GPIO Driver
User Guide
V1.00.01

GPIO Driver

1.1. GPIO introduction

Up to 24 General Purpose I/O pins are available on the ISD91xx series. These are shared peripheral special function pins under control of the alternate configuration registers. These 24 pins are arranged in 2 ports named with GPIOA, and GPIOB. GPIOA has sixteen pins and GPIOB has eight. Each one of the 24 pins is independent and has corresponding register bits to control the pin mode function and data.

The I/O type of each GPIO pin can be independently configured as an input, output, open-drain or in a quasi-bidirectional mode. Upon chip reset, all GPIO pins are configured in quasi-bidirectional mode and port data register resets high.

When device is in deep power down (DPD) mode, all GPIO pins become high impedance.
GPIO can generate interrupt signals to the core as either level sensitive or edge sensitive inputs. Edge sensitive inputs can also be de-bounced.

In quasi-bidirectional mode, each GPIO pin has a weak pull-up resistor which is approximately 110KΩ~300KΩ for VDD from 5.0V to 2.5V.

1.2. GPIO Feature

- Each one of the GPIO pins is independent and has the corresponding register bits to control the pin mode function and data.
- The I/O type of each of I/O pins can be independently software con figured as input, output, open-drain or quasi-bidirectional mode.

1.3. Type Definition

DRVGPIO_PORT

Enumeration identifier	Value	Description
GPA	0	Define GPIO Port A
GPB	1	Define GPIO Port B

DRVGPIO_IO

Enumeration identifier	Value	Description
IO_INPIT	0	Set GPIO as Input mode
IO_OUTPUT	1	Set GPIO as Output mode
IO_OPENDRAIN	2	Set GPIO as Open-Drain mode
IO_QUASI	3	Set GPIO as Quasi-bidirectional mode

DRVGPIO_INT_TYPE

Enumeration identifier	Value	Description
IO_RISING	0	Set interrupt enable by Ris ing Edge or Level High
IO_FALLING	1	Set interrupt enable by Falling Edge or Level Low
IO_BOTH_EDGE	2	Set interrupt enable by Both Edges(Rising and Falling)

DRVGPIO_INT_MODE

Enumeration identifier	Value	Description
MODE_EDGE	0	Set interrupt mode is Edge trigger
MODE_LEVEL	1	1 Set interrupt mode is Level trigger

DRVGPIO_DBCLKSRC

Enumeration identifier	Value	Description
DBCLKSRC_HCLK	0	De-bounce counter c lock source is from HCLK.
DBCLKSRC_10K	1	De-bounce counter c lock source is from internal 10 KHz.

DRVGPIO_FUNC

Enumeration identifier	Pins assignment	Description
FUNC_GPIO	All GPIO pins	Set all GPIO pins as GPIO functions
FUNC_PWM01/ FUNC_PWM01B	GPA[12]~]13]/ GPA[4]~[5]	Enable PWM functions
FUNC_I2C0/ FUNC_I2C1/ FUNC_I2C2	GPA[10]~[11]/ GPA[1]~[3]/ GPA[2]~[3]	Enable I2C functions
FUNC_I2S0/ FUNC_I2S1	GPA[4]~[7]/ GPA[8]~[11]	Enable I2S functions
FUNC_SPI0/ FUNC_SPI1	GPA[0]~[3]/ GPB[0], [2]~[4]	Enable SPI functions
FUNC_ACMP0 / FUNC_ACMP1	GPB[0]~[7]/ GPB[6]~[7]	Enable ACMP functions
FUNC_UART0	GPA[8]~[11]	Enable UART functions
FUNC_TMR0/ FUNC_TMR1	GPA[14]/ GPA[15]	Enable Timer functions
FUNC_MCLK0/ FUNC_MCLK1	GPA[0]/ GPB[1]	Enable MCLK functions
FUNC_DMIC0/ FUNC_DMIC1	GPA[14]~[15]/ GPA[14]~[15]	Enable Digital MIC functions
FUNC_SPK	GPA[12]~[13]	Enable Speaker functions

FUNC_NONE	---	User configures GPIO functions, manually.

1.4. Functions

DrvGPIO_Open

Prototype

```
int32_t DrvGPIO_Open (
    DRVGPIO_PORT   port,
    int32_t        i32Bit,
    DRVGPIO_IO     mode
    )
```

Description

To configure the specified GPIO pin to the specified GPIO operation mode.

