32-bit microcontrollers

HC 32 F 460 Series IC Bus I C²

Applicable objects

F Series	HC 32F460

Table of Contents   Table of Contents

1   Abstract

This application note introduces the Inter-Integrated Curcuit (I 2C) module of HC32F460 series chips, and briefly explains how to use the I2C module by showing sample code for reading and writing to E2PROM.

2   I2C Introduction

I2C protocol is a two-wire serial communication protocol proposed by Philips, which supports half-duplex data communication and has been widely used in electronic circuit products. Common devices that use I2C communication protocol are LCD display drivers, memory E2PROM, audio decoder CODEC, etc.

The SMBus bus is optimized based on the I2C protocol and provides a control bus for system and power control related management tasks. Because it is used in control systems, SMBus is more focused on real-time response during communication than the I2C protocol.

3   I2C of HC32F460 Series

The integrated circuit bus (I2C) modules of the HC32F460 series support the I2C standard protocol and can also be configured to support the SMBus Ver2.0 bus protocol. This series of chips has 3 independent I2C peripheral modules, I2C1~I2C3.

The basic functions of the I2C module are as follows:

- Support multi-master and multi-slave

- Supports standard mode up to 100Kbps and fast mode up to 400Kbps.

- 2 slave addresses can be set, 7-bit address format and 10-bit address format can be set, and broadcast call addresses can be detected.

- I2C protocol handshaking function, arbitration function, SCL clock synchronization function.

- Input signal digital filtering and analog filtering function

- 2 types of reset

- 4 types of interrupts and event outputs

- DMA communication is possible

The following functions are supported for the SMBus bus mode:

- Data transfer speed support 10Kbps~100Kbps

- SCL level timeout measurement

- Fast ACK/NACK function

3.1 System Block Diagram

The I2C module is mounted on the chip peripheral bus APB4 and uses the peripheral clock PCLK3 to divide the frequency to generate the baud rate clock. The system block diagram of the single c h a n n e l I2C module is shown in F i g u r e 1.

The module provides output control and input filtering functions for data port SDA and clock port SCL; when data is output, output register I2CDTR transmits data through shift register I2CDSR; when data is input, input register I2CDRR gets input data from shift register I2CDSR; the module also contains ACK control unit, arbitration unit, and baud rate control Single

