AN0131
应用笔记

Modbus on AT32 MCU

前言

本应用笔记介绍了如何将FreeMODBUS协议栈移植到AT32F43x单片机方法。本文档提供的源代码演示了使用Modbus的应用程序。单片机作为Modbus从机，可通过RS485或RS232与上位机相连，与Modbus Poll调试工具（Modbus主机）进行通讯。

注：本应用笔记对应的代码是基于雅特力提供的V2.x.x 板级支持包（BSP）而开发，对于其他版本BSP，需要注意使用上的区别。

支持型号列表：

支持型号	AT32F435xx
	AT32F437xx
	AT32F425xx

目录

表目录

图目录

1 概述

Modbus是一种串行通信协议，是Modicon公司（现施耐德电气）于1979年为使用可编程逻辑控制器（PLC）通信而发表。如今Modbus已成为工业领域通信协议的业界标准，并且是工业电子设备之间常用的连接方式。

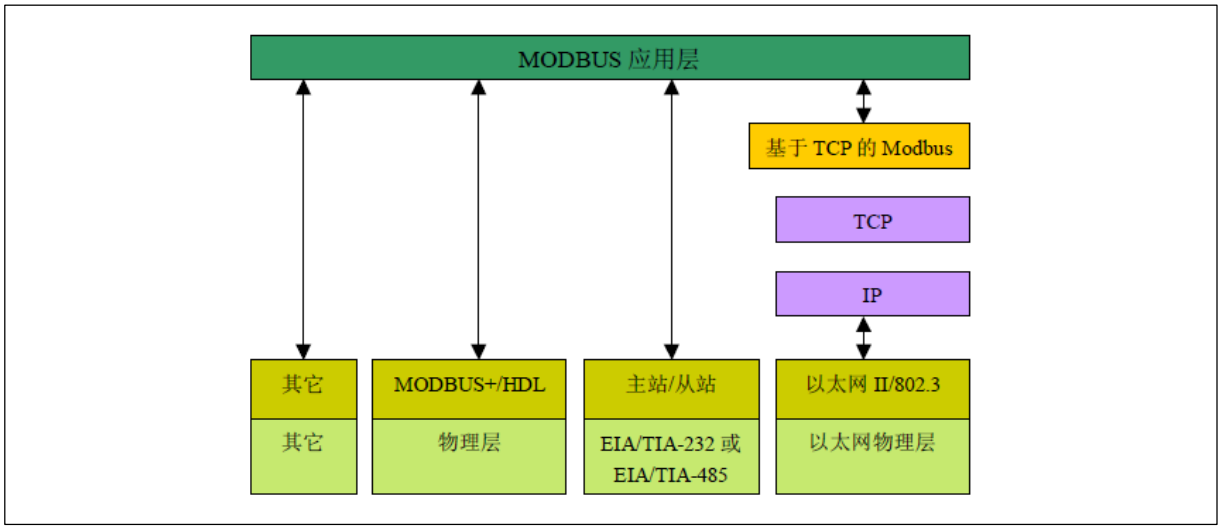
1.1 关于 Modbus 协议

Modbus协议使用的是主从的通讯技术，即由主设备主动查询和操作从设备。一般将主控设备方所使用的协议称为Modbus Master，从设备方所使用的协议称为Modbus Slave。典型的主设备包括工控机和工业控制器等；典型的从设备如可编程逻辑控制器（PLC）等。

MODBUS是OSI模型第7层上的应用层报文传输协议，它在连接至不同类型总线或网络的设备之间提供客户机/服务器通信。

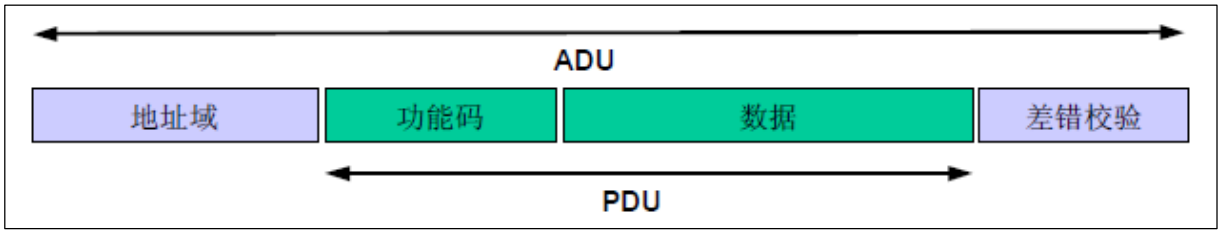
Modbus通讯物理接口可以选用串口（包括RS232、RS485等），也可以选择以太网口等。

图 1. Modbus 通信栈



MODBUS协议定义了一个与基础通信层无关的简单协议数据单元（PDU）。特定总线或网络上的MODBUS协议映射能够在应用数据单元（ADU）上引入一些附加域。

图 2. 通用 Modbus 帧



当服务器对客户机响应时，它使用功能码域来指示正常（无差错）响应或者出现某种差错（称为异常响应）。需要管理超时，以便明确地等待可能不会出现的应答。

图 3. Modbus 事务处理（无差错）

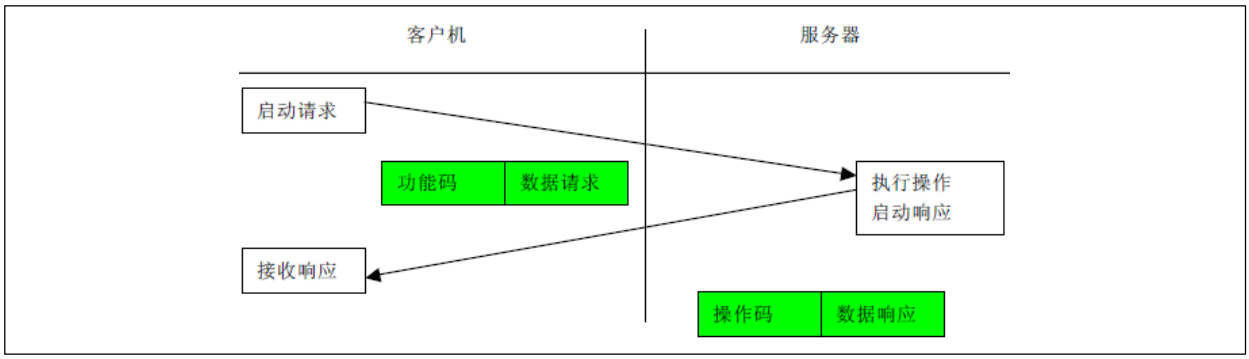
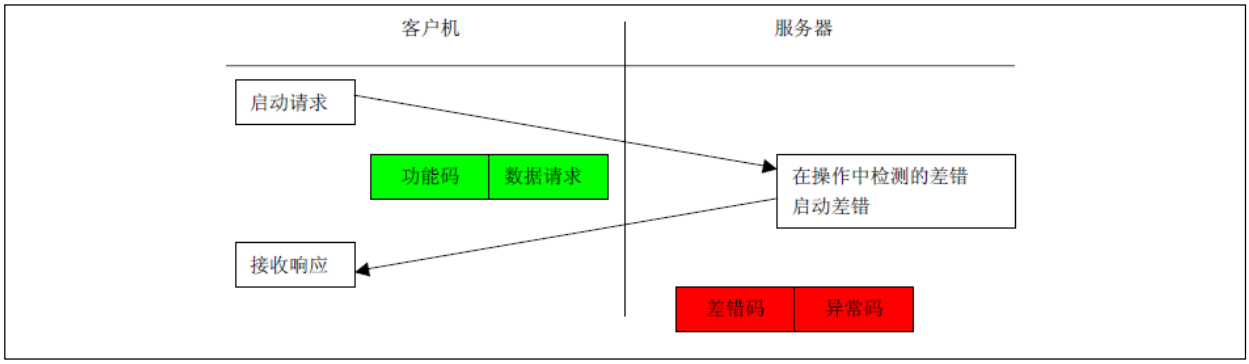


