

32 位微控制器

HC32L15 系列的 I²C 总线接口

适用对象

系列	产品型号
HC32L15	HC32L150KATA
	HC32L150JATA
	HC32L150FAUA
	HC32L156KATA
	HC32L156JATA

目 录

1	摘要	3
2	I²C 总线简介	3
3	HC32L15 系列的 I²C 总线接口	4
3.1	简介	4
3.2	功能特点	4
3.3	拓扑结构	4
3.4	I ² C 总线接口配置	5
3.4.1	引脚使用配置	5
3.4.2	波特率计算	6
3.4.3	寄存器列表	6
3.4.4	工作流程介绍	7
4	样例代码	9
4.1	代码介绍	9
4.1.1	主机模式代码	9
4.1.2	从机模式代码	10
4.2	代码运行	12
5	注意事项	15
6	总结	15
7	版本信息 & 联系方式	16

1 摘要

本篇应用笔记主要介绍 HC32L15 系列的 I²C 总线模块的特点及使用说明，以帮助读者快速熟悉该芯片的此模块。

2 I²C 总线简介

什么是 I²C 总线？

I²C（Inter-Integrated Circuit）总线是由 PHILIPS 公司开发的两线式串行总线，用于连接微控制器及其外围设备。

（引自‘百度百科’，‘互动百科’，‘维基百科’）

I²C 总线的重要特征？

I²C 总线是微电子通信控制领域广泛采用的一种总线标准。它是同步通信的一种特殊形式，具有接口线少、控制方式简单、器件封装形式小、通信速率较高等优点。

3 HC32L15 系列的 I²C 总线接口

3.1 简介

华大 HC32L15 系列单片机内部集成 I²C 总线接口，挂载于 AHB-APB（APB1）总线，最高可工作至 32MHz，支持配置为 I²C 主机设备和 I²C 从机设备模式。

3.2 功能特点

I²C 总线接口具有以下功能特点：

通道数量：6 通道

数据位长度：8 位

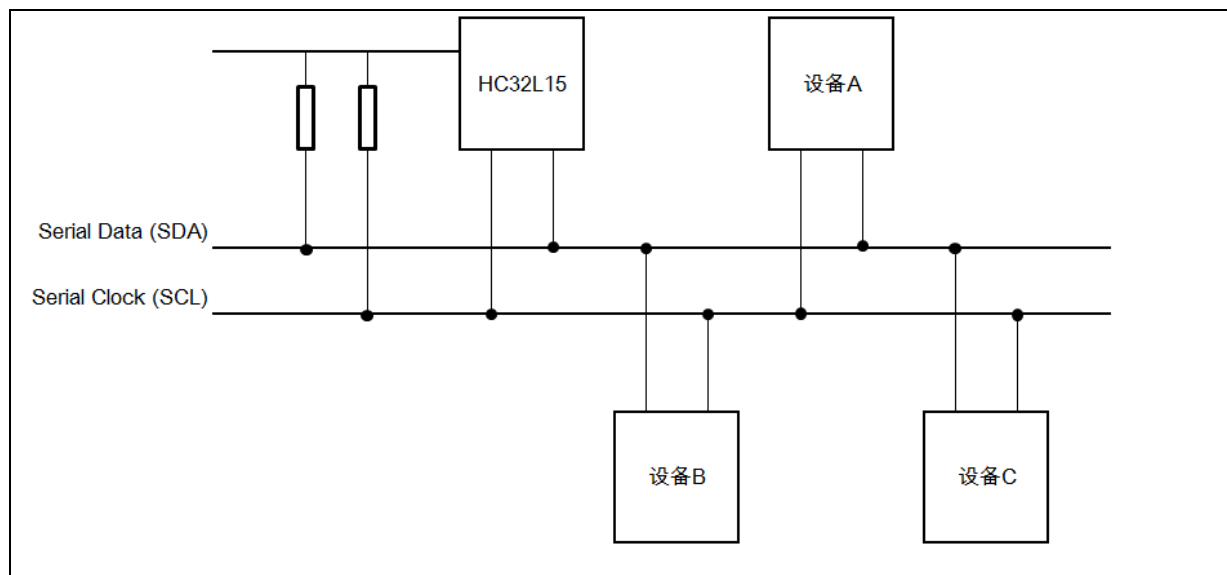
传输速率：100Kbps/400Kbps

中断请求：数据收发完成、总线仲裁丢失

寻址方式：7 位地址、10 位地址、广播地址

3.3 拓扑结构

下图为 I²C 总线连线的拓扑结构。



3.4 I²C 总线接口配置

3.4.1 引脚使用配置

HC32L15 系列 MCU 的 I²C 接口并不局限于 MCU 的固定引脚，为方便用户进行 PCB 设计，支持了端口重映射功能，即每一个通道的 SDA 和 SCL 可映射至多个端口。下面以 HC32L156KATA，LQFP64 封装芯片的通道 0 为例进行说明*1。

