Document Number: AN4652
Rev 0, 01/2013

基于中断和阻塞机制的 MQX I2C 驱动程序

作者: 郭 嘉

内容

1 简介

本应用笔记介绍了基于中断和阻塞机制的 MQX 上的 I2C 驱动程序，屏蔽了 I2C 模块工作的细节并大大简化了应用代码。它包括主机和从机模式。

文中代码已在 K60N512-TWR 开发板上经过验证。

2 驱动程序设计方法

该驱动程序完全由中断驱动。以下介绍了驱动程序在主机和从机模式下的工作情况。

- 主机模式：当用作主机并发送数据时，应用任务将缓冲区的地址和长度传输至驱动程序，然后它将被阻塞。驱动程序完成缓冲区内数据的传输后，将唤醒应用任务。为了减少存储器消耗，驱动程序不会生成自己的缓冲区并将数据从应用复制到缓冲区。该驱动程序直接使用应用层中的缓冲区。同样，当主机接收数据时，它将缓冲区的地址和长度传输给驱动程序，然后将由驱动程序挂起，直到驱动程序完成读取操作。请参见图 1 。
- 从机模式：用作从机时，存在不同之处。和主机不同的是，从机以被动方式工作。因此，当其发送数据时，它将保持阻塞状态，直到主机将其读取。当其读取数据时，它将保持阻塞状态，直到主机向其发送数据。请参见图 1 。