The I2C communication function is implemented by the timeout control unit, the interrupt control unit and the transmit-receive control unit.
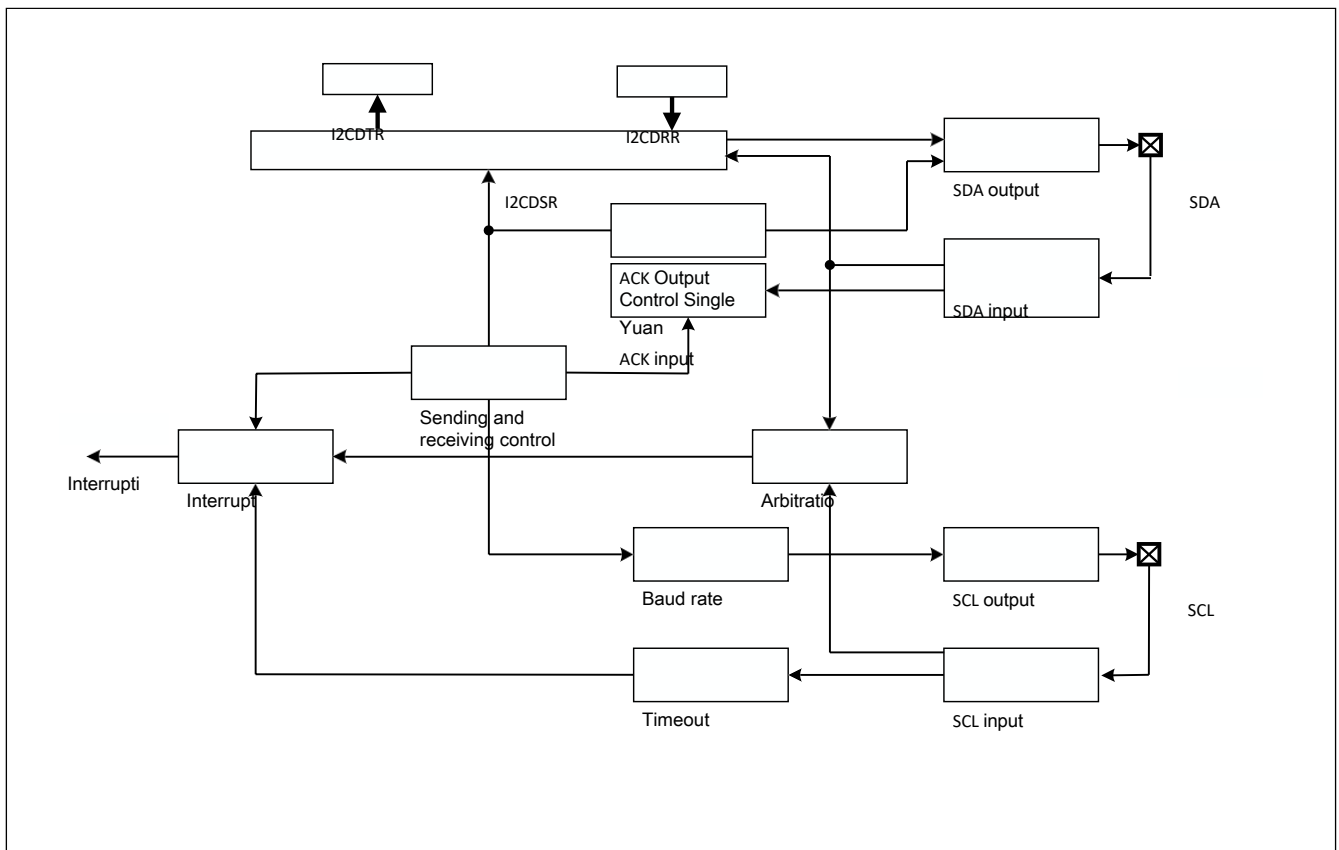

Figure 1 I2C system block diagram

3.2 Hardware Connection

This series of I2C modules supports multi-master and multi-slave bus protocols, and the recommended bus structure is shown in F i g u r e 2. The pull-up resistor resistance value of the bus is determined by the system design.
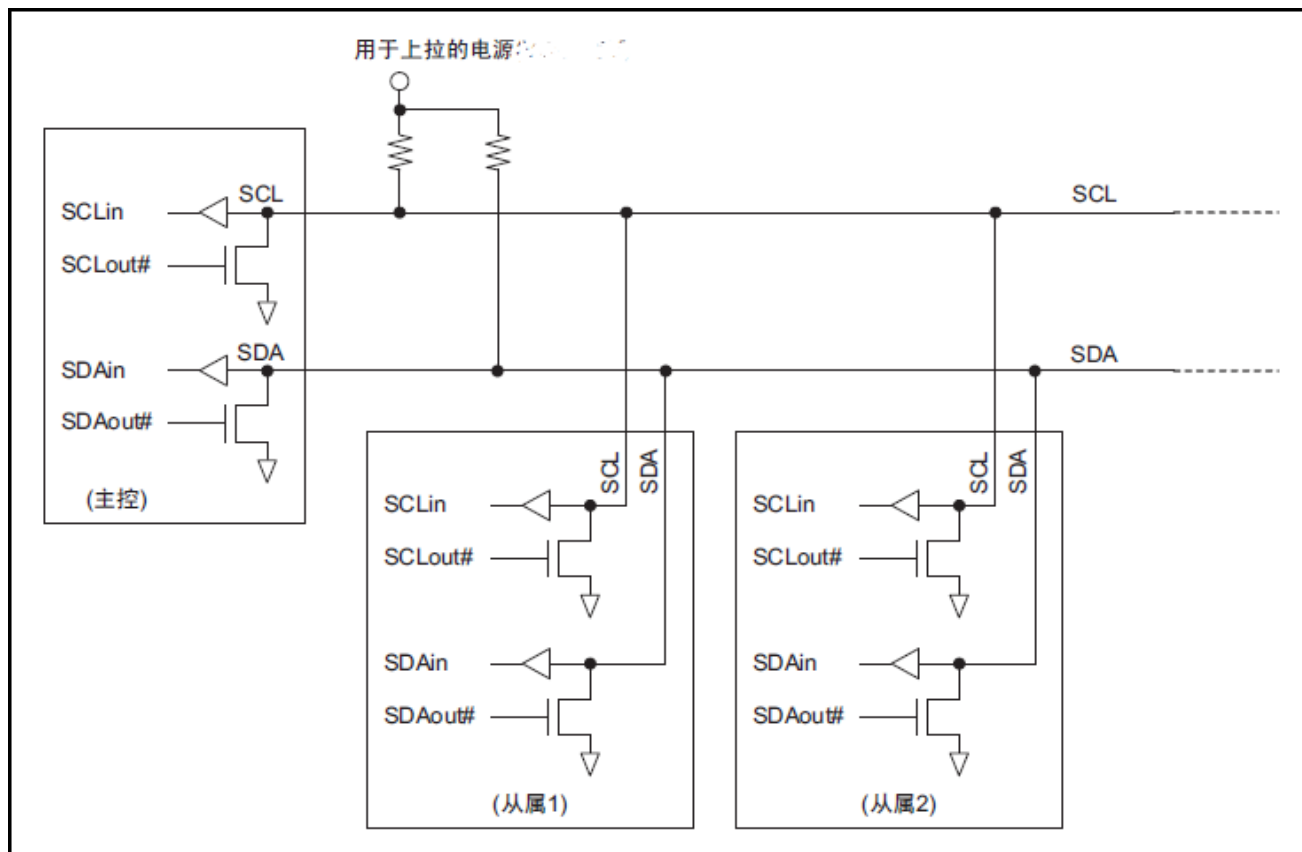