图 4. Modbus 事务处理（异常响应）



MODBUS使用一个‘big-Endian’表示地址和数据项。这意味着当发射多个字节时，首先发送最高有效位。例如，寄存器大小16-bit的值为0x1234，先发送的第一字节为0x12，然后为0x34。

MODBUS的数据模型以一系列具有不同特征表格上的数据模型为基础。四个基本表格为：

表 1. Modbus 数据模型

基本表格	对象类型	访问类型	内容
离散量输入	单个bit	只读	I/O系统提供这种类型数据
线圈	单个bit	读写	通过应用程序改变这种类型数据
输入寄存器	16bit 字	只读	I/O系统提供这种类型数据
保持寄存器	16bit 字	读写	通过应用程序改变这种类型数据

MODBUS有三类功能码：公共功能码、用户定义功能码、保留功能码。

公共功能码是较好地被定义的功能码，保证是唯一的、公开证明的，具有可用的一致性测试。

图 5. 公共功能码定义

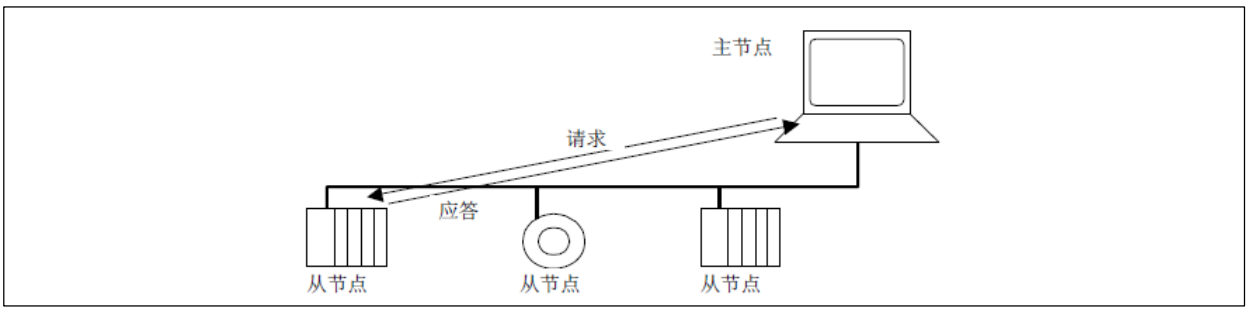
				功能码		
				码	子码	（十六进制）
数据访问	比特访问	物理离散量输入	读输入离散量	02		02
		内部比特或物理线圈	读线圈	01		01
			写单个线圈	05		05
			写多个线圈	15		0F
	16 比特访问	输入存储器	读输入寄存器	04		04
		内部存储器或物理输出存储器	读多个寄存器	03		03
			写单个寄存器	06		06
			写多个寄存器	16		10
			读/写多个寄存器	23		17
			屏蔽写寄存器	22		16
	文件记录访问		读文件记录	20	6	14
			写文件记录	21	6	15
封装接口			读设备识别码	43	14	2B

Modbus 串行链路协议是一个主从协议。在同一时刻，只有一个主节点连接于总线，一个或多个子节点（最大编号为247）连接于同一个串行总线。Modbus通信总是由主节点发起。子节点在没有收到来自主节点的请求时，从不会发送数据。子节点之间从不会互相通信。主节点在同一时刻只会发起一个Modbus事务处理。

主节点以两种模式对子节点发出Modbus 请求：

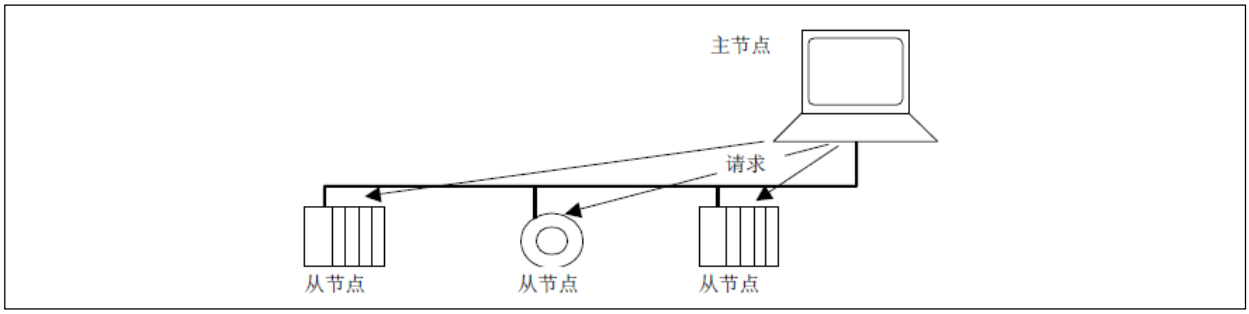
1、 单播模式，主节点以特定地址访问某个子节点，子节点接到并处理完请求后，子节点向主节点返回一个报文(一个'应答')。每个子节点必须有唯一的地址 (1 到 247)，这样才能区别于其它节点被独立的寻址。