请参见下图，了解设计图示。

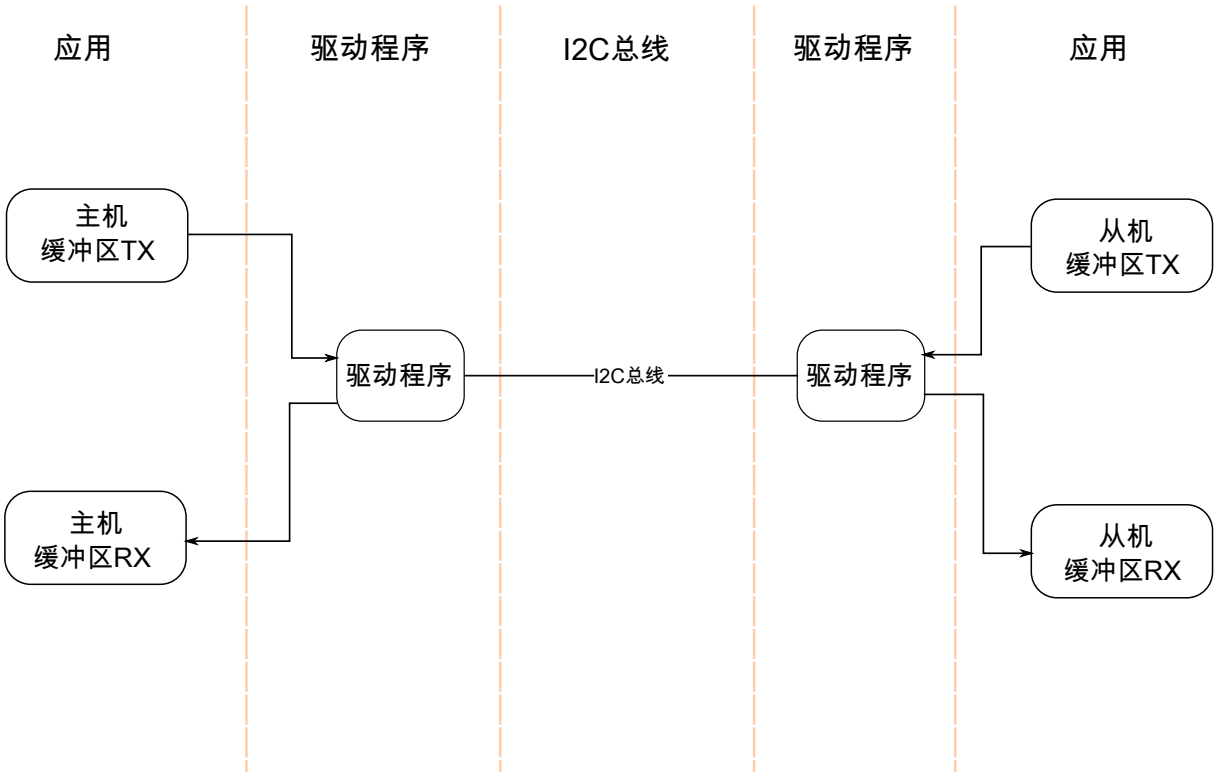
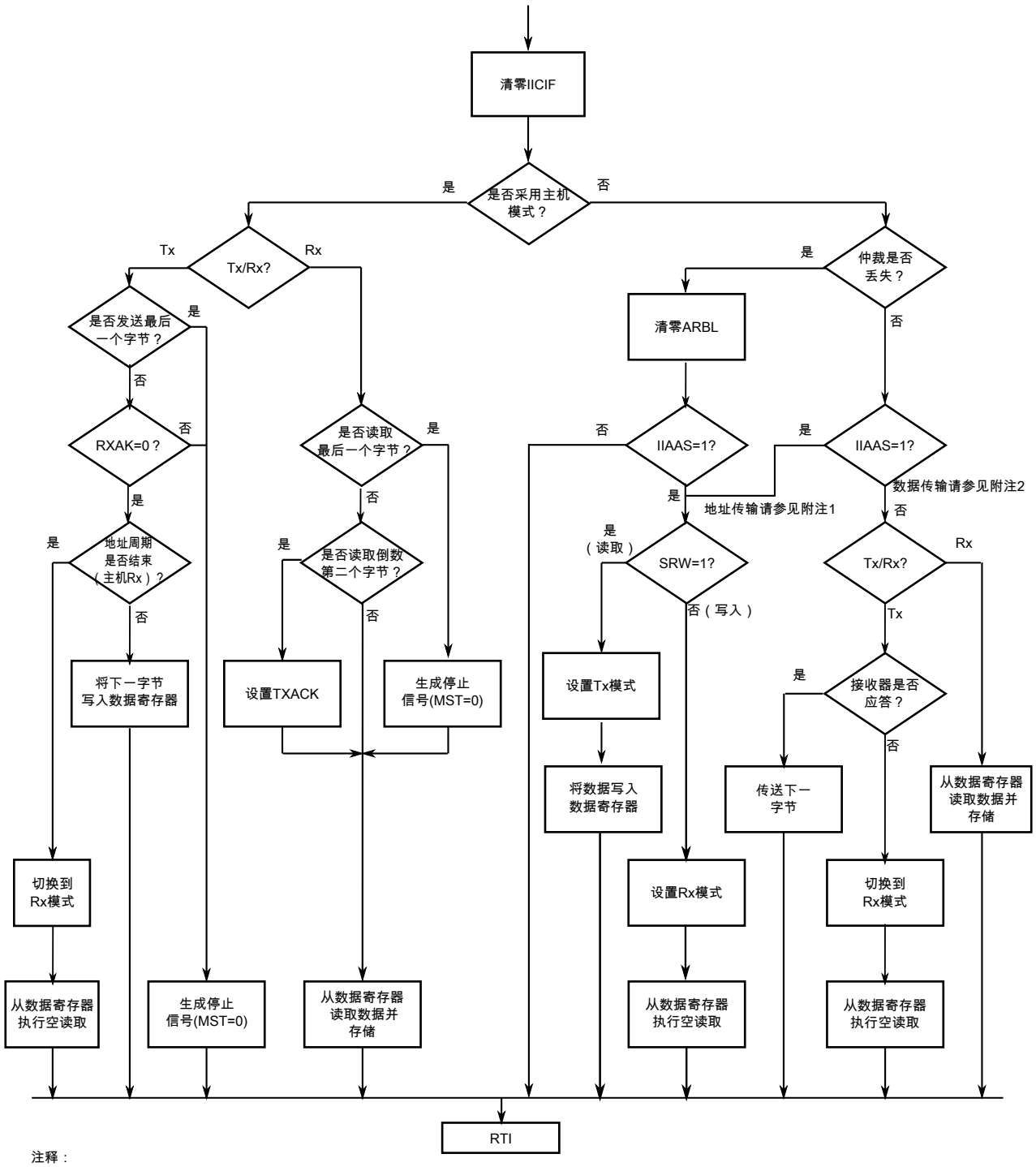


