

## 免责声明

---



### Firmware Disclaimer Information

1. The customer hereby acknowledges and agrees that the program technical documentation, including the code, which is supplied by BEST HEALTH ELECTRONIC Inc., (hereinafter referred to as BestHealth) is the proprietary and confidential intellectual property of BestHealth, and is protected by copyright law and other intellectual property laws.
2. The customer hereby acknowledges and agrees that the program technical documentation, including the code, is confidential information belonging to BestHealth, and must not be disclosed to any third parties other than BestHealth and the customer.
3. The program technical documentation, including the code, is provided and for customer reference only. After delivery by BestHealth, the customer shall use the program technical documentation, including the code, at their own risk. BestHealth disclaims any expressed, implied or statutory warranties, including the warranties of merchantability, satisfactory quality and fitness for a particular purpose.

**Copyright (C) BEST HEALTH ELECTRONIC Inc.**  
**All rights reserved**

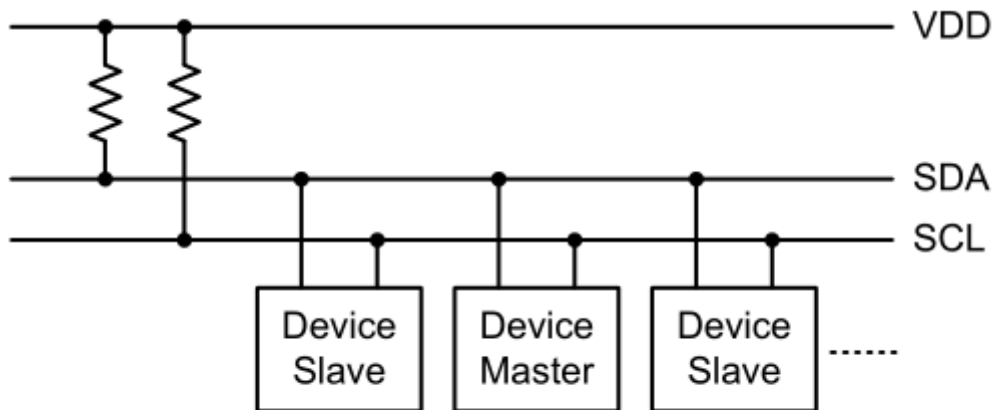
---

## I2C 说明

I2C 可以和传感器、EEPROM 内存等外部硬件接口进行通信。最初是由飞利浦公司研制，是适用于同步串行数据传输的双线式低速串行接口。I2C 接口具有两线通信，非常简单的通信协议和在同一总线上和多个设备进行通信的能力的优点，使之在很多的场合中大受欢迎。

I2C 串行接口是一个双线的接口，有一条串行数据线 SDA 和一条串行时钟线 SCL。由于可能有多个设备在同一条总线上相互连接，所以这些设备的输出都是开漏型输出。因此应在这些输出口上都应加上拉电阻。

