LIVIC – Laboratoire de recherche sur les Intéractions Véhicule - Infrastructure - Conducteur

INRETS – Institut National de Recherche sur les Transports et leur Sécurité
LCPC – Laboratoire Central des Ponts et Chaussées
Ministères Recherche & Transport

CANOpen Memento

Francis Dupin, November 2005

Version 1.2

上传该文，涵盖测试一个节点或配置一个CANopen网络之小技巧。
若你不知道CANopen如何工作，则该文无任何帮助。
可随便分享，，如有错误请告知francis.dupin@inrets.fr

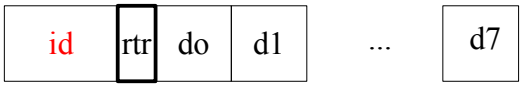
I have put on this document some of the tips to test a node or to configure a CANOpen network.
If you do not know how works CANOpen, this document will not help you at all.
Feel free to redistribute this paper ... and to report the errors and missings to francis.dupin@inrets.fr

翻译 ： 李龙胜
QQ ： 21561401
Email: dragon2001@163.com

如果有翻译不当的地方请告知我，以便及时更正，谢谢！

注意：数据基本为hex格式，部分为二进制，注意识别！

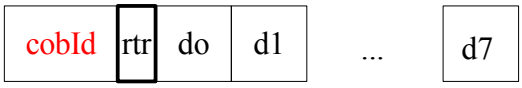
CAN报文基本格式



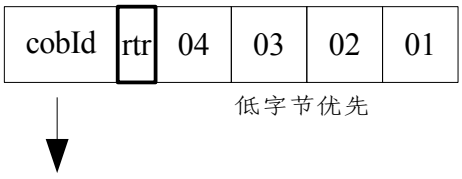
Id: 某个消息的CAN标识符，通常为11bit。
rtr: 0: 普通消息;
1: 远程传输请求消息，不能携带数据。
dn: 数据字节，一个普通消息可以携带0-8个字节数据。

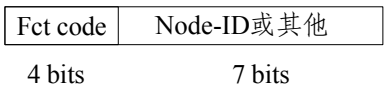
Id : the CAN identifiant of the message. Usually on 11 bits.
rtr : 0 : normal message
1 : Remote Transmit Request message. Cannot contain data dn : data byte. A normal message can contain 0 to 8 bytes of data

在CANOpen里的CAN报文



放入CAN帧里的数据，以LSB优先(ie little endian).
例如 :数据0x01020304 应如下组织 :





4 bits 7 bits

Fct Code (二进制) : EMCY 0001
功能码 PDO : 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010
SDOrx : 1011
SDOtx : 1100
NMT error control : 1110
NMT : 0000
SYNC : 0001
TIME STAMP : 0010

Note： 在该文档中,
- 所有的数据均为十六进制符号，除特殊登明外.
- 若未特指，CAN消息均为«normals»，即(rtr = 0)

NMT protocol

注意：对于NMT协议，
11-bit的CAN报文ID中只携带高4-bit的功能码，
Node-ID放在数据域的D1处；
具体的NMT命令编码放在数据域的D0处！

将某个节点设为operational 模式

		D0	D1
000	0	01	nodeId

将某个节点设为stop 模式

		D0	D1
000	0	02	nodeId

将某个节点设为 pre-operational模式

		D0	D1
000	0	80	nodeId

将某个节点设为reset-application 模式

		D0	D1
000	0	81	nodeId

将某个节点设为reset-communication模式

		D0	D1
000	0	82	nodeId

Note : 意命令所有节点，则nodeId = 00

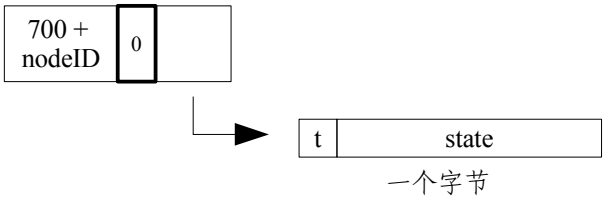
例如：

将节点0x06设为operational模式 : 000 01 06
将所有节点设为pre-operational模式 : 000 80 00

Node guard protocol

意获悉某个节点状态，master发送:

节点应答 :



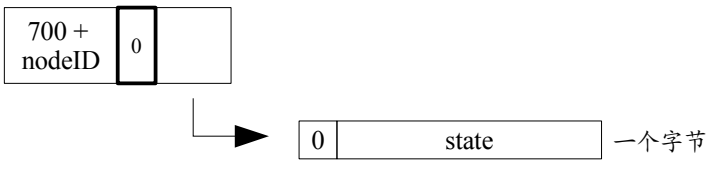
t : 翻转位 0,1,0,1....
state : 0 = 初始化
1 = 断开连接
2 = 连接
3 = 准备
4 = stopped
5 = operational
7F = pre-operational