图 1. 设计图示

3 程序流程

图 2 显示主机和从机模式下 I2C 驱动程序工作的程序流程图。如需了解更多信息，请参阅 K60P120M100SF2RM: K60 参考手册（可从 **freescale.com** 获取）。



注释：
1. 如果已使能通用调用，则检查以确定接收到的地址是否是通用调用地址(0x00)。
如果接收到的地址是通用调用地址，则通用调用必须由用户软件处理 。
2. 如果10位寻址进行的是从机寻址，则在扩展地址的首个字节后，该从机会遭遇中断。
对于此中断，必须确保忽略数据寄存器的内容，且不将其当作有效数据传输。

图 2. 典型的 I2C 中断例程

4 通信时序

本章节介绍了 I2C 驱动程序（包括读取和写入操作）在主机和从机模式下的典型通信时序。

4.1 典型时序

通信时序参考了 MMA7660 器件，它是一种可以满足多种应用需求的典型序列。对于其他具有特殊要求的时序，需要稍微调整驱动程序，以适合新的应用。请参阅图 3 主机写入序列和图 4 主机读取序列。

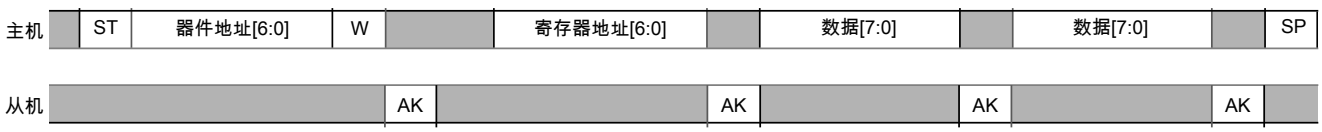


图 3. 写入序列

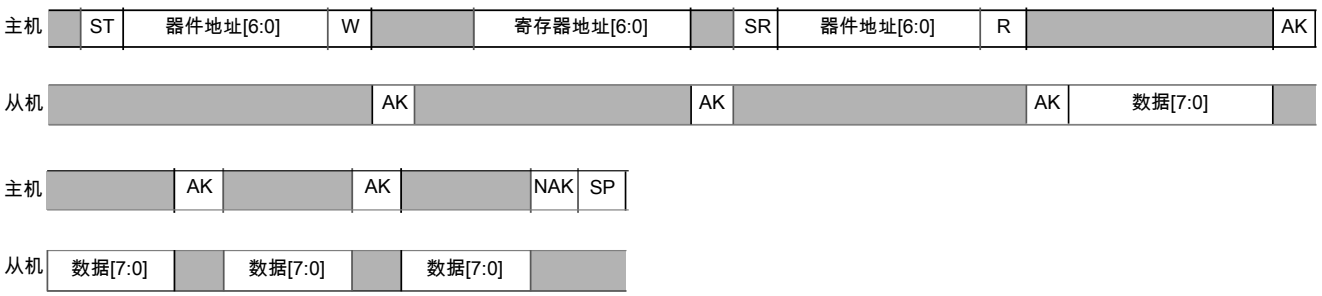


图 4. 读取序列

4.2 典型时序

在某些 Kinetis 器件上，如果 I2C 模块处于从机状态，当其接收到主机发送的 STOP 信号时，它将无法产生中断。因此，驱动程序无法知道何时将数据传输至应用任务。为了解决此问题，需要占用一条命令。当寄存器地址为 0xFF 时，从机将假定其为 STOP 信号并将数据传输至任务。

5 主要函数和宏说明

下表列出了相关的函数和宏，及其相关说明：

函数/宏名称	说明
_ki2c_isr	I2C 中断入口函数
_isr_ii2c_slave	从机中断服务例程:
_isr_ii2c_master	主机中断服务例程
_ki2c_int_fb_tx	应用接口，用于发送数据，已映射至 **fwrite**。
_fb_tx_master	对于主机，发送数据

下一页继续介绍此表...

