



Manoj Kumar Santha Mohan

摘要

本应用手册描述了 C2000™ I2C 如何使用轮询方式 (或中断方式) 与 EEPROM 进行通信。本文档实现了不同的 EEPROM 写入协议, 如字节写入、字写入、分页写入操作和不同的 EEPROM 读取操作, 如字节读取、字读取和分页读取操作。

本示例使用的 EEPROM 为 AT24C256 (写入周期时间为 6ms, 分页操作为 64 字节)。

I2C EEPROM 示例代码 位于以下路径中 :

轮询方式示例 : <C2000Ware>\driverlib\<device>\examples\c28xi2cli2c_ex4_eeprom_polling

中断方式示例 : <C2000Ware>\driverlib\<device>\examples\c28xi2cli2c_ex6_eeprom_interrupt

内容

| | |
|--------------------------------|----|
| 1 引言..... | 2 |
| 2 硬件连接..... | 2 |
| 3 C2000 I2C 源代码..... | 3 |
| 3.1 I2CHandle 说明..... | 3 |
| 3.2 I2CBusScan..... | 3 |
| 3.3 I2C_MasterTransmitter..... | 4 |
| 3.4 I2C_MasterReceiver..... | 4 |
| 4 EEPROM 字节写入..... | 5 |
| 5 EEPROM 字节读取..... | 6 |
| 6 EEPROM 字写入..... | 7 |
| 7 EEPROM 字读取..... | 8 |
| 8 EEPROM 分页写入..... | 9 |
| 9 EEPROM 分页读取..... | 10 |

插图清单

| | |
|---------------------------|----|
| 图 4-1. EEPROM 字节写入命令..... | 5 |
| 图 5-1. EEPROM 字节读取命令..... | 6 |
| 图 6-1. EEPROM 字写入命令..... | 7 |
| 图 7-1. EEPROM 字读取命令..... | 8 |
| 图 8-1. EEPROM 分页写入命令..... | 9 |
| 图 9-1. EEPROM 分页读取命令..... | 10 |

表格清单

| | |
|----------------------------------------------------|---|
| 表 3-1. 工程文件名和描述..... | 3 |
| 表 3-2. i2c_ex6_eeprom_interrupt 示例中使用的 I2C 中断..... | 3 |
| 表 3-3. I2CHandle 的详细信息..... | 3 |

商标

C2000™ is a trademark of Texas Instruments.

所有商标均为其各自所有者的财产。

1 引言

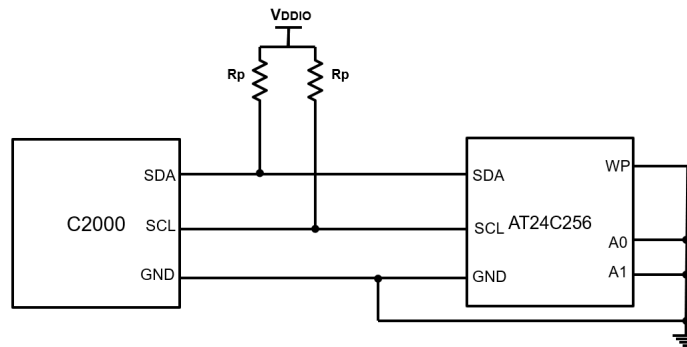
本应用手册中使用的 C28x-I2C 模块具有以下特性：

- 符合 NXP 半导体 I2C 总线规范 (2.1 版) :
 - 支持 8 位格式传输
 - 7 位和 10 位寻址模式
 - 常规调用
 - START 字节模式
 - 支持多个主发送器和从接收器
 - 支持多个从发送器和主接收器
 - 组合主器件发送/接收和接收/发送模式
 - 数据传输速率从 10kbps 到 400kbps (快速模式)
- 接收 FIFO 和发送器 FIFO (16 深 x 8 位 FIFO)
- 支持两个 ePIE 中断 :
 - I2Cx 中断 - 以下任何事件都可以配置为生成 I2Cx 中断 :
 - 发送数据准备好
 - 接收数据准备好
 - 寄存器访问准备好
 - 没有接收到确认
 - 仲裁丢失
 - 检测到停止条件
 - 被寻址为从器件
 - I2Cx_FIFO 中断 :
 - 发送 FIFO 中断
 - 接收 FIFO 中断
- 模块启用/禁用能力
- 自由数据格式模式