Parameter

port [in]

Specified GPIO port. It could be GPA, GPB

i32Bit [in]

Specified bit of the IO port. It could be 0~15 in GPA and 0~7 in GPB.

mode [in]

Set the IO to be IO_INPUT , IO_OUTPUT ,IO_OPENDRAIN or IO_QUASI

Include

Driver/ DrvGPIO.h

Return Value

E_SUCCESS: Operation successful

E_DRVGPIO_ARGUMENT: Incorrect arguments

Example

```
/* Configure GPA[0] to GPIO output mode and GPA[1] to GPIO input mode*/
DrvGPIO_Open (GPA, 0, IO_OUTPUT);
DrvGPIO_Open (GPA, 1, IO_INPUT);
```

DrvGPIO_Close

Prototype

int32_t DrvGPIO_Close (DRVGPIO_PORT port, int32_t i32Bit)

Description

To close the opened IO and reset its configurations

Parameter

port [in]

Specified GPIO port. It could be GPA, GPB.

i32Bit [in]

Specified bit of the IO port. It could be 0~15 in GPA and 0~7 in GPB.

Include

Driver/ DrvGPIO.h

Return Value

E_SUCCESS: Operation successful

E_DRVGPIO_ARGUMENT: Wrong arguments

Example

```
/* Close GPA[0] function */
DrvGPIO_Close (GPA, 0);
```

DrvGPIO_SetBit

Prototype

int32_t DrvGPIO_SetBit (DRVGPIO_PORT port, int32_t i32Bit)

Description

Set the specified GPIO pin to 1.

Parameter

port [in]

Specified GPIO port. It could be GPA, GPB

i32Bit [in]

Specified bit of the IO port. It could be 0~15 in GPA and 0~7 in GPB.

Include

Driver/ DrvGPIO.h

Return Value

E_SUCCESS: Operation successful

E_DRVGPIO_ARGUMENT: Incorrect arguments

Example

```
/* Configure GPA[0] as GPIO output mode*/
DrvGPIO_Open (GPA, 0, IO_OUTPUT);
/* Set GPA[0] to 1(high) */
DrvGPIO_SetBit (GPA, 0);
```

DrvGPIO_GetBit

Prototype

int32_t DrvGPIO_GetBit (DRVGPIO_PORT port, int32_t i32Bit)

Description

Get the pin value from the specified input GPIO pin.

Parameter

port [in]

Specified GPIO port. It could be GPA, GPB

i32Bit [in]

Specify pin of the GPIO port. It could be 0~15 in GPA and 0~7 in GPB.

Include

Driver/ DrvGPIO.h

Return Value

The specified input pin value: 0 / 1

E_DRVGPIO_ARGUMENT: incorrect arguments.

Example

```
int32_t i32BitValue;
/* Configure GPA[1] as GPIO input mode*/
DrvGPIO_Open (GPA, 1, IO_INPUT);
i32BitValue = DrvGPIO_ GetBit (GPA, 1);
if      (i32BitValue == 1)
{
     printf("GPA[1] pin status is high.\n¨);
}
else
{
     printf("GPA[1] pin status is low.\n¨);
}
```

DrvGPIO_ClrBit

Prototype

int32_t DrvGPIO_ClrBit (DRVGPIO_PORT port, int32_t i32Bit)

Description

Set the specified GPIO pin to 0.

Parameter

port [in]

Specified GPIO port. It could be GPA, GPB

i32Bit [in]

Specified bit of the IO port. It could be 0~15 in GPA and 0~7 in GPB..

Include

Driver/ DrvGPIO.h

Return Value

SUCCESS: Operation successful

E_DRVGPIO_ARGUMENT: Incorrect arguments

Example

```
/* Configure GPA[0] as GPIO output mode*/
```

```
DrvGPIO_Open (GPA, 0, IO_OUTPUT);
/* Set GPA[0] to 0(low) */
DrvGPIO_ClrBit (GPA, 0);
```

DrvGPIO_SetPortBits

Prototype

int32_t DrvGPIO_SetPortBits (DRVGPIO_PORT port, int32_t i32Data)

Description

Set the output port value to the specified GPIO port.

Parameter

port [in]

Specified GPIO port. It could be GPA, GPB.

i32Data [in]

The data to write to the specified IO port.