函数/宏名称	说明
_fb_rx_master	对于主机，接收数据
_ki2c_int_fb_rx	应用接口，用于接收数据，已映射至 **fread**。
_fb_tx_slave	对于从机，发送数据
_fb_rx_slave	对于从机，接收数据
#define DEBUG_ENABLE	该宏可以输出信息以便调试。对于正常用途，它将被屏蔽。

6 使用此驱动程序的演示代码

以下子章节介绍了在主机和从机模式下使用 I2C 驱动程序的演示代码。

6.1 主机演示代码

```
struct STRU_I2C_BUFFER
{
    char dst_addr;
    char reg_addr;
    char data[100]; // Here is the buffer for sending or
                // receiving data
};

struct STRU_I2C_BUFFER i2c_buf_rx;
struct STRU_I2C_BUFFER i2c_buf_tx;
int_32 i2c_fb_test_master(void)
{
    uint_32 param;
    int i;
    int len;

    file_iic0 = fopen("ii2c0fb:", NULL);
    if (file_iic0 == NULL)
    {
        printf("\nOpen the IIC0 driver failed!!!\n");
        return IIC_ERR;
    }

    param = 100000;
    ioctl(file_iic0, IO_IOCTL_I2C_SET_BAUD, &param);

    i2c_buf_tx.dst_addr = 0x50; // The I2C slave address
    i2c_buf_tx.reg_addr = 0;    // You may view this as a register
                                // address or a command to slave.
    i2c_buf_rx.dst_addr = 0x50; // The same as above, for receiving

    i2c_buf_rx.reg_addr = 0;    // The same as above, for receiving


    for(i=0;i<8;i++)            // initialize data
        i2c_buf_tx.data[i] = 0xb0+i;

    while(1)
    {
        printf("---------------------\n");
        // send 4 bytes to slave
        len = fwrite(&i2c_buf_tx, 1, 4, file_iic0);
        if( len < 4 )
```

```
            printf("send failed, len = %d \n", len);
        else
            printf("send ok, len = %d \n", len);

        // clear receiving buffer
        memset(i2c_buf_rx.data, 0, 4);
        // receive 4 bytes from slave
        len = fread(&i2c_buf_rx, 1, 4, file_iic0);
        printf("get i2c data, len = %x\n", len);
        for(i=0; i<len; i++)
            printf("%x \n", i2c_buf_rx.data[i]);
        _time_delay(10);
    }
}
```

