLKD_FLAG: Block transfer done flag

eDRVPDMA_WRA_EMPTY_FLAG: Current transfer finished flag

eDRVPDMA_WRA_HALF_FLAG: Current transfer half complete flag

Include

Driver\DrvPDMA.h

Return Value

TRUE: Interrupt status is set.

FALSE: Interrupt status is clear.

Example

```
/* Get channel 3 transfer done interrupt status */
int32_t i32Channel3TransferDone;
/* Enable INT*/
DrvPDMA_EnableInt(eDRVPDMA_CHANNEL_3, eDRVPDMA_BLKD);
.
/* Check channel 3 transfer done interrupt flag */
if(DrvPDMA_PollInt(eDRVPDMA_CHANNEL_3, eDRVPDMA_BLKD_FLAG)==TRUE);
        printf("Channel 3 block transfer done interrupt flag is set!!\n");
else
        printf("Channel 3 block transfer done interrupt flag is not set!!\n");
```

DrvPDMA_GetCurrentSourceAddr

Prototype

uint32_t

DrvPDMA_GetCurrentSourceAddr(

E_DRVPDMA_CHANNEL_INDEX eChannel

);

Description

The function is used to get current source address from specified channel.

Parameters

eChannel [in]

Specify eDRVPDMA_CHANNEL_0~3

Include

Driver\DrvPDMA.h

Return Value

Current source address register indicates the source address where the PDMA transfer is just occurring.

Example

/* Get channel 0 current source address */

uint32_t u32Channel0CurrentSourceAddress;

u32Channel0CurrentSourceAddress=DrvPDMA_GetCurrentSource Addr(eDRVPDMA_CHANNEL_0);

DrvPDMA_GetCurrentDestAddr

Prototype

uint32_t

DrvPDMA_GetCurrentDestAddr(

E_DRVPDMA_CHANNEL_INDEX eChannel

);

Description

The function is used to get current destination address from specified channel.

Parameters

eChannel [in]

Specify eDRVPDMA_CHANNEL_0~3

Include

Driver\DrvPDMA.h

Return Value

Current destination address which indicates the destination address where PDMA is just occurring.

Example

/* Get channel 0 current destination address */

uint32_t u32Ch0CurrDestAddr;

u32Ch0CurrDestAddr=DrvPDMA_GetCurrentDest Addr(eDRVPDMA_CHANNEL_0);

DrvPDMA_GetCurrentTransferCount

Prototype

uint32_t

DrvPDMA_GetCurrentTransferCount(

E_DRVPDMA_CHANNEL_INDEX eChannel

);

Description

The function is used to get current transfer byte count of specified channel.

Parameters

eChannel [in]

Specify eDRVPDMA_CHANNEL_0~3

Include

Driver\DrvPDMA.h

Return Value

Current transfer byte count from channel.

Example

/* Get channel 0 current transfer byte count */

uint32_t u32CurrentTransferByteCount;

u32CurrentTransferByteCount= DrvPDMA_GetCurrentTransferCount(eDRVPDMA_CHANNEL_0);

DrvPDMA_GetInternalBufPointer

Prototype

uint32_t

DrvPDMA_GetInternalBufPointer(

E_DRVPDMA_CHANNEL_INDEX eChannel

);

Description

The function is used to get internal buffer pointer for specified channel.

Parameters

eChannel [in]

Specify eDRVPDMA_CHANNEL_0~3

Include

Driver\DrvPDMA.h

Return Value

E_DRVPDMA_ERR_PORT_INVALID: invalid port

0x01: internal pointer point to byte1(one byte remained in PDMA buffer)

0x03: internal pointer point to byte2(two byte remained in PDMA buffer)

0x07: internal pointer point to byte3(three byte remained in PDMA buffer)

0x0F: internal pointer point to byte4(There is no more data remained in PDMA buffer)

Example

```
uint32_t u32PdmaInternalBufferPoint;
u32PdmaInternalBufferPoint= DrvPDMA_GetInternalBufPointer(eDRVPDMA_CHANNEL_0);
if(u32PdmaInternalBufferPoint==0x01)
        printf(“There is only one byte data remained in PDMA buffer!”);
else if(u32PdmaInternalBufferPoint==0x03)
        printf(“There is two bytes data remained in PDMA buffer!”);
else if(u32PdmaInternalBufferPoint==0x07)
        printf(“There is three bytes data remained in PDMA buffer!”);
else if(u32PdmaInternalBufferPoint==0x0f)
        printf(“There is no data in PDMA buffer!”);
```

DrvPDMA_SetTransferSetting

Prototype

```
int32_t
DrvPDMA_SetTransferSetting(
        E_DRVPDMA_CHANNEL_INDEX eChannel,
        S_DRVPDMA_CH_ADDR_SETTING* psSrcAddr,
        S_DRVPDMA_CH_ADDR_SETTING* psDestAddr,
        uint32_t u32TransferLength
);
```

Description

The function is used to set transfer setting for specified channel.

