

该程序只是部分处理程度，为大家提供参考设计。

主程序

```
void main(void)
{
    BYTE c;
    BYTE cCntr;
    bool bProcData;
    bool bRcvFinish;
    SystemInit();           //系统初始化
    EA = 1;
    cCntr = 0;
    cThirdBuf[0]=1;        //初始化接收数据缓冲不相等，第一次数据到来切相等才显示
    cThirdBuf[2]=2;
    cThirdBuf[4]=3;
    for(;;)                //主循环体
    {
        if(!bEmpty)        //判断是否有数据收到
        {
            c = GetData();
            cpBuf[cByteCntr] = c;
            bRcvFinish = false;

            if(!cByteCntr)  //收到第一个数据
            {
                bReverse = false;
                if(0x50 == (c & 0xF0))    // 判断是否是倒车模式
                {
                    bReverse = true;      //模式标志为置 1
                    bBaCheFlag=false;
                    cByteCntr++;          //接收数据缓冲加 1
                }
                else                // 扒车模式
                {
                    cpBuf[1] = 0;
                    cpBuf[2] = 0;
                    bBaCheFlag=true;
                    bRcvFinish = true;
                    bReverse = false;
                }
            }
        }
        else                //数据处理
        {
            if(bReverse)        //是倒车模式
```

```
{
    cByteCntr++;
    if(3 == cByteCntr)          //3 字节接收完 cByteCntr = 0;
    {
        bRcvFinish = true;
        cByteCntr = 0;
    }
}
else                            //是拔车模式
{
    cByteCntr = 0;
}
}

if(bRcvFinish)                  //数据接收完成处理标志
{
    cThirdBuf[cCntr] = cpBuf[2];
    cCntr++;
    cThirdBuf[cCntr] = cpBuf[1];
    cCntr++;
    if(cCntr == 6)
    {
        cCntr = 0;
    }
    if(cThirdBuf[0] == cThirdBuf[2]) //判断接收的数据是否上次相等，不相等
    {                                //显示不改变，必须两次相等才显示。防止
        cpBuf[2] = cThirdBuf[0]; //显示因干扰跳变。
        cpBuf[1] = cThirdBuf[1];
        bProcData = true;
    }
    else
    if(cThirdBuf[0] == cThirdBuf[4])
    {
        cpBuf[2] = cThirdBuf[0];
        cpBuf[1] = cThirdBuf[1];
        bProcData = true;
    }
    else
    if(cThirdBuf[4] == cThirdBuf[2])
    {
        cpBuf[2] = cThirdBuf[2];
        cpBuf[1] = cThirdBuf[3];
        bProcData = true;
    }
}
```

```

        else
        {
            bRefresh = false;
            bProcData = false;
        }
    }
}
if(bProcData)                //数据处理标志
{
    ProcessData();           //数据处理子程序
    bRefresh = true;
    bProcData=false;
}
if(bTimer)                   //定时时间到刷新显示
{
    bTimer = false;
    Beep();                  //驱动嗒鸣器
    RefreshDisp();          //刷新扫描程序
}
else if(bRefresh)           //有新的数据需要显示时刷新显示
{
    bRefresh=false;
    RefreshDisp();          //刷新扫描程序
}
}
}

//数据处理解码子程序
void ProcessData(void)
{
    bool bPark, bStopFlag;   //局部变量
    BYTE cSensor[4];
    BYTE n, n1, t1, t, c,n2;
    if(bReverse) // 倒车模式
    {
        bPark = false;
        ThirdByte = cpBuf[2]; //将倒车模式的第三字节取出（距离参数）
        cHighByte = 0x60 & ThirdByte; //显示的高位（2.05 米中的 2）
        cHighByte = cHighByte>>5;
        n1 = cpBuf[0] & 0x0C; // 取道危险等级最高的通道号
        n1 >>= 2;
        c = cpBuf[1]; //
        for(n = 0; n < 4; n++)
    }
}

```

```

    {
        t = c & 3;
        cSensor[3 - n] = DoubleLedTab[3 - n][t];    // 各通道的危险程度指示灯 ( 双色
灯 )

        if((3 - n) == n1)
        {
            for(t1 = 0x38, n2 = 0; n2 != t; n2++)
            {
                t1 <<= 1;
            }
            cDispBuf[3] = t1;
            t1 = ~cpBuf[0];
            t1 &= 3;
            cDispBuf[3] |= t1;
            cBeepType = t;
            if(cBeepType == 3)
            {
                bPark = true;
            }
        }
        c >>= 2;
    }
    cDispBuf[4] = (cSensor[0] | cSensor[2] | cSensor[3] | cSensor[1]);

    if(bPark)
    {
        bStopFlag=true;
        cStopCounter=0;
    }
    else
    {
    }

    if(bDC0)                                // 显示最低位为 0 或为 5 ( 2.05 米中的 5 )
    {
        cDispBuf[2] = 0x92;
    }
    else
    {
        cDispBuf[2] = 0xC0;
    }
    ThirdByte >>= 1;
    ThirdByte &= 0x0F;
    if(bStopFlag)                            //在停车状态下

```

```
{
    if(!cHighByte) && (ThirdByte < 6)
    {
    }
    else
    {
        cStopCounter++;
        if(cStopCounter>9) //9 次大于 0.6 米才退出停车状态
        {
            bStopFlag=false;
            cStopCounter=0;
        }
    }
}
else
{
//    cStopCounter--;
    switch(cHighByte) // 显示高位显示驿码
    {
        case 0:
            cDispBuf[0] = 0x40; //0
            break;
        case 1:
            cDispBuf[0] = 0x79; //1
            break;
        case 2:
            cDispBuf[0] = 0x24; //2
            break;
        case 3:
            cDispBuf[0] = 0x30; //3
            break;
        default:
            cDispBuf[0] = 0xFF; //不显示 ;
            break;
    }
    if(0x0F == ThirdByte) //显示 ---
    {

        cDispBuf[0] = Minus;
        cDispBuf[1] = Minus;
        cDispBuf[2] = Minus;
        bMUTE = true;
    }
}
else
```

```

    {
        if(cHighByte>=2)          //大于 2 米不响
        {
            bMUTE = true;
        }
        else
        {
            bMUTE = false;
        }
//        bMUTE = false;          //大于 2 米响
        if(ThirdByte < 10)
        {
            cDispBuf[1] = DispSeg[ThirdByte];
        }
    }

    if(bStopFlag)                //显示—P—
    {
        cDispBuf[0] =Minus;
        cDispBuf[1] = CharP;
        cDispBuf[2] = Minus;
        cBeepType = LevelStop;
        cBeepCntr = BeepFreq8Hz;
        bBeep = SOUND;
        bMUTE = false;
    }
}          // 倒车模式

else    // 扒车模式
{
    cDispBuf[3] = 0xFF;
    cDispBuf[0] = DispSeg[8];
    cDispBuf[1] = DispSeg[8];
    cDispBuf[2] = DispSeg[8];
    c = cpBuf[0];
    c &= 0x0F;
    cBeepType = LevelStop;
    bMUTE = false;
    switch(c)
    {
        case 0:
            cDispBuf[4]=0x55;
            bMUTE = true;

```

```
        break;
    case 1:
        cDispBuf[4]=0x65;    //4 号探头
        break;
    case 2:
        cDispBuf[4]=0x59; //3 号探头
        break;
    case 4:
        cDispBuf[4]=0x95;    //2 号探头
        break;
    case 8:
        cDispBuf[4]=0x56;    //1 号探头
        break;
    }
}
}
```