图 6. 单播模式



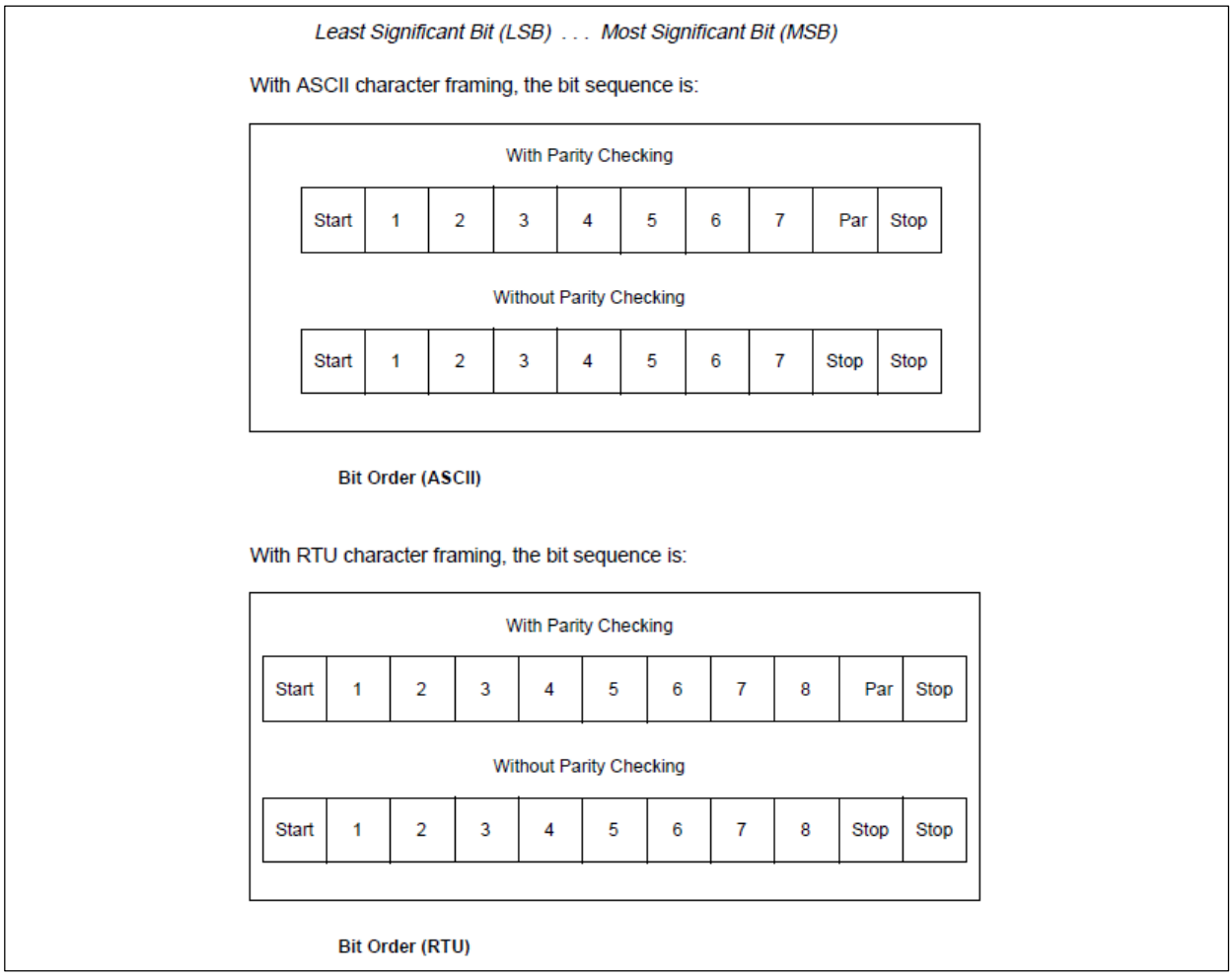
2、 广播模式，主节点向所有的子节点发送请求。对于主节点广播的请求没有应答返回。广播请求一般用于写命令。所有设备必须接受广播模式的写功能。地址 0 是专门用于表示广播数据的。

图 7. 广播模式



Modbus有两种串行传输模式被定义：RTU 模式（默认） 和 ASCII 模式。

图 8. RTU 和 ASCII 模式的位序列



Modbus 串行链路上所有设备的传输模式 (和串行口参数) 必须相同。

有关Modbus 实现和特性的详细信息，请访问Modbus 官网：https://modbus.org。

1.2 关于 FreeModbus 协议栈

FreeMODBUS是一个针对通用的Modbus协议栈在嵌入式系统中应用的实现。它提供了RTU/ASCII传输模式及TCP协议支持。FreeModbus遵循BSD许可证，这意味着用户可以将FreeModbus应用于商业环境中。目前FreeMODBUS只免费提供了一个Modbus从机节点的协议栈。该协议栈使用ANSI C编写，并且支持多个变量。

本应用指南将介绍如何在AT32F435单片机上，通过FreeMODBUS协议栈实现Modbus从机节点的主要功能，并提供基于AT32F43x_StdPeriph_Lib和FreeMODBUS协议栈的源代码。如结合AT32-Comm-EV Board和AT-START Board可以方便快速的搭建起基于RS485的Modbus从机节点。

1.3 关于 Modbus Poll 调试软件

Modbus Poll是一个Modbus主机模拟器，支持Modbus RTU、ASCI、TCP/IP传输模式。它可以协助开发人员调试Modbus从设备，测试和模拟Modbus协议的通信。通过多文档界面接口，可以同时监控多个Modbus从机设备、数据域等。每个窗口可简单地设定从设备ID、功能码、地址、长度和轮询间隔。支持Modbus数据模型的四个基本表格，支持Modbus的多个公共功能码。

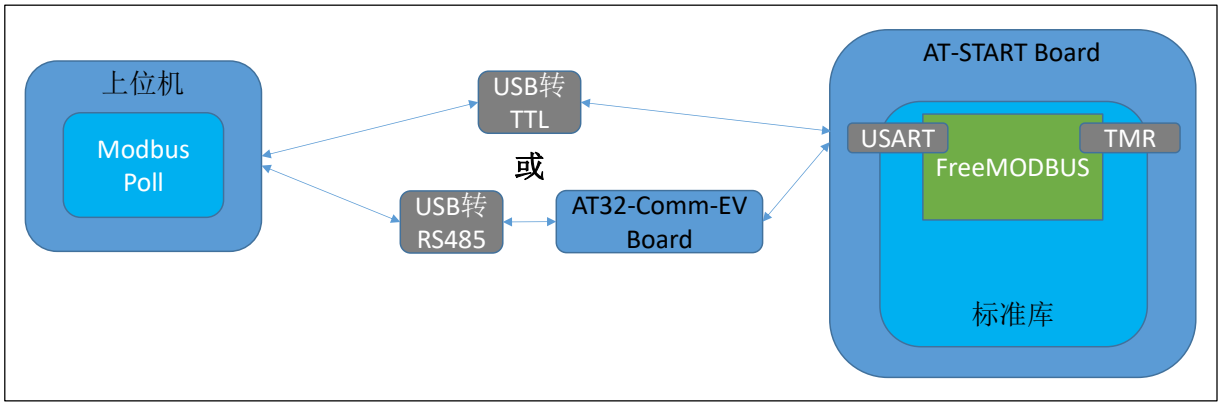
本应用指南中将下载安装Modbus Poll在PC机上作为Modbus主机，与AT-START Board（作为Modbus从机）通过USB转RS485的模块相连，实现一个完整的、可测试的Modbus通信网络。

