

Ethercat解析之命令行工具的使用教程



liidd777
赞同
IT技术分享及教程

说明：EtherCAT为了方便用户空间对主站进行调试，因此提供一套用户空间使用的工具来设置从站参数，观察调试信息等等。正常情况下，每个主站的实例都会生成一个字符设备，名字为：/dev/EtherCATx。

欲想深入了解其他命令，可通过执行ethercat -help命令来查看详细使用方法。

下面具体的介绍了一下各参数及命令的使用，其中[]中为必选参数，< >为可选参数。

1.1 设置别名地址

命令：ethercat alias [OPTIONS] < ALIAS >

参数：

- * -alias -a : 匹配从站的别名；
- * -position -p : 匹配从站的绝对位置；
- * -force -f : 匹配所有从站；

例子：

```
sudo ethercat alias -position 0 0x2000
```

解析：将在bus总线上对应的从站0的别名（默认为0）为0x2000；

注意：必须有从站连接才能使用此命令。

1.2 显示总线配置

命令：ethercat config [OPTIONS]

参数：

- * -alias -a : 匹配从站的别名；
- * -position -p : 匹配从站的绝对位置；
- * -verbose -v : 显示详细信息；

例子：

```
sudo ethercat config -v
```

解析：显示所有从站的详细配置信息。

注意：必须启动应用程序才能使用此命令查看。

1.3 以C语言的形式输出PDO信息

说明：生成的PDO信息可以直接被应用层的ecrt_slave_config_pdos()函数调用。

命令：ethercat cstruct [OPTIONS]

参数：

- * -alias -a : 匹配从站的别名；
- * -position -p : 匹配从站的绝对位置；

例子：

```
sudo ethercat cstruct -a 100
```

解析：输出别名为100的从站的PDO信息。

注意：必须有从站连接才能使用此命令。

1.4 显示过程数据

说明：输出二进制的过程数据。

命令：ethercat data [OPTIONS]

参数：

- * -domain -d : 域的索引值，假如不填写参数则显示所有过程数据。

例子：

```
sudo ethercat data
```

解析：显示所有PDO过程数据。

注意：必须启动应用程序才能使用此命令查看。

1.5 设置主站调试级别

说明：设置主站的调试级别，调试信息将输出在/var/log/syslog文件中。

命令：ethercat debug

其中可有以下情况：

- * 0 : 无任何调试信息输出
- * 1 : 输出部分调试信息
- * 2 : 输出所有的帧的内容（由于输出信息较多，请谨慎使用）

例子：

```
sudo ethercat debug 1
```

解析：打开部分调试信息输出

1.6 配置域

说明：显示域的信息。

命令：ethercat domains [OPTIONS]

参数：

- * -domain -d : 根据索引号，匹配域；
- * -verbose -v : 显示域的详细信息（FMMU和过程数据的信息）；

例子：

```
① sudo ethercat domains执行后显示
```

```
Domain0:LogBaseAddr 0x00000000, Size 12, WorkingCounter 0/3
```

以上各字段的含义：

- * LogBaseAddr：逻辑寻址的逻辑基地址；
- * Size：域交换数据的字节数；
- * WorkingCounter：第一个数字是WKC的当前值，第二个数字是WKC的期望值；

解析：显示域的基本信息。

```
② sudo ethercat domains -v执行后显示
```

```
Domain0:LogBaseAddr 0x00000000, Size 12, WorkingCounter 3/3
SlaveConfig 0:0, SM2 ( Output ), LogAddr 0x00000000, Size 6 06 00 9d aa 00 00
SlaveConfig 0:0, SM3 ( Input ), LogAddr 0x00000000, Size 6 31 0a 9d aa 00 00
```

以上各字段的含义：

- * SlaveConfig：从机配置信息，主要包含别名和地址（绝对地址或相对地址）；
- * SM2：同步管理器2；
- * LogAddr：FMMU映射的地址；
- * Size：映射地址的大小；

* 数据位：十六进制显示的过程数据；

解析：显示域的详细信息（FMMU和过程数据的信息）。
注意：必须启动应用程序才能使用此命令查看。

1.7 访问SDO

说明：向从站写一条PDO条目。

命令：ethercat download [OPTIONS]