通道	引脚名称	功能描述
0	SOT0_0*2	串行数据引脚（SDA）
	SOT0_1	
	SOT0_2	
	SCK0_0	串行时钟引脚（SCL）
	SCK0_1	
	SCK0_2	

*1：针对于此功能，关于该系列 MCU 的每个产品型号，请参考芯片数据手册以获取更多信息。

*2：SOTx_y 格式解释：

x：通道序号

y：可分配端口

x 相同的情况下，y 可以任意组合；如 SOT0_0 与 SCK0_1 配合使用

3.4.2 波特率计算

通信波特率计算公式如下：

$$\text{波特率 (bps)} = \text{PCLK} / (\text{BRS} + 1)$$

PCLK：总线时钟

BRS：波特率发生器重装值

3.4.3 寄存器列表

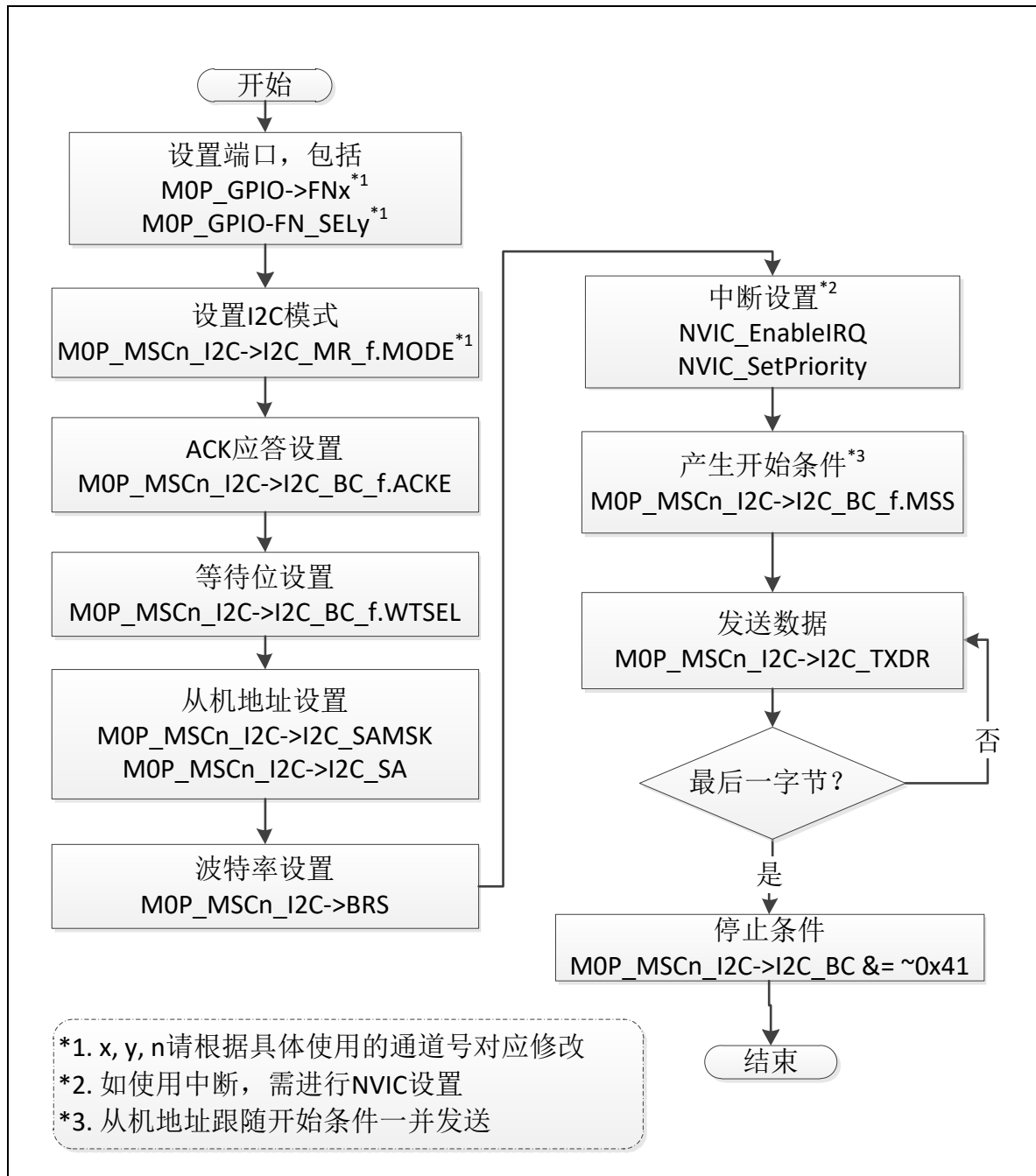
对于 AES 模块的操作主要通过以下寄存器进行：

英文说明（缩写）	中文说明
Bus Control Register（BC）	I ² C 总线控制寄存器
Mode Register（MR）	模式寄存器
Bus Status Register（SR）	I ² C 总线状态寄存器
Receive/Transmit Data Register（RXDR/TXDR）	接收/发送数据寄存器
Extend Bus Control Register（EBCR）	I ² C 总线扩展控制寄存器
Baudrate Setting Register（BRS）	波特率设置寄存器
Slave Address Mask Register（SAMSK）	从机地址屏蔽寄存器
Slave Address Register（SA）	从机地址寄存器

