

8bit & 32bit

ABOV I2C API

User Guide

Application Note

Version 1.00

Contents

1	Overview	3
2	Function block diagram	4
2.1	I2C Master Mode.....	4
2.2	I2C Slave Mode.....	5
3	How to use in main application	6
3.1	Public Macro	6
3.2	I2C Initialization.....	6
3.3	I2C Master – Write / Read	7
3.4	I2C Slave – Set Transmit Buffer.....	7
3.5	Example of use in main application	8
4	How to port to other device	9
4.1	Main (main.c)	9
4.2	Header file (ABOV_USI_I2C.h).....	9
4.3	Source file (ABOV_USI_I2C.c)	10

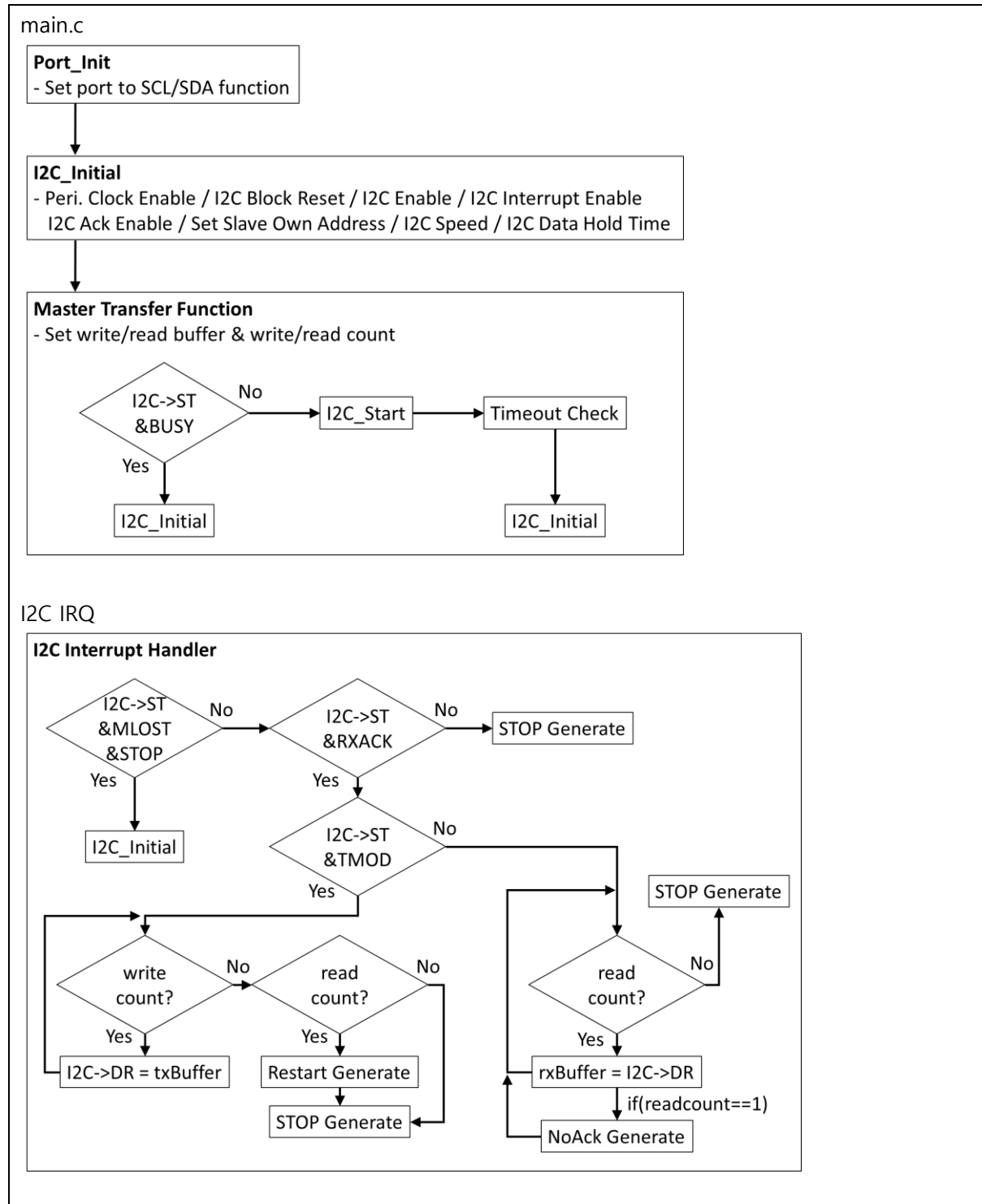
1 Overview

This document describes the firmware for the ABOV_I2C_Driver.