参数：

(1) 可选参数：
* INDEX：16位无符整型的SDO索引；
* SUBINDEX：8位无符整型的SDO子索引；
* VALUE：需写入的SDO的值

(2) [OPTIONS]参数：
* -alias -a：匹配从站的别名；
* -position -p：匹配从站的绝对位置；
* -type -t：SDO条目的数据类型；

type可使用的类型有：

bool、int8、int16、int32、int64、uint8、uint16、uint32、uint64、float、double、string、
octet_string、unicode_string

对于sign-and-magnitude coding有：
sm8、sm16、sm32、sm64

例子：
sudo ethercat download -t int16 -p 0 0x6060 00 08
解析：向从站0的索引号为0x6060（16位），子索引号为00（8位）的地址写入PDO条目
值“0x08”；

1.8 访问SDO

说明：向从站读取一个SDO条目。

命令：ethercat upload [OPTIONS]

参数：

(1) 可选参数：
* INDEX：16位无符整型的SDO索引；
* SUBINDEX：8位无符整型的SDO子索引；

(2) [OPTIONS]参数：
* -alias -a：匹配从站的别名；
* -position -p：匹配从站的绝对位置；
* -type -t：SDO条目的数据类型；

type可使用的类型有：

bool、int8、int16、int32、int64、uint8、uint16、uint32、uint64、float、double、string、
octet_string、unicode_string

对于sign-and-magnitude coding有：
sm8、sm16、sm32、sm64

例子：
sudo ethercat upload -t int16 -p 0 0x6060 00
解析：读取从站0中索引号为0x6060（16位），子索引号为00（8位）的SDO条目。

注意：必须有从站连接才能使用此命令。

1.9 输出EOE统计信息

说明：显示EOE的统计信息，包括主站的发送率和接收率（Byte/s）。

命令：ethercat eoe

参数：无

1.10 FOE通信

(1) 通过FOE读取从站的文件。

命令：ethercat foe_read [OPTIONS]

参数：

① 可选参数

*：为从站中的源文件；

② [OPTIONS]参数
* -output -file -o：为读取数据的存储文件，缺省的话数据将读取到stdout；
* -alias：匹配从站的别名；
* -position：匹配从站的绝对位置；

(2) 通过FOE向从站存储文件。

命令：ethercat foe_write [OPTIONS]

① 可选参数

*：为要发送的文件（指定路径+文件名），也可以为“-”，表示从标准输入中读取文件；

② [OPTIONS]参数
* -output -file -o：为从站存储的目标文件名，假如不指定名字，则为从站发送 FILENAME的去掉路径的名字；
* -alias：匹配从站的别名；
* -position：匹配从站的绝对位置；

1.11 创建一个拓扑图形

说明：输出总线拓扑图。

命令：ethercat graph [OPTIONS]

例子：
sudo ethercat graph | dot -Tsvg > ~/Desktop/bus.svg
解析：将总线拓扑图输出到桌面。

1.12 主站和以太网设备

说明：显示主站和以太网设备信息。

命令：ethercat master [OPTIONS]

参数：

* -master -m：indices为主站的索引。默认显示所有的设备信息；

例子：
sudo ethercat master
解析：显示所有主机的设备信息（发送帧、接收帧、参考时钟、应用时间）。

注意：欲想显示应用时间需启动应用程序。

1.13 同步管理，PDOs，PDO条目

说明：显示出同步管理器的参数和PDO任务和映射信息。

命令：ethercat pdos [OPTIONS]

参数：

* -alias -a ：匹配从站的别名；
* -position -p ：匹配从站的绝对位置；
* -skin -s ：“ skin” 可选择“ default ”和“ etherlab ”；

例子：

sudo ethercat pdos -p 0 -s default执行后显示：

```
SM2 : PhysAddr 0x1400, DefaultSize 64, ControlRegister 0x34, Enable 1
RxPDO 0x1600  "Receive PDO1 Mapping"
PDO entry 0x6040:00, 16 bit, "  "
PDO entry 0x607a:00, 32 bit, "  "
```

以上各字段的含义：

(1) 同步管理器信息

* SM2：同步管理器2；
* PhysAddr：物理地址开始地址；
* DefaultSize：默认数据大小；
* ControlRegister：控制寄存器；
* Enable：使能字