2 硬件连接

下面的原理图显示了如何将 EEPROM 器件连接到 C2000 I2C 模块。本应用报告中使用的 EEPROM 为 AT24C256。在 AT24C256 中，称为器件地址引脚的用户可配置引脚 (A0、A1) 可用于对同一 I2C 总线上多达四个的 AT24C256 器件寻址。这些 A0、A1 引脚被下拉，使得 EEPROM 的从器件地址 = 0x50。写保护输入引脚需要接地，以允许 EEPROM 写操作。

有关选择上拉电阻的信息，请参阅 [I2C 总线上拉电阻计算](#)。



3 C2000 I2C 源代码

表 3-1 中提供的 C2000Ware 软件示例显示了如何使用 I2C 模块通过 I2C 总线与 EEPROM 通信。本示例是为 EEPROM AT24C256 开发的，后者需要 2 个字节为 EEPROM 存储器（从器件地址为 0x50）寻址。表 3-2 显示了基于 EEPROM 中断的示例中使用的 I2C 中断。

表 3-1. 工程文件名和描述

| 源代码 | 说明 |
|--------------------------------|-----------------------------------------|
| i2c_ex4_eeprom_polling.c | 该程序将展示如何使用 I2C 轮询方式执行不同的 EEPROM 写入和读取命令 |
| i2cLib_FIFO_polling.c | 使用轮询的 FIFO 的 C28x-I2C 库源文件 |
| i2cLib_FIFO_polling.h | 使用轮询的 FIFO 的 C28x-I2C 库头文件 |
| i2c_ex6_eeprom_interrupt.c | 该程序将展示如何使用 I2C 中断方式执行不同的 EEPROM 写入和读取命令 |
| i2cLib_FIFO_master_interrupt.c | 用于 FIFO 中断的 C28x-I2C 库源文件 |
| i2cLib_FIFO_master_interrupt.h | 用于 FIFO 中断的 C28x-I2C 库头文件 |

表 3-2. i2c_ex6_eeprom_interrupt 示例中使用的 I2C 中断

| | |
|---------|------------|
| 停止条件 | 寄存器访问就绪 |
| 被寻址为从器件 | TX FIFO 中断 |
| 仲裁失败 | RX FIFO 中断 |
| NACK 条件 | |

3.1 I2CHandle 说明

表 3-3 提供了示例代码中定义的 I2CHandle 结构的详细信息。

表 3-3. I2CHandle 的详细信息

| 结构成员 | 说明 |
|----------------------|----------------------------------------------------|
| base | 使用的 I2C 模块的基址 |
| SlaveAddr | EEPROM 的从器件地址 |
| pControlAddr | 指向存储 EEPROM 地址的变量的指针 |
| NumOfAddrBytes | EEPROM 所需的地址字节数 |
| pTX_MsgBuffer | 指向 TX 消息缓冲区的指针 |
| pRX_MsgBuffer | 指向 RX 消息缓冲区的指针 |
| NumOfDataBytes | I2C 事务中的数据字节数 |
| currentHandlePtr | 指向 I2CHandle 对象的指针 |
| numofSixteenByte | 数据包中的 16 字节数 注意：FIFO 深度为 16。因此，我们需要计算 16 字节的数量。 |
| remainingbytes | 小于 16 字节的字节数 |
| WriteCycleTime_in_us | 基于 EEPROM 定义的写入周期时间（以 us 为单位） |

