OTG15X 系列I²C软件用户手册V2.7

2012年07月04日

◆ OTG15X的一般规则说明

OTG15X是一般指在OTG15系列模块之中使用的处理器。

用户主机是一般指用户自行开发的单片机，与OTG15X之间通过标准折I²C总线通讯。

如果是两个字节组成16位的参数，则第1个字节为低位，第2个字节为高位。

如果是4个字节组成32位的参数，则第1个字节为低位，第4个字节为高位。

0xnn表示所描述的值不确定，可能为任意值。但其值为原先约定的范围，例如指令长度为2～137。

B7表示位于字节的第7位，B6表示位于字节的第6位，以此类推。

用户主机写入OTG15X的I²C地址为0x5e即01011110B，读取的I²C地址为0x5f即01011111B。

寄存器长度一般为8位，用户主机只需要一个字节的读写即可。另外标注字节长度的寄存器，则需要多个字节读写的，应根据需要进行多字节的读写。

OTG15X除了用于I²C的SCL及SDA接口之后，额外提供一个INT输出接口，用于在OTG15X内部状态改变后通知用户主机之用。

正常工作时INT为高电平，由OTG15X之内的上拉电阻拉高，可以被外部电路拉低，但不能被外部电路拉高，与8051单片机的接口相同。

当OTG15X内部有合适的中断产生时，OTG15X置低INT。在用户主机消除了相应的中断状态后OTG15X会将INT变高。

用户主机可以随时检测INT的电平变低，在检测到INT为低后读取INTRD寄存器。INTRD为16位的寄存器，每一个位用来表示一种中断，共有INT0～INT15种中断。每个中断可以由INTEN寄存器设置或清除。当有读取到中断后，可以设置相应的中断号清除产生的中断。

中断机制只是提供一种快速的处理模式，用户主机可以依靠中断号读取相应的寄存器，这样可以大大提高I2C总线的使用时间。

在只读或读写寄存器之后带有“INTx”的寄存器，一般建议是在用户主机读取到相应中断后再读取，这样获得的信息是最及时的，如果在没有发生中断就读取，一般只是读到上次未更新的信息。

◆ OTG15X的系统设置类寄存器

地址	名称	描述
0x00	INTCLR	清除中断寄存器（只写） 中断号说明: 字节0与字节1组成16位的中断号，字节0的B0为INT0，字节1的B7为INT15，以此类推。 INT0=1，OTG15X初始化，可防止用户与OTG15X不同步上电; INT1=1，USB/CF卡接口插拔中断，需要读取”DISKSTA”寄存器; INT2=1，USB/CF内文件结构扫描已完成，可读取相关信息; INT3=1，文件信息完成，可读取“FILEINFO”寄存器、文件名或ID3; INT4=1，曲目播放时间变化; INT5=1，曲目播放停止; INT6=1，找不到指定的文件; INT7=1，文件无法播放; INT8=1，曲目录音停止; INT9=1，U盘/CF卡空间已满/写保护; INT10=0，保留 INT11=1，字库读取完成标志，可读取相关信息; INT12=1，实时时钟变化; INT13=0，保留 INT14=1，OTG15X与电脑及网络通讯接收中断; INT15=1，附加功能中断;
0x01	INTRD	读中断寄存器（只读） 中断号与清除中断寄存器相同。 **注意：读取后应清掉相应的中断号，否则将一直中断。寄存器写入1可清除相应的中断。**
0x02	INTENA	中断允许寄存器（只写） 设置相应的中断允许，OTG15X在状态改变时，将产生相应的中断并变低INT脚，用户主机需检测INT脚，读取中断值并作相应的处理。 中断号与读/写中断寄存器相对应，当相应的位为1时允许相对应中断。为0时禁止相对应中断。
0x03	DISKSTA	接口状态寄存器（只读 / INT1） B0=1，USB接口0有U盘插入或者拔出动作; B5=1，CF卡有插入或者拔出动作; B6=1，USB器件与PC有插入或者拔出动作; B7=1为接口插入类型，B7=0为接口拔出类型;
0x08	APPSTACLR	清除附加功能中断寄存器（只写） **注意：读取后应清掉相应的中断号，否则将一直中断。寄存器写入1可清除相应的中断。**
0x09	APPSTA	附加功能中断寄存器（只读 / INT15） B0=1，OTG15X温度变化; B1=1，选择器件接口时接口上没有器件; B2=1，选择的当前接口上的器件被拔出; B5=1，列表数据读取完成标志，可读取相关信息; B6～B7，保留为0;
0x0c	SETRTC	系统实时时钟设置（只写） 字节0～字节6分别为秒/分/时/星期/日/月/年;
0x0d	READRTC	系统实时时钟读取（只读 / INT12） 字节0～字节6分别为秒/分/时/星期/日/月/年;

0x0f	SYSVER	系统版本信息读取（只读） 字节0～字节3为软件版本生成的小时/日/月/年; 字节4为OTG15X模块硬件代码：=0x15为OTG15E，=0x16为OTG15F，=0x18为OTG15H。