Include

Driver/ DrvGPIO.h

Return Value

SUCCESS: Operation successful

E_DRVGPIO_ARGUMENT: Incorrect arguments

Example

```
/* Set the output value of GPA port to 0x1234 */
DrvGPIO_SetPortBits (GPA, 0x1234);
```

DrvGPIO_GetPortBits

Prototype

int32_t DrvGPIO_GetPortBits (DRVGPIO_PORT port)

Description

Get the data of the specified IO port.

Parameter

port [in]

Specified GPIO port. It could be GPA, GPB.

Include

Driver/ DrvGPIO.h

Return Value

The IO pin value of the specified IO port.

E_DRVGPIO_ARGUMENT: Incorrect argument

Example

```
/* Get the GPA port input data value */
int32_t i32PortValue;
i32PortValue = DrvGPIO_GetPortBits (GPA);
```

DrvGPIO_GetDoutBit

Prototype

int32_t DrvGPIO_GetDoutBit (DRVGPIO_PORT port, int32_t i32Bit)

Description

Get the value of the specified IO bit from GPIO Dout register. It's meaning the pin is output data to high. Otherwise, it is output data to low.

Parameter

port [in]

Specified GPIO port. It could be GPA, GPB.

i32Bit [in]

Specify pin of the GPIO port. It could be 0~15 in GPA and 0~7 in GPB.

Include

Driver/ DrvGPIO.h

Return Value

The bit value of the specified register: 0 / 1

E_DRVGPIO_ARGUMENT: Incorrect arguments

Example

```
/* Get the GPA[1] data output value */
int32_t i32BitValue;
i32BitValue = DrvGPIO_GetDoutBit (GPA, 1);
```

DrvGPIO_GetPortDoutBits

Prototype

int32_t DrvGPIO_GetPortDoutBits (DRVGPIO_PORT port)

Description

Get the Dout register value of the specified IO port. If the corresponding bit of the return port value is 1, it means the corresponding bit is output data to high. Otherwise, it is output data to low.

Parameter

port [in]

Specified GPIO port. It could be GPA, GPB.

Include

Driver/ DrvGPIO.h

Return Value

The value of the GPIO DOUT register value.

E_DRVGPIO_ARGUMENT: Incorrect argument.

Example

```
/* Get the GPA port data output value */
int32_t i32PortValue;
i32PortValue = DrvGPIO_GetPortDoutBits (GPA);
```

DrvGPIO_SetBitMask

Prototype

int32_t DrvGPIO_SetBitMask (DRVGPIO_PORT port, int32_t i32Bit)

Description

This function is used to protect the write data function of the corresponding GPIO pin. When set the bit mask, the write signal is masked and write data to the protect bit is ignored.

Parameter

port [in]

Specified GPIO port. It could be GPA, GPB.

i32Bit [in]

Specified bit of the IO port. It could be 0~15 in GPA and 0~7 in GPB.

Include

Driver/ DrvGPIO.h

Return Value

SUCCESS: Operation successful

Example

```
/* Protect GPA[0] write data function */
DrvGPIO_SetBitMask (GPA, 0);
```

DrvGPIO_ClrBitMask

Prototype

int32_t DrvGPIO_ClrBitMask (DRVGPIO_PORT port, int32_t i32Bit)

Description

This function is used to remove the write protect function of the corresponding GPIOpin. After remove the bit mask, write data to the corresponding bit is workable.

Parameter

port [in]

Specified GPIO port. It could be GPA, GPB.

i32Bit [in]

Specified bit of the IO port. It could be 0~15 in GPA and 0~7 in GPB

Include

Driver/ DrvGPIO.h

Return Value

SUCCESS: Operation successful

Example

```
/* Remove the GPA[0] write protect function */
DrvGPIO_ClrBitMask (GPA, 0);
```

DrvGPIO_SetPortMask

Prototype

int32_t DrvGPIO_SetPortMask(DRVGPIO_PORT port, uint32_t u32Mask)

Description

This function is used to protect the write data function of the corresponding GPIO pins. When set the bits are masked, write data to the protect bits are ignored.

Parameter

port [in]

Specified GPIO port. It could be GPA, GPB.

u32Mask [in]

The mask data for the specified IO port.

Include

Driver/ DrvGPIO.h

Return Value

E_SUCCESS: Operation successful

E_DRVGPIO_ARGUMENT: Incorrect arguments

Example

```
/* Protect GPA[0]/[4] write data function */
DrvGPIO_SetPortMask (GPA, 0x11);
```

DrvGPIO_GetPortMask

Prototype

int32_t DrvGPIO_GetPortMask (DRVGPIO_PORT port)

Description

Get the port value from the specified Data Output Write Mask Register. If the corresponding bit of the return port value is 1, it's meaning the bits are protected. And write data to the bits are ignored.