(2) 显示PDO方向，索引值，PDO名字

* RxPDO：代表从站发送数据的方向（从站接收数据）；
* 0x1600：PDO的索引值；
* "Receive PDO1 Mapping"：PDO的名字；

(3) 显示PDO条目的索引和子索引（都是以16进制的形式现实的），显示位宽和描述

* 0x6040:00：表示索引和子索引；
* 16bit：表示该条目的位宽；
* " "：表示该位的描述；

1.14 寄存器访问

(1) 获取对应从站寄存器的内容

命令：ethercat reg_read [OPTIONS]

[SIZE]

参数：

① 可选参数

* ADDRESS：16位无符号的寄存器地址；
* SIZE：要读取的对应寄存器字节数（16位无符号值）；[SIZE] + ADDRESS不能超过64K，假如type参数隐含要读取的字节数，则可以忽略掉[SIZE]参数；

② [OPTIONS]参数

* -alias -a ：匹配从站的别名；
* -position -p ：匹配从站的绝对位置；
* -type -t ：匹配数据类型；

type可使用的类型有：

bool、int8、int16、int32、int64、uint8、uint16、uint32、uint64、float、double、string、octet_string、unicode_string

对于sign-and-magnitude coding有：

sm8、sm16、sm32、sm64

例子：

sudo ethercat reg_read -p 6 -t sm32 0x092c

解析：获取从站6的0x092C寄存器所存储的值。

(2) 将内容写入指定从站寄存器

命令：ethercat reg_write [OPTIONS]

参数：

① 可选参数

* ADDRESS：16位无符号的寄存器地址；
* DATA：要写入寄存器的数据；假如制定了“ type” 数据类型，那么“ DATA” 根据指定的数据类型对数据进行解析；假如未指定“ type” 数据类型，则“ DATA” 可以为指定的文件或
将“ DATA” 设置为“ - ”，表示从标准输入中获得数据；

② [OPTIONS]参数

* -alias -a ：匹配从站的别名；
* -position -p ：匹配从站的绝对地址；
* -type -t ：匹配数据类型；
* -emergency -e：以紧急的方式请求写入文件；

例子：

sudo ethercat reg_write -p 5 -t sm32 0x092c 200

解析：向从站5的寄存器0x092c写入数据200。

1.15 SDO字典

说明：列出SDO字典（SDO信息和SDO条目信息）。

命令：ethercat sdos [OPTIONS]

参数：

* -alias -a ：匹配从站的别名；
* -position -p ：匹配从站的绝对位置；
* -quiet -q：只输出PDOs，不输出PDO条目信息；

例子：

sudo ethercat sdos执行后显示

```
SDO 0x1000, " Device type"
0x1000:0, r-r-r-, uint32, 32 bit, "Device type"
1
2
SDOs : SDO 0x1000, " Device type"
* 0x1000 : SDO索引值；
* "Device type" : SDO名字；
```

SDO条目：0x1000:0, r-r-r-, uint32, 32 bit, "Device type"

* 0x1000:0：索引值及子索引值；
* r-r-r-：表示访问权限；
* uint32：表示该条目的数据类型；
* 32bit：表示该条目的位宽；

* "Device type" : 对该条目的描述；

1.16 SII访问

(1) 读取从站的SII内容

命令：ethercat sii_read [OPTIONS]

参数：

* -alias -a : 匹配从站的别名；

* -position -p : 匹配从站的绝对位置；

* -verbose -v : 分类显示数据文本内容；

例子：

① sudo ethercat sii_read -p 0 -v

解析：以分类数据文本形式显示sii的内容；

② sudo ethercat sii_read -p 0 | hexdump

解析：以16进制形式显示sii的内容；

③ sudo ethercat sii_read -p 0 > Backup.bin

解析：将sii的内容备份到Back.bin文件下；

注意：必须有从站连接才能使用此命令。

(2) 向从站写入sii内容

命令：ethercat sii_write [OPTIONS]

参数：

* -alias -a : 匹配从站的别名；

* -position -p : 匹配从站的绝对位置；

* -force -f : 不进行有效检查；

例子：

sudo ethercat sii_read -p 0 Backup.bin

解析：将Backup.bin的内容写入从站0的SII中。

注意：必须有从站连接才能使用此命令。

1.17 显示从站的信息

说明：显示总线上的从站的信息。

命令：ethercat slaves [OPTIONS]