3.2 I2CBusScan

函数名称：I2CBusScan(I2CA_BASE, pAvailableI2C_slaves)

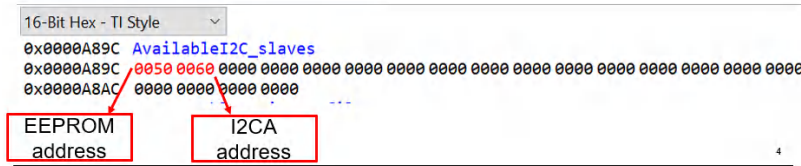
说明：此函数可用于扫描所有连接到 I2C 总线的 I2C 器件，方法是发送不同的从器件地址并检查 ACK/NACK 条件

参数：

base：base 是使用的 I2C 实例的基地址

pAvailableI2C_slaves : 存储 I2C 总线上可用 I2C 地址的数组地址

返回 : I2C 事务的状态



```

16-Bit Hex - TI Style
0x0000A89C AvailableI2C_slaves
0x0000A89C 0050 0060 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0x0000A8AC 0000 0000 0000 0000
EEPROM address I2CA address
  
```

3.3 I2C_MasterTransmitter

函数名称 : I2C_MasterTransmitter(I2CA_BASE, struct I2CHandle *pl2C_Params)

说明 : 此函数可用于发送在 pl2C_Params 中定义的“N”个字节。此函数可用于执行以下 EEPROM 写入操作。

1. EEPROM 字节写入

设置 EEPROM.NumOfDataBytes = 1, 因为需要发送一个字节

2. EEPROM 字写入

设置 EEPROM.NumOfDataBytes = 2, 因为需要发送两个字节

3. EEPROM 分页写入

设置 EEPROM.NumOfDataBytes = “N”, 因为需要发送“N”个字节 - 页面大小取决于选择的 EEPROM

以下设置适用于上述所有 EEPROM 写入操作

设置 EEPROM.NumOfAddrBytes = 2, 因为需要发送高地址字节和低地址字节

参数 :

base : base 是使用的 I2C 实例的基地址

pl2C_Params : 指向从器件的 I2C 地址句柄的指针

返回 : I2C 事务的状态

3.4 I2C_MasterReceiver

函数名称 : I2C_MasterReceiver(I2CA_BASE, struct I2CHandle *pl2C_Params)

说明 : 此函数可用于接收在 pl2C_Params 中定义的“N”个字节。此函数还可用于执行以下 EEPROM 读取操作。

1. EEPROM 字节读取

设置要读取的字节数 (EEPROM.NumOfDataBytes = 1)

2. EEPROM 字读取

设置要读取的字节数 (EEPROM.NumOfDataBytes = 2) - 两个字节组成一个字

3. EEPROM 分页读取

设置要读取的字节数 (EEPROM.NumOfDataBytes = “N”)。分页读取没有页面大小限制。

以下设置适用于上述所有 EEPROM 读取操作

设置 EEPROM.NumOfAddrBytes = 2, 因为需要发送高地址字节和低地址字节

参数 :

base : base 是使用的 I2C 实例的基地址

pl2C_Params : 指向从器件的 I2C 地址句柄的指针

返回 : I2C 事务的状态

4 EEPROM 字节写入

图 4-1 显示了 AT24C256 中是如何定义 EEPROM 字节写入协议的。C2000 I2C 在主发送器模式 (I2CMDR.MST = 1 , I2CMDR.TRX = 1) 中进行配置，以发送 EEPROM 地址 (高地址字节，低地址字节)，后跟数据字节。