◆ OTG15X的播放/录音类寄存器

地址	名称	描述
0x10	DISKSEL	盘符接口选择寄存器（只写） =0x00，使用USB接口0; =0x05，使用CF卡接口; 注意，这个寄存器在对所有操作之中有效，需要提前选择。
0x11	CIRCLEMODE	循环播放模式选择寄存器（只写） =0x00，全部顺序循环播放; =0x01，单目录顺序循环播放; =0x02，单目录顺序播放,播放完毕后停止; =0x03，单目录无序循环播放; =0x04，单曲循环播放; =0x05，单曲播放后停止; 注意，这个寄存器在对所有操作之中有效，需要提前选择。
0x12	PLAYMODE	播放、暂停、停止、录音、停止录音模式选择（只写） =0x00，停止模式; =0x01，播放模式; =0x02，暂停模式; =0x03，录音模式; =0x04，暂停录音; =0x05，停止录音。
0x17	GOTOTIME	跳到指定时间播放（只写）
0x14	SPECPLAY	指定曲目播放（只写） (说明：目录及曲目的排序都是按照创建时间的先后顺序进行排列) 字节0～字节1为曲目号; 字节2～字节3为目录号;
0x16	AUDMODE	声音及通道模式选择 （只写） =0x00，停止模式时为静音; =0x01，停止模式时为选择ADC的声音; =0x02，停止模式时为选择内置FLASH播放; =0xff，停止模式时静音; =0x10，播放模式时解除静音; =0x11，播放模式时打开静音;
0x18	RECMODE	录音格式设定（只写） 字节0～字节3具体值请参照附录一“录音格式设定”
0x19	CHECKLIST	列表查询（只写） station NO x = x list NO xy = z Playfile xy [] = {0xa1,0xa2,0xa3,...} 字节0～字节1为x; 字节2为y(十进制，最大为255); 字节3为>1时，表示回传指定位置的单个数据;=0时，表示回传整个数组的数据。

		z为单个数组的总单元数(十进制，最大为255); 例: station NO 1 = 1 list NO 001001 = 3 Playfile 001001 [] = {0x01,0x02,0x03,} list NO 001002 = 4 Playfile 001002 [] = {0x04,0x05,0x06,0x07,} station NO 2 = 2 list NO 002001 = 3 Playfile 002001 [] = {0xa1,0xa2,0xa3,} list NO 002002 = 4 Playfile 002002 [] = {0xa4,0xa5,0xa6,0xa7,}
0x1a	PLAYLIST	列表播放（读/写） I2C 传输用户所选择的位于CF 卡根目录下Playlist.txt文档中列表的站数号;寄存器记录用户选择播放的列表号;（注: Playlist.txt文档内容要按规定格式书写，列表播放文件放在AUDIO文件夹中）。列表格式（如下）: [1] station NO 1= 1 // 代表站数 list NO 1=2 // 代表本站包括语音条目数量 playfile 1= 0001.mp3+000a.mp3 // 本站歌曲名; 由4个英文字母或阿拉伯数字组成
0x1b	NEWFOLDER	新建文件夹寄存器（只写） 直接发送文件夹名称进行新建，最多支持8个字节的文件名
0x1d	CHECKLIST DATA	查询列表所得数据回传（只读） 为双字节的字符串，以0x0000为结束（只读 / APPSTA/B5=1）

◆ OTG15X的文件信息类寄存器

地址	名称	描述
0x20	NEWFILE	新建文件寄存器（只写） 在U盘内建立一个新文件。 =0x00，新建文件，默认的文件名为NEW_XX.15X，其中XX为从01开始的递增的文件序列号。 =0x01，关闭文件，关闭并保存新建的文件。
0x21	TRACKTOL	U盘内的总曲目数（只读 / INT2）
0x22	COPYFILE	复制文件寄存器（只写）
0x23	DIRTOL	U盘内的总目录数（只读 / INT2）
0x24	DELFILE	删除文件寄存器（只写） 删除指定位置的文件(其中目录及文件的排序都是按照创建时间的先后顺序进行排列)，16位寄存器，高8位为目录号，低8位为文件号。
0x25	DIRTRACK	查询指定目录内的总曲目数（读写） 8位寄存器。
0x27	PLAYTIME	播放时间（只读/ INT4） 说明：曲目的播放时间。整除60为分钟数，取余60为秒。
0x29	FILEINFO	正在播放的文件信息（只读 / INT3） 字节0～字节1为曲目号; 字节2～字节3为目录号; 字节4～字节5为曲目总时间; 字节6为曲目类型; 字节7为曲目码流率; 字节8为曲目采样率; 曲目的总时间整除60为分钟数，取余60为秒。 字节6文件类型=0x01，为MP3文件；=0x02，为WMA文件; 字节7码流率若文件类型为MP3，则字节7的值乘以8为曲目的码流率；若文件类型为WMA，该值直接为码流率。 字节8采样率=0x08，为8K；=0x0b为11.025KHz;=0x0c 为12K；=0x10为16K；=0x16为22.05KHz;=0x18为24K；=0x20为32K；=0x2c为44.1KHz;=0x30 为48K。
0x2b	FILENAME	曲目文件名内码，为双字节的字符串，以0x0000为结束（只读 / INT3）
0x2d	ID3NAME	曲目ID3内码，为双字节的字符串，以0x0000为结束（只读 / INT3）