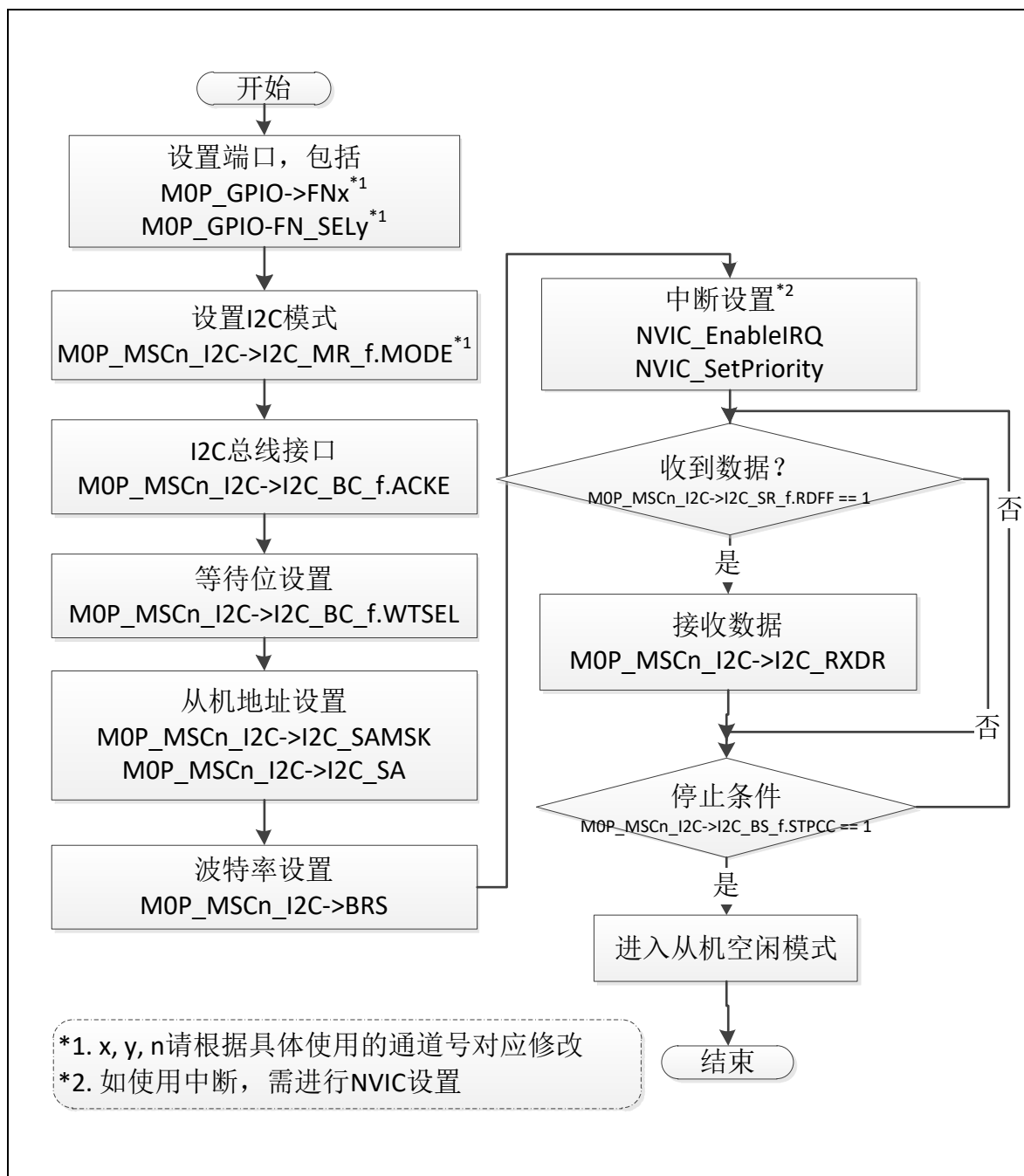
3.4.4 工作流程介绍

I²C 总线接口分为主机模式和从机模式，下面分别介绍两个模式的工作流程。

主机模式发送数据工作流程图



从机模式接收数据工作流程图



4 样例代码

4.1 代码介绍

用户可以根据上述的工作流程编写自己的代码来学习验证该模块，也可以直接通过华大半导体的网站下载设备驱动库（Device Driver Library，DDL）的样例代码，并使用其中的 I²C 总线接口 Example 进行验证。

4.1.1 主机模式代码

- 1) 设置从机地址：

```
#define I2C_DEV_ADDR      (0x3Au)
#define I2C_DEV_ADDR_W    ((I2C_DEV_ADDR<<1) | 0u)
#define I2C_DEV_ADDR_R    ((I2C_DEV_ADDR<<1) | 1u)
```

- 2) 选择所使用的 I²C 总线接口通道：

```
static uint8_t u8I2cCh = I2CCH2;
```

- 3) 查看数据手册，根据所选择的通道，配置端口：

```
Gpio_SetFunc_SOT2_0();
Gpio_SetFunc_SCK2_0();
```

- 4) 如果使用中断，请定义并初始化中断回调函数配置结构体：

```
stc_i2c_irq_cb_t stcI2cIntCb;

stcI2cIntCb.pfnTxIrqCb = I2cTxCallback;
stcI2cIntCb.pfnRxIrqCb = I2cRxCallback;
stcI2cIntCb.pfnStopDetectIrqCb = I2cStopDetectCallback;
stcI2cIntCb.pfnTxRxIrqCb = I2cIntCallback;
```

- 5) 定义 I²C 总线接口配置结构体，并赋值，初始化：

```
stc_i2c_config_t stcI2c0Config;

stcI2c0Config.enMsMode = I2cMaster;
stcI2c0Config.u32BaudRate = 100000u;
stcI2c0Config.bWaitSelection = FALSE;

I2c_Init(u8I2cCh, &stcI2c0Config);
```

