|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 文件编号 | |  | 密 级 |  | |
| 文件类型 | |  | 作 者 |  | |
| 项目代码 | |  | 适用范围 |  | |
| **Leddar-Linux应用程序开发说明** | | | | | |
| 关联文档 | | | | | |
| **文件编号** | **文件名称** | | | | **版本** |
|  | 《EasyARM-iMX28x\_Linux.pdf》 | | | |  |
|  | 《Modbus\_over\_serial\_line\_V1\_02》 | | | |  |
|  | 《 54A0028-1 LeddarVu User Guide》 | | | |  |
|  |  | | | |  |
|  |  | | | |  |
|  |  | | | |  |
|  |  | | | |  |
|  |  | | | |  |
|  |  | | | |  |
|  |  | | | |  |
|  |  | | | |  |
|  |  | | | |  |
|  |  | | | |  |
|  |  | | | |  |
|  |  | | | |  |
|  |  | | | |  |
|  |  | | | |  |

**修订记录**

| **版本** | **修订内容说明** | **修订人** | **修订日期** |
| --- | --- | --- | --- |
| 1.0 | 新建 | 梁敦辉 | 2017/2/24 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 串口配置

参考文档《EasyARM-iMX28x\_Linux.pdf》5.3的串口编程。

## 串口配置

串口属于终端设备，其接口属性用termios结构描述

struct termios {

tcflag\_t c\_cflag /\* 控制标志\*/

tcflag\_t c\_iflag; /\* 输入标志\*/

tcflag\_t c\_oflag; /\* 输出标志\*/

tcflag\_t c\_lflag; /\* 本地标志\*/

tcflag\_t c\_cc[NCCS]; /\* 控制字符\*/

};

具体结构体的成员的属性，请参考参考文档。

控制雷达的串口，具体串口参数是可以设置的，默认参数是 ：

波特率： 115200

数据位： 8

奇偶校验位： 0

停止位： 1

流控： 无

因此，我们在linux中的串口配置可以使用serial\_open(LEDDAR\_DEV\_NAME, 115200, 0, 1)；函数来打开串口设备文件，并且配置串口参数，此函数函数体位于uart.c文件中，下面函数主要通过termios结构体写进参数。更多串口配置详细内容，请参考《EasyARM-iMX28x\_Linux.pdf》5.3的串口编程。

# Modbus协议详解

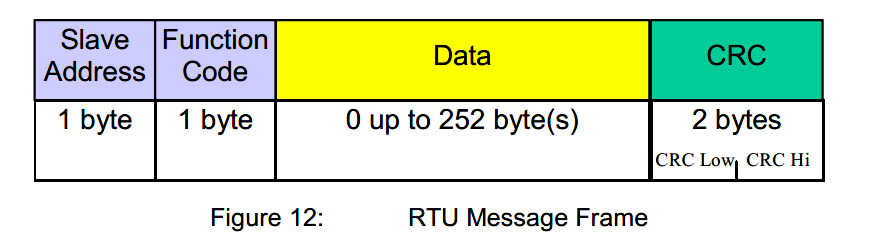
参考文档《Modbus\_over\_serial\_line\_V1\_02》

## Modbus简介

Modbus是由Modicon（现为施耐德电气公司的一个品牌）在1979年发明的，是全球第一个真正用于工业现场的总线协议。主要包含两种模式：一种模式是ASCII（美国信息[交换码](http://baike.baidu.com/view/5481005.htm)），另一种模式是RTU（远程[终端设备](http://baike.baidu.com/view/34654.htm)）。雷达使用的是RTU模式，因此这里不介绍ASCII模式，详情请参考modbus官方文件《Modbus\_over\_serial\_line\_V1\_02》