参数：

* -alias -a : 匹配从站的别名；

* -position -p : 匹配从站的绝对地址；

* -verbose -v : 显示从站的详细信息；

例子：

sudo ethercat slaves -v

解析：显示所有从站的详细信息。

1.18 SOE IDN访问

说明：读取从站的SOE IDN。

命令：ethercat soe_read [OPTIONS]

参数：

① 可选参数

* DRIVE：驱动号【0 - 7】，缺省默认为0；

* IDN:

② [OPTIONS]参数

* -alias -a : 匹配从站的别名；

* -position -p : 匹配从站的绝对地址；

* -type -t : 匹配数据类型；

type可使用的类型有：

bool、int8、int16、int32、int64、uint8、uint16、uint32、uint64、float、double、string、octet_string、unicode_string

对于sign-and-magnitude coding有：sm8、sm16、sm32、sm64

1.19 请求应用层转换状态机

说明：请求应用层转换状态。

命令：ethercat states [OPTIONS]

参数：

(1) 可选参数

* STATE：可选的参数有INIT、PREOP、BOOT、SAFEOP、OP；

(2) [OPTIONS]参数

* -alias -a : 匹配从站的别名；

* -position -p : 匹配从站的绝对地址；

例子：

sudo ethercat states -p 0 OP

解析：将从站0的状态切换为OP；

注意：必须有从站连接才能使用此命令。

1.20 显示主站版本

说明：显示主站的版本。

命令：ethercat version [OPTIONS]

1.21 生成从站配置描述

说明：生成从站信息描述文件。

命令：ethercat xml [OPTIONS]

参数：

* -alias -a : 匹配从站的别名；

* -position -p : 匹配从站的绝对地址；

例子：

sudo ethercat xml -p 0

解析：生成从站0的从站信息描述文件并显示出来

开启系统后，用户主要需要进行从站配置与PDO映射、域操作功能，然后就可以进行PDO交换了，当然，这个过程是基于主站状态机在后台完成了庞大的扫描工作后我们才得以进行这些操作的：

1. 1.ec_master_t *ecrt_request_master(unsigned int master_index);

2. 2.ec_domain_t *ecrt_master_create_domain(ec_master_t *master);

3. 3.ec_slave_config_t *ecrt_master_slave_config(ec_master_t *master,uint16_t alias, uint16_t

position, uint32_t vendor_id, uint32_t product_code);

4. ____uint8_t ecrt_slave_config_reg_pdo_entry(ec_slave_config_t *sc,uint16_t index, uint8_t subindex,ec_domain_t *domain,unsigned int *bit_position);

5. ____uint8_t ecrt_master_activate(ec_master_t *master);

6. uint8_t *ecrt_domain_data(ec_domain_t *domain);

7. void ecrt_domain_queue(ec_domain_t *domain);

8. void ecrt_master_send(ec_master_t *master);

以上，就是配置PDO基本需要使用到的函数，当然，主站程序运行在内核空间，应用层需要一种方式实现对内核调用。通过open主站创建的字符设备/dev/EtherCAT就是这一目的。

首先第一个函数ecrt_request_master(),负责打开字符设备文件，传入参数为使用主站索引(使用/etc/init.d/ethercat开启系统的时候可以一次性传入多个MAC地址从而开启多个主站，不同主站创建不同字符设备，如：/dev/EtherCAT0、/dev/EtherCAT1...)，而这个索引就是用于开启对应的字符设备，从而使用不同的主站，使用open接口打开，然后返回文件描述符，该文件描述符用于ioctl调用对应主站功能。另外在申请到主站后就不在允许进行总线的重新扫描，固化了主站连接的从站链表，且设置从站状态当前申请状态为PREOP；

第二个函数ecrt_master_create_domain()创建域，domain与PDO映射有密切联系，之后在说到主站激活的位置细谈这部分；至于这个函数具体内容就是直接在内核空间中创建一块ec_domain_t空间并将该空间连接到主站domains链表尾部；

