FreeCars2.1 双电磁方案

FreeCars 电子科技
Http://FreeCars.taobao.com
Http://FreeCars.GitHub.IO
QQ 讨论群：384273254,149168724
2015-3-2

目录

一、 必要的硬件

使用 FreeCars 双电磁方案，您可能需要以下的硬件模块

1.FreeCars 电磁组三蓝牙通信套装

2.FreeCars 电磁组四蓝牙通信方案

3.FreeCars 鸳鸯测距模块

二、 预备知识

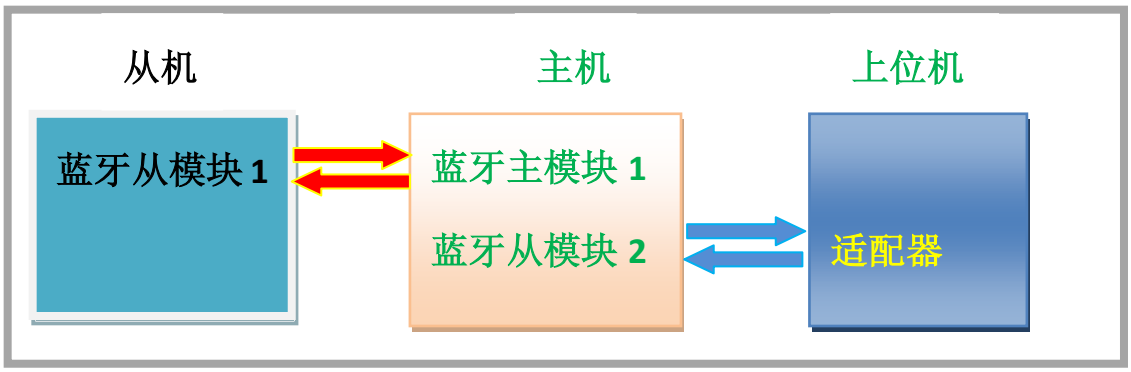
1. 学习使用 FreeCars 双电磁方案，您必须先学会使用 FreeCars 示波器的使用，请先使用单蓝牙、单个单片机板子通信方式，调好后再来学习使用三蓝牙方案。
2. 请先学会蓝牙模块密码、主从模式的配置，理解其含义
3. 学会一般的串口调试方法，学会用调试助手查看串口中的数据
4. 您需要足够的耐心和毅力来学习 FreeCars 三蓝牙方案，否则请选择 FreeCars 四蓝牙方案