BH 8bit MCU 硬件只能作为 I2C 从机。若需要作为主机则需要使用 IO 模拟。详细可参考 I2C\_Master 相关范例。



I<sup>2</sup>C 主从总线连接图

# example 说明

此范例演示了 BH 8bit MCU 作为 I2C 从机，使用中断收发数据的范例。

范例模拟将 BH 8bit MCU 作为一个功能模块，master 可以读写相关寄存器的读写操作。

范例主机部分参考范例 I2c\_Master\_IO 部分

## 程序说明

### 1. I2c\_Master\_IO 主机部分代码如下

```

25 #define DEVICE_I2C_ADDRESS 0xA0 ① 从机I2C地址
26 typedef struct {
27     uint8_t regAddress;
28     uint8_t regData[3];
29 } deviceReg_t;
30 deviceReg_t deviceReg; ② 自定义数据类型
31
32 void main()
33 {
34     // 判断是否为上电复位或者非正常情况下的复位，如果是上电复位，执行上电复位初始化，反之执行WDT
35     if (_to == 0 || _pdf == 0) {
36         // config sys clock
37         Oscillators_Cfg();
38         // read default data
39         deviceReg.regAddress = 0xD0; ③ 读出从机地址0xD0后连续3个byte的数据
40         I2c_Master_ReadData(DEVICE_I2C_ADDRESS, &deviceReg.regAddress, 3);
41         // write new data
42         deviceReg.regData[0] = 0x01;
43         deviceReg.regData[1] = 0x02;
44         deviceReg.regData[2] = 0x03; ④ 修改从机地址0xD0后连续3个byte的数据为1~3
45         I2c_Master_WriteData(DEVICE_I2C_ADDRESS, &deviceReg.regAddress, 3);
46         deviceReg.regData[0] = 0x00;
47         deviceReg.regData[1] = 0x00;
48         deviceReg.regData[2] = 0x00;
49     } else {
50         // WDT溢出复位初始化
51         GCC_CLRWD();
52     }
53     while (1) {
54         GCC_DELAY(10000);
55         GCC_CLRWD();
56         // read new data ⑤ 读出从机地址0xD0连续3个byte的数据，确认写入是否正确
57         I2c_Master_ReadData(DEVICE_I2C_ADDRESS, &deviceReg.regAddress, 3);
58     }
59 }

```

### 2. I2c\_Slave\_Interrupt 从机部分代码如下

```

DEFINE_SFR(uint8_t, RegExample0, 0xD0); ① 定义寄存器位置，比如RegExample0,地址为0xD0
DEFINE_SFR(uint8_t, RegExample1, 0xD1);
DEFINE_SFR(uint8_t, RegExample2, 0xD2);

volatile uint8_t rxBuf[20];
void main()
{
    // 判断是否为上电复位或者非正常情况下的复位，如果是上电复位，执行上电复位初始化，反之执行WDT溢出初始化
    if (_to == 0 || _pdf == 0) {
        // config sys clock
        Oscillators_Cfg();
        // config I2c
        I2c_Cfg_t Cfg;
        Cfg.addr = 0xA0; ③ I2c从机地址定义
        Cfg.debounce = I2C_DEBOUNCE_NONE;
        Cfg.wakeUp = true; ④ 开启I2c地址匹配唤醒功能
        Cfg.timeOut = false;
        Cfg.timeOutMs = 10;
        I2c_Cfg(&Cfg);
        I2c_Enable();
        I2c_Isr_Enable();
        // enable emi
        _emi = 1;

        // init data
        RegExample0 = 0x51; ② 给寄存器赋初值
        RegExample1 = 0x52;
        RegExample2 = 0x53;
    } else {
        // WDT溢出复位初始化
        GCC_CLRWD();
    }

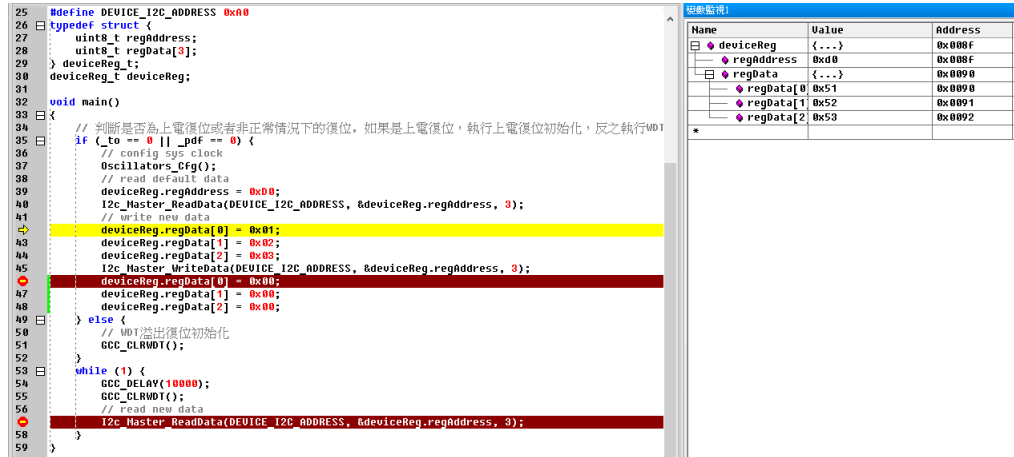
    while (1) {
        GCC_CLRWD();
    }
}