Figure 2 I2C bus structure diagram

The chip I2C function pin resource contains 3 channels of SDA and SCL ports, and the I2C port of this series chip can be assigned to any port in the function group. Please refer to the pin configuration and function chapter in the datasheet of this series of chips for the principle and method of port configuration, or refer to the sample and driver library provided by the manufacturer for configuration.

I2C Channels	Function Port	Direction
I2C1	I2C1_SDA I2C1_SCL	I/O
I2C2	I2C2_SDA I2C2_SCL	I/O

I2C3	I2C3_SDA	I/O
	I2C3_SCL	

3.3 I2C Protocol

For I2C bus protocols, please refer to the detailed description in Chapter 27.3 of this series chip user manual, including I2C protocol start conditions, address transfer, data transfer, stop conditions, restart conditions, SCL clock synchronization, arbitration, handshaking, etc.

3.4 I2C module configurable functions

The following 3.4.1 to 3.4.4 are the basic configuration functions of the I2C module:

3.4.1 Filtering function configuration

The receive data filtering function is divided into digital filtering function and analog filtering function.

The filtering function of the I2C module is disabled by default, when you need to use it, please pass the register I2C_FLTR before the I2C function is enabled.

Perform the configuration.

3.4.2 Baud rate configuration

Depending on the digital filtering function and the I2C baud rate divider clock setting, the baud rate calculation method for each case is shown below:

DNFE	FREQ	Baudrate
0	000	$\dfrac{1}{\dfrac{(SHIGHW + 3) + (SLOWW + 3)}{\Phi I2C} + SCL\ 上升时间 + SCL\ 下降时间}$
1	000	$\dfrac{1}{\dfrac{(SHIGHW + 3 + 滤波能力) + (SLOWW + 3 + 滤波能力)}{\Phi I2C} + SCL\ 上升时间 + SCL\ 下降时间}$
0	Non 000	$\dfrac{1}{\dfrac{(SHIGHW + 2) + (SLOWW + 2)}{\Phi I2C} + SCL\ 上升时间 + SCL\ 下降时间}$
1	Non 000	$\dfrac{1}{\dfrac{(SHIGHW + 2 + 滤波能力) + (SLOWW + 2 + 滤波能力)}{\Phi I2C} + SCL\ 上升时间 + SCL\ 下降时间}$

ΦI2C: I2C baud rate clock frequency, obtained by PCLK3 dividing frequency.