三、 关于三蓝牙方案和四蓝牙方案

1. 三蓝牙方案

三蓝牙方案就是使用三个蓝牙模块，一个适配器实现双车双向通信以及双车和电脑的双向通信。其中两个蓝牙模块实现双车的通信，一个蓝牙模块实现与上位机的通信。

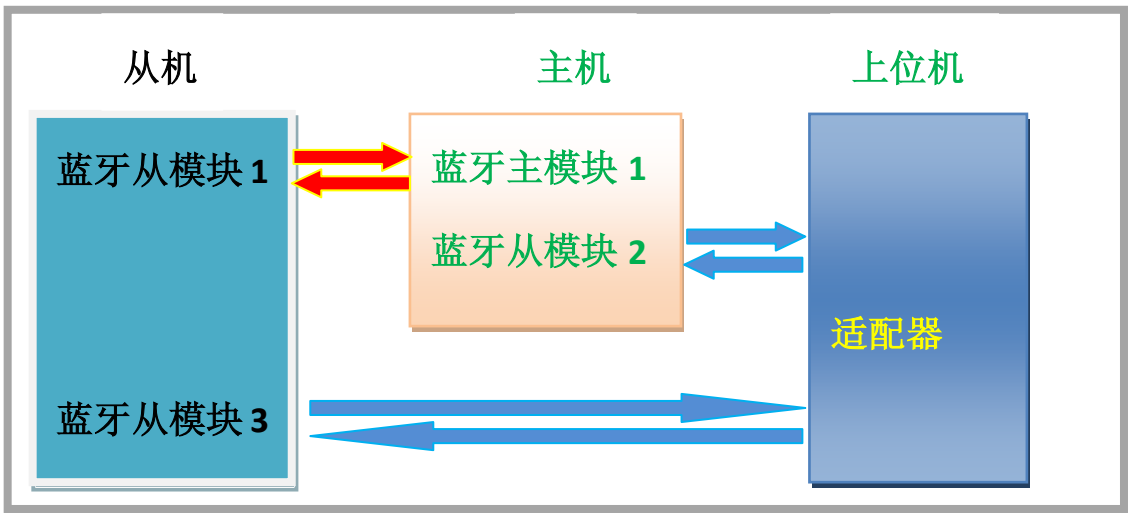
核心：从机的数据经过主机转发给上位机。



2. 四蓝牙方案

四蓝牙方案使用四个蓝牙模块，一个适配器实现双车双向通信以及双车和电脑的双向通信。其中两个蓝牙模块实现双车通信，两个蓝牙模块分别和电脑连接实现和上位机的通信。我店适配器可以同时连接多个蓝牙设备。

核心：将两个车和电脑的通信分开，分别和电脑通信。方案简单，程序易懂好实现



3. 关于 24L01 和无线串口

我们认为这两种硬件都可以用在双车通信中，而且 24L01 十分便宜。但是我们却没有推荐这两种通信方式，原因主要是这样的：第一，他们都是半双工的，无法实现实时双向传输；第二，24L01 的程序比较麻烦，新手不好使用；第三，使用现成串口通信，比较方便。

四、 三蓝牙方案

FreeCars 上位机默认配置是三蓝牙方案。三蓝牙方案可以节省一个蓝牙，但是由于从机数据需要主机转发，因此协议比较繁多复杂，请初学者三思后取舍。

4.1 FreeCars 软件配置说明

第一次调协议，没有调出来前，不要修改任何配置，都使用默认即可。

为了方便，FreeCars2.1 的三蓝牙模式将电感矩阵数据从示波器中分离出来，额外添加了协议，分为主机电感矩阵协议和从机电感矩阵协议。两者基本一样。但是上位机配置的电感数据必须和软件配置的电感数量一致才可以正确通信。

4.2 主机（主车 main）电感矩阵协议

由于一般 AD 都是 12 位的，因此用一个 16 位数（uint16）表示一个电感的 AD 值，也就是两个 8 位数。假设主车电感数量为 4 个，也就是

```
#define  MainCarInductorNum  4
```

那么协议就是

0xFA+0x37+AD1+AD2+...+ADx+SUM

0xFA+0x37 是数据头，告知上位机上来的数据是从车的电感矩阵

ADx(ADx.1,ADx.2)是一个 16 进制数据，要分为两个 8 位数据

因此完整的协议是

0xFA+0x37+AD1.1+AD1.2+AD2.1+AD2.2+...+ADx.1+ADx.2+SUM

其中 SUM =(0xFA+0x37+AD1.1+AD1.2+AD2.1+AD2.2+...+ADx.1+ADx.2)%256

代码如下

```
void sendMainCarBufferToMagViewer(void)
{
  uint8 i,sum=0;
  USendOneByte(FreeCarsUARTPort,0xFA);//数据头
  USendOneByte(FreeCarsUARTPort,0x37);
  sum += 0xFA + 0x37;//全部数据加入校验
  for(i=0;i<magMainDataNum;i++)
  {
    USendOneByte(FreeCarsUARTPort,magMainBuffer[i]);
    sum += magMainBuffer[i];
  }
  USendOneByte(FreeCarsUARTPort,sum);//发送校验位
}
```

