摘　要: 介绍一种以 DALLAS DS5002FP 单片机为主控制器" title="主控制器">主控制器的 MDB/ICP 协议实现。从硬件和软件描述了如何控制 MDB 总线,并给出一个较成熟的控制外设" title="外设">外设会话的总线驱动程序。该协议实现清晰、简单,采用 MCS-51 兼容单片机降低成本,充分体现了 MDB/ICP 的优越性。
关键词: 单片机　自动售卖系统　主控制器

随着国民经济和技术的发展,自动售卖系统由于其方便性、易管理性和低成本,正得到越来越广泛的应用。而自动售卖系统的外设也越来越复杂,从投币机到读钞机到非现金交易(如 IC 记帐卡),导致对主控制器的要求越来越高。为了简化设计,采用一个简单、稳定的内部通讯总线协议" title="总线协议">总线协议非常必要。

本文介绍了欧洲售货机制造者协会(EVMMA)制定的 MDB/ICP 总线协议。该协议简洁明了,功能强大,可扩展性强,并且对外挂设备数目没有限制,是理想的自动售卖系统内部总线协议。本文从主控制器的角度给出了对 MDB/ICP 总线进行操作控制的硬件和软件实现。

图 1　位传递顺序

1 MDB/ICP 协议简介

MDB/ICP 协议(Multi-Drop Bus / Internal Communication Protocol)是欧洲售货机制造者协会制定的一套用于协调自动售卖机的主控制器(VMC)与多个外设之间通信的协议。硬币机和读钞机的标准首先是由 Coinco 在美国可口可乐公司的指定下开发的。这个标准于 1993 年被 NAMA 协会采用,经过一个专门的工作组修订后,于 1994 年被 EVMMA 采用。第二阶段的非现金交易标准由 Debitek 代表 NAMA 开发,并于 1994 年被 NAMA 采用。EVMMA 加入了一些兼容的指令后于 1995 年采用。

MDB 接口实际上是工作于 9600 波特率的主从型串行总线接口,所有外围设备(例如硬币机、读钞机、读卡器等)均为主控制器(传统上称为售货机控制器——VMC)的从机。所有外围设备与 VMC 之间的通信方式都一致。

MDB 协议的串行位格式为:1 个起始位,8 个数据位,1 个方式位与 1 个停止位,共 11 位。位传递的顺序如图 1 所示。其中方式位根据传递的方式不同置 0 或置 1。在 MDB 总线上,VMC 通过广播方式向外发送命令。第一字节为地址字节(实际上只有高 5 位寻址信息,低 3 位为对外设的指令),该字节被所有的外设读取,但只

有符合地址字节所指定的外设才处理其后的数据字节,并做出反应。在 VMC 到外设的数据中,地址字节的方式位被置 1,数据字节的方式位被置 0,外设通过检验接收到的方式位确认是地址命令还是数据。当数据从外设发送到主机时,最后送出的字节方式位被置 1,标志着数据发送完毕。

VMC 向外设传送的指令由一个地址字节、一些可选的数据字节与一个校验和(CHK)字节构成。发送指令后,外设应答 VMC 的通信块可以由一个数据块" title="数据块">数据块和一个 CHK 字节组成,或者一个应答字节(ACK),或者一个无应答字节(NAK)。如果外设应答数据块的话,VMC 将通过一个应答字节(ACK)、无应答字节(NAK)或重发字节(RET)应答外设传回的数据。