The SCL rise time and SCL fall time in the equation are system design dependent and can be corrected for the output baud rate by correctly substituting the actual SCL rise and fall times.

3.4.3 Slave Address Configuration

Applicati

There are two slave address registers, I2C_SLR0 and I2C_SLR1, which can support two slave addresses at the same time, 7-bit or 10-bit slave addresses respectively.

3.4.4 Broadcast address matching function

The broadcast address matching function can be enabled by the register flag bit I2C_CR1.GCEN. When enabled, the slave I2C device can receive a broadcast command with address 0 and set the I2C_SR.GCF flag bit.

The following 3.4.5~3.4.7 are the configurable functions in SMBus mode.

3.4.5 SMBus Host Address Matching Enable

SMBus mode enable allows you to configure host address configuration enable, alarm response address match enable, and default address match enable when in slave receive mode. For details, please refer to the ""27.3.2 Address Matching" section in the user manual of this chip series.

3.4.6 SMBus Timeout Measurement

The timeout measurement of SMBus includes SCL level timeout measurement, timeout measurement of slave, and timeout measurement of host. The hardware of this I2C module supports SCL level timeout measurement function, and the timeout function can be configured by TMOUTEN, HTMOUT and LTMOUT registers, please refer to the chapter ""27.3.2.3 SMBus Host Address Matching" in the user manual of this chip series for detailed description. The slave timeout measurement and host timeout measurement can be implemented by software.

3.4.7 Fast ACK Configuration

In SMBus receive mode, SCL needs to be held low on the 8th falling edge of the clock, and then the CPU will write the ACK bit to release the SCL low state according to the calculation result after processing the datagram error code (PEC) hosting. This function can be enabled by the I2C_CR3.FACKEN flag bit.

3.5 Interrupt source and event signal output

The interrupt source output is shown below.

Name	Interrupt source	Interruption flags	Interrupt conditions
I2C_EEI	Communication errors/communication events	ARLOF	ARLOF=1&ARLOIE=1
		SLADDR0F	SLADDR0F=1& SLADDR0IE=1
		SLADDR1F	SLADDR1F=1& SLADDR1IE=1
		SMBALRTF	SMBALRTF =1& SMBALRTIE=1
		SMBHOSTF	SMBHOSTF =1& SMBHOSTFIE=1
		SMBDEFAULTF	SMBDEFAULTF =1& SMBDEFAULTIE=1
		GENCALLF	GENCALLF =1& GENCALLIE=1
		NACKF	NACKF=1&NACKIE -1
		TMOUTF	TMOUTF=1&TMOUTIE=1
		STARTF	STARTF=1&STARTIE=1
		STOPF	STOPF=1&STOPIE=1
I2C_RXI	Receiving data full	RFULLF	RFULLF=1&RFULLIE=1
I2C_TXI	Send data empty	TEMPTYF	TEMPTYF=1&TEMPTYIE=1
I2C_TEI	End of sending	TENDF	TENDF=1&TENDIE=1

The event signal input is shown below. The event signal output flag enables the DMA data receiving and transmitting function, please refer to the application note of DMA function of this series chip for the usage.

Name	Event Source	Event Conditions
I2C_EEI	Communication error/communication time	ARLOF=1
		SLADDR0F=1
		SLADDR1F=1
		SMBALRTF=1
		SMBHOSTF=1
		SMBDEFAULTF=1
		GENCALLF=1
		NACKF=1
		TMOUTF=1
		STARTF=1
		STOPF=1
I2C_RXI	Receiving data full	RFULLF=1
I2C_TXI	Send data empty	TEMPTYF=1
I2C_TEI	End of sending	TENDF=1

3.6   Register Description

I2C1   Base   Address:

0x4004E000   I2C2   Base

Address: 0x4004E400 I2C3

Base Address: 0x4004E800

Register Name	Symbols	Offset Address	Bit width	Reset value
I2C control register 1	I2C_CR1	0x00	32	0x0000 0040
I2C control register 2	I2C_CR2	0x04	32	0x0000 0000
I2C control register 3	I2C_CR3	0x08	32	0x0000 0006
I2C slave address register 0	I2C_SLR0	0x10	32	0x0000 1000
I2C slave address register 1	I2C_SLR1	0x14	32	0x0000 0000
I2C Status Register	I2C_SLTR	0x18	32	0xFFFF FFFF
I2C Status Register	I2C_SR	0x1C	32	0x0000 0000
I2C Status Register	I2C_CLR	0x20	32	0x0000 0000
I2C data transmission register	I2C_DTR	0x24	8	0xFF
I2C Data Receive Register	I2C_DRR	0x28	8	0x00
I2C Baud Rate Control Register	I2C_CCR	0x2C	32	0x0000 1F1F
I2C Baud Rate Control Register	I2C_FLTR	0x30	32	0x0000 0010

The above is the register distribution of the 3 independent channels of the I2C module of this series chip, please refer to the relevant chapter of the user manual for detailed register description.

3.7 Initialization process

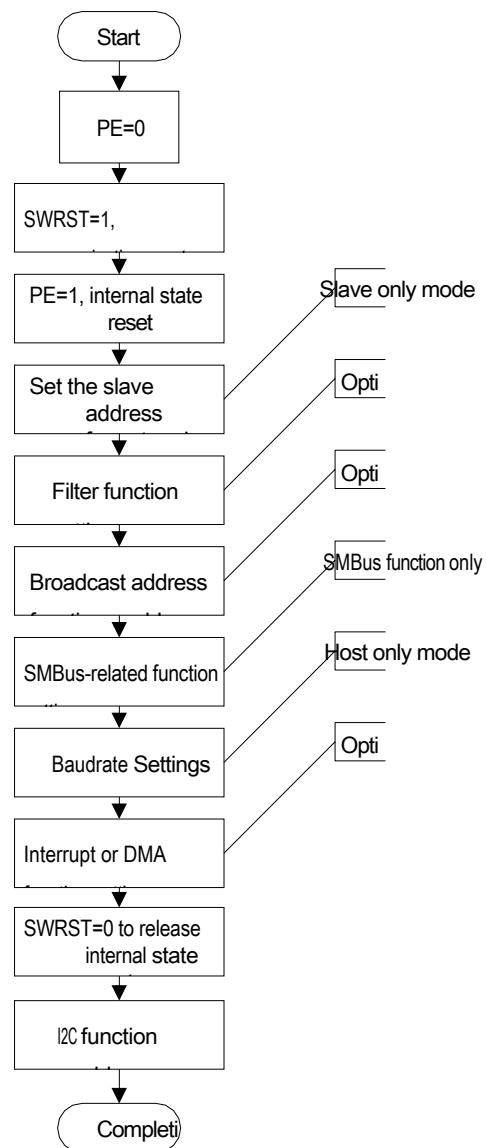The recommended initialization process for the I2C module is as follows:

Figure 3 I2C module initialization flow

4   Sample Code

4.1   Code Introduction

Users can write their own code to learn and verify the module according to the above workflow, or download the sample code of Device Driver Library (DDL) directly from the website of UW Semiconductors and use the I2C sample for verification.

The following section briefly describes the sample E2PROM read/write for the DDL-based I2C module.

1) Test buffer, test IO initialization and system clock initialization

```
uint8_t u8TxBuf[PAGE_LEN];
uint8_t u8RxBuf[PAGE_LEN];
uint32_t i.
uint8_t u8Ret = I2C_RET_OK;
stc_port_init_t stcPortInit.

for(i=0; i<PAGE_LEN; i++)
{
      u8TxBuf[i] = i+1.
}
memset(u8RxBuf, 0x00, PAGE_LEN).

/* Initialize system clock*/
SysClkIni().

/*initiallize LED port*/
MEM_ZERO_STRUCT(stcPortInit);
stcPortInit.enPinMode = Pin_Mode_Out;
stcPortInit.enExInt = Enable; stcPortInit.
enPullUp = Enable.

/* LED0 Port/Pin initialization */
PORT_Init(LED0_PORT, LED0_PIN, &stcPortInit).
/* LED1 Port/Pin initialization */
PORT_Init(LED1_PORT, LED1_PIN, &stcPortInit).
```