4.3 从机（从车 slave）电感矩阵协议

这个协议和主车的基本一样，不同的就是数据头不相同，用来告知上位机发来的数据是。数据头是 0xFB+0x38，因此代码是：

```
void sendSlaveCarBufferToMagViewer(void)
{
  uint8 i,sum=0;
  USendOneByte(FreeCarsUARTPort,0xFB);//数据头
  USendOneByte(FreeCarsUARTPort,0x38);
  sum += 0xFB + 0x38;//全部数据加入校验
  for(i=0;i<magSlavDataNum;i++)
  {
    USendOneByte(FreeCarsUARTPort,magSlavBuffer[i]);
    sum += magSlavBuffer[i];
  }
  USendOneByte(FreeCarsUARTPort,sum);//发送校验位
}
```

4.4 主机示波器协议补充说明

FreeCars2.1 的示波器协议没有修改，与 FreeCars2.0 是一致的。但是三蓝牙方案中，需要观察的从车的数据发送给了主机，占用了主机示波器的一些通道。因此主机部分通道是给了从机用的。距离可以查看 Debug.c 中的 slaveScopeDataToFreeCars 函数，默认 SlaveScopeDataNum 是 8

```
uint8 scopeDataSent = 0;
void slaveScopeDataToFreeCars(void)
{
  int i;
  for(i=0;i<SlaveScopeDataNum;i++)
  {
    push(i,slaveScopeDataBuffer[i]);
  }
}
```

4.5 参数调试补充说明

我们在调试时候，可能也同时需要调试从车的参数，比如舵机以及电机的 PID。我们没有提供主车到从车的参数数据的发送协议，需要这个功能的用户可以参考我们的协议进行自行编写。我们不提供的主要目的是，不让这个方案看起来更复杂！

4.6 从机示波器数据到主机的协议以及解析

我们需要观察从车上的数据，比如车子速度等，就需要将从车的数据经过主车发送到上位机。因此存在从车的发送协议以及主车解析这个数据的两个过程。
示波器的数据是 Int16 类型的，也就是-32768~32767，因此使用一个 16 位数来保存，与电感矩阵的协议基本一致，只不过数据头是 0xFC+0x31 以告知主机，从机发送过来的是示波器数据，默认从机通道数量为 8，例程如下:(uart.c,uart.h)

```
void sendDataToScope(void)
{
  uint8 i,sum=0;
  //使用轮询的方式发送数据，当数据未发送，程序停在此处直到发送完成
  USendOneByte(MainSlaveRxTxUARTPort,0xFC);
  USendOneByte(MainSlaveRxTxUARTPort,0x31);
  sum+=(0xFC);       //全部数据加入校验
  sum+=(0x31);
  for(i=0;i<FreeCarsDataNum;i++)
  {
    USendOneByte(MainSlaveRxTxUARTPort,uSendBuf[i]);
    sum+=uSendBuf[i];          //全部数据加入校验
  }
  USendOneByte(MainSlaveRxTxUARTPort,sum);
}
```

主机需要解析从机发送过来的数据，就需要在中断里面处理发送过来的数据。具体步骤如下：接收到一个数，将其放入缓冲区，如果发现数据总量大于等于一帧的总数据量，那么就进行判断。用上面的例子来说就是，总数据量是 2 + 8*2+1(2 个数据头，8 个 16 位数据，1 个数据尾)=19,如果发现第一个数据为 0xFC 且第二个数是 0x31，那么就认为接收到从机示波器数据，进一步求所有数据的和，得到 SUM，如果 SUM 和最后一个数相等，那么就认为接收到的数据完全是对的。例程如下(建议直接看 k60 工程)：