Parameter

port [in]

Specified GPIO port. It could be GPA, GPB.

Include

Driver/ DrvGPIO.h

Return Value

The port value of the specified register.

Example

```
/* Get the port value from GPA Data Output Write Mask Resister */
int32_t i32MaskValue;
i32MaskValue = DrvGPIO_GetPortMask (GPA);
/* If (i32 MaskValue = 0x11), its meaning GPA [0]/ [4] are protected */
```

DrvGPIO_EnableDebounce

Prototype

int32_t DrvGPIO_EnableDebounce (DRVGPIO_PORT port, uint32_t u32Bit)

Description

Enable the de-bounce function of the specified GPIO input pin.

Parameter

port [in]

Specified GPIO port. It could be GPA, GPB.

u32Bit [in]

Specify pin of the GPIO port. It could be 0~15 in GPA and 0~7 in GPB.

Include

Driver/ DrvGPIO.h

Return Value

SUCCESS: Operation successful

E_DRVGPIO_ARGUMENT: Incorrect arguments

Example

```
/* Enable GPA[0] interrupt de-bounce function */
DrvGPIO_EnableDebounce (GPA, 0);
```

DrvGPIO_DisableDebounce

Prototype

int32_t DrvGPIO_DisableDebounce (DRVGPIO_PORT port, uint32_t u32Bit)

Description

Disable the de-bounce function of the specified GPIO input pin.

Parameter

port [in]

Specified GPIO port. It could be GPA, GPB.

u32Bit [in]

Specified bit of the IO port. It could be 0~15 in GPA and 0~7 in GPB.

Include

Driver/ DrvGPIO.h

Return Value

SUCCESS: Operation successful.

E_DRVGPIO_ARGUMENT: Incorrect arguments.

Example

```
/* Disable GPA[0] interrupt de-bounce function */
DrvGPIO_DisableDebounce (GPA, 0);
```

DrvGPIO_SetDebounceTime

Prototype

```
int32_t DrvGPIO_SetDebounceTime (
  uint32_t u32DebounceClk,
  DRVGPIO_DBCLKSRC clockSource
)
```

Description

Set the interrupt de-bounce sampling time based on the de-bounce counter clock source. If the de-bounce clock source is from internal 10 KHz and sampling cycle selection (u32DebounceClk) is 4. The target de-bounce time is (2^4)*(1/(10*1000))s = 16*0.0001 s = 1600 us, and system will sampling interrupt input once per 1600 us.

Parameter

u32DebounceClk [in]

The number of sampling cycle selection, the range of value is from 0 ~ 15. The target de-bounce time is (2^(u32DebounceClk))*(clockSource) second.

clockSource [in]

The debounce clock source can be DBCLKSRC_HCLK or DBCLKSRC_10K.

Include

Driver/ DrvGPIO.h

Return Value

SUCCESS: Operation successful

Example

```
/* Set de-bounce sampling time to 1600 us. (2^4)*(10 KHz) */
DrvGPIO_SetDebounceTime (4, DBCLKSRC_10K);
```

DrvGPIO_GetDebounceTime

Prototype

int32_t DrvGPIO_GetDebounceTime (void)

Description

This function is used to get the number of de-bounce timing setting.

Parameter

None

Include

Driver/ DrvGPIO.h

Return Value

The debounce time setting.

Example

```
int32_t i32DebounceTime;
i32DebounceTime = DrvGPIO_GetDebounceTime ();
/* If i32DebounceTime is 4 and clock source from 10 KHz.      */
/* It's meaning to sample interrupt input once per 16*100us.      */
```

DrvGPIO_EnableInt

Prototype

```
int32_t DrvGPIO_EnableInt (
    DRVGPIO_PORT port,
    uint32_t u32Bit,
    DRVGPIO_INT_TYPE triggerType,
    DRVGPIO_INT_MODE mode
)
```

Description

Enable the interrupt function of the specified GPIO pin. Except for GPB[0] and GPB[1] pins.