## RTU模式

Modbus的RTU模式的数据帧格式如下：



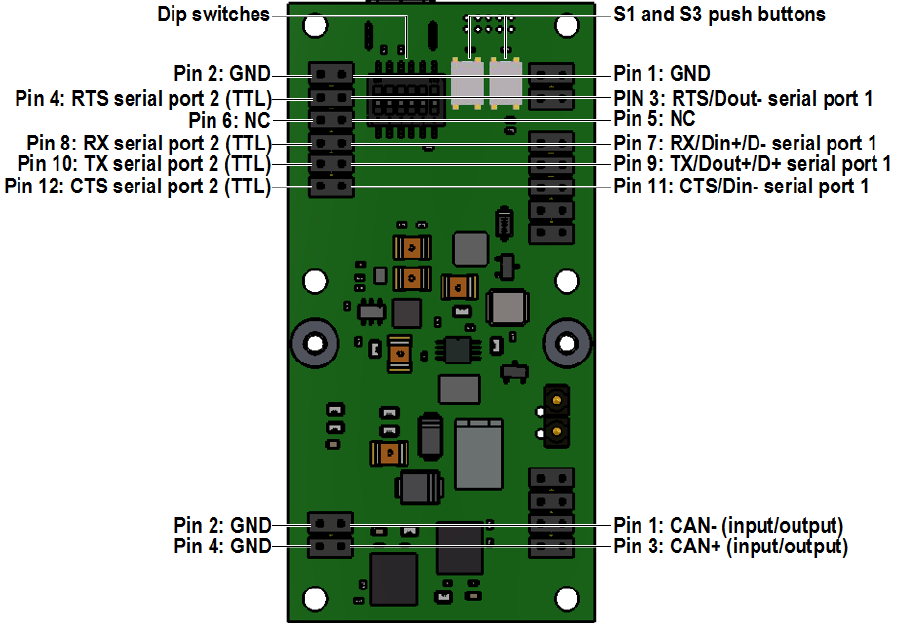
第一个是从机地址，第二个是功能吗，第三个是帧的数据，第四个是2字节CRC校验，低8位在前面。好比如雷达的读取距离命令（参考《54A0028-1 LeddarVu User Guide》）