图 5 显示此演示的运行结果。

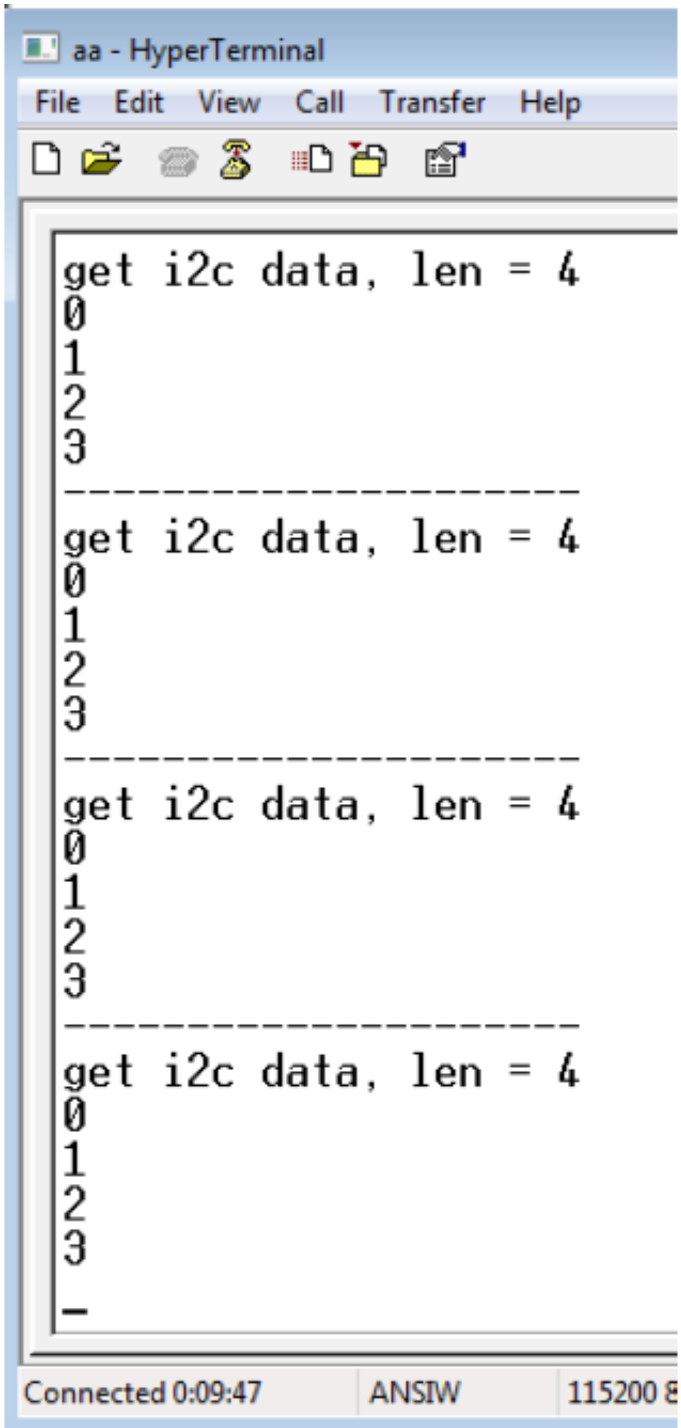


图 5. 用于主机读取的控制台输出

6.2 从机演示代码

在此演示代码下，将会创建两个任务，以单独进行发送和接收。但是，这并不是必须的；用户也可以在相同任务中进行发送和接收。

注

由于驱动程序与任务共用同一缓冲区，请确保在驱动程序工作时数据不得由其他任务或中断修改。

```
#define I2C0_SLAVE_ADDRESS 0x50
char buf_i2c_rx[256];
char buf_i2c_tx[256];

void task_slave_rx(uint_32 initial_data)
{
    uint_32 param;
    uint_32 len;
    int i;

    printf("I2C slave demo for FB. ******************\n");
    file_iic0 = fopen("ii2c0fb:", NULL);

    // Set to slave mode with specified slave address
    param = I2C0_SLAVE_ADDRESS;
    ioctl(file_iic0, IO_IOCTL_I2C_SET_SLAVE_MODE, &param);

    _task_create(0, TASK_I2C_SLAVE_TX, 0);

    while(1)
    {
        // clearing and reading
        memset(buf_i2c_rx, 0, 256);
        len = fread(buf_i2c_rx, 1, 256, file_iic0);
        if(len > 0)
        {
            // show data received
            printf("get i2c data, len: %d\n", len);
            for(i=0;i<len;i++)
                printf("%x\n", buf_i2c_rx[i]);
        }
        else
            printf("read failed.\n");
     }
}

void task_slave_tx(uint_32 initial_data)
{
    int i;
    int len;
    for(i=0;i<256;i++)
        buf_i2c_tx[i] = i;

    while(1)
    {
        len = fwrite(buf_i2c_tx, 1, 4, file_iic0);
        if(len < 4)
            printf("send fail. len = %d \n", len);
        else
            printf("send OK. len = %d \n", len);
    }
}
```