- 6) 完成初始化后，发送开始条件，从机地址，数据方向为主机到从机：

```
I2c_Start(I2C_DEV_ADDR_W);
```

7) 向从机发送数据:

```
I2c_Write(au8I2cMasterTxBuf, SAMPLE_I2C_MASTER_TX_BUFFSIZE);
```

8) 数据发送结束后, 发起停止条件:

```
I2c_Stop();
```

9) 下面开始接受从机数据, 先发送开始条件以及从机地址, 数据方向为从机到主机:

```
I2c_Start(I2C_DEV_ADDR_R);
```

10) 接下来开始接收从机发来的数据:

```
I2c_Read(au8I2cMasterRxBuf, SAMPLE_I2C_MASTER_RX_BUFFSIZE);
```

11) 接收完所有预期个数的数据后, 发送停止条件:

```
I2c_Stop();
```

通过以上代码即可完成一次 I²C 主机设备发送、接收数据的操作流程。

4.1.2 从机模式代码

1) 设置从机地址:

```
#define I2C_DEV_ADDR      (0x3Au)
#define I2C_DEV_ADDR_W    ((I2C_DEV_ADDR<<1) | 0u)
#define I2C_DEV_ADDR_R    ((I2C_DEV_ADDR<<1) | 1u)
```

2) 选择所使用的 I²C 总线接口通道:

```
static uint8_t u8I2cCh = I2CCH2;
```

3) 查看数据手册, 根据所选择的通道, 配置端口:

```
Gpio_SetFunc_SOT2_0();
Gpio_SetFunc_SCK2_0();
```

4) 定义 I²C 总线接口配置结构体, 并赋值, 初始化:

```
stc_i2c_config_t stcI2c0Config;

stcI2c0Config.enMsMode = I2cSlave;
stcI2c0Config.u32BaudRate = 100000u;
stcI2c0Config.u8SlaveAddr = I2C_DEV_ADDR;
stcI2c0Config.u8SlaveMaskAddr = 0x00u;
stcI2c0Config.bWaitSelection = FALSE;

I2c_Init(u8I2cCh, &stcI2c0Config);
```