◆ OTG15X的文件编辑类寄存器

地址	名称	描述
0x30	FONTTYPE	字库格式选择（只写） 字节0～字节1为限制字库缓冲区字节长度，为0x0000则不限制长度; 字节2为字库格式选择，具体值请参照附录二“字库格式说明”; 字节3为字库内码选择，0为GBK，1为UNICODE码。 字节4为字体选择，0为自动，1为简体，2为繁体。 这个寄存器一般收到在INTRD->系统异常-> OTG15X初始化时写入一次。
0x31	FONTLENG	有效字库总字节长度（只读 / INT11） 字节0～字节1为有效字库总字节长度，在“GETFONT” 引发INTRW产生中断之后可以读取。
0x32	GETFONT	获取指定字符串的字库（只写） 字节0为输入字符串的路径选择。 字节0为0xff时: 从已经获取的FONTBUFF字库指定偏移位置; 字节1～字节2为偏移位置; 字节0为0xfe时: 字节1为输入内码字符串的长度; 字节2～字节n为输入内码字符串，用户主机可以输入任意字符; 字节0为0xfc时: 使用当前曲目文件名内码为输入内码字符串; 字节0为0xfb时: 使用当前曲目ID3内码为输入内码字符串; 字节0小于0x80时，与“接口选择寄存器目录号”相同: 字节1～字节2为曲目号; 字节3～字节4为目录号; 注意：这时会引发INTRW产生中断，用户主机必须等待“字库读取完成标志”再读取“FONTBUFF寄存器”。
0x33	FONTBUFF	字库缓冲区（只读 / INT11） 字节0～字节n-1为字库数据; 字节n为校验和，其值等于从字节0到字节n-1所有字节相加之和;
0x34	TRACK NAME	查询指定曲目名称 高8位为目录号，低8位为曲目号。 其他操作如读取当前播放文件名一样。

附录一:

◆ 录音格式设定

字节0为指定录音格式，默认为MP3格式128Kbps/44.1KHz/立体声;

字节1为指定录音停止方式，0为录音长度到达后停止，1为录音长度到达后文件名号码加1后继续录为下一首;

字节2～字节3为指定录音文件的长度，单位为MB ， 0x0000为默认50MB，0x0001为1MB;

选用MP3压缩格式录音，录音扩展名为*.MP3	
0x00: 320kbps/44.1kHz/立体声 0x04: 192kbps/44.1kHz/立体声 0x07: 160kbps/44.1kHz/立体声 0x0a: 128kbps/44.1kHz/立体声 0x10: 64kbps/44.1kHz/立体声	
选用WAVE无损压缩格式录音，录音扩展名为*.WAV	
0x30: 44.1kHz/立体声	

注意:

1 录音文件的长度越大，录音初始化所需要的时间越长。
2 1MB = 1,048,576字节。
3 若以录音格式是128Kbps/44.1KHz/立体声，可录音约1小时（位率/8为该位率录音1秒钟的字节数，单位为KB）。
4 双声道即不相关的两个通道。
5 VBR为可变码率录音方式。
6 默认值为128kbps/44.1kHz/立体声。
7 录音前须指定相应的码流率及采样率，否则以默认值进行录音。

附录二:

◆ 字库格式说明及使用方法

OTG15X支持任何字库排列方式。

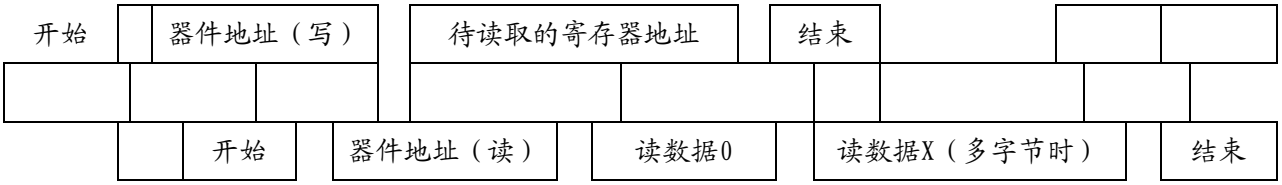
附录三:

◆ OTG15X读、写寄存器说明

OTG15X写寄存器示意图:

开始	器件地址（写）	寄存器地址	写数据0	写数据X（多字节时）	结束

OTG15X读寄存器示意图:



先使用写的器件地址写入待读取的寄存器地址，再使用读的器件地址读入相应的数据。

在对I^2C写入每个字节包括数据及地址时，需要接收第9位ACK位，ACK位由OTG15X输出0。用户主机依靠ACK可以获得OTG15X是否正常工作的信息。

在对I^2C读取时每个字节时，需要发送第9位ACK位，ACK位由用户主机输出0。但最后一个字节则需要发送第9位NAK位，NAK位由用户主机输出1。

附录四:

◆ OTG15X使用IO口模拟I²C时序源代码说明

```
void main() {
    BYTE gLocal_1;
    // 其他的初始化;
    // I²C总线及INT初始化
    while (1) {
        if (pMED_INT) {
        gLocal_1 = MMED_ReadByte(cADD_INTRD);      // 读取    中断
            if (gLocal_1 & 0x01) {                          // B0系统异常
                MMED_WriteByte(cADD_INTCLR, 0x01);      //  清除相应的中断
                // 其他处理
            }
            else if (gLocal_1 & 0x02) {       // B1及其他处理
                // 其他相应处理
            }
        }
       // 其他处理
    }
}

// 以下函数可以在所有I²C总线之中使用
void MMED_WriteByte(BYTE gLocal_1, BYTE gLocal_2) {       // 写单字节的寄存器
    MMEDStart();                                // I2C开始
    MMEDWrite(0x5e);                            // 写入器件地址写地址
     MMEDWrite(gLocal_1);                           // 写寄存器地址
    MMEDWrite(gLocal_2);                        // 写寄存器值
    MMEDStop();                                 // I2C停止
    return;
}

BYTE MMED_Read_BYTE(BYTE gLocal_1) {          // 读8位寄存器
    MMEDStart();                                // I2C开始
    MMEDWrite(0x5e);                            // 写入器件地址写地址
     MMEDWrite(gLocal_1);                           // 写寄存器地址
    MMEDStop();                   // 这时OTG15X准备相应的数据用于读取

    MMEDStart();                                // I2C开始
    MMEDWrite(0x5e+1);                          // 写入器件地址写读地址
    gLocal_1 = MMEDRead(1);       // 读寄存器ACK位, 非应答 ( NAK ) 为1停止读取
    MMEDStop();                                 // I2C停止
    return gLocal_1;                            // 回传寄存器值
}
```

```
BOOL MMEDWrite(BYTE gLocal_1){          // I2C写字节，gLocal为待写数据
    BOOL FLocal_1;                      // 返回ACK/NAK标志
    BYTE gLocal_2;                      // 位计数器

    gLocal_2 = 8;                       // 8位，高位先出
    do {
        pMED_SCL(0);                    // 置低I2C时钟线，以改变数据
        if (gLocal_1 & 0x80){           // 如果数据位为1
            pMED_SDA(1);                // 置高I2C数据线
        }
        else {                          // 如果数据位为0
            pMED_SDA(0);                // 置低I2C数据线
        }
        MUSDELAY(5);                    // 延时5微秒
        gLocal_1 <<= 1;                 // 准备下一位数据
        pMED_SCL(1);                    // 置高I2C时钟线，保持数据稳定
    MUSDELAY(5);                        // 延时5微秒
    } while (--gLocal_2 != 0);          // 完成8位数据

    pMED_SCL(0);                        // 置低I2C时钟线，准备ACK位
    pMED_SDA(1);                        // 置高I2C数据线，ACK位为1
    MUSDELAY(5);                        // 延时5微秒
    pMED_SCL(1);                        // 置高I2C时钟线
    MUSDELAY(5);                        // 延时5微秒
    FLocal_1 = 0;                       // NCK标志表示不成功
    if (!pMED_SDA_HIGH){
        FLocal_1 = 1;                   // ACK标志表示成功
    }
    pMED_SCL(0);
    return FLocal_1;                    // 返回ACK/NAK标志
}


void MMEDStart(){                       // I2C开始
    pMED_SDA(1);                        // 置高数据线，数据线空闲
    pMED_SCL(1);                        // 置高时钟线，时钟线空闲
    MUSDELAY(5);                        // 延时5微秒，保持状态稳定
    pMED_SDA(0);        // 数据线在时钟为高电平时由高电平向低电平切换表示开始
    MUSDELAY(5);                        // 延时5微秒
    pMED_SCL(0);                        // 置低时钟线，准备接收或发送数据
    return;
}
```

```
BYTE MMEDRead(BOOL FLocal_NAK) {                    // I2C读
    BYTE gLocal_1;                                  // 数据暂存器
    BYTE gLocal_2;                                  // 位计数器

    pMED_SDA(1);                                    // SDA准备输入
    MUSDELAY(5);                                    // 延时5微秒
    gLocal_2 = 8;                                       // 8位
    do{
        pMED_SCL(1);                // 置高I2C时钟线，时钟线为高电平时数据方有效
        gLocal_1 <<= 1;                             // 如果数据线为高电平，则数据暂存器赋1
        if (pMED_SDA_HIGH) gLocal_1 |= 0x01;
        MUSDELAY(5);                                // 延时5微秒，保持状态稳定
        pMED_SCL(0);                                // 置低I2C时钟线，接收下一位
        MUSDELAY(5);                                // 延时5微秒，保持状态稳定
    } while (--gLocal_2 != 0);                      // 循环8次，完成一个字节的接收

    if (!FLocal_NAK){                                   // 如果ACK标志为0
        pMED_SDA(0);                                // 置低数据线，ACK为0，继续读取
    }
    else {                                              // 如果ACK标志为1
        pMED_SDA(1);                                // 置高数据线，ACK为1，读取结束
    }
    MUSDELAY(5);                                    // 延时5微秒
    pMED_SCL(1);                                    // 发送ACK/NAK位
    MUSDELAY(5);                                    // 延时5微秒
    pMED_SCL(0);                                    // 置低I2C时钟线
    return gLocal_1;                                // 返回接收到的数据
}

void MMEDStop() {                                   // I2C停止
    pMED_SDA(0);                                    // 置低数据线，准备停止
    MUSDELAY(5);                                    // 延时6微秒，保持状态稳定
    pMED_SCL(1);                                    // 置高时钟线
    MUSDELAY(5);                                    // 延时6微秒，保持状态稳定
    pMED_SDA(1);                // 数据线在时钟为高电平时由低电平向高电平切换表示结束
    return;
}
void MUSDELAY(BYTE gLocal_1) {
    // 根据用户主机的情况作DELAY
    return;
}
```