图 4-1. EEPROM 字节写入命令

代码流程：

1. 启动条件 + 发送从器件地址 (0x50) + 写入位 + ACK 位 (来自从器件)
2. 发送 EEPROM 高地址字节 + ACK 位 (来自从器件)
3. 发送 EEPROM 低地址字节 + ACK 位 (来自从器件)
4. 发送数据字节 + ACK 位 (来自从器件)
5. 生成停止条件

```
EEPROM.SlaveAddr = 0x50;
EEPROM.base = I2CA_BASE;
EEPROM.pControlAddr = &ControlAddr;
EEPROM.NumOfAddrBytes = 2;
EEPROM.pTX_MsgBuffer = TX_MsgBuffer;
EEPROM.pRX_MsgBuffer = RX_MsgBuffer;
EEPROM.NumOfAttempts = 5;
EEPROM.Delay_us = 10;
EEPROM.WriteCycleTime_in_us = 6000;
```

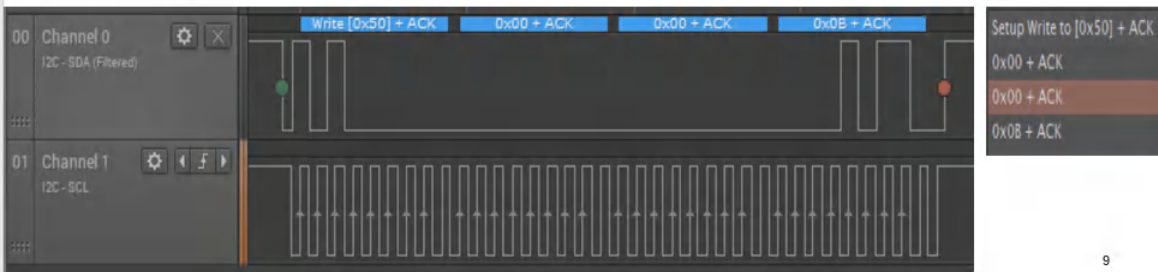
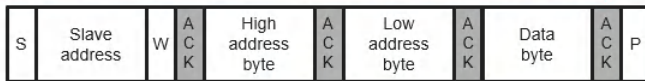
```
//Example 1: EEPROM Byte Write
//Write 11 to EEPROM address 0x0
```

```
ControlAddr = 0;
EEPROM.NumOfDataBytes = 1;
TX_MsgBuffer[0] = 11;
status = I2C_MasterTransmitter(&EEPROM);
```

Depends upon EEPROM used. Refer EEPROM datasheet

```
//Wait for EEPROM write cycle time
//This delay is not mandatory. User can run their application code
//It is however important to wait for EEPROM write cycle time before
//another read / write transaction
```

```
DEVICE_DELAY_US(EEPROM.WriteCycleTime_in_us);
```



5 EEPROM 字节读取

图 5-1 显示了 AT24C256 中是如何定义 EEPROM 字节读取协议的。C2000 I2C 在主发送器模式 (I2CMDR.MST = 1, I2CMDR.TRX = 1) 中进行配置, 以发送 EEPROM 地址 (高地址字节, 低地址字节), 随后在主接收器模式 (I2CMDR.MST = 1, I2CMDR.TRX = 0) 中生成重复启动条件, 以从 EEPROM 接收数据字节。