例如:

意获悉节点5状态，master发送一帧CAN请求(rtr = 1) : 705
若节点5处于stopped模式，则应答 : 705 05
master发送一帧NMT命令节点进入pre-operational模式: 000 80 05
master发送一帧新CAN请求 : 705
节点5此时处于pre-operational模式，应答 : 705 FF (因为toggle = 1)

Heartbeat protocol

节点周期性发送其状态，无需任何请求帧 :



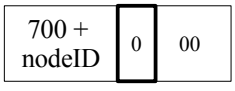
state : 0 = Boot-up
4 = stopped
5 = operational
7F = pre-operational

例如

节点5，处于pre-operational模式，周期性发送: 705 7F
注意 : 这里没有翻转位t.

Bootup protocol

当某个节点在initializing模式之后，进入pre-operational模式，将发送 :



当一个心跳节点启动后，它的Boot-up报文，就是其第一个心跳报文;

SDO protocol

SDO用于读/写某个节点的对象字典。
发出读/写请求的节点称为client节点。
数据被读/写的节点称为server节点。

Read = upload protocol
Write = download protocol
若读/写的数据不超过4个字节，则最简方式为使用 SDO upload/download expedited protocol.
所有SDO均为相同CAN帧长度：8个数据字节，且rtr = 0.

SDO Download expedited protocol

意 write 数据 0xd0d1... 到server节点的对象字典中，则client节点发送:

client请求:

Data length = 1 byte

600 + Serv NodeId	0	2F	Index	Sub index	d0	x	x	x

X : undefined. Put 0

client请求:

Data length = 2 bytes

600 + Serv NodeId	0	2B	Index	Sub index	d1	d0	x	x

X : undefined. Put 0

client请求:

Data length = 3 bytes

600 + Serv NodeId	0	27	Index	Sub index	d2	d1	d0	x

X : undefined. Put 0

client请求:

Data length = 4 bytes

600 + Serv NodeId	0	23	Index	Sub index	d3	d2	d1	d0

server应答(若成功) :

580 + Serv NodeId	0	60	Index	Sub index	00	00	00	00

server应答(若失败) :

580 + Serv NodeId	0	80	Index	Sub index	SDO abort code error

|------ 4个字节 -----|

例如:

意写一个字节数据：0xFD到节点5的对象字典的主索引0x1400、子索引2处，发送:
605 2F 00 14 02 FD 00 00 00
若成功，节点5应答:
585 60 00 14 02 00 00 00 00

意写4个字节数据：0x60120208到节点5的对象字典的主索引0x1603、子索引1处，发送:
605 23 03 16 01 08 02 12
60 若成功，节点5应答:
585 60 03 16 01 00 00 00 00

SDO Upload expedited protocol

意 read server节点对象字典中的某处数据0xd0d1...，client节点发送：

client 请求：

600 + Serv NodeId	0	40	Index	Sub index	00	00	00	00

server 应答 (若成功)：

Data length = 1 byte

580 + Serv NodeId	0	4F	Index	Sub index	d1	x	x	x

X : undefined. Should be 0

server应答 (若成功)：

Data length = 2 bytes

580 + Serv NodeId	0	4B	Index	Sub index	d1	d0	x	x

X : undefined. Should be 0

server应答 (若成功)：

Data length = 3 bytes

580 + Serv NodeId	0	47	Index	Sub index	d2	d1	d0	x

X : undefined. Sould be 0

server应答 (若成功)：

Data length = 4 bytes

580 + Serv NodeId	0	43	Index	Sub index	d3	d2	d1	d0

server应答 (若失败)：

580 + Serv NodeId	0	80	Index	Sub index	SDO abort code error

|------ 4个字节 -----|

例如：

意read节点5对象字典主索引0x1400、子索引2处的数据0xFD，则client发送：
605 40 00 14 02 00 00 00 00
若成功，则节点5应答：
585 4F 00 14 02 FD 00 00 00

意read节点5对象字典主索引0x1400、子索引1处的数据0x60120208，则client发送：
605 40 03 16 01 00 00 00 00
若成功，则节点5应答：
585 43 03 16 01 08 02 12 60