第三个函数ecrt_master_slave_config(),从站配置函数，说到从站配置函数，有个地方需要明确一下，那就是主站模块中的从站链表，他们之间的关系并不像我们想的那么紧密，首先，从站配置是应用层通过ecrt_master_slave_config()函数添加到主站的configs链表中的，而从站链表slaves则是通过ethercat主站状态机通过扫描ethercat总线得到的，两者的共通之处就在于从站配置过程的alias、position、VID和PID,这些在主站扫描到的从站链表中都是有一份参数的，当应用层设置配置和主站扫描到的从站信息相匹配的时候，那就会从站与从站配置相联系(这时候会将主站扫描到的SI信息关于sync关联的PDO信息复制到从站配置中，也就是默认PDO配置)；这样，就可以通过从站配置寻找到主站模块中从站配置信息，进而获取到从站信息。

第四个函数ecrt_slave_config_reg_pdo_entry(),该函数是从站配置PDO函数，首先需要注意一点，那就是在主站正式激活之前，所有配置PDO相关的其实并没有相对应的空间申请操作，一切都是进行的计算，计算PDO需要多大空间，计算domains的逻辑位置，计算FMMU的逻辑位置，计算PDO的逻辑位置，直到主站正式激活，主站会根据自身已经申请的域链表domains，获取所有域需要的空间大小，而后对每个域进行逻辑地址分配，每个域中再对每个FMMU进行逻辑地址分配。这样。言归正传，继续该函数研究，首先了解一下接口：

```
1. int ecrt_slave_config_reg_pdo_entry(ec_slave_config_t *sc,
2. uint16_t index, /* 对象字典索引 */
3. uint8_t subindex, /* 对象字典子索引 */
4. ec_domain_t *domain, /* 使用的域 */
5. unsigned int *bit_position)/* 一个Byte中的偏移量，当PDO映射逻辑地址不是完整的Byte时有用 */
```

首先，通过从站应用层配置索引可以获取到主站的从站配置链表中对应的从站配置，而后根据配置目标domain的索引获取到主站的domains链表对应的域，这样就可以实现应用层的从站配置和domain切换到主站对应的从站配置和domain，然后进行pdo映射操作。另外每次进行PDO映射的时候，并不是只对设置的一个对象进行处理，而是直接将整个sync关联的PDO都进行批次化处理：具体操作如下

1. 1.找到PDO对象所在的sync同步管理器；

2. 2.找到PDO对象在该同步管理器中的相对起始地址，如果该起始地址并不是整Byte数据，那么启用bit_position，如果没有，进行报警；

3. 3.进行FMMU配置计算(FMMU管理映射是以一个sync关联的所有PDO进行映射的，因此只要配置的PDO对象的domain和sync是同一个，那就说明已经配置过对应sync，因此也就不必要新建FMMU去配置映射PDO)，这里有几个空间块关系和逻辑地址需要明确一下，先说空间块，在主站没有激活之前，各个domains是无关的，domain内部由多个FMMU组成，每个FMMU配置映射是基于sync关联的PDO得到的(且经过字节对齐处理);然后就是domain->data_size，这个量只表明当前需要的空间大小，那么这个量也可以看作下一个需要配置的FMMU的起始逻辑地址，而对于PDO的逻辑地址，则可以通过在变量sync关联PDO过程进行获取，也就是第二步完成的操作；

所以，本质上来讲，PDO配置映射就是确定PDO的逻辑地址位置。

第五个函数ecrt_master_activate(),该函数为主站正式激活的函数，该函数完成以下功能：

1. 1.主站注册的domains链表整体大小计算：经过PDO注册映射之后，对主站中关联的每个domain都是相互独立的，并且每个domains都只是一个计算得到的值，其实并没有需要的空间申请，因此主站激活的时候会将全部domains需要的空间进行统一申请。具体申请到的PDO映射逻辑空间存放在字符设备打开文件的priv指针处，关于字符设备的file->priv在字符设备打开ecdev_open的时候进行指定。然后按照主站连接的domains链表顺序对每个domain起始地址进行处理设置：主要是domain的逻辑起始地址，domain的大小，以及domain中的FMMU的逻辑地址重新设置，最终，domain和FMMU的逻辑起始地址都是在主站全部domains环境下的逻辑起始地址。为了进行PDO映射，在主站激活过程中还对每个domain需要的子报文进行空间申请。在计算子报文需要个数的时候，各个domain计算各自需要的子报文需要，这样就可以满足不同domain的PDO交换周期是可以单独设置的；在domain计算自己需要的子报文个数的时候，计算是以FMMU为一个基本单位进行计算的，每个子报文可承载的数据大小为1486Byte，然后根据一个子报文中FMMU的输入输出情况设置子报文类型，也就是EC_DATAGRAM_LRD类型子报文，EC_DATAGRAM_LRR类型子报文以及EC_DATAGRAM_LRW类型子报文。然后将这些子报文添加到domain的datagram_pairs链表中。当然，整体过程中，domain映射空间和子报文使用的数据空间都是同一段空间。