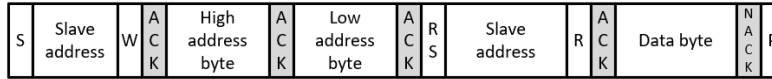


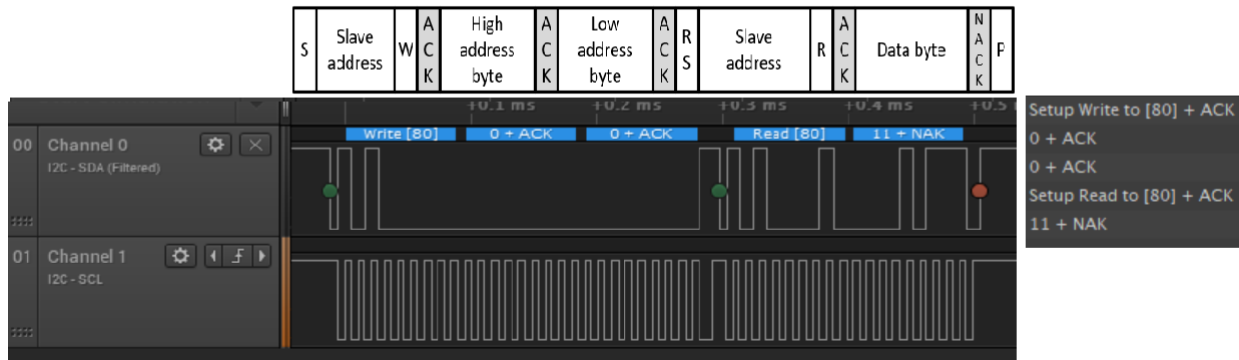
图 5-1. EEPROM 字节读取命令

代码流程:

1. 启动条件 + 发送从器件地址 (0x50) + 写入位 + ACK 位 (来自从器件)
2. 发送 EEPROM 高地址字节 + ACK 位 (来自从器件)
3. 发送 EEPROM 低地址字节 + ACK 位 (来自从器件)
4. 重复启动条件 + 发送从器件地址 (0x50) + 读取位 + ACK 位 (来自从器件)
5. 接收数据字节 + ACK 位 (来自从器件)
6. 生成停止条件

```
//Example 2: EEPROM Byte Read
//Make sure 11 is written to EEPROM address 0x0
ControlAddr = 0;
EEPROM.pControlAddr = &ControlAddr;
EEPROM.NumOfDataBytes = 1;
status = I2C_MasterReceiver(&EEPROM);

while(I2C_getStatus(EEPROM.base) & I2C_STS_BUS_BUSY);
```



6 EEPROM 字写入

图 6-1 显示了 AT24C256 中是如何定义 EEPROM 字写入协议的。C2000 I2C 在主发送器模式 (I2CMDR.MST = 1, I2CMDR.TRX = 1) 中进行配置, 以发送 EEPROM 地址 (高地址字节, 低地址字节), 后跟两个数据字节。

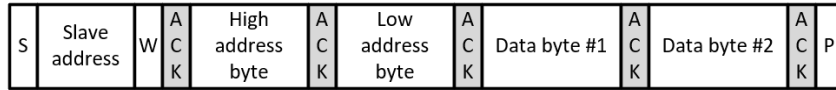


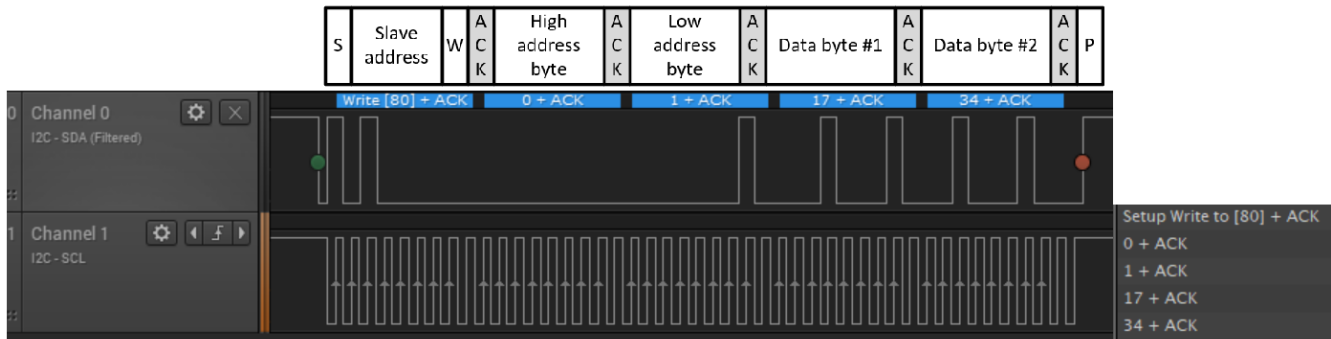
图 6-1. EEPROM 字写入命令

代码流程 :