2 AT32 硬件准备

硬件主要由AT32-Comm-EV Board和AT-START Board组成。

本应用指南提供的demo使用到的外设资源有USARTTMR等，用户可根据具体需要灵活配置和修改，使用RS232或RS485来实现Modbus物理层的连接。

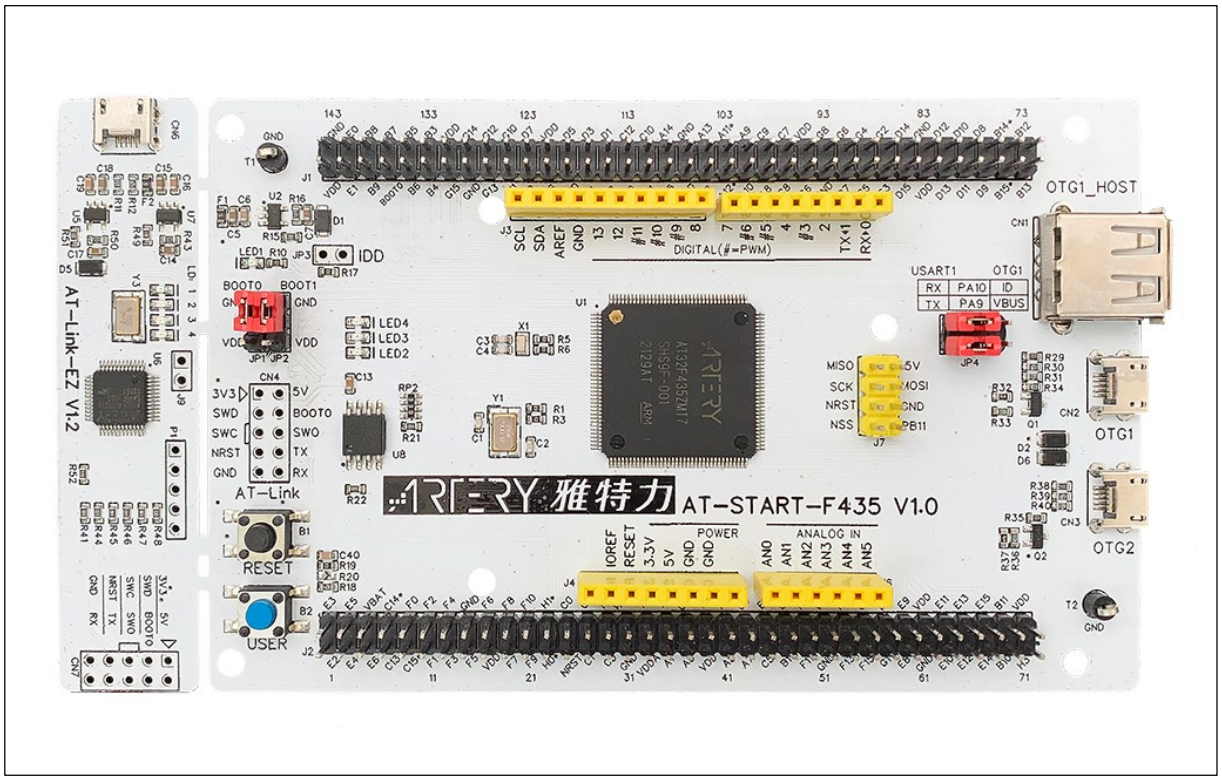
图 9. AT32 Modbus 结构原理图



- AT-START Board

当前提供例程基于AT-START-F435。可提供基于RS232的Modbus通信。

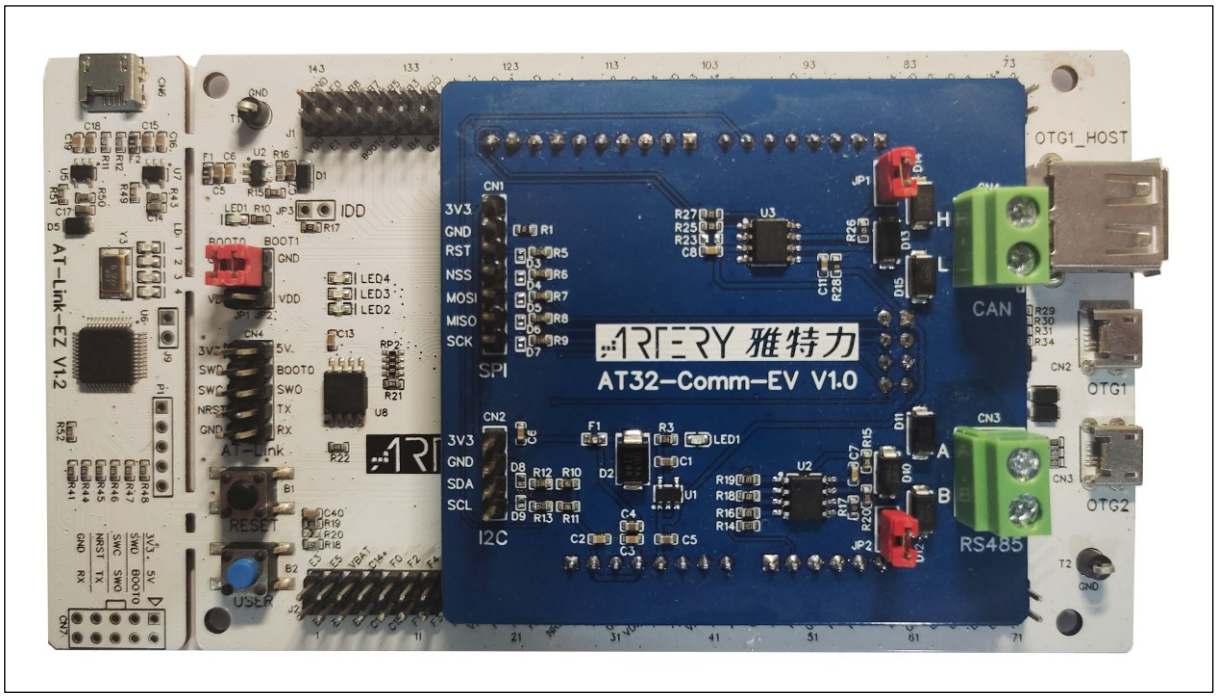
图 10. AT-START-F435 V1.0 实验板



- AT32-Comm-EV Board

可提供例程基于RS485的Modbus通信支持。

图 11. AT-START 结合 AT32-Comm-EV 使用



3 将FreeMODBUS移植到AT32上

3.1 基础工程准备