图 6 显示此演示的运行结果。

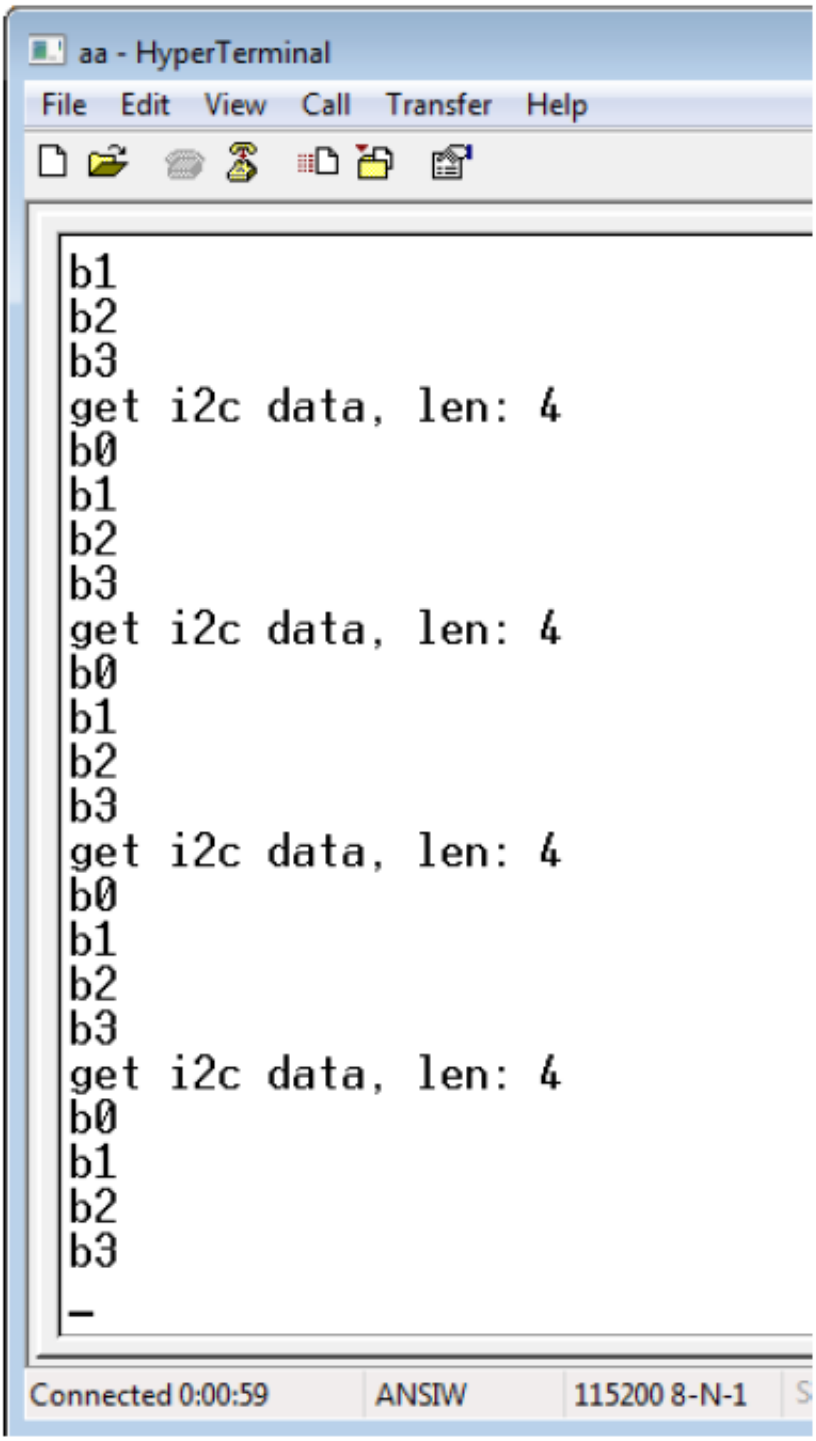


图 6. 用于从机读取的控制台输出

7 驱动程序安装

要安装此驱动程序，请遵照以下步骤：

1. 请将随本应用笔记一起发布的文件 **i2c_int_k_fb.c**（包含在 **freescale.com** 上的 AN4655SW.zip 中）复制到文件夹 <mqx_install>\mqx\source\io\i2c\int 中，并将其添加到 BSP 工程。请参见下图。