需要发送的是 0x01 0x41 0xc0 0x10 ，0x01为雷达地址，0x41为功能码，0xc0、0x10分别为CRC 校验的低8位和高8位。

# 雷达读写

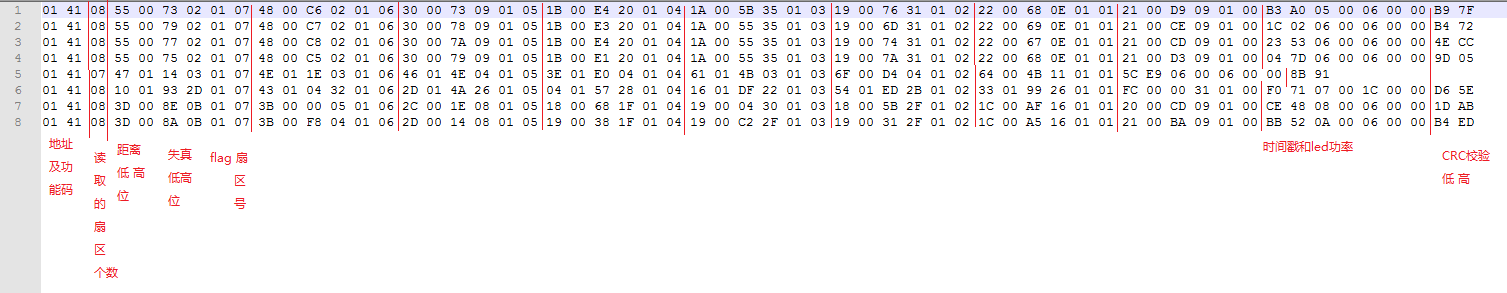
参考文档《54A0028-1 LeddarVu User Guide》

## 雷达的串口测试

雷达的引脚图如下：

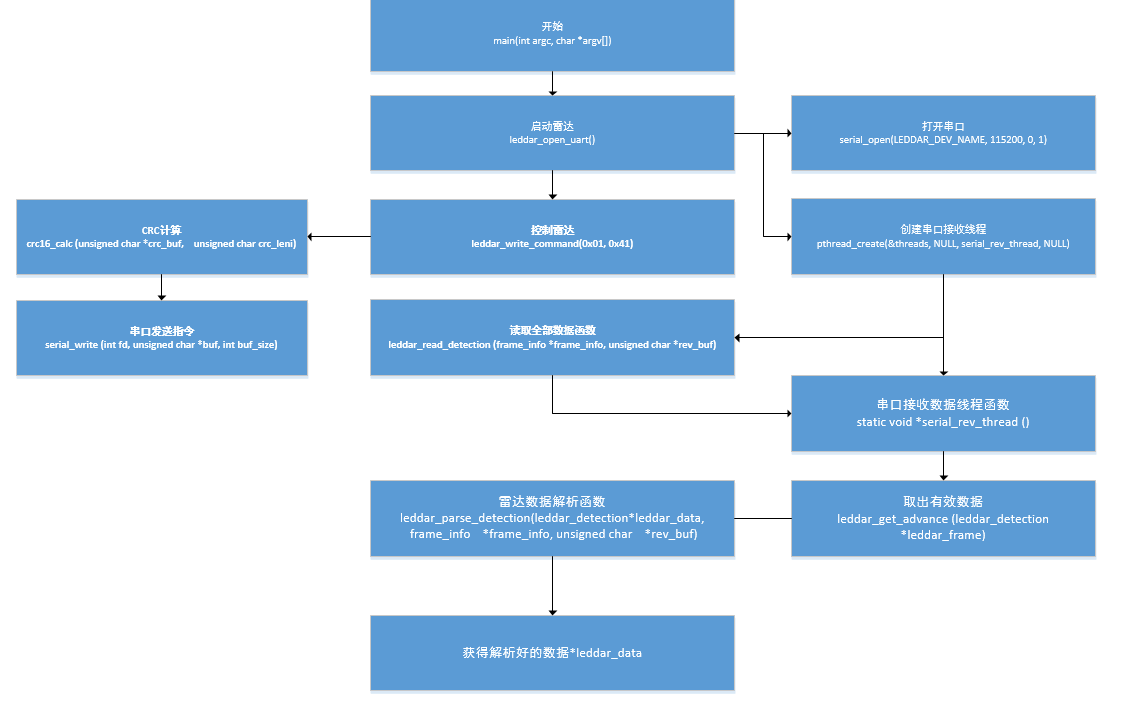
 通过Pin8 与Pin9分别连接到USB串口转换工具的 TX 和RX 处，打开串口调试助手，配置如下：

根据《54A0028-1 LeddarVu User Guide.pdf》P44页末，读取雷达距离，衰减等数据所用的fuction code 为 0x41，根据Modbus协议，发送的命令为 ：0x01 0x41 0xc0 0x10 四位十六进制数据，0x01为雷达在总线中的通讯，0x41为功能吗，最后两位分别为CRC校验（低位）和CRC校验（高位）。

通过串口调试助手发送数据给雷达后，根据Modbus协议返回数据，数据详情：《54A0028-1 LeddarVu User Guide》P44-45，下图为多次读取数据的返回：

数据格式标明，原图也在该目录下。

## 雷达基于linux平台的读写

流程图如下：

程序采取异步读取功能，首先初始化串口，配置串口参数，并且创建读线程，读线程通过select()函数来监视串口是否有数据可读，当读取到数据的时后，select()退出阻塞，并且调用读取数据存放在数据缓存区，然后进行数据解析，当数据帧不完整的时候，继续select()阻塞，再次读取，知道完整的帧，然后解析出数据的含义，同时要在主线程中发送雷达0x41命令，串口的读取数据线程才有数据的返回。

在linux中，串口设备的read函数可能读取数据不足：好比如：发送0x41命令去读取雷达的时候，正常情况下返回的数据是60字节，但有可能存在着一下情况，第一次调用read的时候，只读取了30个，还有15个在缓存中，还有15个正在接收，因此可能要调用多次read才能正确的读取命令。因此采用异步线程select这种方式来进行读取。