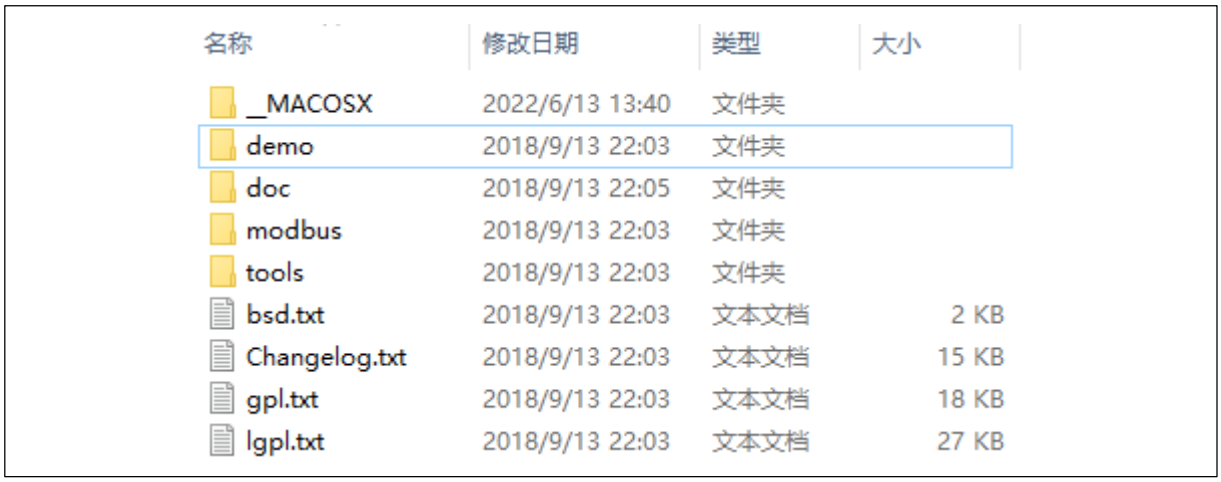
下载最新版本BSP&PACK文件，按照其应用指南进行安装及配置，本文档及例程均基于AT32F4xx_StdPeriph_Lib_V2.x.x的BSP&PACK文件进行开发。可借用at_start_f435文件夹下的temple工程来进行修改，更改文件夹及工程名为freemodbus，并准备在该工程内添加FreeMODBUS源码。

3.2 工程内添加 FreeMODBUS 源码

用户需前往FreeMODBUS官网或Github上下载最新版源码。源码包解压后，内有如下文件。

本文及所有移植例程均基于freemodbus-v1.6版本进行移植。

图 12. FreeMODBUS 源码文件



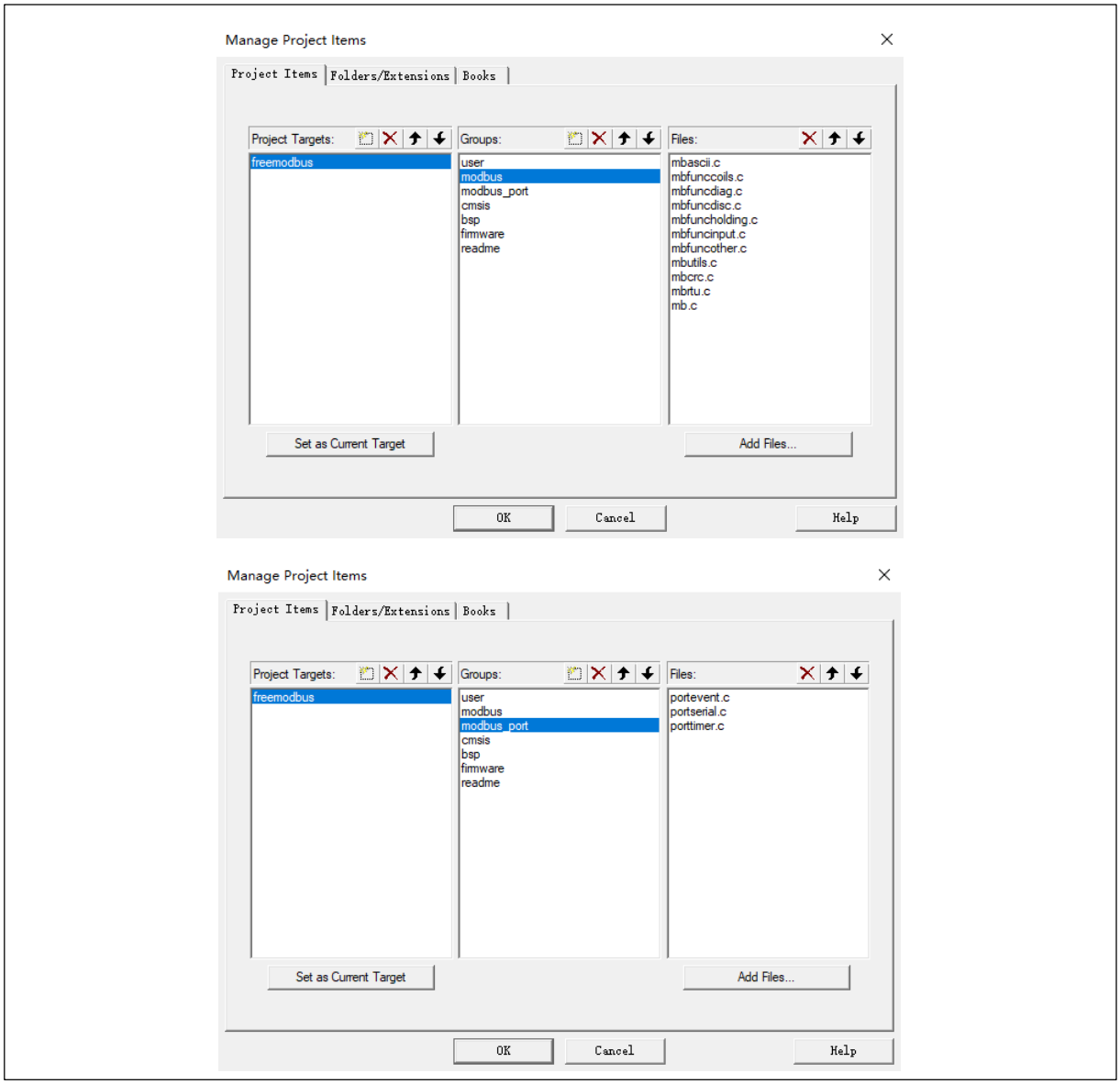
将源码包解压后，复制modbus文件夹和demo\BARE\port文件夹到前面的基础工程freemodbus目录下面，将port文件夹改名为modbus_port。如下图所示。