5) 根据主机请求, 向主机发送或者从主机接收数据

```
if(TRUE == I2c_GetStatus(u8I2cCh, I2cDevAddrMatch))
{
    if(i2c_slave_tx_master_rx == I2c_GetDataDir(u8I2cCh))//Tx
    {
        delay1ms(1);
        I2c_SlaveWriteData(au8I2cSlaveTxBuf, &WriteLength);
    }
    else//Rx
    {
        I2c_SlaveReadData(au8I2cSlaveRxBuf,&ReadLength);
        memset(au8I2cSlaveTxBuf, 0, sizeof(au8I2cSlaveTxBuf));
        memcpy(au8I2cSlaveTxBuf, au8I2cSlaveRxBuf, ReadLength);
    }
}
```

通过以上代码即可完成 I²C 主机设备发送、接收数据的操作流程。

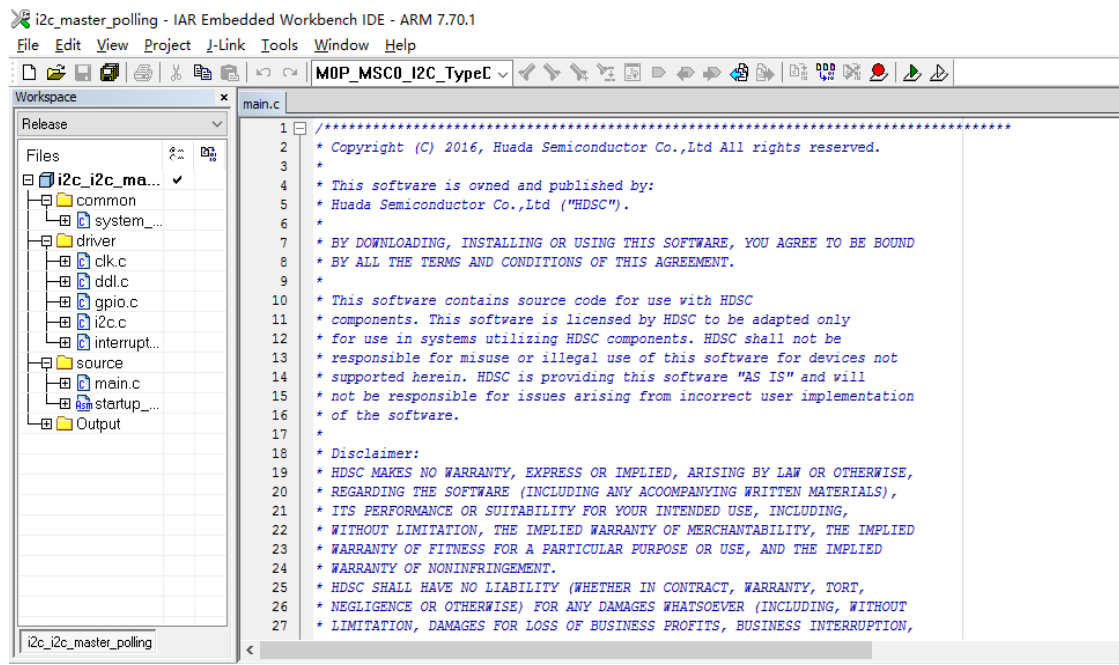
4.2 代码运行

用户可以通过使用 DDL→Example→i2c 文件夹内的样例代码，并配合学习板（比如‘SK-HC32L156-64L V10’）运行相关代码学习使用 I²C 总线接口模块，此样例分为主机部分和从机部分，两套代码配合使用，主机发送数据至从机，从机接收到主机数据并保存；主机再从从机读回刚才发送的数据，并作比较判断通信是否成功。

以下部分主要介绍如何在‘SK-HC32L156-64L V10’学习板上运行 I²C 总线接口样例代码并观察结果：

- 确认安装正确的 IAR Embedded Workbench for ARM 7.70 工具（www.iar.com 下载并安装）；
- 获取‘SK-HC32L156-64L V10’学习板两块；
- 将两块学习板的 I²C 通道 2 的 SCL 和 SDA 连接，分别位于 CN6 的 Pin9 和 Pin10
- 从华大半导体网站下载 I²C 样例代码；
- 下载并运行样例代码：