图 2 至图 5 为几个典型的会话例子,其中“*”表示传送时方式位置 1,“ADD”表示地址字节,“CHK”表示传送数据的校验和。

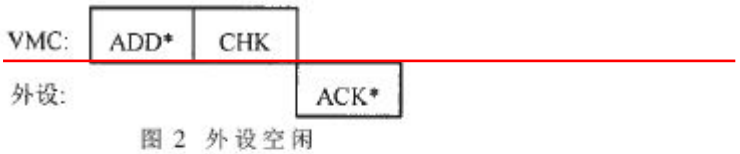

图 2 外设空闲

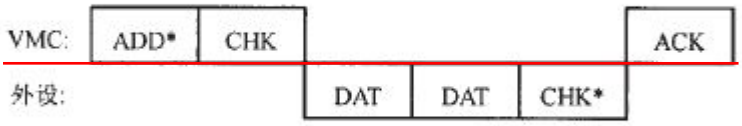

图 3 外设有数据返回

图 4 VMC 有数据要发送

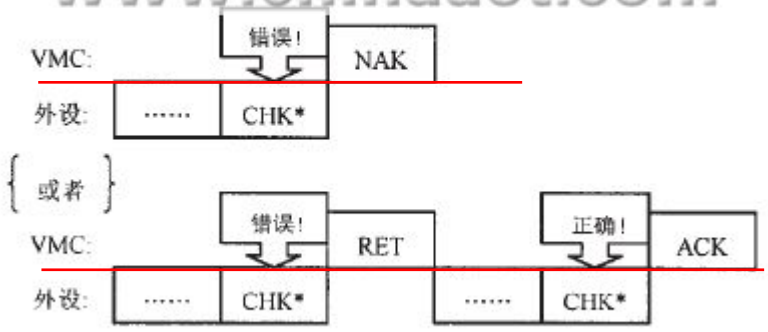

图 5 VMC 发现校验和错误(有两种处理方式)

2 硬件系统结构

在 MDB/ICP 总线协议基础上,笔者构建了一个实际的自动售卖系统,系统原理图如图 6 所示。VMC 通过 MDB 总线与投币机、读钞机和读卡器交互,控制外设的运转,并从外设获得用户的支付情况。同时,VMC 也负责处理与用户之间的交互,驱动液晶显示,处理键盘输入,驱动符合用户需要的电机运转出货。在本项目中,VMC 还需通过 MODEM 自动拨号连接公司总部的服务器,将销售数据传送回公司。

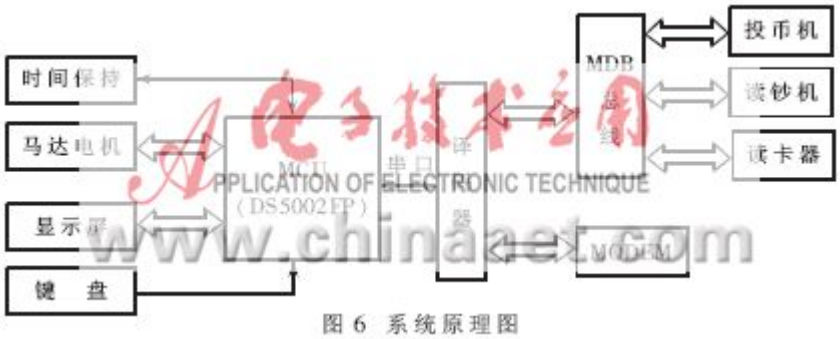

图 6 系统原理图

如图 6 所示,所有 MDB 外设均挂在同一总线上,由总线提供 24V 和 5V 的电源,数据线与单片机的串口" title="串口">串口相连。

本项目中选用的单片机为 DALLAS DS5002FP。该单片机代码级兼容 MCS-51 系列单片机,在使用外存的情况下仍然可以使用 4 个 PO 作为数据 I/O,带外设的能力较强。但它只有一个串口,控制 MDB 总线和控制 MODEM 都需要使用串口,因此必须将串口进行复用。通过一个译码器实现选通功能。

3 MDB/ICP 会话控

将 VMC 与外设之间通过 MDB 总线的会话分为四个层次:

(1)串口初始化

初始化串口时,主要工作是设置波特率(9600)和传输方式(11 位)。为顺利获得 9600 波特率,单片机采用的晶振为 11.0592MHz。源代码如下(采用 Franklin C51 语言编写):

```
// 一些常量定义
#define uchar unsigned char
#define T_RESPONSE 0xee       /*5.0 毫秒*/
#define MAX_BLOCK_SIZE 36   /*数据块最大长度*/
#define ERR_TIME_OUT 0x81   /*超时*/ 0x81=16*8+1=129毫秒
#define ERR_NO_MODE_BIT 0x82/*未收到最后字节*/
#define ERR_CHECKSUM 0x83 /*校验和错*/
#define ACK 0x00
#define RET 0xaa
#define NAK 0xff
// 初始化串口
void InitSerialPort()
{
    SCON = 0xd0;        // 设置串口为方式 3(9bit)
    TMOD &= 0x0f;
    TMOD |= 0x20;       // 设置定时器 1 为方式 2
    TR1 = 1;            // 定时器 1 使能
    TH1 = 0xfd;         // 设定波特率: 9600 NRZ
```

```
    PCON &= 0x7f;       // Set SMOD=0
    RI = 0;
    TI = 0;
}
```