SDO abort protocol

Abort code (hexa)	
0503 0000	翻转位t位翻转
0504 0000	SDO协议超时
0504 0001	Client/server命令标识符无效或未知
0504 0002	无效的block尺寸(block mode only)
0504 0003	无效的次序号(block mode only)
0504 0004	CRC错误(block mode only)
0504 0005	超出memory
0601 0000	Unsupported access to an object
0601 0001	试图读只写对象
0601 0002	试图写只读对象
0602 0000	对象不在对象字典中
0604 0041	对象不能被映射到PDO上
0604 0042	意映射到PDO上的对象个数和长度超出PDO长度
0604 0043	普通参数不兼容原因
0604 0047	普通器件内部不兼容
0606 0000	由于硬件问题导致访问失败
0607 0010	数据类型不匹配，服务参数长度不匹配有
0607 0012	数据类型不匹配，服务参数长度太高有
0607 0013	数据类型不匹配，服务参数长度太低有
0609 0011	Sub-index不存在
0609 0030	参数值超出范围(only for write access)
0609 0031	Value of parameter written too hight
0609 0032	Value of parameter written too low
0609 0036	最大值比最小值还小有
0800 0000	一般错误
0800 0020	Data cannot be transferred or stored to the application
0800 0021	Data cannot be transferred or stored to the application because of local control
0800 0022	Data cannot be transferred or stored to the application because ofthe present device state
0800 0023	对象字典动态生成失败，或当前没有对象字典。

How to configure a PDO Transmit ?

例如：

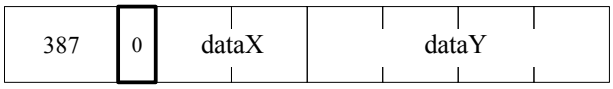
- 配置PDO 0x1800 + n
- 其COBID必须为0x387 (why not ?)
- PDO基于synchro发送
- 在该命令中，必须包含数据：X(2 bytes)和Y(4 bytes)
--- dataX定义在主索引0x6000、子索引03处
--- dataY定义在主索引0x2010、子索引21处

1 – 主索引1800 + n, 子索引01 : 写COBID(4个字节)
2 – 子索引02 : 写传输类型 « t » (1个字节)
t = 1 to 0xF0 : PDO每接收到« t »个SYNC就发送;
t = FD : 在接收到一个请求PDO (rtr = 1)后发送;
t = FF : 基于事件发送。节点自发主动地发送PDO。
3 – 主索引1A00 + n : 定义nth数据的映射关系
子索引0 : 写入该PDO有几种数据(1个字节)。对于该例写« 2 »
子索引1 : 定义第一个数据位于哪里、尺寸(8个字节)
其格式为: 主索引(2 bytes) – 子索引(1 byte) – 位尺寸(1 byte)
对于该例，写« 6000 03 08 »
子索引2 : 定义第二个数据位于哪里、尺寸(8个字节)
对于该例，写« 2010 21 20 »

意配置节点5的PDO 1802每3个synchro发送一次，则SDOs应该发送(尚未测试)：

```
605  23 02 18 01 00 00 07 38
605  2F 02 18 02 03 00 00 00
605  2F 02 1A 00 02 00 00 00
605  23 02 1A 01 08 03 00 60
605  23 02 1A 02 20 21 10 10
```

The PDO :



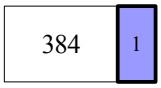
Note : 一个PDO最多只能携带8个字节数据。

What is a PDO Transmited « on request » ?

即一个，必须在接收到一个具有与该PDO相同COBID的远程传输请求(rtr)时，发送的PDO。

例如：

PDO 请求 :



若某个COBID为384的PDO 为« on request », 则其必须被发送。

How to configure a PDO Receive ?

例如 :
- 配置PDO 0x1400 + n
- 其COBID必须为0x183 (why not ?)
- 该PDO为同步传输
- 其必须包含数据: data X (2 bytes) 和data Y (4 bytes) 在该命令中
--- dataX被定义在主索引0x6000 子索引03
--- dataY被定义在主索引0x2010 子索引21

1 – 主索引1400 + n, 子索引01 : 写COBID(4 bytes)
2 – 主索引1400 + n, 子索引02 : 写传输类型« t » (1 byte)
t = 1 to 0xF0 : PDO每接收到« t »个SYNC就发送;
t = FD : 在接收到一个请求PDO (rtr = 1)后发送;
t = FF : 基于事件发送。节点自发主动地发送PDO。
3 – 主索引1600 + n : 定义映射关系.
子索引0 : 写入该PDO有几种数据(1个字节)。对于该例写« 2 »
子索引1 : 定义第一个数据位于哪里、尺寸(8个字节)
其格式为: 主索引(2 bytes) – 子索引(1 byte) – 位尺寸(1 byte)
对于该例，写« 6000 03 08 »
子索引2 : 定义第二个数据位于哪里、尺寸(8个字节)
对于该例，写« 2010 21 20 »

