本资料仅供内部使用！

Canopen从站模块NetIC调试及使用说明

**2013年 8 月 7 日**

修改记录

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 制定日期 | 生效日期 | 制定 / 修订 内容摘要 | 页数 | 版本 | 拟稿 | 审查 | 批准 |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

目 录

[1 简介 1](#_Toc365708913)

[1.1 NetIC 1](#_Toc365708914)

[2 硬件开发 1](#_Toc365708915)

[3 软件开发 2](#_Toc365708916)

[3.1 Modbus协议 2](#_Toc365708917)

[3.2 软件配置 3](#_Toc365708918)

[3.3 Memory Map 4](#_Toc365708919)

[3.3.1 RPDO 5](#_Toc365708920)

[3.3.2 TPDO 6](#_Toc365708921)

[3.3.3 SDO 7](#_Toc365708922)

[3.4 驱动API 7](#_Toc365708923)

[3.5 API使用实例 8](#_Toc365708924)

[3.5.1 接收到FC04 指令 8](#_Toc365708925)

[3.5.2 接收到FC16指令 9](#_Toc365708926)

# 简介

## NetIC

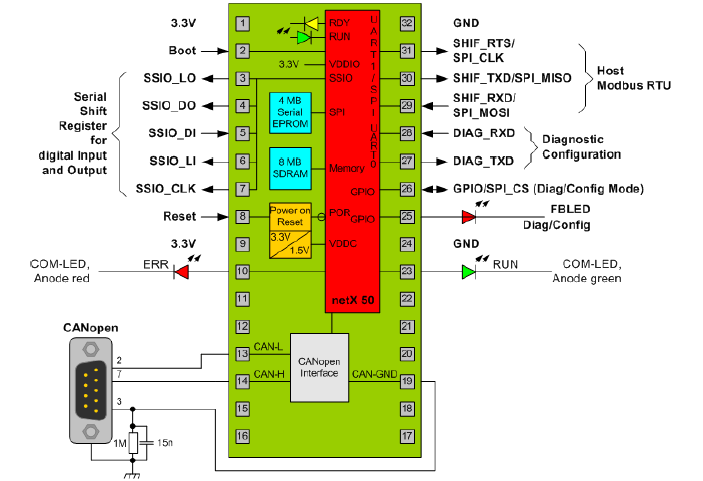
Canopen从站模块NetIC采用Hilscher的netX50高度集成的网络控制器，它具有全新的系统优化结构，适合工业通信和大规模的数据吞吐。支持目前所有主流的实时以太网和现场总线系统（如PROFINET、Ethernet/IP、EtherCAT、Powerlink、SERCOS III、Open Modbus on TCP/IP、AS-Interface、Interbus、CC-Link、CANopen、DeviceNet、Profibus等协议）。

NetIC模块内部已经集成专用的ASIC芯片和固化程序Firmware，由该模块实现实时以太网协议，主机CPU通过嵌入式模块提供的SPI/UART接口接口与模块进行数据通信，用户不需要关心协议的具体实现，只需通过Modbus RTU协议往NetIC的相关寄存器地址读/写数据即可。因此，该模块就像一个内存，使用非常方便。该模块的另一特点是，在同一个硬件模块上就能支持各种不同实时以太网协议（Profinet、EtherCAT、Ethernet/IP、SERCOS III、Powerlink和Modbus TCP），只需下载不同的协议堆栈Firmware。因此，减少了硬件开发成本，只需在软件上更改相关协议的API，减少了开发的再投入成本。

# 硬件开发

NIC 50-COS 管脚连接如下所示：

netIC模块提供了32针管脚，支持SPI/UART接口通过Modbus RTU 协议集成入用户的主机系统。用户CPU实现Modbus RTU协议与netIC模块通讯；29、30、31管脚为串行主机接口，可以与主CPU的UART口直接连接，也可以通过RS485芯片进行连接；13、14、19管脚为CANopen总线接口；27、28管脚为诊断接口，连接PC，通过上位配置软件进行模块参数配置和固件升级；3、4、5、6、7管脚为串行移位寄存器接口，用于连接外部开关量或模拟量信号(暂时未用到)；10、23、25为ＬＥＤ接口，用来显示模块状态；