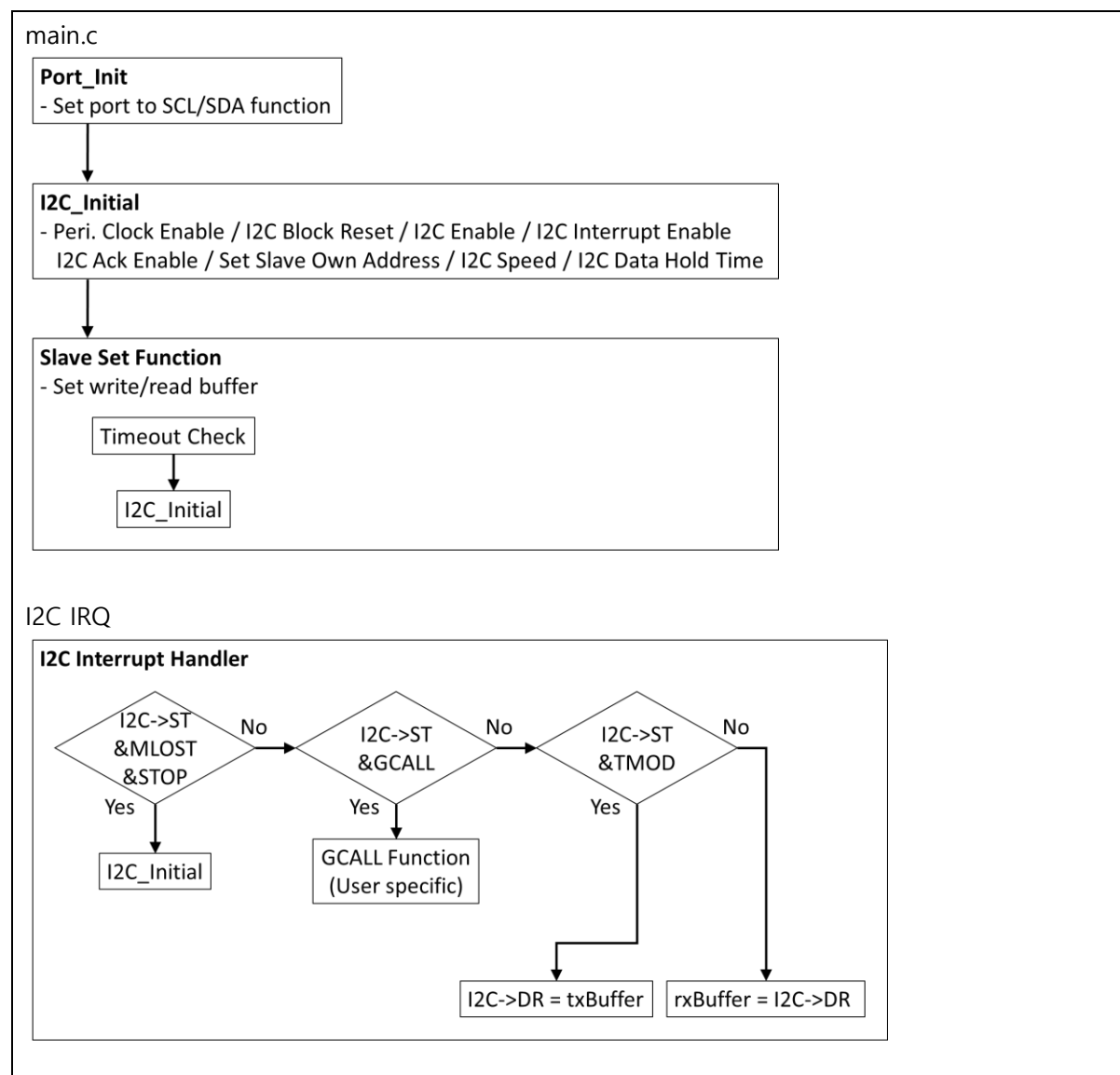
It has the goal of helping the developer to understand how this firmware works, and how to tailor the application.