意配置节点5的PDO 1402为每3个syncchro就被接收下来， 则SDOs应该发送(尚未测试) :
605 23 02 14 01 00 00 07 38
605 2F 02 14 02 03 00 00 00
605 2F 02 16 00 02 00 00 00
605 23 02 16 01 08 03 00 60
605 23 02 16 02 20 21 10 10

The PDO :

Question : Why do we need to configure a « transmission type » for a PDO receive ?
Answer : I don't know

关于PDO接收和发送 (receive/transmit)的重要提示 :
--- PDO不能携带超过8个字节的数据.
--- 没有必要使用节点ID来组成COBID (You are not obliged to make the cobId with the node id)。但这些值需受约束:
最高4位(bit 10 to 7) – 为功能码 - 指示消息为PDO，则必须为以下一种(二进制) :
0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010。
你可以自由选择低7位 (bits 6 to 0)，除了0以外。
因此, **允许用于某个PDO的CobId为(hexa) :**
181 to 57F ， 除了200, 280, 300, 380, 400, 480, 500

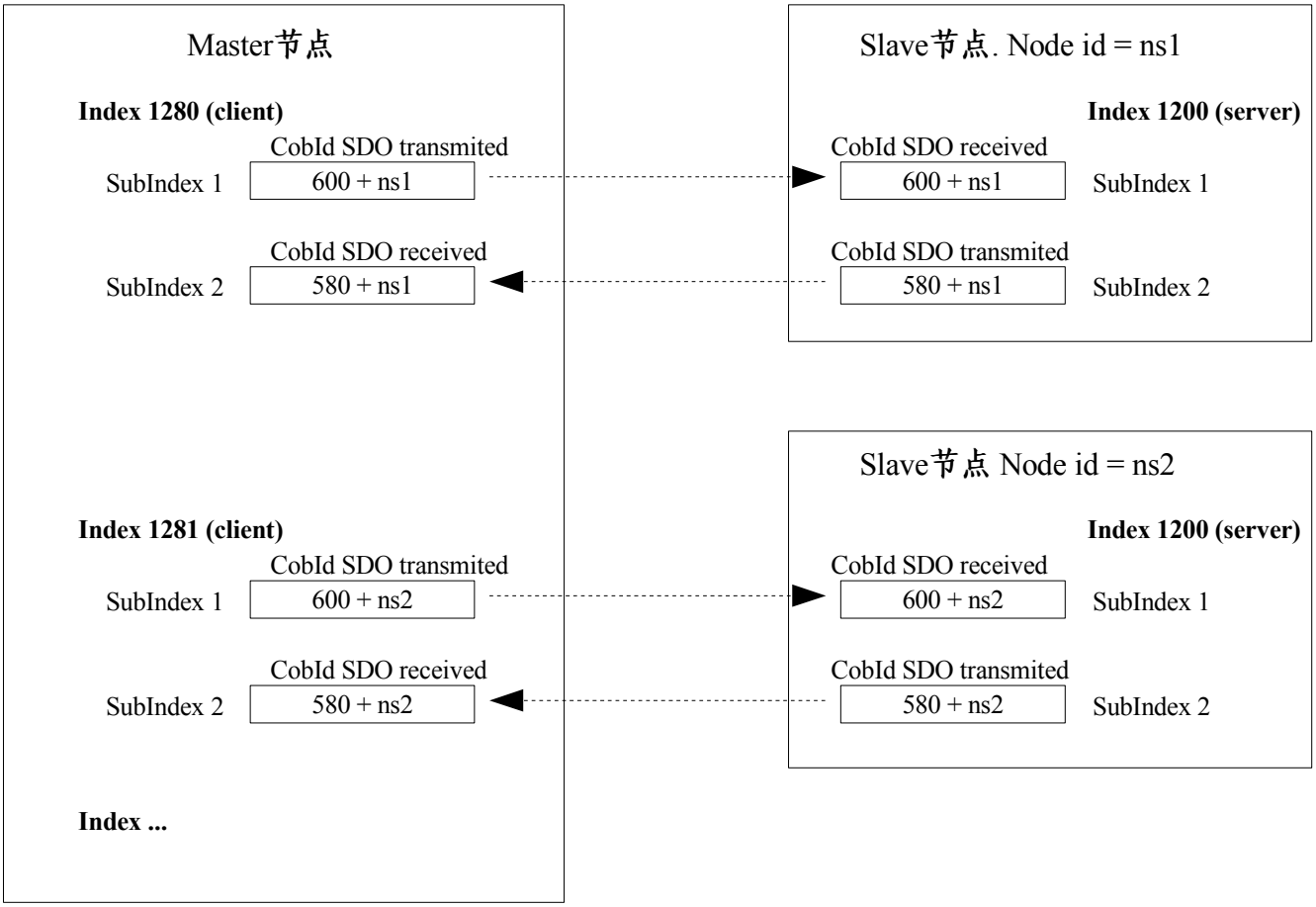
How to configure a SDO client ?