附录五:

◆ 龙珠科技使用C语言书写源代码的一般规则说明

1. 命名规则

所有变量、常量、函数等命名由3部分组成。

例如: FAUD_Mute 分为 F AUD _Mute 共3部分。

第1部分由1个字母或1个字母加1个数字组成，表示所定义的类别。

内容	意义
大写 'M'	表示为函数
大写 'F'	表示为标志变量，1位的变量
小写 'g'	表示8位的变量
小写 'g2'	表示16位变量
小写 'g4'	表示32位变量
小写 'g8'	表示为64位变量
小写 'c'	表示常数
小写 'p'	表示IO口

第2部分一般由三至4个大写英文字母组成。表示这个命名所属的文件，例如在H06_AUD.C之中使用的变量，则其第2部分则为AUD，以下为一此常用的文件。

内容	意义
AUD	表示通用音频的处理文档
VOL	表示多声道的音量处理文档
SUR	带多声道环绕声的处理文档。
SUB	表示在扩充main文档的功能，因为main不要放太多的函数，以免影响可观性。
DOS	带USB主机或硬盘接口的操作系统处理文档。
MED	带MP3等多媒体音频播放处理文档。

第3部分为具体的内容，一般是一个单或多个单词，每个单词只有前一个采用大写。每个单词之间可以加下划线，因为每个单词之间有大写作间隔，一般可以不加下划线。原则是如果可观性差，或者单词本身是压缩的词汇（这个时候一般会是大写），可适当加下划线。

2. 全局及局部变量的规则

内容	意义(与VC++相兼容)	C语言标准
1位标志变量	EXTR BOOL FAUD_Mute	无
8位无符号变量	EXTR BYTE gAUO_Volume	unsigned char
16位无符号变量	EXTR WORD g2AUO_EQ_Mode	unsigned int
32位无符号变量	EXTR DWORD g4AUO_Mute_Timer	unsigned long
指针变量	EXTR BYTE *gpAUD_Pointer	unsigned char
局部变量	EXTR BYTE gLocal_1	unsigned char

局部变量，绝对禁止使用1或几个字母，如 'X' 等做局部变量，这样不利于复制，亦无法看出其位数，所有的书写都是第1次认真命名，使用时采用复制的方式，不主张再次写入相同的名字。

标志局部变量BOOL FLocal_1，8位的局部变量BYTE gLocal_1等。所有局部变量第1部分与全局变量的相同，第2部分采用 'Local_' 字符串表示局部的意思，第3部分由数字1至9及小写 'a' 至 'z' 组成。

附录六:

◆ 标准的I2C通讯功能叙述

I²C总线的概念

I²C总线由两线串行数据（SDA）和串行时钟（SCL）线在连接到总线的器件间传递信息。每个器件都有一个唯一的地址识别（无论是微控制器，LCD驱动器，存储器或键盘接口），而且都可以作为一个发送器或接收器（由器件的功能决定）。很明显，LCD驱动器只是一个接收器，而存储器则既可以接收又可以发送数据。除了发送器和接收器外，器件在执行数据传输时也可以被看作是主机或从机(见表1)。主机是初始化总线的数据传输并产生允许传输的时钟信号的器件。此时，任何被寻址的器件都被认为是从机。

表1 I²C总线术语的定义

术语	描述
发送器	发送数据到总线的器件
接收器	从总线接收数据的器件
主机	初始化发送、产生时钟信号和终止发送的器件
从机	被主机寻址的器件
多主机	同时有多于一个主机尝试控制总线，但不破坏报文
仲裁	是一个在有多个主机同时尝试控制总线，但只允许其中一个控制总线并使报文不被破坏的过程
同步	两个或多个器件同步时钟信号的过程

I²C总线是一个多主机的总线。这就是说可以连接多于一个能控制总线的器件到总线。由于主机通常是微控制器，让我们考虑以下数据在两个连接到I²C总线的微控制器之间传输的情况（见图2）。

这突出了I²C总线的主机—从机和接收起—发送器的关系。应当注意的是：这些关系不是持久的，只由当时数据传输的方向决定。传输数据的过程如下;

1) 假设微控制器A要发送信息到微控制器B:
   - 微控制器A（主机）寻址微控制器B（从机）
   - 微控制器A（主机—发送器）发送数据到微控制器B（从机—接收器）
   - 微控制器A终止传输
2) 如果微控制器A想从微控制器B接收信息:
   - 微控制器A（主机）寻址微控制器B（从机）
   - 微控制器A（主机—接收器）从微控制器B（从机—发送器）接收数据
   - 微控制器A终止传输

甚至在这种情况下，主机（微控制器A）也产生定时而且终止传输。

连接多于一个微控制器到I²C总线的可能性意味着超过一个主机可以同时尝试初始化传输数据。为了避免由此产生混乱，发展出一个仲裁过程。 它依靠线与连接所有I²C总线接口到I²C总线。

