

UM2001A TWI 使用注意事项

版本: V1.1



广芯微电子（广州）股份有限公司

<http://www.unicmicro.com/>

条款协议

本文档的所有部分，其著作权归广芯微电子（广州）股份有限公司（以下简称广芯微电子）所有，未经广芯微电子授权许可，任何个人及组织不得复制、转载、仿制本文档的全部或部分组件。本文档没有任何形式的担保、立场表达或其他暗示，若有任何因本文档或其中提及的产品所有资讯所引起的直接或间接损失，广芯微电子及所属员工恕不为其担保任何责任。除此以外，本文档所提到的产品规格及资讯仅供参考，内容亦会随时更新，恕不另行通知。

1. 本文档中所记载的关于电路、软件和其他相关信息仅用于说明半导体产品的操作和应用实例。
用户如在设备设计中应用本文档中的电路、软件和相关信息，请自行负责。对于用户或第三方因使用上述电路、软件或信息而遭受的任何损失，广芯微电子不承担任何责任。
2. 在准备本文档所记载的信息的过程中，广芯微电子已尽量做到合理注意，但是，广芯微电子并不保证这些信息都是准确无误的。用户因本文档中所记载的信息的错误或遗漏而遭受的任何损失，广芯微电子不承担任何责任。
3. 对于因使用本文档中的广芯微电子产品或技术信息而造成的侵权行为或因此而侵犯第三方的专利、版权或其他知识产权的行为，广芯微电子不承担任何责任。本文档所记载的内容不应视为对广芯微电子或其他人所有的专利、版权或其他知识产权作出任何明示、默示或其它方式的许可及授权。
4. 使用本文档中记载的广芯微电子产品时，应在广芯微电子指定的范围内，特别是在最大额定值、电源工作电压范围、热辐射特性、安装条件以及其他产品特性的范围内使用。对于在上述指定范围之外使用广芯微电子产品而产生的故障或损失，广芯微电子不承担任何责任。
5. 虽然广芯微电子一直致力于提高广芯微电子产品的质量和可靠性，但是，半导体产品有其自身的具体特性，如一定的故障发生率以及在某些使用条件下会发生故障等。此外，广芯微电子产品均未进行防辐射设计。所以请采取安全保护措施，以避免当广芯微电子产品在发生故障而造成火灾时导致人身事故、伤害或损害的事故。例如进行软硬件安全设计（包括但不限于冗余设计、防火控制以及故障预防等）、适当的老化处理或其他适当的措施等。

目录

1	摘要.....	3
2	TWI 操作说明.....	3
2.1	TWI 窗口时间.....	3
2.1.1	OTP 未烧录数据.....	3
2.1.2	OTP 已烧录数据.....	3
2.2	操作 Reg3F 命令说明	4
3	TWI 打开操作流程.....	4
3.1	唤醒（上电）	4
3.2	进入 TWI 模式.....	4
3.3	测试 TWI.....	5
4	参考示例	5
4.1	定义	5
4.2	TWI 读写字节.....	5
4.3	TWI 操作.....	7
5	总结.....	9
6	版本修订	10

1 摘要

本篇应用笔记介绍 TWI 使用注意事项及配置流程。TWI (Two-Wire Interface) 作为 UM2001A 的串行通信接口，在 UM2001A 芯片中有着特定的使用规范和操作流程。确保芯片与外部设备之间稳定、高效的通信至关重要。

本篇应用笔记主要包括：

- TWI操作说明
- TWI打开操作流程
- 参考示例
- 总结

2 TWI 操作说明

2.1 TWI 窗口时间

TWI 窗口时间有 2 种情况，分别为 OTP 未烧录数据和 OTP 已烧录数据。本应用中的 OTP 未烧录数据和 OTP 已烧录数据，仅考虑寄存器 Reg1A/Reg1B/Reg1C/Reg1D 是否已被烧录数据。

2.1.1 OTP 未烧录数据

UM2001A 芯片上电或被唤醒时，芯片会打开 TWI 窗口并一直处于 TWI 窗口时间。

2.1.2 OTP 已烧录数据

UM2001A 芯片上电或被唤醒时，芯片会打开 TWI 窗口，窗口时间由固定时间+寄存器 Reg09[7:6] 配置，固定时间为 125 μ s。

相关寄存器如下：

Reg	Bit	Name	Type	Description
0x09	7:6	starup_wait_time	W/R	00: 0 ms 01: 5 ms 10: 10 ms 11: 15 ms

注：在实现 TWI 接口操作时，建议考虑 TWI 窗口的最小时间，即配置 Reg[7:6]=00b’时，TWI 窗口打开总时间为 125μs。此时，接口能够兼容所有 TWI 窗口时间完成芯片进入 TWI 配置模式。

2.2 操作 Reg3F 命令说明

在读写 Reg3F 命令寄存器时，需要在操作 Reg3F 前需要先复位 TWI，即发送连续 50 个 TWI CLK 时钟，TWI Data 引脚保持低电平后再对 Reg3F 进行操作

3 TWI 打开操作流程

TWI 打开操作流程分为 3 步，分别为唤醒（上电）、进入 TWI 模式、测试 TWI。

3.1 唤醒（上电）

通过拉低 TWI 的 CLK 引脚唤醒芯片，低电平时间需要保持至少 50μs 时间后拉高。芯片唤醒后进入 TWI 窗口时间。

3.2 进入 TWI 模式

芯片唤醒后，连续发送 2ms 的 TWI 打开时序，即 TWI 的 Data 引脚保持低电平，TWI 的 CLK 发送时钟。时钟发送完成后，读取寄存器值 Reg00，确保芯片停留在 TWI 配置模式。