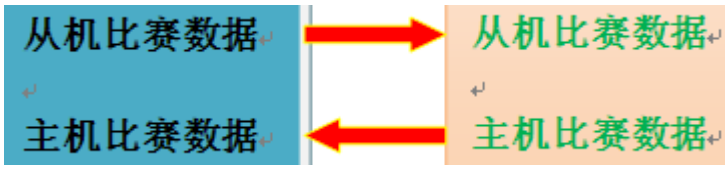
```
void UART1IRQHandler(void)
{
  static int stack=0;
  uint8 data,sum,i;
  if((UART_S1_REG(UART1) & UART_S1_RDRF_MASK))//是接收中断
  {
     data = UartGetOneByte(UART1);
     if(stack < MainSlaveRxBufferLen)
     {
       slaveToMainBuffer[stack++] = data;//放到缓冲区
       if(stack >= slaveScopeTotalNumInOneSeries //大于总数才开始解析
          &&(slaveToMainBuffer[stack - slaveScopeTotalNumInOneSeries] == 0xFC)
            &&(slaveToMainBuffer[stack - slaveScopeTotalNumInOneSeries + 1] == 0x31) //数据头
            )//从机发送过来的 示波器 数据解析，这些数据需要主机转发到电脑
       {
         sum=0;
         for(i=stack - slaveScopeTotalNumInOneSeries;i<stack-1;i++)//不含最后一个校验位
```

```
        {
          sum += slaveToMainBuffer[i];
        }
        if(sum == slaveToMainBuffer[stack-1])//解析成功
        {
          for(i=0;i<SlaveScopeDataNum;i++)
          {
            slaveScopeDataBuffer[i]=(slaveToMainBuffer[stack - slaveScopeTotalNumInOneSeries
+ 2 + i*2]<<8) + slaveToMainBuffer[stack - slaveScopeTotalNumInOneSeries + 3 + i*2];
          }
          //debug
          slaveScopeDataToFreeCars();
          stack=0;//成功了，重新开始
        }
      }
    }else
    {
      stack = 0;//缓冲区满了还没有解析导数据，清零从新开始
    }
  }//是接收中断
}//void
```

4.7 从机电感矩阵到主机的协议以及解析

从机的电感数据要发送到电脑来查看，就需要发送到主机，然后由主机转发给电脑。这个协议和示波器转发协议基本一致，只不过是数据头不一样，用以识别是电感数据而不是示波器数据。具体请看例程。

4.8 主从比赛实时数据的协议以及解析



该部分分为从机发送主机解析和主机发送从机解析两部分。可以看到两部分是对称的，基本一样。要传输的数据是比赛时必须使用的，因此传输速度要快，要做到尽可能实时性。这个协议与示波器协议也是一样的，只不过是协议头不同，实际数据也不同，这里用一个 8 位表示一个数据，用户如果需要发送一个 16 位数据，用两个 8 位即可。关于程序和数据头，请看我们的例程。

