I2C Driver
User Guide
V1.00.01

I2C Driver

1.1. I2C Introduction

I2C is a two-wire, bi-directional serial bus that provides a simple and efficient method of data exchange between devices. The I2C standard is a true multi-master bus including collision detection and arbitration that prevents data corruption if two or more masters attempt to control the bus simultaneously. Serial, 8-bit oriented, bi-directional data transfers can be made up 1.0 Mbps.

Data is transferred between a Master and a Slave synchronously to SCL on the SDA line on a byte-by-byte basis. Each data byte is 8 bits long. There is one SCL clock pulse for each data bit with the MSB being transmitted first. An acknowledge bit follows each transferred byte. Each bit is sampled during the high period of SCL; therefore, the SDA line may be changed only during the low period of SCL and must be held stable during the high period of SCL. A transition on the SDA line while SCL is high is interpreted as a command (START or STOP).

1.2. I2C Feature

The I2C bus uses two wires (SDA and SCL) to transfer information between devices connected to the bus. The main features of the bus are:

- Master/Slave up to 1Mbit/s
- Bidirectional data transfer between masters and slaves
- Multi-master bus (no central master)
- Arbitration between simultaneously transmitting masters without corruption of serial data on the bus
- Serial clock synchronization allows devices with different bit rates to communicate via one serial bus
- Serial clock synchronization can be used as a handshake mechanism to suspend and resume serial transfer
- Built-in a 14-bit time-out counter will request the I2C interrupt if the I2C bus hangs up and timer-out counter overflows.
- External pull-up are needed for high output
- Programmable clocks allow versatile rate control
- Supports 7-bit addressing mode
- I2C-bus controllers support multiple address recognition (Four slave address with mask option)

1.3. Type Definition

E_I2C_PORT

Enumeration identifier	Value	Description
I2C_PORT0	0	I2C port 0
I2C_PORT1	1	I2C port 1

E_I2C_CALLBACK_TYPE

Enumeration identifier	Value	Description
I2CFUNC	0	For I2C Normal condition
ARBITLOSS	1	For Arbitration Loss condition when I2C operates as master mode
BUSERROR	2	For I2C Bus Error condition
TIMEOUT	3	For I2C 14-bit time-out counter time out

1.4. Functions

DrvI2C_Open

Prototype

int32_t DrvI2C_Open (E_I2C_PORT port, uint32_t u32clock_Hz, uint32_t u32baudrate_Hz);

Description

To enable I2C function and set clock divider. I2C clock = PCLK / (4x (divider+1)). The maximum of I2C baud rate is 1MHz.

Parameter

port [in]

Specify I2C interface. (I2C_PORT0)

u32clock_Hz [in]

I2C clock source frequency. The unit is Hz.

u32baudrate_Hz [in]

I2C bit rate. The unit is Hz.

Include

Driver/DrvI2C.h

Return Value

0 Succeed

Example

```
/* Enable I2C0 and set I2C0 bus clock 100 KHz */
DrvI2C_Open(I2C_PORT0, DrvSYS_GetHCLK() * 1000, 100000);
```

DrvI2C_Close

Prototype

int32_t DrvI2C_Close (E_I2C_PORT port);

Description

To disable I2C function and clock source.

Parameter

port [in]

Specify I2C interface. (I2C_PORT0)

Include

Driver/DrvI2C.h

Return Value

0 Succeed

Example

```
/* Disable I2C0 */
DrvI2C_Close (I2C_PORT0);
```

DrvI2C_SetClock

Prototype

```
int32_t DrvI2C_SetClock(E_I2C_PORT port, uint32_t u32clock_Hz, uint32_t u32baudrate_Hz);
```

Description

To Set clock divider. Divider = (uint32_t) (((clock_Hz*10)/(baudrate_Hz * 4) + 5) / 10 - 1);.

The maximum of I2C baud rate is 1MHz.

Parameter

port [in]

Specify I2C interface. (I2C_PORT0)

u32clock_Hz [in]

I2C clock source frequency. The unit is Hz.

u32baudrate_Hz [in]

I2C bit rate. The unit is Hz.