1. 启动条件 + 发送从器件地址 (0x50) + 写入位 + ACK 位 (来自从器件)
2. 发送 EEPROM 高地址字节 + ACK 位 (来自从器件)
3. 发送 EEPROM 低地址字节 + ACK 位 (来自从器件)
4. 发送数据字节 # 1 + ACK 位 (来自从器件)
5. 发送数据字节 # 2 + ACK 位 (来自从器件)
6. 生成停止条件

```
//Example 3: EEPROM word (16-bit) write
//EEPROM address 0x1 = 22 & 0x2 = 33
ControlAddr = 1; //EEPROM address to write
EEPROM.NumOfDataBytes = 2;
TX_MsgBuffer[0] = 0x11;
TX_MsgBuffer[1] = 0x22;
EEPROM.pTX_MsgBuffer = TX_MsgBuffer;
status = I2C_MasterTransmitter(&EEPROM);

//Wait for EEPROM write cycle time
//This delay is not mandatory. User can run their application code instead.
//It is however important to wait for EEPROM write cycle time before you initiate
//another read / write transaction
DEVICE_DELAY_US(EEPROM.WriteCycleTime_in_us);
```



7 EEPROM 字读取

图 7-1 显示了 AT24C256 中是如何定义 EEPROM 字读取协议的。C2000 I2C 在主发送器模式 (I2CMDR.MST = 1, I2CMDR.TRX = 1) 中进行配置, 以发送 EEPROM 地址 (高地址字节, 低地址字节), 随后在主接收器模式 (I2CMDR.MST = 1, I2CMDR.TRX = 0) 中生成重复启动条件, 以从 EEPROM 接收两个数据字节。

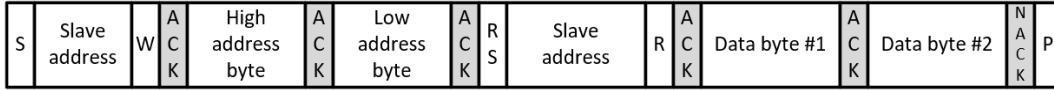


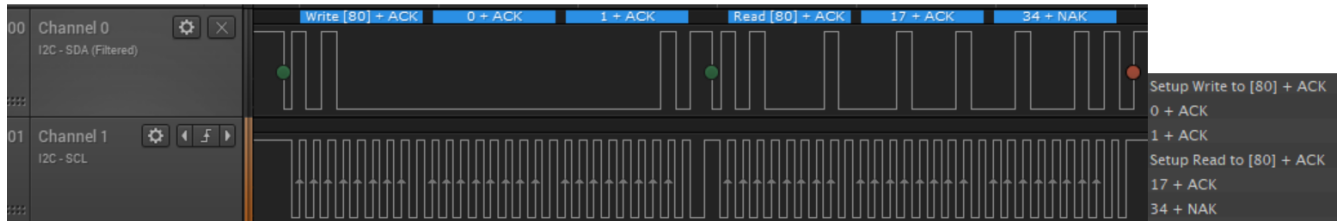
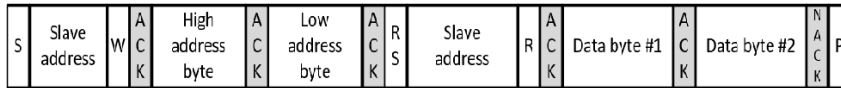
图 7-1. EEPROM 字读取命令

代码流程 :