Parameters

eChannel [in]

Specify eDRVPDMA_CHANNEL_0~3

psSrcAddr [in]

Pointer to SrcAddr

psDestAddr, [in]

Pointer to DestAddr

u32TransferLength [in]

Transfer length in byte

Include

Driver\DrvPDMA.h

Return Value

E_SUCCESS: Success

E_DRVPDMA_FALSE_INPUT: invalid argument

Example

```
/* Set UART0 for TX by channel 1, source is SrcArray. */
S_DRVPDMA_CH_ADDR_SETTING sSrcAddr, sDestAddr;
sSrcAddr.u32Addr =(uint32_t)SrcArray;
sSrcAddr.eAddrDirection= eDRVPDMA_DIRECTION_INCREMENTED;
sDestAddr.u32Addr = UART0_BA;
sDestAddr.eAddrDirection= eDRVPDMA_DIRECTION_FIXED;
DrvPDMA_SetTransferSetting( eDRVPDMA_CHANNEL_1, & sSrcAddr, & sDestAddr, 64);
```

DrvPDMA_GetTransferSetting

Prototype

```
int32_t
DrvPDMA_GetTransferSetting(
    E_DRVPDMA_CHANNEL_INDEX eChannel,
    E_DRVPDMA_TARGET eTarget,
    uint32_t* pu32Addr,
    E_DRVPDMA_DIRECTION_SELECT* peDirection
);
```

Description

The function is used to get transfer setting for specified channel.

Parameters

eChannel [in]

Specify eDRVPDMA_CHANNEL_0~3

eTarget [in]

Specify PDMA source or destination:

eDRVPDMA_TARGET_SOURCE,

eDRVPDMA_TARGET_DESTINATION

pu32Addr [out]

Pointer to return address

peDirection [out]

Pointer to return direction

Include

Driver\DrvPDMA.h

Return Value

E_SUCCESS: Success

E_DRVPDMA_FALSE_INPUT: invalid Argument

Example

unit32_t u32Addr;

E_DRVPDMA_DIRECTION_SELECT eDirection

DrvPDMA_GetTransferSetting(ePDMA_CHANNEL_0, eDRVPDMA_TARGET_SOURCE, &u32Addr, &eDirection);

DrvPDMA_SetCHForAPBDevice

Prototype

```
int32_t
DrvPDMA_SetCHForAPBDevice(
        E_DRVPDMA_CHANNEL_INDEX eChannel,
        E_DRVPDMA_APB_DEVICE eDevice,
        E_DRVPDMA_APB_RW eRWAPB
);
```

Description

The function is used to select PDMA channel for APB device.

Parameters

eChannel [in]

Specify eDRVPDMA_CHANNEL_0~3

eDevice [in]

Channel for APB device. It includes of

eDRVPDMA_SPI0, eDRVPDMA_UART0, eDRVPDMA_ADC, eDRVPDMA_DPWM, eDRVPDMA_I2S

eRWAPB [in]

Specify APB direction

eDRVPDMA_READ_APB: APB to memory

eDRVPDMA_WRITE_APB: memory to APB

Include

Driver\DrvPDMA.h

Return Value

E_SUCCESS: Success

E_DRVPDMA_FALSE_INPUT: Invalid APB device

Example

/* Set PDMA channel 1 to UART0 TX port */

DrvPDMA_SetCHForAPBDevice(eDVPDMA_CHANNEL_1,eDRVPDMA_UART0, eDRVPDMA_WRITE_APB);

```
/* Set PDMA channel 0 to SPI0 RX port */
DrvPDMA_SetCHForAPBDevice(eDVPDMA_CHANNEL_0, eDRVPDMA_SPI0,
eDRVPDMA_READ_APB);
```

DrvPDMA_InstallCallBack

Prototype

```
int32_t
DrvPDMA_InstallCallBack(
    E_DRVPDMA_CHANNEL_INDEX eChannel,
    E_DRVPDMA_INT_ENABLE eIntSource,
    PFN_DRVPDMA_CALLBACK pfncallback
);
```

Description

The function is used to install callback function for specified channel and interrupt source.

Parameters

eChannel [in]

Specify eDRVPDMA_CHANNEL_0~3

eIntSource [in] : Interrupt source

eDRVPDMA_TABORT: Read/Write Target Abort.

eDRVPDMA_BLKD: Block Transfer done .

eDRVPDMA_WRA: Wraparound interrupt enable.

pfncallback [in]

The callback function pointer

Include

Driver\DrvPDMA.h

Return Value

E_SUCCESS: Success

Example

```
/* install PDMA0_Callback function for channel 0 for transfer done. */
DrvPDMA_InstallCallBack(eDRVPDMA_CHANNEL_0,eDRVPDMA_BLKD,
    (PFN_DRVPDMA_CALLBACK) PDMA0_Callback );
```

DrvPDMA_GetVersion

Prototype

```
int32_t
DrvPDMA_GetVersion(void);
```

Description

Return the current version number of driver.

Parameters

None

Include

Driver\DrvPDMA.h

Return Value

Version number:

31:24	23:16	15:8	7:0
00000000	MAJOR_NUM	MINOR_NUM	BUILD_NUM

Example

printf("Driver version:%x\n", DrvPDMA_GetVersion());

2. Revision History

Version	Date	Description
1.00.01	Mar. 2011	Preliminary PDMA Driver User Guide of ISD9160