Include

Driver/DrvI2C.h

Return Value

0 Succeed

Example

```
/* Enable I2C0 and set I2C0 bus clock 100 KHz */
DrvI2C_Open(I2C_PORT0, DrvSYS_GetHCLK() * 1000, 100000);
```

DrvI2C_GetClock

Prototype

```
uint32_t DrvI2C_GetClock(E_I2C_PORT port, uint32_t clock_kHz);
```

Description

To get the I2C bit rate. I2C bit rate = I2C source clock / (4 x (I2CCLK_DIV+1))

Parameter

port [in]

Specify I2C interface. (I2C_PORT0)

clock_kHz [in]

I2C clock source frequency (KHz)

Include

Driver/DrvI2C.h

Return Value

I2C bit rate

Example

```
/* Get I2C0 bus clock */
uint32_t      u32 clock;
u32clock = DrvI2C_GetClock (I2C_PORT0);
```

DrvI2C_SetAddress

Prototype

int32_t DrvI2C_SetAddress (E_I2C_PORT port, uint8_t u8slaveNo, uint8_t u8slave_addr, uint8_t u8GC_Flag);

Description

To set 7-bit physical slave address to the specified I2C slave address and enable/disable General Call function. Four slave addresses supported. The setting takes effect when I2 C operates as slave mode.

Parameter

port [in]

Specify I2C interface. (I2C_PORT0)

u8slaveNo [in]

To select slave address. The u8slaveNo is 0 ~ 3.

u8slave_addr [in]

To set 7-bit physical slave address for selected slave address.

u8GC_Flag [in]

To enable or disable general call function. (1: enable, 0: disable)

Include

Driver/DrvI2C.h

Return Value

0: Succeed

others: Failed

Example

```
DrvI2C_SetAddress(I2C_PORT0, 0, 0x15, 0);  /* Set I2C0 1st slave address 0x15 */
DrvI2C_SetAddress(I2C_PORT0, 1, 0x35, 0);  /* Set I2C0 2nd slave address 0x35 */
DrvI2C_SetAddress(I2C_PORT0, 2, 0x55, 0);  /* Set I2C0 3rd slave address 0 x55 */
DrvI2C_SetAddress(I2C_PORT0, 3, 0x75, 0);  /* Set I2C0 4th slave address 0x75 */
```

DrvI2C_SetAddressMask

Prototype

int32_t DrvI2C_SetAddressMask (E_I2C_PORT port, uint8_t u8slaveNo, uint8_t u8slaveAddrMask);

Description

To set 7-bit physical slave address mask to the specified I2C salve address mask. Four slave address masks supported. The setting takes effect when I2C operates as slave mode.

Parameter

port [in]

Specify I2C interface. (I2C_PORT0)

u8slaveNo [in]

To select slave address mask. The value is 0 ~ 3.

u8slaveAddrMask [in]

To set 7-bit physical slave address mask for selected slave address mask. The corresponding address bit is Don't care¨.

Include

Driver/DrvI2C.h

Return Value

0: Succeed

others: Failed

Example

```
DrvI2C_SetAddress (I2C_PORT0, 0, 0x15, 0);          /* Set I2C0 1st slave address 0x15 */
DrvI2C_SetAddress (I2C_PORT0, 1, 0x35, 0);          /* Set I2C0 2nd slave address 0x35 */
/* Set I2C0  1st slave address mask 0x01, slave address 0x15 and 0x14 would be addressed */
DrvI2C_SetAddressMask (I2C_PORT0, 0, 0x01);
```

```
/* Set I2C0   2nd slave address mask 0x04, slave address 0x35 and 0x31 would
be addressed
*/
DrvI2C_SetAddressMask (I2C_PORT0, 1, 0x04);
```

DrvI2C_GetStatus

Prototype

uint32_t DrvI2C_GetStatus (E_I2C_PORT port);

Description

To get the I2C status code. There are 26 status codes.

I2C status code	Description
0x00	Bus error. When a START or STOP condition is present at an illegal position in the formation frame.
0x08	A START has been transmitted.
0x10	A repeated START has been transmitted.
0x18	SLA+W will be transmitted; ACK bit will be received.
0x20	SLA+W will be transmitted; NACK bit will be received.
0x28	Data byte in S1DAT has been transmitted; ACK has been received.
0x30	Data byte in S1DAT has been transmitted; NACK has been received.
0x38	Arbitration lost in SLA+R/W or Data byte.
0x40	SLA+R has been transmitted; ACK has been received.
0x48	SLA+R has been transmitted; NACK has been received.
0x50	Data byte has been received; ACK has been returned.
0x58	Data byte has been received; NACK has been returned.
0x60	Own SLA+W has been received; ACK has been returned.
0x68	Arbitration lost SLA+R/W as master; Own SLA+W has been received; ACK has been returned.
0x70	Reception of the general call address and one or more data bytes; ACK has returned.
0x78	Arbitration lost SLA+R/W as master; and address as SLA by general call; ACK has been returned.
0x80	Previously addressed with own SLA address; Data has been received; ACK has been returned.
0x88	Previously addressed with own SLA address; NACK has been