2 Function block diagram

2.1 I2C Master Mode



2.2 I2C Slave Mode



3 How to use in main application

3.1 Public Macro

The macros on the left side are user specific.

Modification is required for the user application.

#define I2C_DEVICE_ADDRESS	0xA0	#define I2C_ACK_DISABLE	0
#define I2C_SLAVE_OWN_ADDR	0xA0	#define I2C_ACK_ENABLE	1
#define I2C_SPEED	10000	#define I2C_WRITE_MODE	0
#define I2C_MAX_BUFFER_SIZE	20	#define I2C_READ_MODE	1
#define I2C_MAX_CHANNEL	2	#define I2C_IDLE	0
#define I2C_CH0	0	#define I2C_BUSY	1
#define I2C_CH1	1	#define I2C_FALSE	0
		#define I2C_TRUE	1

3.2 I2C Initialization

'USI_I2C_Init()' function is required to use I2C functionality.

This function includes the following.

- Peri. Clock Enable / I2C Block Reset / I2C Interrupt Enable
- I2C Enable / I2C Ack Enable / Set Slave Own Address / Set I2C Speed / Set I2C Data Hold Time

```
void USI_I2C_Init(uint8_t ch, uint32_t speed, uint8_t addr, uint8_t ack)
```

↑ ↑ ↑ ↑
 I2C channel number I2C speed I2C slave own address I2C ack enable/disable

[example of use]

```
USI_I2C_Init(I2C_CH0, I2C_SPEED, I2C_SLAVE_OWN_ADDR, I2C_ACK_ENABLE);
```

3.3 I2C Master – Write / Read

Using 'USI_I2C_MasterTransferData()' function, you can use both I2C Write and I2C Read functions.

This function includes the following parameter.

- I2C channel number / slave device address
- write data buffer / write data length / read data buffer / read data length

```
void USI_I2C_MasterTransferData(uint8_t ch, uint8_t dev_addr, uint8_t *write_data, uint8_t write_len, uint8_t *read_data, uint8_t read_len)
```

[example of use]

(case1) I2C Write 4-byte
→ USI_I2C_MasterTransferData(I2C_CH0, I2C_DEVICE_ADDRESS, *u8TxDat, 4, *u8RxDat, 0);

(case2) I2C Read 4-byte
→ USI_I2C_MasterTransferData(I2C_CH0, I2C_DEVICE_ADDRESS, *u8TxDat, 0, *u8RxDat, 4);

(case3) I2C Write 1-byte , Read 3-byte
→ USI_I2C_MasterTransferData(I2C_CH0, I2C_DEVICE_ADDRESS, *u8TxDat, 1, *u8RxDat, 3);

(case4) I2C Write 3-byte , Read 1-byte
→ USI_I2C_MasterTransferData(I2C_CH0, I2C_DEVICE_ADDRESS, *u8TxDat, 3, *u8RxDat, 1);

3.4 I2C Slave – Set Transmit Buffer

Using 'USI_I2C_SetSlaveData()' function, you can set transmit buffer data.