2) I2C port configuration and I2C peripheral gating enable

```
/* Initialize I2C port*/
PORT_SetFunc(I2C1_SCL_PORT, I2C1_SCL_PIN, Func_I2c1_Scl, Disable);
PORT_SetFunc(I2C1_SDA_PORT, I2C1_SDA_PIN, Func_I2c1_Sda, Disable).

/* Enable I2C Peripheral*/
PWC_Fcg1PeriphClockCmd(PWC_FCG1_PERIPH_I2C1, Enable).
```

3) I2C function initialization configuration

```
/* Initialize I2C peripheral and enable function*/
E2_Initialize().
```

4) E2PROM Write a byte

```
/* E2prom byte write*/
u8Ret = E2_StartOrRestart(GENERATE_START).
u8Ret = E2_SendAdr((E2_ADDRESS<<1)|E2_ADDRESS_W).
JudgeResult(u8Ret).
u8Ret = E2_SendAdr(DATA_TEST_ADDR);
JudgeResult(u8Ret).
u8Ret = E2_WriteData(u8TxBuf, 1);
JudgeResult(u8Ret).
u8Ret = E2_Stop();
JudgeResult(u8Ret).
```

5) 5mS delay and E2PROM read one byte

```
/* 5mS delay for e2prom*/
Ddl_Delay1ms(5).

/* E2prom ramdom read*/
u8Ret = E2_StartOrRestart(GENERATE_START);
JudgeResult(u8Ret).
u8Ret = E2_SendAdr((E2_ADDRESS<<1)|E2_ADDRESS_W).
JudgeResult(u8Ret).
u8Ret = E2_SendAdr(DATA_TEST_ADDR);
JudgeResult(u8Ret).

u8Ret = E2_StartOrRestart(GENERATE_RESTART);
JudgeResult(u8Ret).
u8Ret = E2_SendAdrRevData((E2_ADDRESS<<1)|E2_ADDRESS_R, u8RxBuf, 1);
JudgeResult(u8Ret).
u8Ret = E2_Stop();
JudgeResult(u8Ret).
```

6) Compare read and write data, feedback test results via LED

```
/* Compare the data */
if(0x01 ! = u8RxBuf[0])
{
    /* e2prom byte write error*/
    while(1)
    {
        LED0_TOGGLE().
        Ddl_Delay1ms(500).
    }
}
```

7) E2PROM page operation read/write and LED feedback results (omitted)

4.2 Code Run

Users can download the sample code (i2c_at24c02) of the HC32F460 DDL from the UW website and run the code with the evaluation board (EV-HC32F460-LQFP100-050-V1.1) to learn how to use the i2C module.

The following section focuses on how to run the sample code on the evaluation board and observe the results:

— Verify that the correct IAR EWARM v7.7 tool is installed (please download the appropriate installation package from the official IAR website and refer to the user manual for installation).

— Download the HC32F460 DDL code from the UW Semiconductors website.