1. 启动条件 + 发送从器件地址 (0x50) + 写入位 + ACK 位 (来自从器件)
2. 发送 EEPROM 高地址字节 + ACK 位 (来自从器件)
3. 发送 EEPROM 低地址字节 + ACK 位 (来自从器件)
4. 生成重复启动条件 + 发送从器件地址 (0x50) + 读取位 + ACK 位 (来自从器件)
5. 接收数据字节 # 1 + ACK 位 (来自从器件)
6. 接收数据字节 # 2 + NACK 位 (来自从器件)
7. 生成停止条件

```
//Example 4: EEPROM word (16-bit) read
//Make sure EEPROM address 1 has 0x11 and 2 has 0x22
ControlAddr = 1;
EEPROM.pControlAddr = &ControlAddr;
EEPROM.pRX_MsgBuffer = RX_MsgBuffer;
EEPROM.NumOfDataBytes = 2;

status = I2C_MasterReceiver(&EEPROM);
```



8 EEPROM 分页写入

图 8-1 显示了 AT24C256 中是如何定义 EEPROM 分页写入协议的。C2000 I2C 在主发送器模式 (I2CMDR.MST = 1, I2CMDR.TRX = 1) 中进行配置，以发送 EEPROM 地址 (高地址字节，低地址字节)，后跟 “N” 数据字节。

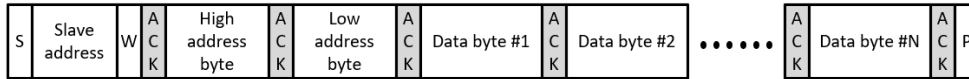


图 8-1. EEPROM 分页写入命令

代码流程：

1. 启动条件 + 发送从器件地址 (0x50) + 写入位 + ACK 位 (来自从器件)
2. 发送 EEPROM 高地址字节 + ACK 位 (来自从器件)
3. 发送 EEPROM 低地址字节 + ACK 位 (来自从器件)
4. 发送数据字节 # 1 + ACK 位 (来自从器件)
5. 发送数据字节 # 2 + ACK 位 (来自从器件)

ooo (发送更多字节)

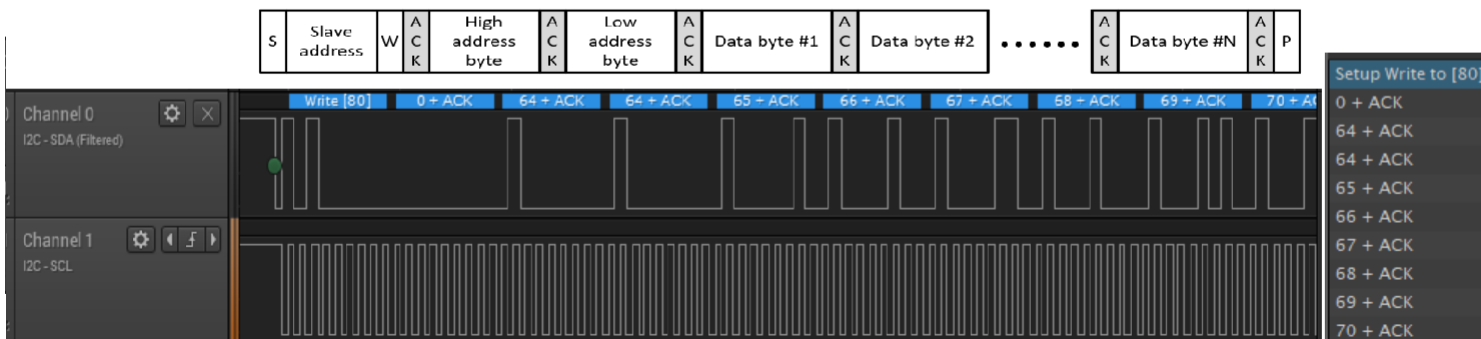
6. 发送数据字节 # N + ACK 位 (来自从器件)
7. 生成停止条件

```
//Example 5: EEPROM Page write
//Program address = data pattern from address 64
```

```
for(i=0;i<MAX_BUFFER_SIZE;i++)
{
    TX_MsgBuffer[i] = i+64;
}
```

```
ControlAddr = 64; //EEPROM address to write
EEPROM.NumOfDataBytes = MAX_BUFFER_SIZE;
EEPROM.pTX_MsgBuffer = TX_MsgBuffer;
status = I2C_MasterTransmitter(&EEPROM);
```

```
//Wait for EEPROM write cycle time
//This delay is not mandatory. User can run their application code instead.
//It is however important to wait for EEPROM write cycle time before you initiate
//another read / write transaction
DEVICE_DELAY_US(EEPROM.WriteCycleTime_in_us);
```