如果两个或多个主机尝试发送信息到总线，在其他主机都产生“0”的情况下，首先产生一个“1”的主机将丢失仲裁。仲裁时的时钟信号是用线与连接到SCL线的主机产生的时钟的同步结合。

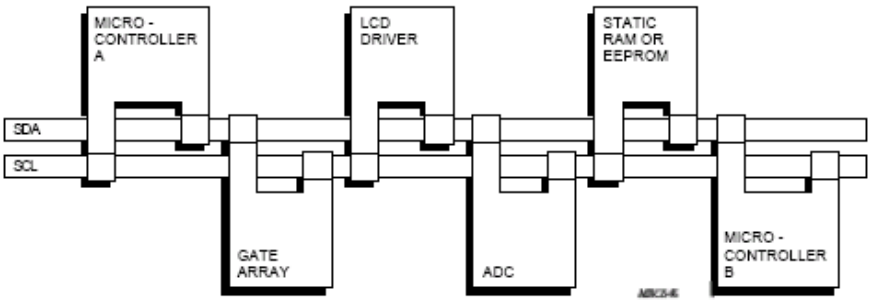


图 2 使用两个微控制器的 I²C 总线配置举例

I²C总线上产生时钟信号通常是主机器件的责任；当在总线上传输数据时，每个主机产生自己的时钟信号。主机发出的总线时钟信号只有在以下的情况才能被改变：慢速的从机器件控制时钟线并延长时钟信号，或者在发生仲裁时被另一主机改变。

总体特征

SDA和SCL都是双向线路，都通过一个电流源或上拉电阻连接到正的电源电压（见图3）。当总线空闲时，这两条线路都是高电平。连接到总线的器件输出级必须是漏极开路或集电极开路才能执行线与的功能。I²C总线上数据的传输速率在标准模式下可达100Kbit/s，在快速模式下可达400Kbit/s，在高速模式下可达3.4Mbit/s。连接到总线的接口数量只由总线电容是400pF的限制决定。

位传输

由于连接到I²C总线的器件有不同种类的工艺（CMOS，NMOS，双极性），逻辑‘0’（低）和‘1’（高）的电平不是固定的，它由V_{DD}的相关电平决定。每传输一个数据位就产生一个时钟脉冲。

数据的有效性

SDA线上的数据必须在时钟的高电平周期保持稳定。数据线的高或低电平状态只有在SCL线的时钟信号是低电平时才能改变（见图4）。

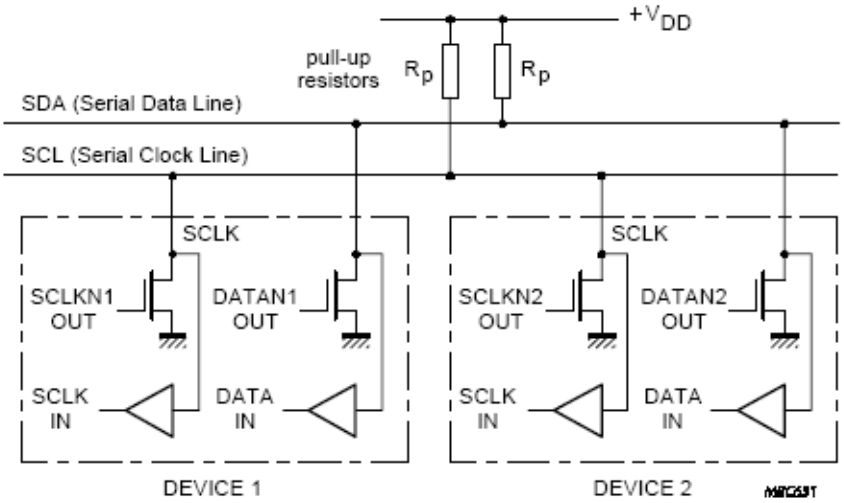


图 3 标准模式器件和快速模式器件连接到 I²C 总线

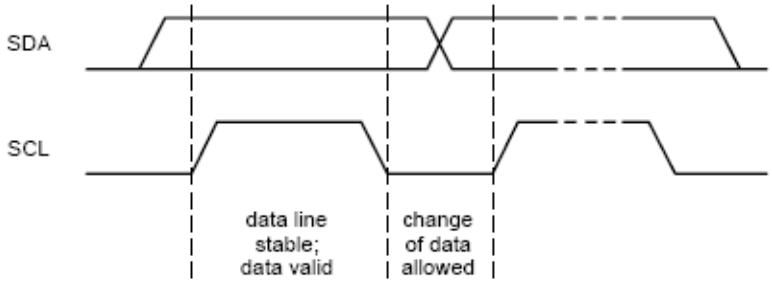


图 4 I²C 总线的位传输

起始和停止条件

在I²C总线中，唯一出现的是被定义为起始（S）和停止（P）条件（见图5）的情况。

其中一种情况是在SCL线是高电平时，SDA线从高电平向低电平切换。这个情况表示起始条件。

当SCL是高电平时，SDA线由低电平向高电平切换表示停止条件。

起始和停止条件一般由主机产生。总线在起始条件后被认为处于忙的状态。在停止条件的某段时间后，总线被认为再次处于空闲状态。

如果产生重复起始(Sr)条件而不产生停止条件，总线会一直处于忙的状态。此时的起始条件（S）和重复起始(Sr)条件在功能上是一样的。因此在本文档的剩余部分，符号S将作为一个通用的术语既表示起始条件又表示重复起始条件，除非有特别声明的Sr。