```

## 现象说明

1. 连接 e-link 和目标板
2. 分别将主机程序和从机程序烧录到目标板上
3. 将主机上的 SDA、SCL、GND 等连接到从机对应的 SDA、SCL、GND 上
4. 下载并运行MCU程序
  - 主机按以下断点依次查看数据

### 1. 读取默认值



```

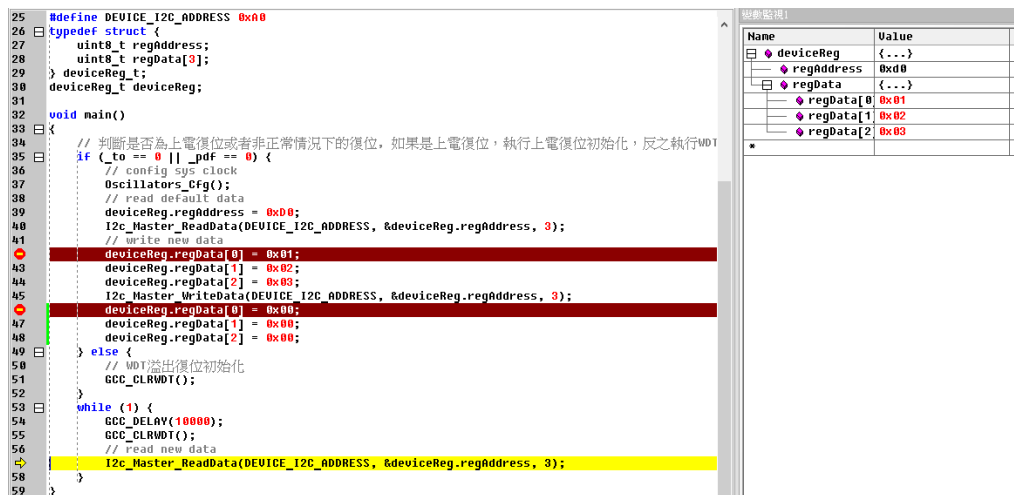
25 #define DEVICE_I2C_ADDRESS 0xA0
26 typedef struct {
27     uint8_t regAddress;
28     uint8_t regData[3];
29 } deviceReg_t;
30 deviceReg_t deviceReg;
31
32 void main()
33 {
34     // 判断是否为上电复位或者非正常情况下的复位，如果是上电复位，执行上电复位初始化，反之执行WDT
35     if (_to == 0 || _pdf == 0) {
36         // config sys clock
37         Oscillators_Cfg();
38         // read default data
39         deviceReg.regAddress = 0x00;
40         I2c_Master_ReadData(DEVICE_I2C_ADDRESS, &deviceReg.regAddress, 3);
41         // write new data
42         deviceReg.regData[0] = 0x01;
43         deviceReg.regData[1] = 0x02;
44         deviceReg.regData[2] = 0x03;
45         I2c_Master_WriteData(DEVICE_I2C_ADDRESS, &deviceReg.regAddress, 3);
46         deviceReg.regData[0] = 0x00;
47         deviceReg.regData[1] = 0x00;
48         deviceReg.regData[2] = 0x00;
49     } else {
50         // WDT溢出复位初始化
51         GCC_CLRWDVT();
52     }
53     while (1) {
54         GCC_DELAY(10000);
55         GCC_CLRWDVT();
56         // read new data
57         I2c_Master_ReadData(DEVICE_I2C_ADDRESS, &deviceReg.regAddress, 3);
58     }
59 }