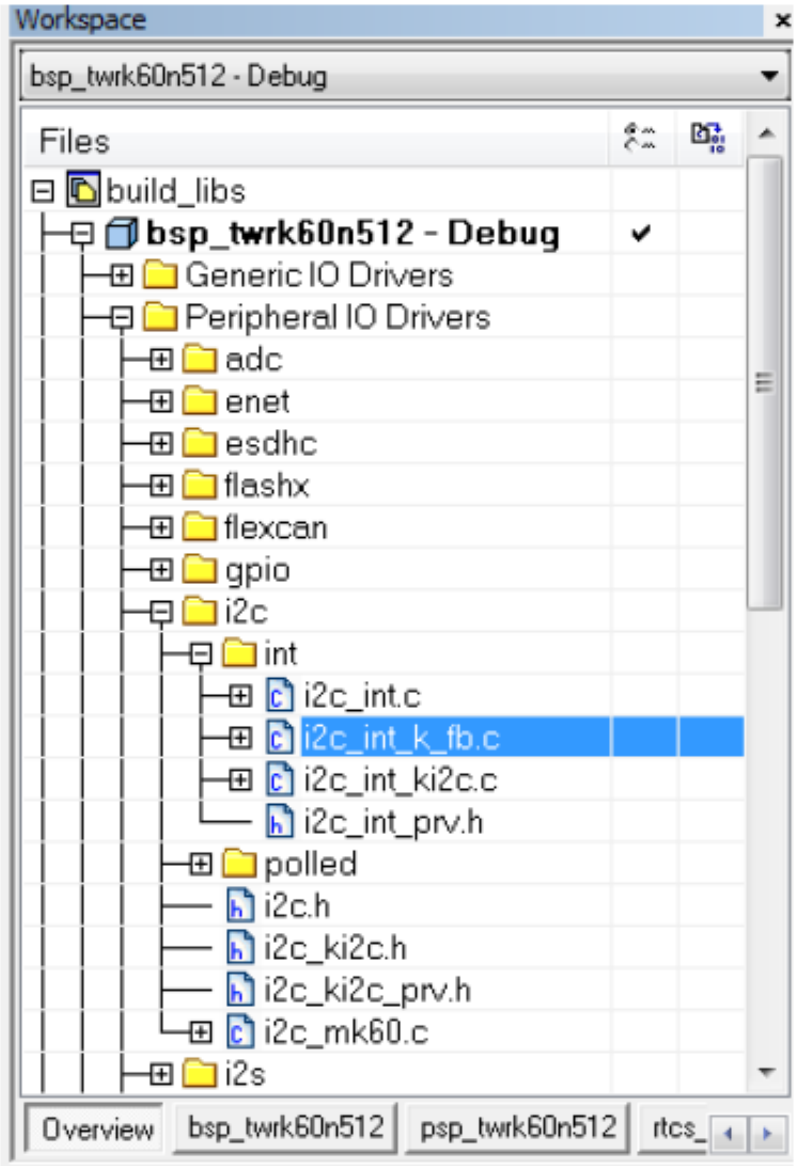


图 7. 将 i2c_int_k_fb.c 添加到 BSP 工程

2. 将以下行添加到 **i2c_ki2c.h**:

```
extern uint_32 _ki2c_int_fb_install (char_ptr, KI2C_INIT_STRUCT_CPTR);
```

3. 在<mqx_install>\mqx\source\bsp\twrk60n512\init_bsp.c 中，添加以下行，与本应用笔记关联的软件 AN4652SW.zip 中有一个示例文件。

```
#if BSPCFG_ENABLE_II2C0_FB
   _ki2c_int_fb_install("ii2c0fb:", &_bsp_i2c0_init);
#endif
#if BSPCFG_ENABLE_II2C1_FB
   _ki2c_int_fb_install("ii2c1fb:", &_bsp_i2c1_init);
#endif
```

4. 在<mqx_install>\config\twrk60n512\user_config.h 中，添加以下行:

```
#define BSPCFG_ENABLE_II2C0_FB   1
#define BSPCFG_ENABLE_II2C1_FB   0
```

5. 然后对其进行编译，驱动程序将在应用工程中可用。

8 结论

本应用笔记介绍了基于中断和阻塞机制的 MQX I2C 驱动程序。首先，介绍其机制，然后介绍了程序流程和时序。为方便读者理解，还介绍了主要函数说明、演示代码和安装步骤。

9 参考

可从 **freescale.com** 获取以下参考文档。

- K60P120M100SF2RM：K60 参考手册
- AN3902：如何开发 MQX I/O 驱动器

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number AN4652
Revision 0, 01/2013