# 软件开发

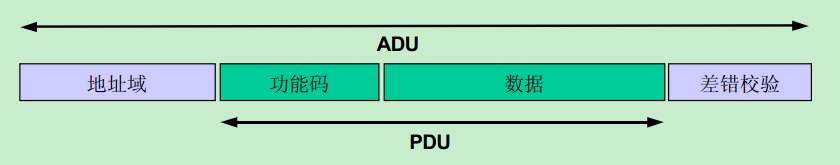
## Modbus协议

NetIC采用Modbus协议与主机进行通讯，模块可配置为Master或Slave模式。

目前作为Modbus Master模式；

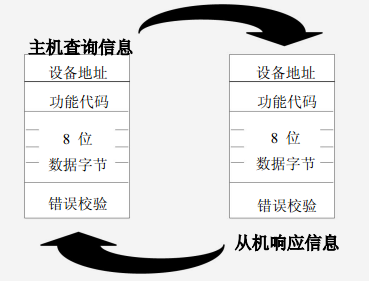
Modbus是OSI模型第7层上的应用层报文传输协议，它在连接至不同类型的总线或网络设备之间提供客户机/服务器通讯。

Modbus是一个请求/应答协议，使用主/从技术，即主机启动数据传输，称为查询，从机返回对主机查询的响应，并处理查询所要求的动作，Modbus报文格式如下所示：



主机可对各从机寻址，发出广播信息，从机返回信息作为对查询的响应，从机对于主机的广播查询，无响应返回，Modbus根据设备地址，请求功能代码，发送数据，错误校验码，建立主机查询格式，从机的相应信息也用modbus协议组织，它包括确认动作的代码，返回数据或错误校验码。若在接收信息时出现一个错误或从机不能执行要求的动作时，从机会组织一个错误信息，并向主机发送作为响应。

查询响应周期如下：



查询：

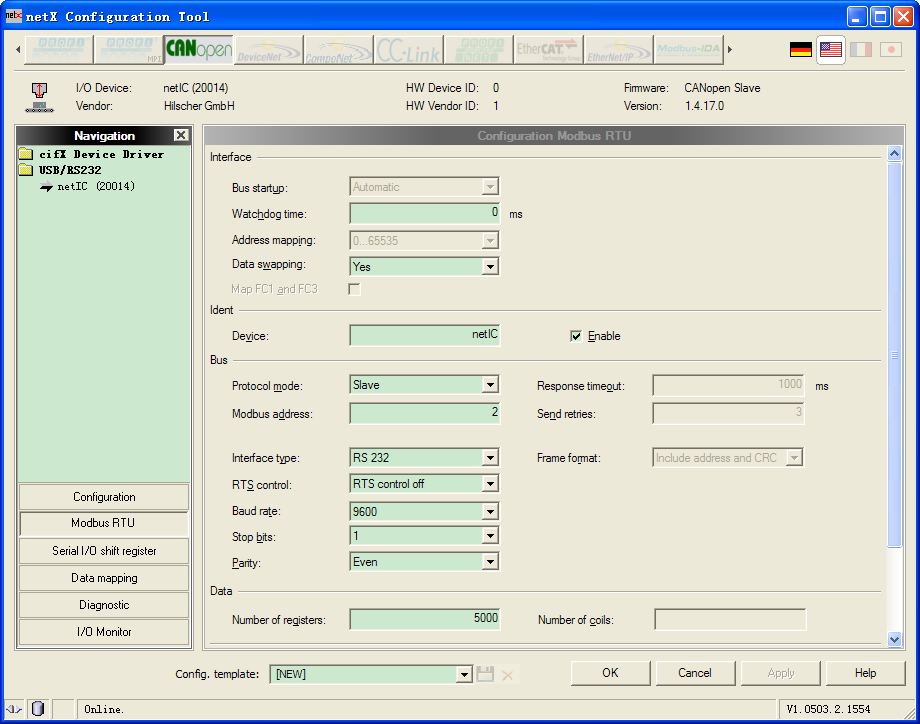
查询中的功能代码为被寻址的从机设备应执行的动作类型，数据字节中包含从机须执行功能的各附加信息，如功能代码03将查询从机，并读保持寄存器。并用寄存器的内容作为响应。该数据区必须含有告之从机读取寄存器的起始地址及数量，错误校验区的一些信息，为从机提供一种校验方法，以保证信息内容的完整性。