图 13. freemodbus 工程目录

打开工程文件，并按以下2个步骤添加到工程文件内。可参考AT提供的例程文件进行添加。

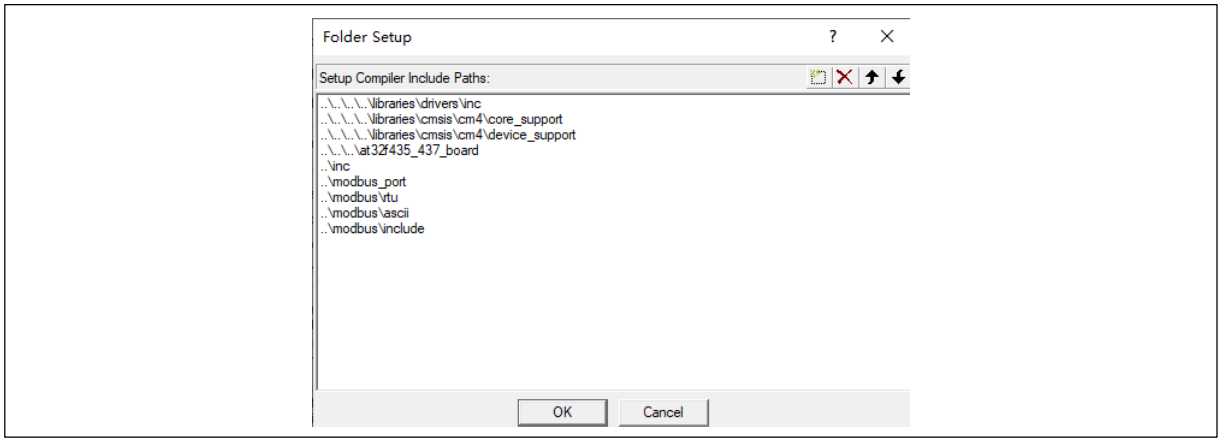
1、添加modbus和modbus_port内的所有.c文件（与tcp相关的除外）到工程项目内。

图 14. freemodbus 工程的项目



2、需将添加的.c文件所对应的.h文件的路径添加到工程的文件夹设置内。

图 15. freemodbus 工程的文件夹设置



3.3 工程代码的修改

1、修改“port.h”文件。在该文件中添加包含关于AT32 MCU的“at32f435_437.h”头文件。补全用于互斥操作的开关中断宏定义。屏蔽掉关于TRUE、FALSE的宏定义，因在AT BSP的头文件中已有定义，避免编译出错。

2、修改“portserial.c”和“porttimer.c”文件。在该文件中添加关于USART和TMR外设的底层驱动部分代码。用户可根据自己的硬件环境来自行修改，所以这里不进行具体说明，也可参考AT所提供的例程文件。

3、需特别注意的是，因USART寄存器DT的值里包含数据位和校验位，而FreeMODBUS源码里在读取USART接收的数据后，会把所有值都当作数据位来处理而出错，这可能是和其他厂家的MCU有差异，所以这里需在“mbascii.c”文件中去修改源码。具体可参考AT所提供的例程文件。

4、在工程中新建并添加“mbtask.c/.h”文件。该文件用于创建Modbus的通信任务（作从机）来调用FreeMODBUS协议栈的API层，并建立Modbus的数据模型（4个基本表格）用于与Modbus Poll（作主机）进行模拟通信测试。

“mbtask.c/.h”文件中实现的调用管理：

- 保持寄存器的读/写
- 输入寄存器的读取
- 线圈的读/写
- 离散量输入的读取

“mbtask.h”文件中定义的Modbus数据模型和建立通信所需的参数：

表 2. Modbus 的配置参数

参数	描述
MB_SLAVE_ADDRESS	设置设备的从机地址
MB_BAUDRATE	设置通信的波特率
REG_INPUT_START	输入寄存器的起始地址
REG_INPUT_NREGS	输入寄存器的数量
REG_HOLDING_START	保持寄存器的起始地址
REG_HOLDING_NREGS	保持寄存器的数量
REG_COILS_START	线圈的起始地址
REG_COILS_SIZE	线圈的数量
REG_DISCRETE_START	离散量输入的起始地址
REG_DISCRETE_SIZE	离散量输入的数量

3.4 设备功能的实现

1、在“mbtask.c”文件中编写void modbus_task(void)函数去调用协议栈的API层，来实现modbus从机任务的功能。

```
void modbus_task(void)
{
  eMBErrorCode       eStatus;

  eStatus = eMBInit(MB_RTU, MB_SLAVE_ADDRESS, 0, MB_BAUDRATE, MB_PAR_NONE);
  if(MB_ENOERR == eStatus)
  {
    printf("modbus init ok\r\n");
    eStatus = eMBEnable();
    if(MB_ENOERR == eStatus)
    {
      printf("modbus enable ok\r\n");
    }
    else
    {
      printf("modbus enable fail, error code: %u\r\n", eStatus);
    }
  }
  else
  {
    printf("modbus init fail, error code: %u\r\n", eStatus);
  }
  if(MB_ENOERR != eStatus)
  {
    printf("exit modbus task.\r\n");
    return;
  }
  printf("start modbus pooling..\r\n");
  for(;;){
    eMBPoll();
  }
}
```