在绝大多数CANopen网络中，每个slave节点仅实现一个SDO server(主索引1200)，以用于接收来自client节点的SDO。默认下，slave节点SDO功能是很好配置的。
通常，任一slave节点无需实现任何SDO client，因为它通常无需发送任何SDO给其他节点。

如果Master节点需要能够配置其他slave节点，就不是件简单的事情了。
意发送SDO给其他slave节点，其就必须有多个SDO clients(每个slave一个).
这些入口必须配置在主索引1280, 1281, ...

配置在主索引0x1280+n的SDO client，用于与节点nsl(slave节点)通讯。

1 – 主索引1280 + n, 子索引 01 : write the cobId transmit (4 bytes) : 600 + nsl
2 – 子索引 02 : write the cobId receive (4 bytes) : 580 + nsl
3 – 子索引 03 : write the slave node id (1 byte) : nsl (可选)



How to configure a node to send the SYNC ?

一个节点可以周期性的发送SYNC信号。这可以由master或某个slave节点来实现。

1 – 主索引1006, 子索引00 : 写入周期，单位微秒(4个字节) :

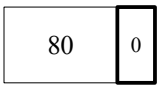
例如 :
意每10ms (即10000us) 发送一个SYNC, 则写入值: 0x2710 若为节点9，则SDO应为:

609 23 06 10 00 10 27 00 00

意停止之，写0。
意启动之，写0x40000080 到主索引0x1005,子索引0处。

Note: 节点必须处于operational模式才能发送SYNC。

What is the SYNC message ?



SYNC仅用于PDO。

How to configure a node to send its heartbeat ?

意发送其心跳每n个ms：：
1 – 位于主索引0x1017 – 子索引00 : 写 « n ». (2个字节) :

例如 :
意每100ms发送一个心跳，则写入值为0x64
若节点号为9，则SDO应为:
609 2B 17 10 00 64 00 00 00
意停止之，写0。

How to configure a node to monitor the heartbeats of several nodes ?

例如 :
该节点期待2个来自其他节点的心跳 :
« nodeid1 », 最少每 n1 ms
« nodeid2 », 最少每 n2 ms

1 – 主索引1016, 子索引00 : 需要检测多少个心跳。(4个字节)。对于该例值为« 2 ».
2 - 子索引01 (4个字节) : 第一个节点值 : 00 – nodeid1 (1 byte) – n1 (2 bytes)
3 - 子索引02 (4个字节) : 第二个节点值 : 00 – nodeid2 (1 byte) – n2 (2 bytes)
|----------4个字节----------|

例如 :
意配置节点9来监测其他节点的心跳:
node 2 : 至少每100(dec) ms = 0x64
node 3 : 至少每400(dec) ms = 0x190
node 7 : 至少每4000(dec) ms = 0xFA0
则SDO应为:
609 2B 16 10 00 03 00 00 00 // 3个入口
609 2B 16 10 01 64 00 02 00 // 写入的数据为 0x00 02 00 64
609 2B 16 10 02 90 01 03 00 // 写入的数据为 0x00 03 01 90
609 2B 16 10 03 A0 0F 07 00 // 写入的数据为 0x00 07 0F A0

意停止之，写0 例如在子索引01 , . .,

What is this message ?

功能码 (ID-bits10-7)	COBID(hex)	Protocol	通讯参数在OD中的索引
0000	000	NMT模块控制	
0001	080	SYNC	1005H, 1006H, 1007H
0001	081-0FF	EMERGENCY	1024H, 1015H
0010	100	TIME STAMP	1012H, 1013H
0011->1010	181-57F	PDO*	多个
1011	581-5FF	SDO response (server->client)	1200H
1100	601-67F	SDO request (client->server)	1200H
1110	701-77F	NMT error control	1016H, 1017H

*除了：200, 280, 300, 380, 400, 480, 500。

其中NM模块控制、SYNC、TIME STAMP为广播对象，其他为对等对象。