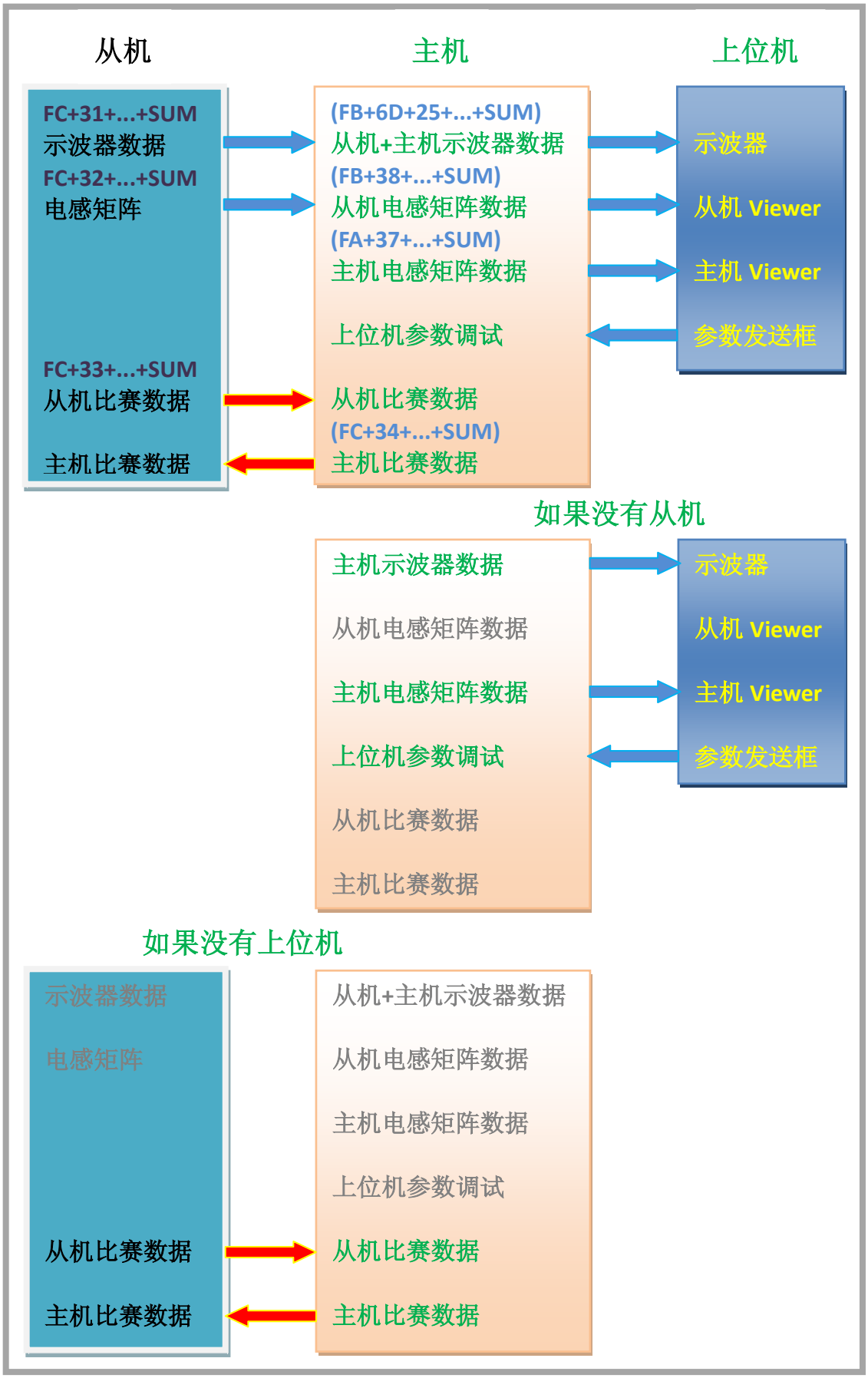


图 1 三蓝牙方案协议示意图

五、 四蓝牙方案

四蓝牙方案其实是一般单车蓝牙通信方案简单的乘以 2，只不过中间多加两个蓝牙实现两个车子的通信。请先在软件配置中配置为四蓝牙方案，再在电磁组模块中点击四蓝牙双电磁方案即可。

此时会再次启动一次 Freecars 上位机，四蓝牙方式，就是使用两个 FreeCars 上位机分别控制小车。这样使得方案及其简单。这里值得注意的是，四蓝牙方案的通信协议依然使用 FreeCars2.0 旧版本协议，无需理会三蓝牙方案中提到的协议，电感依然占用示波器通道。

六、 写在后面的话

FreeCars 双电磁中的三蓝牙方案看起来是比较复杂的，可能很多人会望而生畏。但是我们却把三蓝牙方案作为重中之重，花了很多时间和精力来做上位机和例程，我们不仅仅是为了帮同学们节省一个蓝牙的金钱，更希望同学们可以在智能车制作中学习到更多的东西。我们切实希望选择了三蓝牙方案的同学们，能够不烦躁、不着急，耐心调试，功夫不负有心人，我们一定可以成功！

当然，我们不建议编程能力不够强的新手同学选用三蓝牙方案，以免浪费大家宝贵的时间。选择四蓝牙方案，依然需要一定的时间去学习相关东西，也能学习到很多很多关于通信、调度的东西，这或许对于新手，也是一件十分不错的事情。

比赛并非只有胜负，我们相信你一定行！