响应：

从机正常响应时，响应功能码是查询功能码的应答，数据字节包含从机采集的数据，如寄存器值或状态，如出现错误，则修改功能码，指明错误响应，并在数据字节中含有一个代码，来说明错误，错误检查区允许主机确认有效的信息内容。

## 软件配置

将NetIC模块的Diagnostic接口与PC通过串口相连，打开netX Configuration Tool配置软件，界面如下：



可配置的内容包括：Canopen Node ID、Can Baudrate、Modbus RTU参数等；

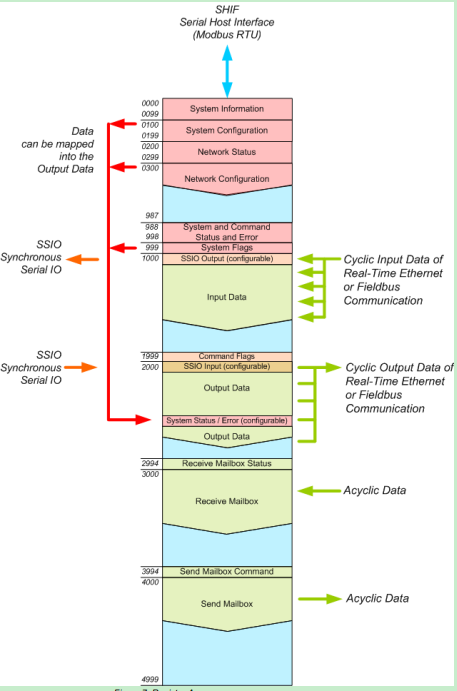
具体配置方法见

《netIC Configuration by netX Configuration Tool OI 09 EN.pdf》

《Canopen从站NetIC配置流程.doc》。

## Memory Map

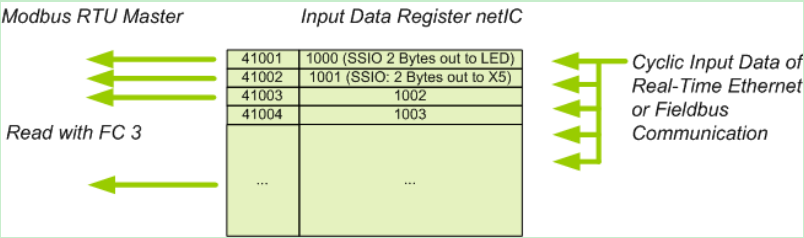
NetIC模块内存寄存器map如下图所示：



该寄存器区域具有固定的结构，分为几个不用的数据区域：Cyclic Input Data、Cyclic Output Data、Acyclic Data(Send mailbox、Receive mailbox)、the shift registers、internal information-、configuration-、and status structures；主机通过Modbus Rtu协议可以读写所有的寄存器内容；

主机通过对Cyclic Input Data、Cyclic Output Data区域的读写可自动转换为PDO的发送与接收，主机对Send Mailbox 、 Receive Mailbox的读写可自动转换为SDO的发送与接收；

### RPDO

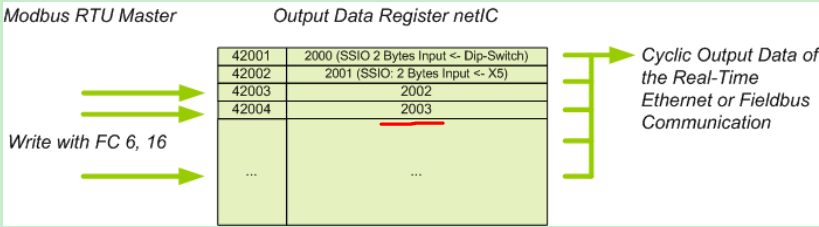


RPDO通讯机制如上图所示：

NetIC接收到PDO后，根据PDO的内容会修改1000之后相应的寄存器，NetIC作为Modbus Master，在接收到该条PDO之后，立即发送Modbus指令到Slave，功能码为FC16(写寄存器指令)，修改ARM中相对应寄存器的值。