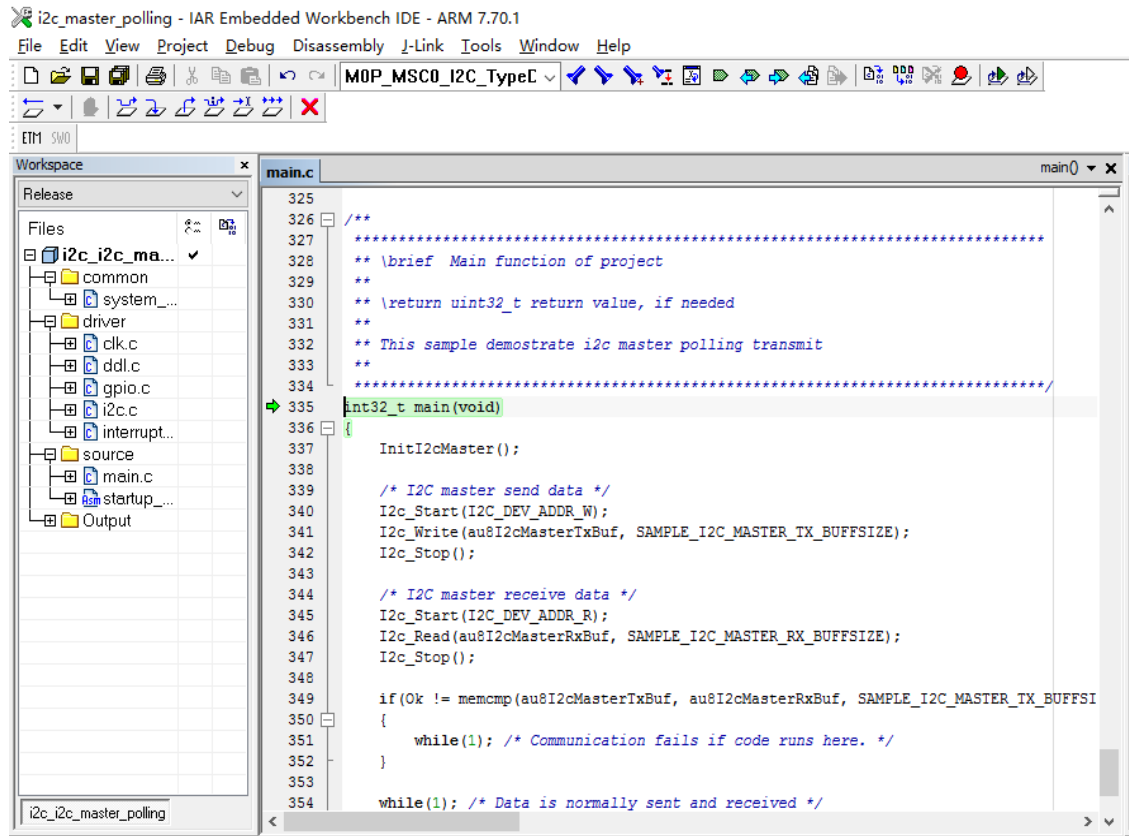
1) 主机项目：打开 i2c_master_polling 项目，并打开‘main.c’如下视图：