2、在“main.c”文件中，由int main(void) 主函数去调用modbus_task()任务函数即可。

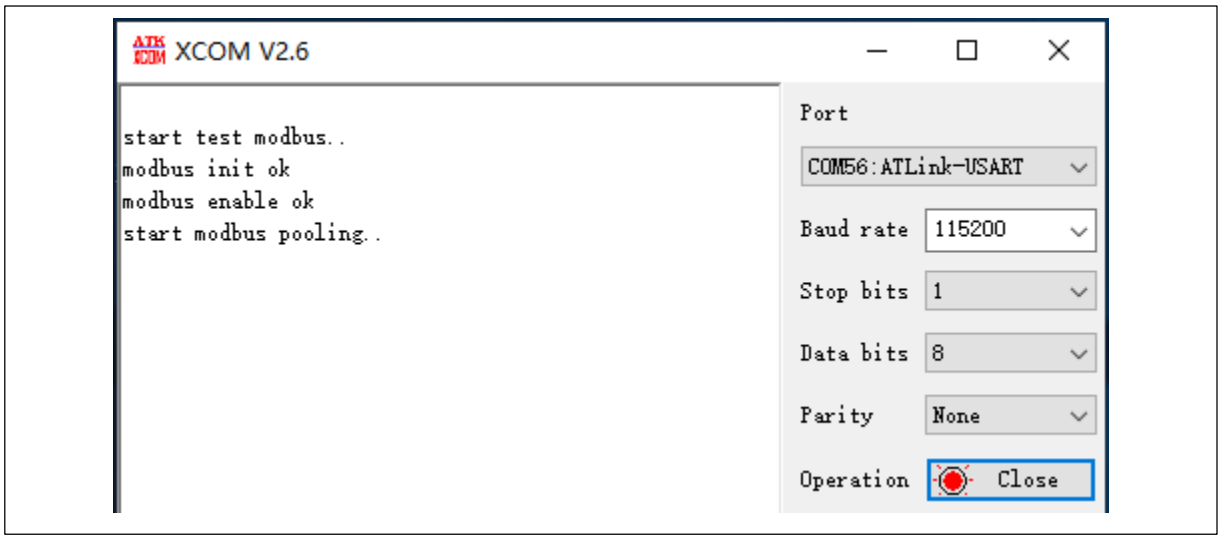
```
int main(void)
{
  system_clock_config();
  at32_board_init();
  uart_print_init(115200);
  printf("\r\nstart test modbus..\r\n");

  modbus_task();
  while(1);
}
```