如果连接到总线的器件合并了必要的接口硬件，那么用它们检测起始和停止条件十分简便。但是，没有这种接口的微控制器在每个时钟周期至少要采样SDA线两次来判别有没有发生电平切换。

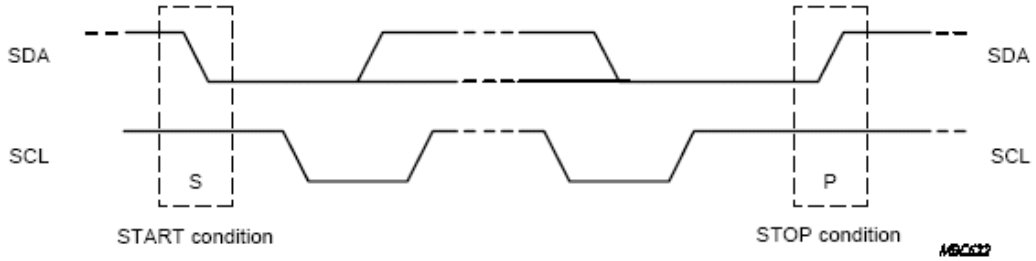


图 5 起始和停止条件

传输数据

①字节格式

发送到SDA线上的每个字节必须为8位。每次传输可以发送的字节数量不受限制。每个字节后必须跟一个响应位。首先传输的是数据的高位（MSB）（见图6）。如果从机要完成一些其他功能后（例如一个内部中断服务程序）才能接收或发送下一个完整的数据字节，可以使时钟线SCL保持低电平迫使主机进入等待状态。当从机准备好接收下一个数据字节并释放时钟线SCL后，数据传输继续。

在一些情况下，可以用与I^2C总线格式不一样的格式（例如兼容CBUS的器件）。甚至在传输一个字节时，用这样的地址起始的报文可以通过产生停止条件来终止。此时不会产生响应。

②响应

数据传输必须带响应。相关的响应时钟脉冲由主机产生。在响应的时钟脉冲期间，发送器释放SDA线（高）。

在响应的时钟脉冲期间，接收器必须将SDA线拉低，使它在这个时钟脉冲的高电平期间保持稳定的低电平（见图7）。当然，必须考虑建立和保持时间。

通常，被寻址的接收器在接收到的每个字节后，除了用CBUS地址开头的报文，必须产生一个响应。

当从机不能响应从机地址时（例如它正在执行一些实时函数不能接收或发送），从机必须使数据线保持高电平。主机然后产生一个停止条件终止传输或者产生重复起始条件开始新的传输。

如果主机—接收器响应了从机地址但是在传输了一段时间后不能接收更多数据字节，主机必须再一次终止传输。这个情况用从机在第一个字节后没有产生响应来表示。从机使数据线保持高电平，主机产生一个停止或重复起始条件。

如果传输中有主机接收器，它必须通过在从机不产生时钟的最后一个字节不产生一个响应，向从机—发送器通知数据结束。从机—发送器必须释放数据线，允许主机产生一个停止或重复起始条件。