9 EEPROM 分页读取

图 9-1 显示了 AT24C256 中是如何定义 EEPROM 分页读取协议的。C2000 I2C 在主发送器模式 (I2CMDR.MST = 1, I2CMDR.TRX = 1) 中进行配置, 以发送 EEPROM 地址 (高地址字节, 低地址字节), 随后在主接收器模式 (I2CMDR.MST = 1, I2CMDR.TRX = 0) 中生成重复启动条件, 以从 EEPROM 接收 “N” 数据字节。

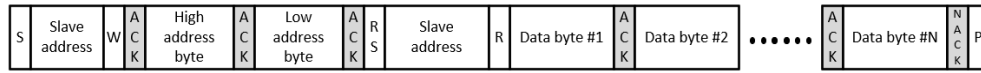


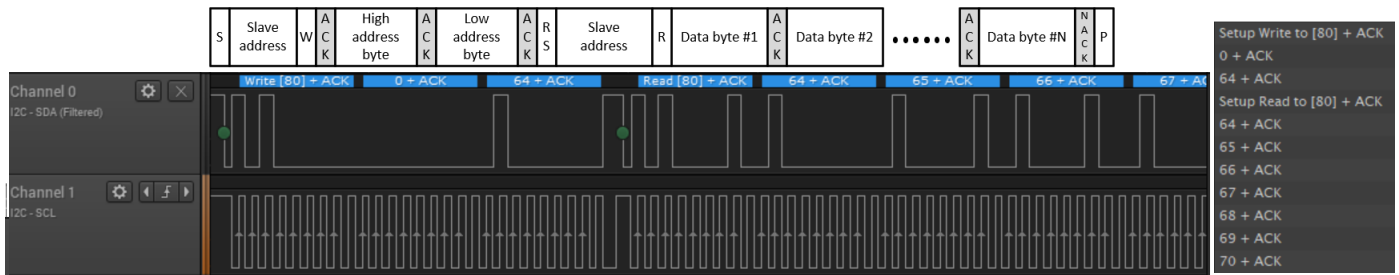
图 9-1. EEPROM 分页读取命令

代码流程：

1. 启动条件 + 发送从器件地址 (0x50) + 写入位 + ACK 位 (来自从器件)
2. 发送 EEPROM 高地址字节 + ACK 位 (来自从器件)
3. 发送 EEPROM 低地址字节 + ACK 位 (来自从器件)
4. 生成重复启动条件 + 发送从器件地址 (0x50) + 读取位 + ACK 位 (来自从器件)
5. 接收数据字节 # 1 + ACK 位 (来自从器件)
6. 接收数据字节 # 2 + NACK 位 (来自从器件)
- o o o (接收更多字节)
7. 接收数据字节 # N + ACK 位 (来自从器件)
8. 生成停止条件

```
//Example 6: EEPROM word Paged read
ControlAddr = 64;
EEPROM.pControlAddr = &ControlAddr;
EEPROM.pRX_MsgBuffer = RX_MsgBuffer;
EEPROM.NumOfDataBytes = MAX_BUFFER_SIZE;

status = I2C_MasterReceiver(&EEPROM);
```



重要声明和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做任何明示或暗示的担保，包括但不限于对适销性、某特定用途方面的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他功能安全、信息安全、监管或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的应用。严禁对这些资源进行其他复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 [TI 的销售条款](#) 或 [ti.com](#) 上其他适用条款/TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265

Copyright © 2022，德州仪器 (TI) 公司