4 设备的测试

至此，官方例程移植完毕，编译并下载，开打与AT-Link相连的串口，可看到如下打印信息。

图 16. 串口打印信息

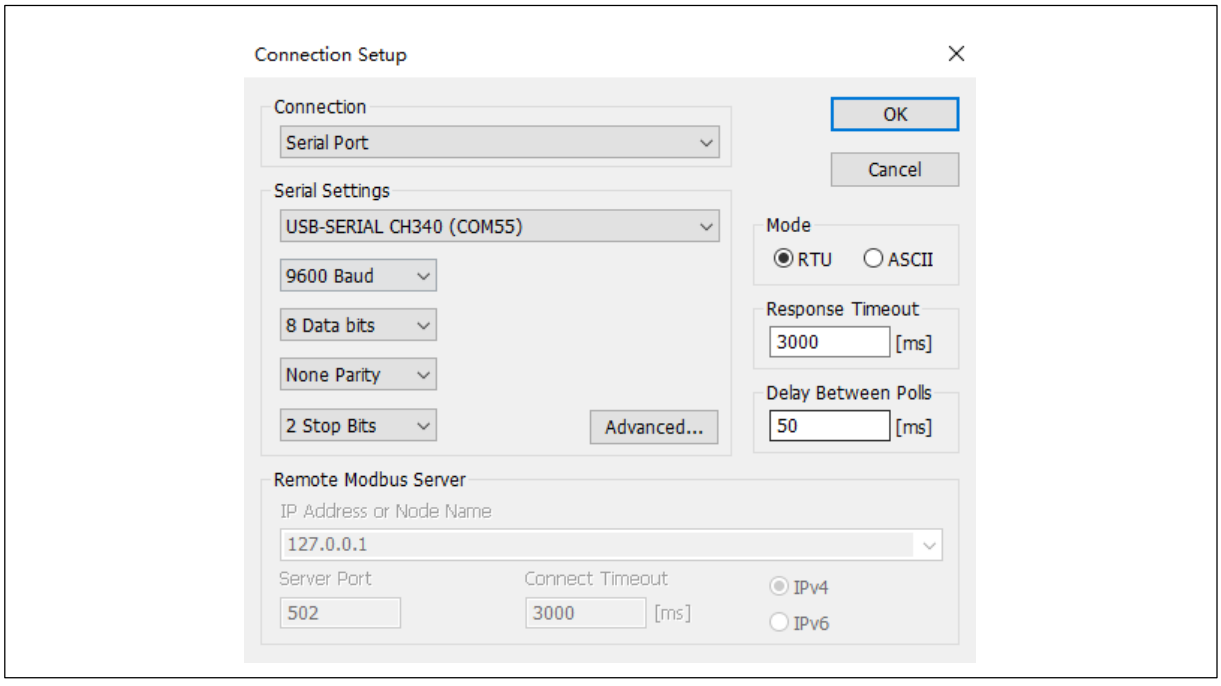


从打印信息可以看到，从设备已经正常的运行起来。

这时我们需要将此设备与上位机相连接，再打开Modbus Poll软件，模拟主设备来进行单播通信，即发送请求并接收应答。

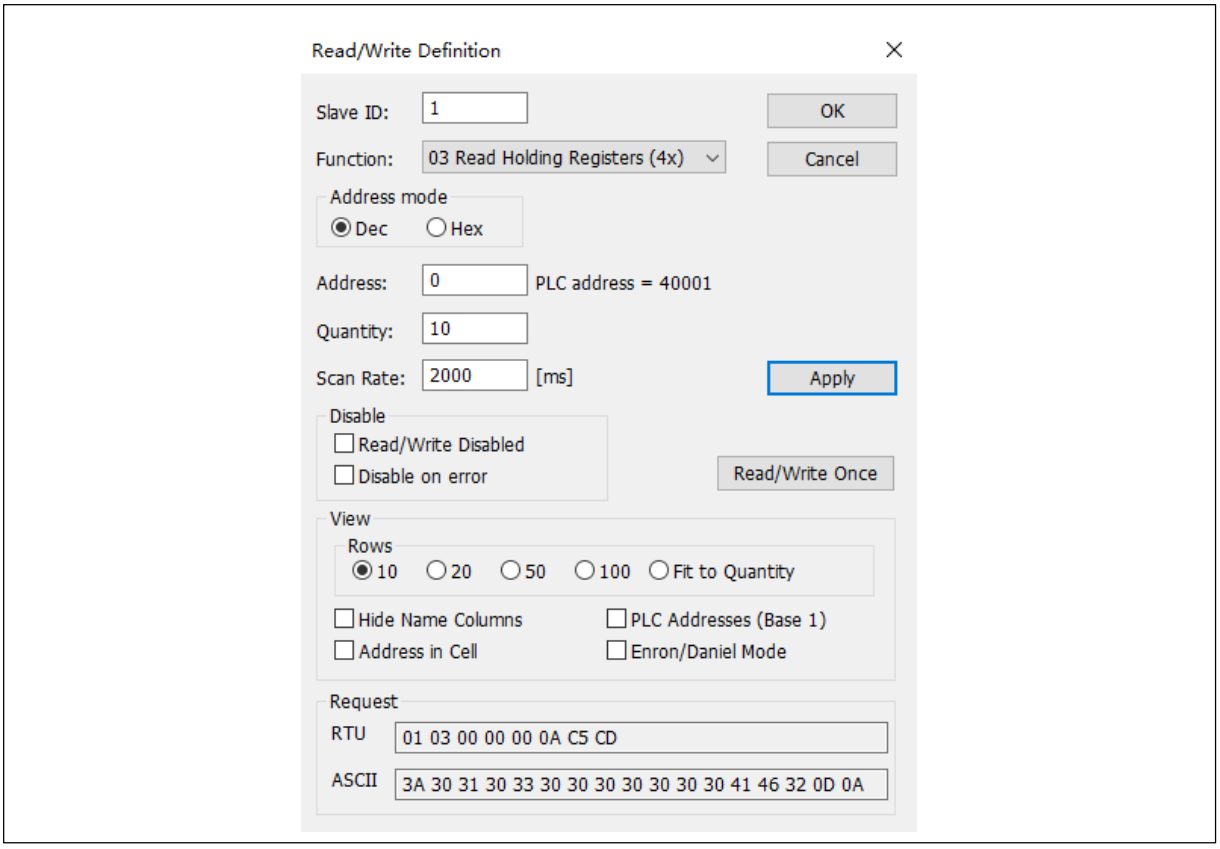
1. 首先对Modbus Poll软件进行连接设置，选择与从设备相同的传输模式（RTU Mode），并配置相应串口的参数，也须与从设备相同。

图 17. Modbus Poll 连接设置



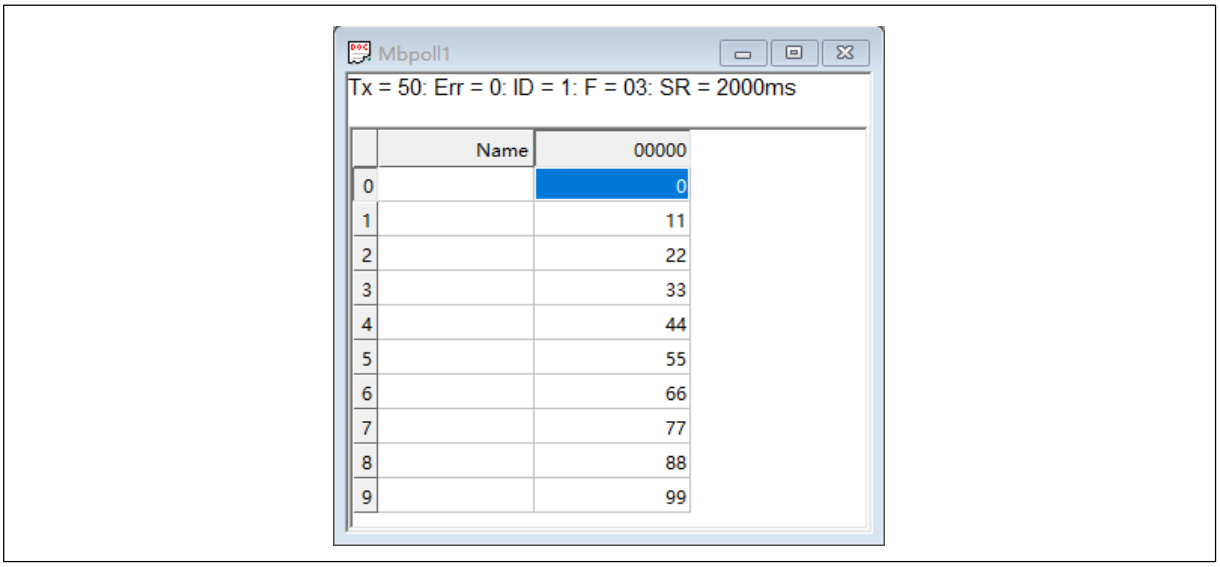
2. 再对Modbus Poll软件进行读写命令的定义，下面仅以功能码03（读保持寄存器）为例来讲解，用户可自行用同样的方式测试其他功能码。

图 18. Modbus Poll 读/写定义



3. 在Modbus Poll软件的文档界面中，可以看到已成功读回保持寄存器的值，并与从设备的程序中初始化保持寄存器时的值一致，测试通过。

图 19. Modbus Poll 文档界面



5 版本历史

表 3. 文档版本历史

日期	版本	变更
2022.06.13	2.0.0	最初版本

重要通知 - 请仔细阅读

买方自行负责对本文所述雅特力产品和服务的选择和使用，雅特力概不承担与选择或使用本文所述雅特力产品和服务相关的任何责任。

无论之前是否有过任何形式的表示，本文档不以任何方式对任何知识产权进行任何明示或默示的授权或许可。如果本文档任何部分涉及任何第三方产品或服务，不应被视为雅特力授权使用此类第三方产品或服务，或许可其中的任何知识产权，或者被视为涉及以任何方式使用任何此类第三方产品或服务或其中任何知识产权的保证。

除非在雅特力的销售条款中另有说明，否则，雅特力对雅特力产品的使用和/或销售不做任何明示或默示的保证，包括但不限于有关适销性、适合特定用途(及其依据任何司法管辖区的法律的对应情况)，或侵犯任何专利、版权或其他知识产权的默示保证。

雅特力产品并非设计或专门用于下列用途的产品：(A) 对安全性有特别要求的应用，如：生命支持、主动植入设备或对产品功能安全有要求的系统；(B) 航空应用；(C) 汽车应用或汽车环境；(D) 航天应用或航天环境，且/或(E) 武器。因雅特力产品不是为前述应用设计的，而采购商擅自将其用于前述应用，即使采购商向雅特力发出了书面通知，风险由购买者单独承担，并且独力负责在此类相关使用中满足所有法律和法规要求。

经销的雅特力产品如有不同于本文档中提出的声明和/或技术特点的规定，将立即导致雅特力针对本文所述雅特力产品或服务授予的任何保证失效，并且不应以任何形式造成或扩大雅特力的任何责任。