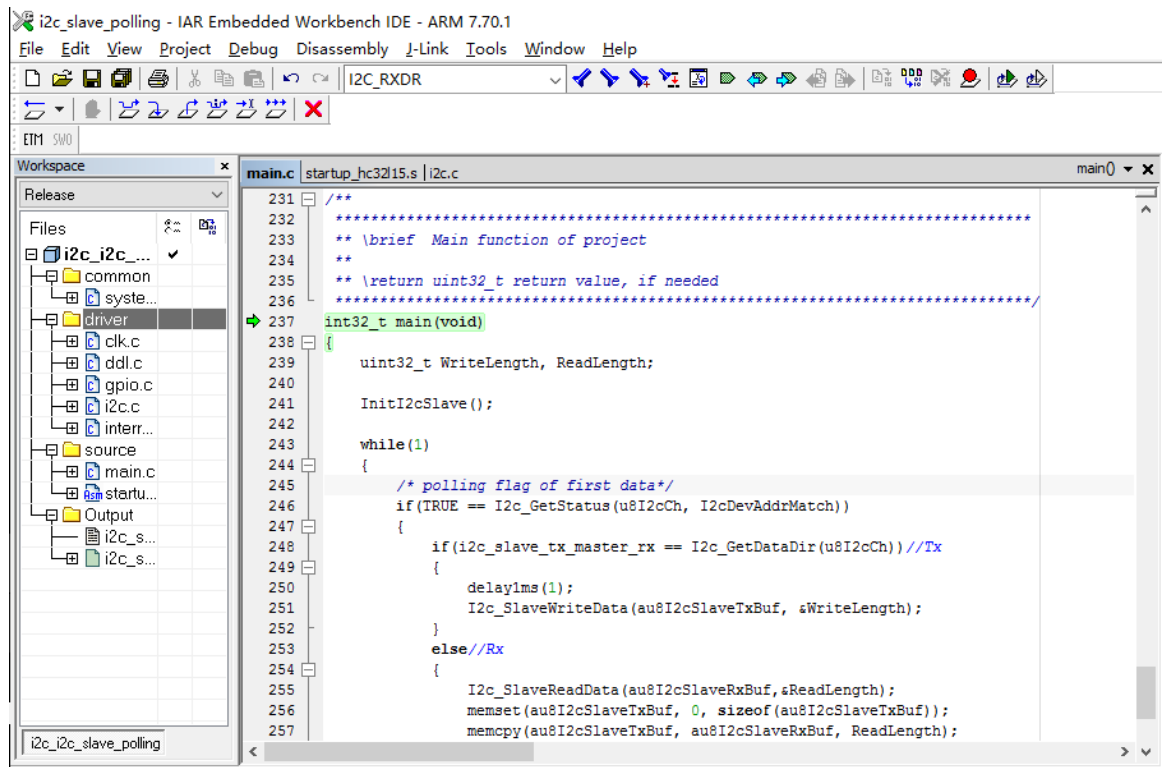
2) 点击  重新编译整个项目；


3) 点击  将代码下载到学习板上；

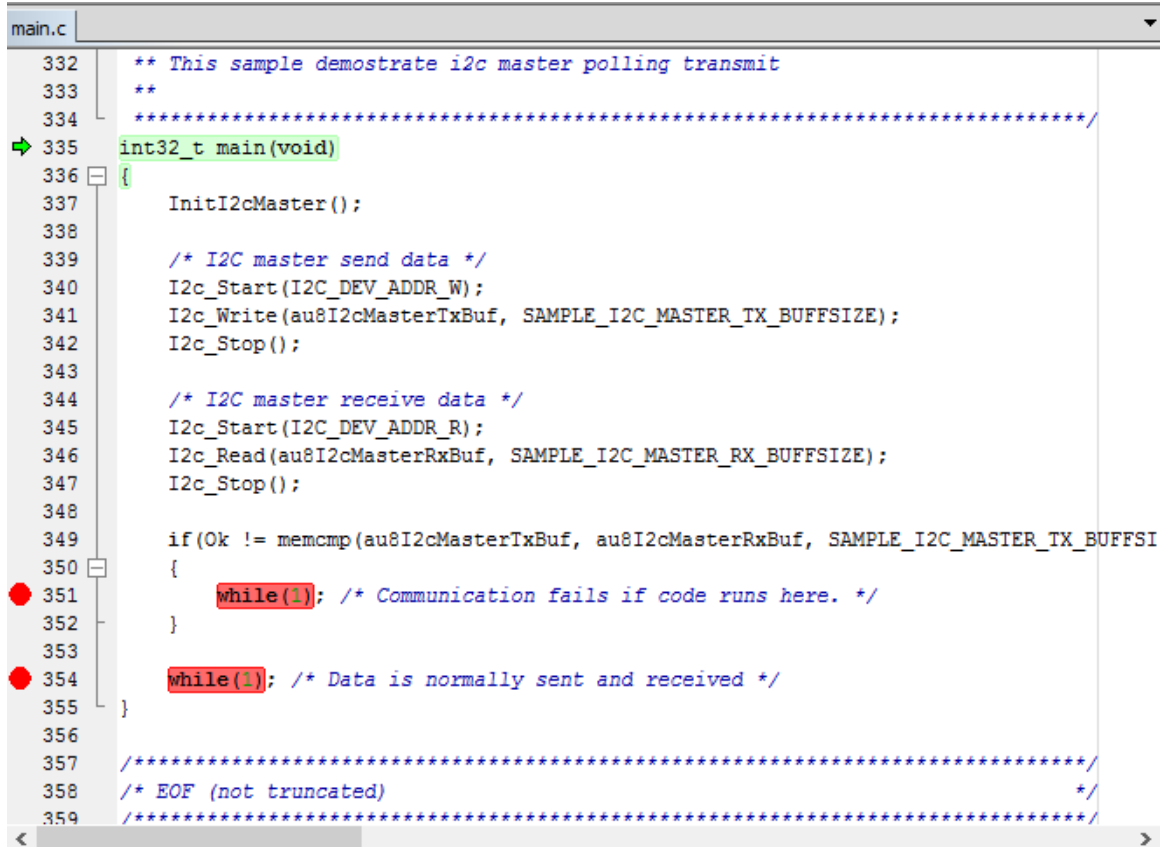
4) 可以看见类似如下的视图:



5) 使用同样的方法，将‘i2c_slave_polling’项目代码编译并下载至第二块学习板上，如下图所示:



- 6) 选择‘i2c_slave_polling’项目调试环境，点击或按快捷键‘F5’，运行从机项目代码；
- 7) 在‘i2c_master_polling’项目的 main.c 文件中的两个 while(1)处分别设置断点，如下图所示：



```

332  /** This sample demonstrate i2c master polling transmit
333  **
334  *****/
335  int32_t main(void)
336  {
337      InitI2cMaster();
338
339      /* I2C master send data */
340      I2c_Start(I2C_DEV_ADDR_W);
341      I2c_Write(au8I2cMasterTxBuf, SAMPLE_I2C_MASTER_TX_BUFFSIZE);
342      I2c_Stop();
343
344      /* I2C master receive data */
345      I2c_Start(I2C_DEV_ADDR_R);
346      I2c_Read(au8I2cMasterRxBuf, SAMPLE_I2C_MASTER_RX_BUFFSIZE);
347      I2c_Stop();
348
349      if(Ok != memcmp(au8I2cMasterTxBuf, au8I2cMasterRxBuf, SAMPLE_I2C_MASTER_TX_BUFFSI
350      {
351          while(1); /* Communication fails if code runs here. */
352      }
353
354      while(1); /* Data is normally sent and received */
355  }
356
357  *****/
358  /* EOF (not truncated) */
359  *****/
  
```

- 8) 通过 IAR 菜单栏‘View-Watch-Watch 1’激活‘Watch 1’窗口，并将全局变量‘au8I2cMasterRxBuf’添加至‘Watch 1’窗口来观察，如下图所示：

Watch 1				x
Expression	Value	Location		T
au8I2cMasterRxBuf	<array> " "	0x200000AC		u
[0]	'\0' (0x00)	0x200000AC		u
[1]	'\0' (0x00)	0x200000AD		u
[2]	'\0' (0x00)	0x200000AE		u
[3]	'\0' (0x00)	0x200000AF		u
[4]	'\0' (0x00)	0x200000B0		u
[5]	'\0' (0x00)	0x200000B1		u
[6]	'\0' (0x00)	0x200000B2		u
[7]	'\0' (0x00)	0x200000B3		u
[8]	'\0' (0x00)	0x200000B4		u
[9]	'\0' (0x00)	0x200000B5		u
<click to add>				

- 9) 选择‘i2c_master_polling’项目调试环境，点击  或按快捷键‘F5’，运行主机项目代码，直至代码运行至断点处，观察‘au8I2cMasterRxBuf’的值，如下图所示：

Watch 1				x
Expression	Value	Location		T
au8I2cMasterRxBuf	<array>"..."	0x200000AC		u
[0]	'0' (0x30)	0x200000AC		u
[1]	'1' (0x31)	0x200000AD		u
[2]	'2' (0x32)	0x200000AE		u
[3]	'3' (0x33)	0x200000AF		u
[4]	'4' (0x34)	0x200000B0		u
[5]	'5' (0x35)	0x200000B1		u
[6]	'6' (0x36)	0x200000B2		u
[7]	'7' (0x37)	0x200000B3		u
[8]	'8' (0x38)	0x200000B4		u
[9]	'9' (0x39)	0x200000B5		u
<click to add>				

- 10) 至此，该样例演示结束，用户亦可通过修改代码中的数据长度、内容、I²C 总线接口通道等参数，来学习体验该模块的功能。

5 注意事项

- I²C 总线为开漏端口，为了保证通信正常，请分别接上拉电阻至 SDA 和 SCL 信号线上。
- 关于通道重定位功能，请查看数据手册以及技术手册的 GPIO 章节对应描述。
- 最高支持 400Kbps 通信速率。
- I²C 总线模块工作时钟不低于 8MHz。

6 总结

以上章节简要介绍了 I²C 总线接口，详细说明了 HC32L15 系列的 I²C 总线接口模块，演示了如何使用相关的样例代码进行 I²C 主从设备进行通信，在开发中用户可以根据自己的实际需要使用该模块。

7 版本信息 & 联系方式

日期	版本	修改记录
2018/8/9	Rev1.0	初版发布。



如果您在购买与使用过程中有任何意见或建议，请随时与我们联系。

Email: mcu@hdsc.com.cn

网址: www.hdsc.com.cn

通信地址: 上海市张江高科园区碧波路 572 弄 39 号

邮编: 201203