	returned.
0x90	Previously addressed with general call; data has been received; ACK has been returned.
0x98	Previously addressed with general call; data has been received; NACK has been returned.
0xA0	A STOP or repeated START has been received while still addressed as SLV/REC.
0xA8	Own SLA+R has been received; ACK has been returned.
0xB0	Arbitration lost SLA+R/W as master; Own SLA+R has been received; ACK has been returned.
0xB8	Data byte in S1DAT has been transmitted; ACK has been received.
0xC0	Data byte or Last data byte in S1DAT has been transmitted; NACK has been received.
0xC8	Last data byte in S1DAT has been transmitted; ACK has been received.
0xF8	No serial interrupt is requested.

Parameter

port [in]

Specify I2C interface. (I2C_PORT0)

Include

Driver/DrvI2C.h

Return Value

I2C status code

Example

```
uint32_t      u3 2status;
u32status = DrvI2C_GetStatus (I2C_PORT0);        /* Get I2C0 current status
code */
```

DrvI2C_WriteData

Prototype

```
void DrvI2C_WriteData(E_I2C_PORT port, uint8_t u8data);
```

Description

To set a byte of data to be sent.

Parameter

port [in]

Specify I2C interface. (I2C_PORT0)

u8data [in]

Byte data.

Include

Driver/DrvI2C.h

Return Value

None

Example

DrvI2C_WriteData (I2C_PORT0, 0x55); /* Set byte data 0x55 into I2C0 data register */

DrvI2C_ReadData

Prototype

uint8_t DrvI2C_ReadData(E_I2C_PORT port);

Description

To read the last data from I2C data register.

Parameter

port [in]

Specify I2C interface. (I2C_ PORT0)

Include

Driver/DrvI2C.h

Return Value

Data from I2C Data register.

Example

```
uint8_t      u8data;
u8data = DrvI2C_ReadData (I2C_PORT0); /* Read out byte data fro m I2C0 data register */
```

DrvI2C_Ctrl

Prototype

void DrvI2C_Ctrl(E_I2C_PORT port, uint8_t u8start, uint8_t u8stop, uint8_t u8intFlag, uint8_t u8ack);

Description

To set I2C control bit include STA, STO, AA, SI in control register.

Parameter

port [in]

Specify I2C interface. (I2C_PORT0)

u8start [in]

To set STA bit or not. (1: set, 0: don't set). If the STA bit is set, a START or repeat

START signal will be generated when I2C bus is free.

u8stop [in]

To set STO bit or not. (1: set, 0: don't set). If the STO bit is set, a STOP signal will be generated. When a STOP condition is detected, this bit will be cleared by hardware automatically.

u8intFlag [in]

To clear SI flag (I2C interrupt flag). (1 : clear, 0: don't work)

u8ack [in]

To enable AA bit (Assert Acknowledge control bit) or not. (1: enable, 0: disable)

Include

Driver/DrvI2C.h

Return Value

None

Example

DrvI2C_Ctrl (I2C_PORT0, 0, 0, 1, 0); /* Set I2C0 SI bit to clear SI flag */

DrvI2C_Ctrl (I2C_PORT0, 1, 0, 0, 0); /* Set I2C0 STA bit to send START signal */

DrvI2C_GetIntFlag

Prototype

uint8_t DrvI2C_GetIntFlag(E_I2C_PORT port);

Description

To get I2C interrupt flag status.

Parameter

port [in]

Specify I2C interface. (I2C_PORT0)

Include

Driver/DrvI2C.h

Return Value

Interrupt status (1 or 0)

Example

uint8_t u8flagStatus;

u8flagStatus = DrvI2C_GetIntFlag (I2C_PORT0); /* Get the status of I2C0 interrupt flag */

DrvI2C_ClearIntFlag

Prototype

void DrvI2C_ClearIntFlag (E_I2C_PORT port);

Description

To clear I2C interrupt flag if the flag is set 1.