— Download and run the project file in i2c_at24c02\ at

    1) Open the i2c_at24c02\ project and open the 'main.c' view as follows:
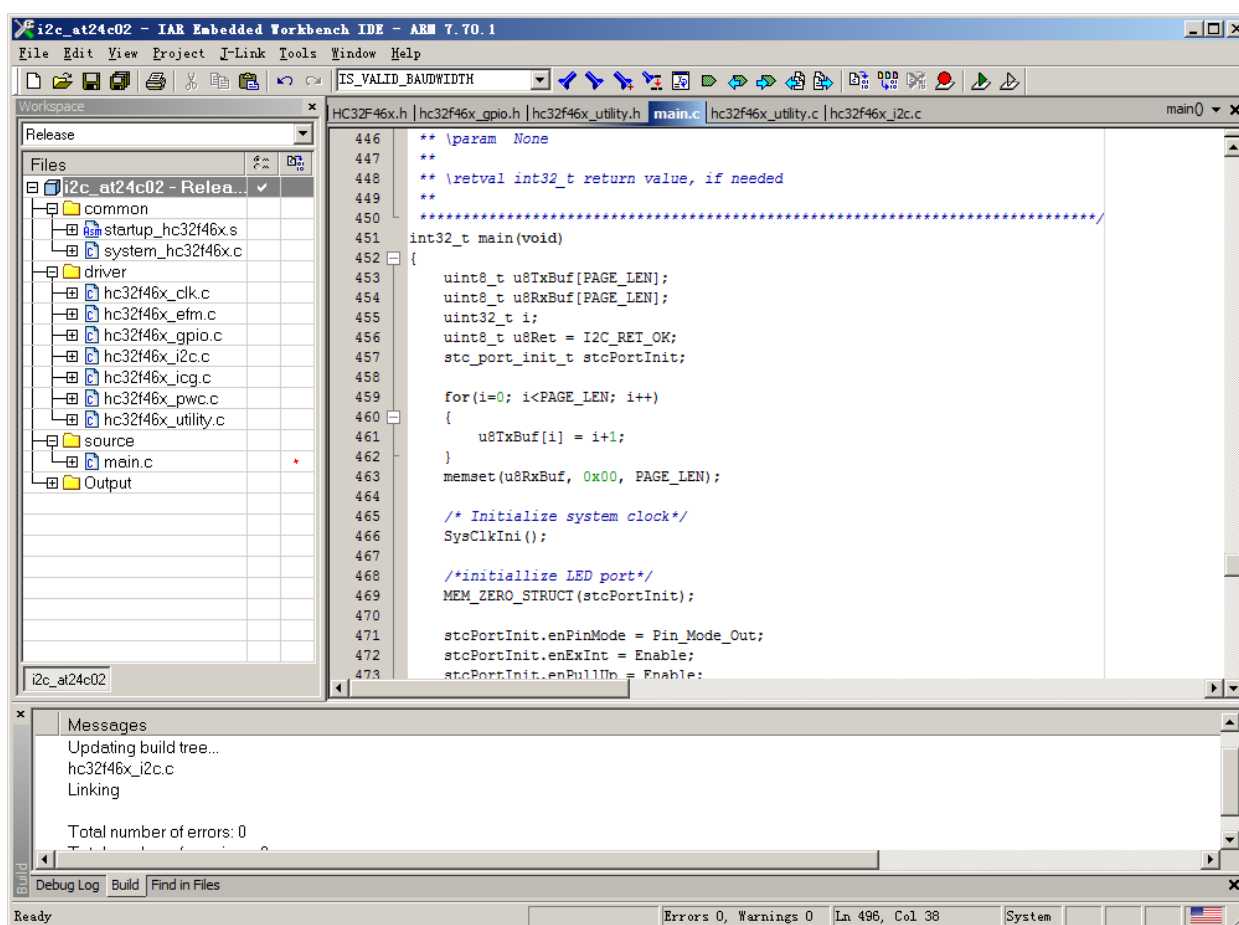
    2) Click [icon] to recompile the entire project.

    3) Click [icon] Download the code to the evaluation board and run it at full speed.

Applicati

4) Observe the LED light change to determine whether the E2PROM read/write is successful. After the operation is finished, the green light blinks to indicate successful reading and writing and data comparison, while the red light blinks to indicate I2C communication failure.

5   Summary

The above section briefly introduces the HC32F460 series I2C, explains the register and initialization operation flow of the I2C module, and demonstrates how to use the sample code, so that users can configure and use the I2C module according to their needs in the actual development.

6   Version Information & Contact

Date	Versions	Modify records
2019/3/15	Rev1.0	Initial Release
2020/8/26	Rev1.1	Update "Name" in 3.5 form; update support for models

If you have any comments or suggestions in the process of purchase and use, please feel free to contact us.

Email: mcu@hdsc.com.cn

Website: http://www.hdsc.com.cn/mcu.htm

Address: 10/F, Block A, 1867 Zhongke Road, Pudong

New Area, Shanghai, 201203, P.R. China