Parameter

port [in]

Specified GPIO port. It could be GPA, GPB.

u32Bit [in]

Specify pin of the GPIO port. It could be 0~15 in GPA and 2~7 in GPB. But the GPB[0]/[1] is only used for external interrupt 0/1.

triggerType [in]

Specified trigger type. It could be IO_RISING, IO_FALLING or IO_BOTH_EDGE and it's meaning the interrupt function enable by rising edge/high level, falling edge/low level or both rising edge and falling edge. If the interrupt mode is MODE_LEVEL and interrupt type is BOTH_EDGE, then calling this API is ignored.

mode [in]

DRVGPIO_INT_MODE, specify the interrupt mode. It could be MODE_EDGE or MODE_LEVEL to control the interrupt is by edge trigger or by level trigger. If the interrupt mode is MODE_LEVEL and interrupt type is BOTH_EDGE, then calling this API is ignored.

Include

Driver/ DrvGPIO.h

Return Value

E_SUCCESS: Operation successful

E_DRVGPIO_ARGUMENT: Incorrect arguments

Example

```
/* Enable GPA[3] interrupt function and its rising and edge trigger. */
DrvGPIO_EnableInt (GPA, 3, IO_RISING, MODE_EDGE);
```

DrvGPIO_DisableInt

Prototype

int32_t DrvGPIO_DisableInt (DRVGPIO_PORT port, uint32_t u32Bit)

Description

Disable the interrupt function of the specified GPIO pin. Except for GPB[0] and GPB[1] pins.

Parameter

port [in]

Specified GPIO port. It could be GPA, GPB.

u32Bit [in]

Specify pin of the GPIO port. It could be 0~15 in GPA and 2~7 in GPB. But the GPB[0]/[1] is only used for external interrupt 0/1.

Include

Driver/ DrvGPIO.h

Return Value

E_SUCCESS: Operation successful

E_DRVGPIO_ARGUMENT: Incorrect arguments

Example

```
/* Disable GPA[3] interrupt function. */
DrvGPIO_DisableInt (GPB, 3);
```

DrvGPIO_SetIntCallback

Prototype

```
void DrvGPIO_SetIntCallback (
    GPIO_GPAB_CALLBACK pfGPABCallback,
)
```

Description

Install the interrupt callback function for GPA/GPB port, except GPB [0] and GPB [1] pins.

Parameter

pfGPABCallback [in],

The callback function of GPA and GPB interrupts.

Include

Driver/ DrvGPIO.h

Return Value

None

Example

/* Set GPA/B and GPC/D/E interrupt callback functions */

DrvGPIO_SetIntCallback (GPABCallb ack);

DrvGPIO_EnableEINT0

Prototype

void DrvGPIO_EnableEINT0 (

DRVGPIO_INT_TYPE triggerType,

DRVGPIO_INT_MODE mode,

GPIO_EINT0_CALLBACK pfEINT0Callback

)

Description

Enable the interrupt function for external GPIO interrupt from INT0 (GPB [0]) pin.

Parameter

triggerType [in]

Specified trigger type. It could be IO_RISING, IO_FALLING, IO_BOTH_EDGE and it's meaning the interrupt function enable by rising edge/high level, falling edge/low level or both rising edge and falling edge. If the interrupt mode is MODE_LEVEL and interrupt type is BOTH_EDGE, then calling this API is ignored.

mode [in]

Specified the interrupt mode. It could be MODE_EDGE or MODE_LEVEL to control the interrupt is by edge trigger or by level trigger. If the interrupt mode is MODE_LEVEL and interrupt type is BOTH_EDGE, then calling this API is ignored

pfEINT0Callback [in]

It's the function pointer of the external INT0 callback function.

Include

Driver/ DrvGPIO.h

Return Value

None

Example

/* Enable external INT0 interrupt as falling and both-edge trigger. */

DrvGPIO_EnableEINT0 (IO_BOTH_EDGE, MODE_EDGE, EINT0Callback);

DrvGPIO_DisableEINT0

Prototype

void DrvGPIO_DisableEINT0 (void)

Description

Disable the interrupt function for external GPIO interrupt from INT0 (GPB[0]) pin.

Parameter

None

Include

Driver/ DrvGPIO.h

Return Value

None

Example

/* Disable external INT0 interrupt function. */

DrvGPIO_DisableEINT0 ();

DrvGPIO_EnableEINT1

Prototype

void DrvGPIO_EnableEINT1 (

DRVGPIO_INT_TYPE triggerType,

DRVGPIO_INT_MODE mode,

GPIO_EINT1_CALLBACK pfEINT1Callback

)

Description

Enable the interrupt function for external GPIO interrupt from INT1(GPB[1]) pin.