The buffer data can be modified by the user.

```
void USI_I2C_SetSlaveData(uint8_t ch)
{
    .....
    receiveNum[ch] = 0;
    transmitNum[ch] = 0;

    .....
    for(i=0; i<I2C_MAX_BUFFER_SIZE; i++)
    {
        transmitBuffer0[i] = (0x00 + i);
        receiveBuffer0[i] = 0x00;
    }
    .....
}
```

[example of use]

```
USI_I2C_SetSlaveData(I2C_CH0);
```

3.5 Example of use in main application

This is example of use in main application.

```
int main( void )
{
    unsigned char i;
    unsigned char u8TxDat[8];
    unsigned char u8RxDat[8];

    Init_Port();
    Init_Clock();

    #if 1    /* Master mode */
    USI_I2C_Init(I2C_CH0, I2C_SPEED, I2C_SLAVE_OWN_ADDR, I2C_ACK_ENABLE);

    for(i=0;i<8;i++)
    {
        u8TxDat[i] = (i + 0x00);
        u8RxDat[i] = 0x00;
    }

    while(1)
    {
        USI_I2C_MasterTransferData(I2C_CH0, I2C_DEVICE_ADDRESS, u8TxDat, 8, u8RxDat, 0);
    }

    #else    /* Slave mode */
    USI_I2C_Init(I2C_CH1, I2C_SPEED, I2C_SLAVE_OWN_ADDR, I2C_ACK_ENABLE);

    while(1)
    {
        USI_I2C_SetSlaveData(I2C_CH1);
    }
    #endif
    return 0;
}
```


4 How to port to other device

4.1 Main (main.c)

1) Include header file → #include "ABOV_USI_I2C.h"

2) Call 'USI_I2C_IRQHandler()' function at the I2C IRQ.

→ Example) void I2C0_Handler(void){ USI_I2C_IRQHandler(I2C_CH0); }
void I2C1_Handler(void){ USI_I2C_IRQHandler(I2C_CH1); }

3) Use functions as below in user application.

→ Example) USI_I2C_Init(I2C_CH0, I2C_SPEED, I2C_SLAVE_OWN_ADDR, I2C_ACK_ENABLE);
USI_I2C_MasterTransferData(I2C_CH0, I2C_DEVICE_ADDRESS, *u8TxDat, 4, *u8RxDat, 0);
USI_I2C_SetSlaveData(I2C_CH1);

4.2 Header file (ABOV_USI_I2C.h)

It needs to change public typedef as below.

I2C Control Register and I2C Status Register may differ from product to product,

So the corresponding bits must be checked.

```
enum i2c_control_flags{ // I2C Control Register Flags
    fI2CnEN      = (1<<7),
    fTXDLVENBn   = (1<<6),
    fI2CnIEN     = (1<<5),
    fI2CnIFLAG   = (1<<4),
    fACKnEN      = (1<<3),
    fIMASTERn    = (1<<2),
    fSTOPCn      = (1<<1),
    fSTARTCn     = (1<<0),
};
enum i2c_status_flags{ // I2C Status Register Flags
    fGCALL      = (1<<7),
    fTEND       = (1<<6),
    fSTOPD      = (1<<5),
    fSSEL       = (1<<4),
    fMLOST      = (1<<3),
    fBUSY        = (1<<2),
    fTMODE      = (1<<1),
    fRXACK      = (1<<0),
};
```

4.3 Source file (ABOV_USI_I2C.c)

1) Exchange device header file → #include "user_device_name.h"

2) Exchange name of register as below

- SCUCG relevant registers and their sub bits
- I2C relevant registers and their sub bits
- Interrupt relevant registers and their sub bits

ABOV Disclaimer

IMPORTANT NOTICE – PLEASE READ CAREFULLY

ABOV Semiconductor ("ABOV") reserves the right to make changes, corrections, enhancements, modifications, and improvements to ABOV products and/or to this document at any time without notice. ABOV does not give warranties as to the accuracy or completeness of the information included herein. Purchasers should obtain the latest relevant information of ABOV products before placing orders. Purchasers are entirely responsible for the choice, selection, and use of ABOV products and ABOV assumes no liability for application assistance or the design of purchasers' products. No license, express or implied, to any intellectual property rights is granted by ABOV herein. ABOV disclaims all express and implied warranties and shall not be responsible or liable for any injuries or damages related to use of ABOV products in such unauthorized applications. ABOV and the ABOV logo are trademarks of ABOV. All other product or service names are the property of their respective owners. Information in this document supersedes and replaces the information previously supplied in any former versions of this document.

© 2020 ABOV Semiconductor – All rights reserved