Parameter

port [in]

Specify I2C interface. (I2C_PORT0)

Include

Driver/DrvI2C.h

Return Value

None

Example

DrvI2C_ClearIntFlag (I2C_PORT0); /* Clear I2C0 interrupt flag (SI) */

DrvI2C_EnableInt

Prototype

int32_t DrvI2C_EnableInt (E_I2C_PORT port);

Description

To enable I2C interrupt function.

Parameter

port [in]

Specify I2C interface. (I2C_PORT0)

Include

Driver/DrvI2C.h

Return Value

0 Succeed

Example

DrvI2C_EnableInt (I2C_PORT0); /* Enable I2C0 interrupt */

DrvI2C_DisableInt

Prototype

int32_t DrvI2C_DisableInt (E_I2C_PORT port);

Description

To disable I2C interrupt function.

Parameter

port [in]

Specify I2C interface. (I2C_PORT0)

Include

Driver/DrvI2C.h

Return Value

0 Succeed

Example

DrvI2C_DisableInt (I2C_PORT0); /* Disable I2C0 interrupt */

DrvI2C_InstallCallBack

Prototype

int32_t DrvI2C_InstallCallBack (E_I2C_PORT port,
E_I2C_CALLBACK_TYPE Type,
I2C_CALLBACK callbackfn);

Description

To install I2C call back function in I2C interrupt handler.

Parameter

port [in]

Specify I2C interface. (I2C_PORT0)

Type [in]

There are four types for call back function. (I2CFUNC / ARBITLOSS / BUSERROR /
TIMEOUT)

I2CFUNC: For normal I2C condition

ARBITLOSS: For master mode when arbitration loss occurs. The status code is 0x38.

BUSERROR: For bus error condition. The status code is 0x00.

TIMEOUT: For 14 -bit time-out counter overflow.

callbackfn [in]

Call back function name for specified interrupt event.

Include

Driver/DrvI2C.h

Return Value

0: Succeed

others: Failed

Example

```
/* Install I2C0 call back function .I2C0_Callback_Normal. for I2C normal condition */
DrvI2C_InstallCallback (I2C_PORT0, I2CFUNC, I2C0_Callback_Normal);
/* Install I2C0 call back function .I2C0_Callback_ BusErr. for Bus Error condition */
DrvI2C_InstallCallback (I2C_PORT0, BUSERROR, I2 C0_Callback_BusErr);
```

DrvI2C_UninstallCallBack

Prototype

```
int32_t DrvI2C_UninstallCallBack (E_I2C_PORT port,
E_I2C_CALLBACK_TYPE
Type);
```

Description

To uninstall I2C call back function in I2C interrupt handler.

Parameter

port [in]

Specify I2C interface. (I2C_PORT0)

Type [in]

There are four types for call back function. (I2CFUNC / ARBITLOSS / BUSERROR /

TIMEOUT)

I2CFUNC: For normal I2C condition

ARBITLOSS: For master mode when arbitration loss occurs. The status code is 0x38.

BUSERROR: For bus error condition. The status code is 0x00.

TIMEOUT: For 14 -bit time-out counter overflow.

Include

Driver/DrvI2C.h

Return Value

0: Succeed

others: Failed

Example

```
/* Uninstall I2C0 call back function for I2C normal condition */
DrvI2C_UninstallCallBack (I2C_PORT0, I2CFUNC);
/* Uninstall I2C0 call back function for Bus Error condition */
DrvI2C_UninstallCallBack (I2C_PORT0, BUSERROR);
```

DrvI2C_ClearTimeoutFlag

Prototype

```
void      DrvI2C_ClearTimeoutFlag (E_I2C_PORT port);
```

Description

To clear I2C TIF flag if the flag is set 1.

Parameter

port [in]

Specify I2C interface. (I2C_PORT0)

Include

Driver/DrvI2C.h

Return Value

None

Example

DrvI2C_ClearTimeoutFlag (I2C_PORT0); /* Clear I2C0 TIF flag */

DrvI2C_GetVersion

Prototype

uint32_t DrvI2C_GetVersion (void);

Description

Get this module's version.

Parameter

None

Include

Driver/DrvI2C.h

Return Value

Version number:

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

I2C0_IRQHandler

Prototype

void I2C0_IRQHandler(void)

Description

Install ISR to handle interrupt event.

Parameter

None

Include

Driver/ DrvI2C.h

Return Value

None.

2. Revision History

Version	Date	Description
1.00.01	Mar. 2011	Preliminary I2C Driver User Guide of ISD9160