见：[3.5.2 接收到FC16指令](#_接收到FC16指令)

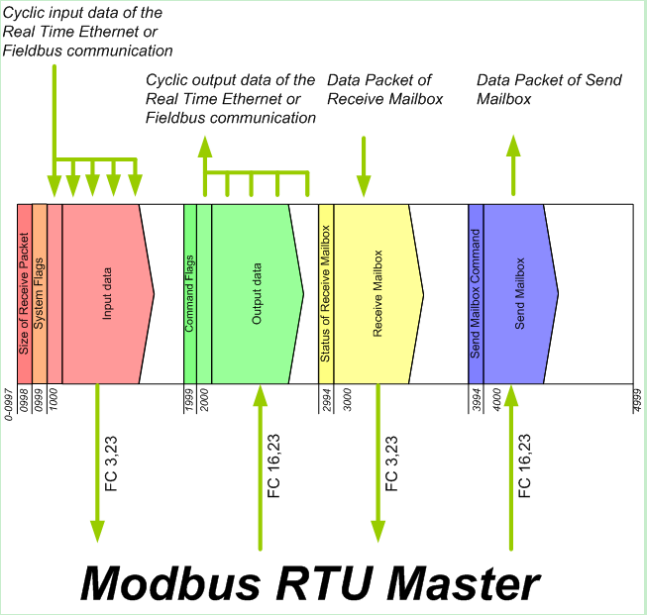
### TPDO



由于ARM作为Modbus Slave，不能主动发送指令，所以NetIC需要通过Modbus指令（FC03）主动查询ARM中存储的感兴趣的值，查询周期可通过软件设置，当查询到的值发生变化之后，NetIC会把该值写到如上图所示相应的寄存器中，并通过TPDO指令发送到Can总线上去；

见：[3.5.1接收到FC04 指令](#_接收到FC04_指令)

### SDO



SDO通讯机制如上图所示：

主机Modbus Master通过FC16或FC23对4000寄存器开始的Send MailBox进行写操作，NetIC转换为SDO发送到CAN总线上；主机通过FC03或FC23指令读取3000开始的Receive Mailbox 寄存器可以获取接收到的SDO指令内容；

由于目前SDO主要是Canopen主站对从站进行初始化配置，主机不需要实现SDO内容。

## 驱动API

uint8\_t **Mb\_Receive\_Response** (uint8\_t **Mb\_slave\_id,**

uint8\_t\* **rece\_buf,**

uint8\_t  **rece\_length**,

uint8\_t\*  **g\_variable**)

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 意义 | 取值范围 |
| **Mb\_slave\_id** | uint8\_t | Modbus 从站地址 | 1~247（0广播地址） |
| **rece\_buf** | uint8\_t\* | 接收到要解析的Modbus指令 | —— |
| **rece\_length** | uint8\_t | 接收到的指令长度 | 0~255 |
| **g\_variable** | uint8\_t\* | ARM内部供读写的寄存器 | —— |
| 返回值 | uint8\_t | 成功：0  失败：1 | 0、1 |

## API使用实例

void **modbus\_salve2\_task**(void \*p\_arg)

{

(void)p\_arg;

while (1)

{

**OSSemPend**(**MB\_SEM\_Reacived**,0,&err); //阻塞等待 **Mb\_Receive\_Response**(**mb\_slave\_id**,**MB\_ReceBuf**,**MB\_Receivce\_length**, **g\_Variable**);

}

}

### 接收到FC04 指令

接收到FC04指令，读取从0x00地址开始的8个寄存器的值，如果值有变化则NetIC以TPDO指令的形式发送到总线上；

接收到的Modbus指令如下：

 地址0x00处的数据由0x15变为0x16，NetIC发现数据改变后，会将该数据封装进TPDO发送到Can总线上，如下：

Can总线数据：



### 接收到FC16指令

当NetIC接收到ComX发送的 202 80 80 80 80 80 80 80 80 这条指令时，将数据写入到自己的内存中相应的区域，通过与之前的数据比较发现数据已经发生变化，便将该内存区域的数据已Modbus 指令的方式发送给ARM；捕获的数据如下：

主站发送的TPDO：



NetIC发送的Modbus指令：

设置从32开始的8个寄存器的值为 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80