2. 2.完成主站的domains域空间处理，之后就停止主站的空闲阶段线程(等待空闲阶段线程退出);

3. 3.开启主站的操作阶段线程，该线程和空闲阶段线程相比不同之处在于将主站子报文队列组合发送移动到用户手中进行处理。

第六个函数ecrt_domain_data(),返回domain在逻辑空间的逻辑地址；

第七个函数ecrt_domain_queue(),就是将对应domain空间使用的子报文排列到主站子报文发送链表中；

第八个函数ecrt_master_send(),就是将主站的子报文发送链表的子报文进行发送(在发送过程中，默认的会将从站状态机子报文和主站状态机子报文进行排队发送);

以上，就是应用层执行PDO映射的基本原理。

EtherCAT igh源码的ecrt_slave_config_dc ()函数的理解。

• ethercat

总结一下自己对igh的ecrt_slave_config_dc () 函数的理解。参考了igh的example里的“dc_user例程”。

例程里有这样一段代码：

```
// configure SYNC signals for this slave
ecrt_slave_config_dc(sc, 0x0700, PERIOD_NS, 4400000, 0, 0);
```

在slave_config.c文件里可以查看到函数的定义：

```
/** Configure distributed clocks.
 *
 * Sets the AssignActivate word and the cycle and shift times for the sync
 * signals.
 *
 * The AssignActivate word is vendor-specific and can be taken from the XML
 * device description file (Device -> Dc -> AssignActivate). Set this to zero,
 * if the slave shall be operated without distributed clocks (default).
 *
 * This method has to be called in non-realtime context before
 * ecrt_master_activate().
 *
 * \attention The \a sync1_shift time is ignored.
 */
void ecrt_slave_config_dc(
    ec_slave_config_t *sc, /**< Slave configuration. */
    uint16_t assign_activate, /**< AssignActivate word. */
    uint32_t sync0_cycle, /**< SYNC0 cycle time [ns]. */
    int32_t sync0_shift, /**< SYNC0 shift time [ns]. */
    uint32_t sync1_cycle, /**< SYNC1 cycle time [ns]. */
    int32_t sync1_shift /**< SYNC1 shift time [ns]. */
);

void ecrt_slave_config_dc(ec_slave_config_t *sc, uint16_t assign_activate,
    uint32_t sync0_cycle_time, int32_t sync0_shift_time,
    uint32_t sync1_cycle_time, int32_t sync1_shift_time)
{
    ec_ioctl_config_t data;
    int ret;


    data.config_index = sc->index;
    data.dc_assign_activate = assign_activate;
    data.dc_sync[0].cycle_time = sync0_cycle_time;
    data.dc_sync[0].shift_time = sync0_shift_time;
    data.dc_sync[1].cycle_time = sync1_cycle_time;
    data.dc_sync[1].shift_time = sync1_shift_time;

    ret = ioctl(sc->master->fd, EC_IOCTL_SC_DC, &data);
    if (EC_IOCTL_IS_ERROR(ret)) {
        fprintf(stderr, "Failed to set DC parameters: %s\n",
            strerror(EC_IOCTL_ERRNO(ret)));
    }
}
```

发布于 05-28

开放式 IEC 61131 控制系统设计 (书籍) 终端命令

文章被以下专栏收录



IT技术专栏

IT技术分享及教程

推荐阅读

如何使用SLURM？

提交SBATCH脚本在HPC上运行任务的主要方法是通过sbatch命令提交一个脚本。例如：sbatch MyJobScript.sh在MyJobScript.sh中的命令会在第一个被找到的、可用的、满足资源要求的compute node...

风影忍着发表于数据分析之...



Linux 爱好者 Linux 从命令行轻松将文本片段上传到类似 Pastebin 的服务中

还没有评论

写下你的评论...

赞同

添加评论

分享

喜欢

收藏

申请转载

...