注：若芯片已烧录 OTP 数据，建议 TWI 的时钟在 400kHz 以上，确保在最短的 TWI 窗口时间（125μs）内能完成相应的操作。若芯片未烧录 OTP 数据，唤醒后将一直处于 TWI 窗口时间内，则

TWI 的时钟频率不受上述限制。

3.3 测试 TWI

测试 TWI 接口是否可以正常读取寄存器值，需要先进行 TWI 复位，即发送 50 个 TWI CLK 时钟，TWI 的 DATA 引脚保持低电平。然后读取寄存器 Reg3F，判断寄存器 Reg3F 的值低四位是否为 0x07，若是则说明芯片已进入 TWI 配置状态，可以正常配置芯片。

4 参考示例

4.1 定义

```
#define TWI_CLK_DIR_OUTPUT      ** /* 设置为输出 */
#define TWI_CLK_LEVEL_LOW      ** /* 输出低电平 */
#define TWI_CLK_LEVEL_HIGH     ** /* 输出高电平 */
#define TWI_DATA_DIR_INPUT     ** /* 设置为输入 */
#define TWI_DATA_DIR_OUTPUT    ** /* 设置为输出 */
#define TWI_DATA_LEVEL_LOW     ** /* 输出低电平 */
#define TWI_DATA_LEVEL_HIGH    ** /* 输出高电平 */
#define TWI_DATA_LEVEL_READ    ** /* 读取电平 */
```

4.2 TWI 读写字节

```
/******
 * Function      : twi_write_byte
 * Description   : TWI 写一个字节
 * Input        : uint8_t byte
 * Output       : none
 * Return       : none
 *****/

void twi_write_byte(uint8_t byte)
{
    TWI_DATA_DIR_OUTPUT;                /* data 脚设置为输出 */
```

```

    TWI_CLK_LEVEL_HIGH;                                /* 拉高 sclk 脚电平 */
    for(uint8_t i=0;i<8;i++)
    {
        if(byte&0x80)                                  /* 高位如果为 1 则拉高电平 */
        {
            TWI_DATA_LEVEL_HIGH;
        }
        else
        {
            TWI_DATA_LEVEL_LOW;                        /* 为 0 则拉低电平 */
        }
        TWI_CLK_LEVEL_LOW;                             /* 拉低电平 */
        TWI_CLK_LEVEL_HIGH;                           /* 拉高电平 */
        byte <<= 1;                                    /* 左移 1 位 */
    }
    TWI_CLK_LEVEL_HIGH;
    TWI_DATA_LEVEL_HIGH;
}

/*****
* Function      : twi_read_byte
* Description    : twi 读一个字节
* Input         : none
* Output        : none
* Return        : uint8_t
*****/
uint8_t twi_read_byte(void)
{
    TWI_DATA_DIR_INPUT;
    TWI_CLK_LEVEL_LOW;                                /* 拉低时钟电平 */

    uint8_t value = 0;
    for(uint8_t i=0;i<8;i++)
    {
        value <<= 1;
        TWI_CLK_LEVEL_LOW;
        if(TWI_DATA_LEVEL_READ == 1)
        {

```

```

        value++;
    }
    TWI_CLK_LEVEL_HIGH;
}
TWI_DATA_DIR_OUTPUT;           /* data 脚设置为输出 */
TWI_DATA_LEVEL_HIGH;
TWI_CLK_LEVEL_HIGH;           /* 拉高 clk 脚电平 */
return value;
}

/*****
* Function      : twi_reset
* Description    : TWI 复位
* Input         : none
* Output        : none
* Return        : none
*****/
void twi_reset(void)
{
    TWI_DATA_LEVEL_LOW;
    for(uint8_t i=0;i<50;i++)
    {
        TWI_CLK_LEVEL_LOW;       /* 拉低电平 */
        TWI_CLK_LEVEL_HIGH;      /* 拉高电平 */
    }
}

```

4.3 TWI 操作

```

/*****
* Function      : twi_on
* Description    : 打开 TWI
* Input         : none
* Output        : none
* Return        : uint8_t
*****/
uint8_t twi_on(void)
{

```



```

uint8_t count = 10;

/* 唤醒 */
TWI_DATA_LEVEL_LOW;
TWI_CLK_LEVEL_LOW;
delay_us(50);
TWI_CLK_LEVEL_HIGH;

/* 2ms CLK */
TWI_DATA_LEVEL_LOW;
for(uint16_t i=0;i<680;i++)
{
    TWI_CLK_LEVEL_LOW;          /* 拉低电平 */
    TWI_CLK_LEVEL_HIGH;        /* 拉高电平 */
}

twi_write_byte(0x80);
twi_read_byte();

while(--count)
{
    twi_reset();
    twi_write_byte(0xBF);
    if((twi_read_byte() & 0x0F) == 0x07)
    {
        break;
    }
}

if(count == 0)
{
    return 1;
}
return 0;
}

/*****
* Function      : twi_off

```

```
* Description   : 关闭 twi
* Input         : none
* Output        : none
* Return        : none
```

```
*****/
```

```
void twi_off(void)
```

```
{
```

```
    twi_reset();
```

```
    twi_write_byte(0xFF);
```

```
    twi_write_byte(0x02);
```

```
}
```

5 总结

1. 操作 Reg3F 前需要发射时序复位 TWI 接口。
2. 芯片唤醒后，需要在 TWI 窗口内发送打开时序，使芯片进入 TWI 模式。
3. TWI 打开需要读取 Reg3F 低四位来判断是否可正常读写寄存器。

6 版本修订

版本	日期	描述
V1.0	2025.03.25	初始版
V1.1	2025.04.11	更新 TWI 操作代码