```

Name	Value	Address
deviceReg	{...}	0x000F
regAddress	0x00	0x000F
regData	{...}	0x0090
regData[0]	0x51	0x0090
regData[1]	0x52	0x0091
regData[2]	0x53	0x0092

### 2. 写入新的值

### 3. 读出写入的值



```

25 #define DEVICE_I2C_ADDRESS 0xA0
26 typedef struct {
27     uint8_t regAddress;
28     uint8_t regData[3];
29 } deviceReg_t;
30 deviceReg_t deviceReg;
31
32 void main()
33 {
34     // 判断是否为上电复位或者非正常情况下的复位，如果是上电复位，执行上电复位初始化，反之执行WDT
35     if (_to == 0 || _pdf == 0) {
36         // config sys clock
37         Oscillators_Cfg();
38         // read default data
39         deviceReg.regAddress = 0x00;
40         I2c_Master_ReadData(DEVICE_I2C_ADDRESS, &deviceReg.regAddress, 3);
41         // write new data
42         deviceReg.regData[0] = 0x01;
43         deviceReg.regData[1] = 0x02;
44         deviceReg.regData[2] = 0x03;
45         I2c_Master_WriteData(DEVICE_I2C_ADDRESS, &deviceReg.regAddress, 3);
46         deviceReg.regData[0] = 0x00;
47         deviceReg.regData[1] = 0x00;
48         deviceReg.regData[2] = 0x00;
49     } else {
50         // WDT溢出复位初始化
51         GCC_CLRWDVT();
52     }
53     while (1) {
54         GCC_DELAY(10000);
55         GCC_CLRWDVT();
56         // read new data
57         I2c_Master_ReadData(DEVICE_I2C_ADDRESS, &deviceReg.regAddress, 3);
58     }
59 }

```

Name	Value	Address
deviceReg	{...}	0x000F
regAddress	0x00	0x000F
regData	{...}	0x0090
regData[0]	0x01	0x0090
regData[1]	0x02	0x0091
regData[2]	0x03	0x0092

5. 从机运行后等待主机写入后停下查看数据

```

21 #include "..\..\driver\wdt.h"
22 #include "..\..\driver\Oscillators.h"
23 #include "..\..\driver\I2c.h"
24
25 DEFINE_SFR(uint8_t, RegExample0, 0x00);
26 DEFINE_SFR(uint8_t, RegExample1, 0x01);
27 DEFINE_SFR(uint8_t, RegExample2, 0x02);
28
29 volatile uint8_t rxBuf[20];
30 void main()
31 {
32     // 判断是否为上电复位或者非正常情况下的复位，如果是上电复位，执行上电复位初始化，反之执行WD
33     if (_to == 0 || _pdf == 0) {
34         // config sys clock
35         Oscillators_Cfg();
36         // config I2c
37         I2c_Cfg_t Cfg;
38         Cfg.addr = 0xA0;
39         Cfg.debounce = I2C_DEBOUNCE_CLOCK2;
40         Cfg.wakeUp = true;
41         Cfg.timeOut = true;
42         Cfg.timeOutMs = 10;
43         I2c_Cfg(&Cfg);
44         I2c_Enable();
45         I2c_Isr_Enable();
46         // enable emi
47         _emi = 1;
48
49         // init data
50         RegExample0 = 0x51;
51         RegExample1 = 0x52;
52         RegExample2 = 0x53;
53     } else {
54         // WDT溢出复位初始化
55         GCC_CLRWDI();
56     }
57     while (1) {
58         GCC_CLRWDI();
59     }
60 }

```

变量监视1

Name	Value
RegExample0	0x01
RegExample1	0x02
RegExample2	0x03
*	

# FAQ

---