(2)字节的传送

这里需要注意的是:发送数据时要根据需要设置方式位,而接收数据时要返回方式位的值,以判断是否收完所有数据。

```
    // 传送字节
void TransmitByte(uchar byte, bit mode)
    // 形参:byte——准备发送的字节
    //       mode——预备要设置的方式位
{
    TB8 = mode;         // 设置方式位
    SBUF = byte;          // 发送字节
    while(!TI);         // 等待发送完毕
    TI = 0;
}
    // 接收字节
#pragma disable
uchar ReceiveByte(uchar *byte, uchar *bMode)
    // 形参:*byte——返回接收到的字节值
    //       *bMode——返回接收到的字节的方式位
    // 返回值:0—超时,1—成功接收
{
    TMOD &= 0xf0;
    TMOD |= 0x01;          // 设置定时器 0 为方式 1(16 位)
    TH0 = T_RESPONSE;
    TL0 = 0x00;           // 设置超时门限
    TF0 = 0;
    TR0 = 1;              // 定时器 0 使能
    while (!RI && !TF0)
       ;                  // 等待接收字节直至超时
    TF0 = 0;
    if (RI) {             // 已接收字节
       RI = 0;
       *byte = SBUF;      // 返回字节值
       *bMode = RB8;      // 返回方式位
       return 1;
    } else {              // 超时
       RI = 0;
       return 0;
    }
}
```

(3) 数据块的传送

这部分与下面的会话部分放在一个函数体内实现。

(4) 会话

这是 MDB 会话控制的核心部分。根据需要传送数据包,计算校验字节,控制方式位,在时序允许的响应时间内接受外设返回的数据,做出 ACK 或 NAK 等反应。并且按照 MDB/ICP 标准中推荐的方式处理异常情况,当接受数据超时或校验和错时,重复发送命令多次,以增强容错性能。将接收到的数据返回给调用者,并返回结果码。源程序如下:

```
//与外设会话,在调用之前确认译码器选通 MDB 总线
uchar Session(uchar add, uchar dat[], uchar count)
// 形参:add——VMC 发送的地址指令字节
//       dat[]——VMC 发送的数据块
//       count——数据块的大小
// 返回值:0——外设应答 ACK
//   非 0 且小于 0x80——外设应答的数据块的大小
//   大于等于 0x80——会话中出错
// 外设应答的数据块存放在全局数组 uchar recBuff[]
{
    uchar data check, i, j, err;
    uchar data mode;

    for (j=0; j<5; j++) {   //最多重复发送命令 5 次
       check = 0;
       err = 0;                    方式位
       TransmitByte(add, 1);          //发送地址字节
       check += add;              //计算 CHK
       for (i=0; i<count; p="" 发送数据字节?<="" ?????="" {="" i++)="">
          TransmitByte(dat[i], 0);
          check += dat[i];
       }
       TransmitByte(check, 0);          //发送 CHK
       for (i=0, check=0, mode=0;
       !mode && i<="" &&="">
       i++)
       { // 反复接收字节直至方式位为 1 或出错
       // 接收到的数据存在全局数组 recBuff[]里
          if (!ReceiveByte(recBuff+i, &mode))
          //超时。外设可用超时表示 NAK
             err = i?ERR_TIME_OUT:NAK;
          else if (i==0 && recBuff[i]==NAK
          && mode)
          // 收到 NAK
             err = NAK;
          else if (!mode)
          //方式位为 0 表示还有数据
```

```
            check += recBuff[i];
        }   // for i

        if (!err) {     // 未发生错误
            if (!mode) {
                    // 收完 36 个字节还未结束
                TransmitByte(NAK, 0);
                err = ERR_NO_MODE_BIT;
            } else if (i>1) {
                    // 收到数据块
                if (check != recBuff[i-1]) {
                    //校验和错
                    TransmitByte(NAK, 0);
                    err = ERR_CHECKSUM;
                } else {
                    //一切正常,发送 ACK 后跳出循环
                    TransmitByte(ACK, 0);
                    break;
                }
            } else
                        //收到外设传来的 ACK
                break;
        } // if (!err)
        Wait(T_RESPONSE);   //防止与外设数据冲突
    } // for j
            //返回接收到的数据块大小或出错代码
    return err?err:(i-1);
}
```

本文使用 DS5002FP 实现了对 MDB 总线的控制与访问。通过将 MDB/ICP 协议进行分解,很好地实现了总线驱动。实践证明该驱动程序稳定、可靠,大大降低了上层界面开发的难度,提高了系统的可维护性,节约了成本