Parameter

triggerType [in]

Specified the interrupt trigger type. It could be IO_RISING, IO_FALLING or IO_BOTH_EDGE and it's meaning the interrupt function enable by rising edge/high level, falling edge/low level or both rising edge and falling edge. If the interrupt mode is MODE_LEVEL and interrupt type is BOTH_EDGE, then calling this API is ignored.

Mode [in]

Specified the interrupt mode. It could be MODE_EDGE or MODE_LEVEL to control the interrupt is by edge trigger or by level trigger. If the interrupt mode is MODE_LEVEL and interrupt type is BOTH_EDGE, then calling this API is ignored.

pfEINT1Callback [in]

It's the function pointer of the external INT1 callback function.

Include

Driver/ DrvGPIO.h

Return Value

None

Example

```
/* Enable external INT1 interrupt as low level trigger. */
DrvGPIO_EnableEINT1 (IO_FALLING, MODE_LEVEL, EINT1Callback);
```

DrvGPIO_DisableEINT1

Prototype

void DrvGPIO_DisableEINT1 (void)

Description

Disable the interrupt function for external GPIO interrupt from INT1(GPB[1]) pin.

Parameter

None

Include

Driver/ DrvGPIO.h

Return Value

None

Example

```
/* Disable external INT1 interrupt function. */
DrvGPIO_DisableEINT1 ();
```

DrvGPIO_GetIntStatus

Prototype

uint32_t DrvGPIO_GetIntStatus (DRVGPIO_PORT port)

Description

Get the port value from the specified Interrupt Trigger Source Indicator Register. If the corresponding bit of the return port value is 1, it's meaning the interrupt occurred at the corresponding bit. Otherwise, no interrupt occurred at that bit.

Parameter

port [in]

Specified GPIO port. It could be GPA, GPB.

Include

Driver/ DrvGPIO.h

Return Value

The port value of the GPIO interrupt status register.

Example

```
/* Get GPA interrupt status. */
uint32_t   u32INTStatus;
u32INTStatus = DrvGPIO_GetIntStatus (GPA);
```

DrvGPIO_InitFunction

Prototype

int32_t DrvGPIO_InitFunction (DRVGPIO_FUNC function)

Description

Initialize the specified function and configure the relative pins for specified function used.

Parameter

function [in]

DRVGPIO_FUNC, specified the relative GPIO pins as special function pins.

It could be:

FUNC_GPIO,
FUNC_PWM01/ FUNC_PWM01B,
FUNC_I2C0/FUNC_I2C1,
FUNC_I2C2,
FUNC_I2S0/ FUNC_I2S1,
FUNC_SPI0/ FUNC_SPI1,
FUNC_ACMP0/ FUNC_ACMP1,
FUNC_UART0,
FUNC_TMR0/ FUNC_TMR1,
FUNC_MCLK0/ FUNC_MCLK1,
FUNC_DMIC0/ FUNC_DMIC1,
FUNC_SPK,
FUNC_NONE

Include

Driver/ DrvGPIO.h

Return Value

E_SUCCESS: Operation successful

E_DRVGPIO_ARGUMENT: Incorrect arguments

Example

```
/* Init UART0 function */
DrvGPIO_InitFunction (FUNC_UART0);
```

DrvGPIO_GetVersion

Prototype

int32_t DrvGPIO_GetVersion (void)

Description

This function is used to return the version number of GPIO driver.

Include

Driver/ DrvGPIO.h

Return Value

The version number of GPIO driver:

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

Example

```
/* Get the current version of GPIO Driver */
int32_t i32GPIOVer;
i32GPIOVer = DrvGPIO_GetVersion ();
```

EINT0_IRQHandler

Prototype

void EINT0_IRQHandler(void)

Description

Install ISR to handle interrupt event.

Parameter

None

Include

Driver/ DrvGPIO.h

Return Value

None.

EINT1_IRQHandler

Prototype

void EINT1_IRQHandler(void)

Description

Install ISR to handle interrupt event.

Parameter

None

Include

Driver/ DrvGPIO.h

Return Value

None.

GPAB_IRQHandler

Prototype

void GPAB_IRQHandler(void)

Description

Install ISR to handle interrupt event.

Parameter

None

Include

Driver/ DrvGPIO.h

Return Value

None.

2. Revision History

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