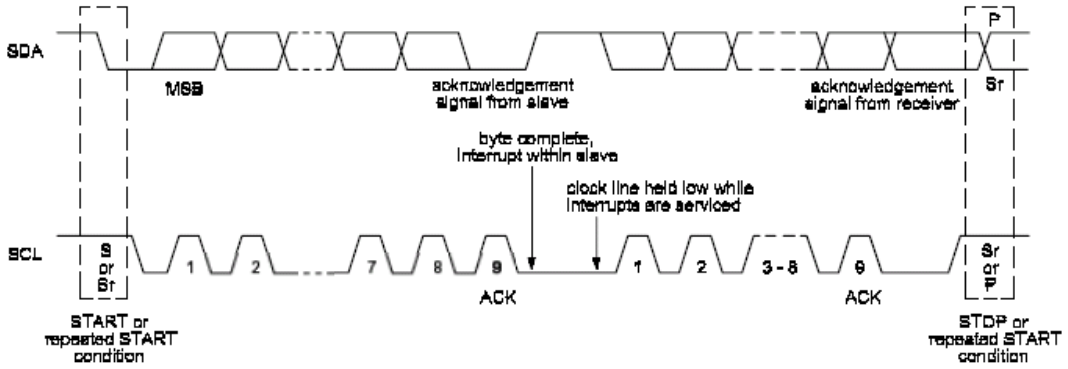


图 6 I²C 总线的数据传输

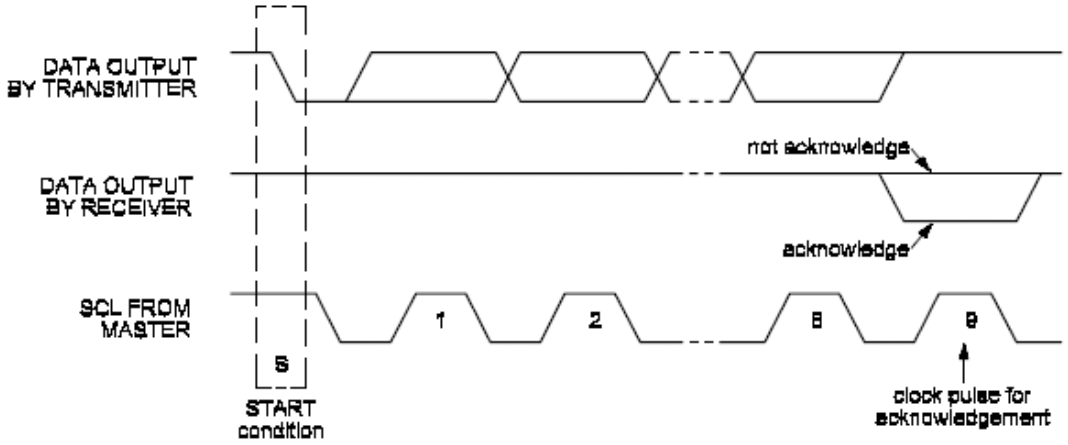
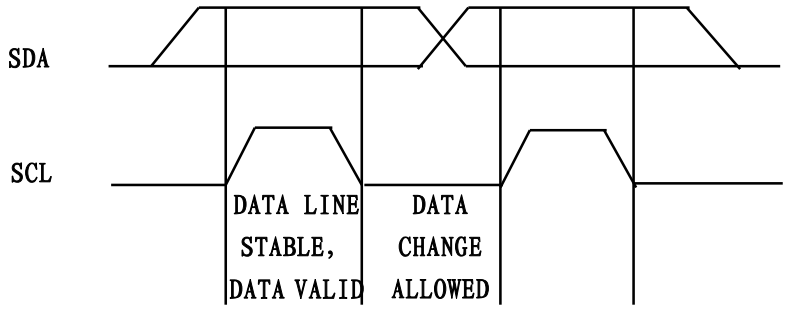


图 7 I²C 总线的响应

资料确认

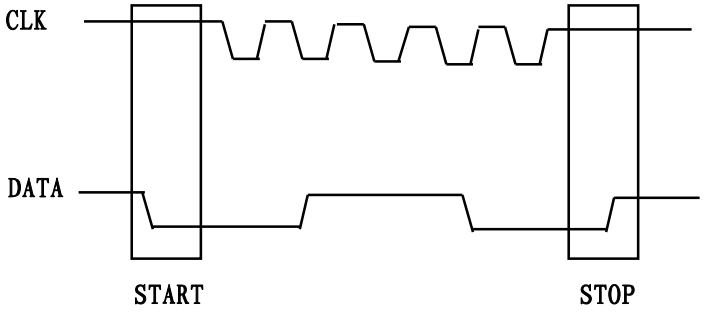
当SCL讯号是在“高准位”时，SDA Line上的资料才会被视为正确且稳定的资料。而只有当SCL讯号在“低准位”时，SDA Line才可做高、低准位的切换。请参阅下图:



开始和结束

（1）当SCL讯号设定在高准位，且SDA讯号由高准位转换成低准位时；则表示序列资料的“开始”。

（2）当SCL讯号设定在高准位，且SDA讯号由低准位转换成高准位时；则表示序列资料的“结束”。



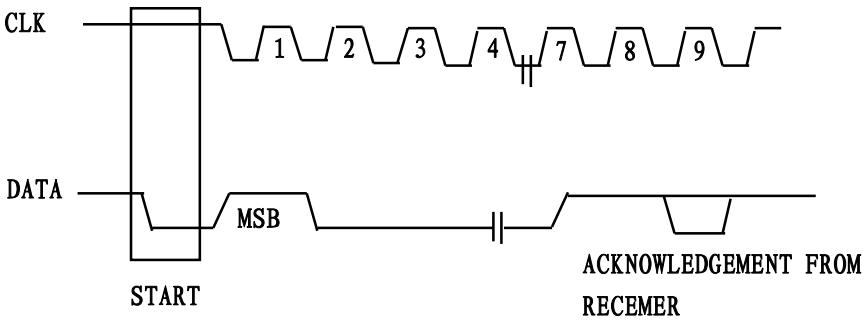
位元组格式

每一个传输到SDA Line的位元组（byte）有八个位元(bit),每一位元组后面需要有一个“认可”位元，且以最大符号位元（MSB）为首的方式传送出去。

应答信号

在第九个时脉时，用户主机先将SDA Line设定为电阻性的高准位，若周边设备（OTG15X）认可此信号，则SDA Line将会被周边设备（OTG15X）拉至低准位，使SDA Line在此时脉中保持一稳定的低准位状态。请参阅下图:

已被定址的OTG15X在收到每一位元组（BYTE）后，即产生一“应答”的动作；否则在第九个时脉（CLOCK）的时间内SDA Line将会一直保持着高准位状态。



无应答信号的传输

如果您想省略OTG15X对“应答”信号的侦测，可使用一较简单的传输方式。其方式为OTG15X在收到每一位元组(byte)后，等待一时脉（clock），不做时脉的确认。如果您使用此种方法，将会有较大的机会造成传输错误，并且会减低对杂讯的免疫力。