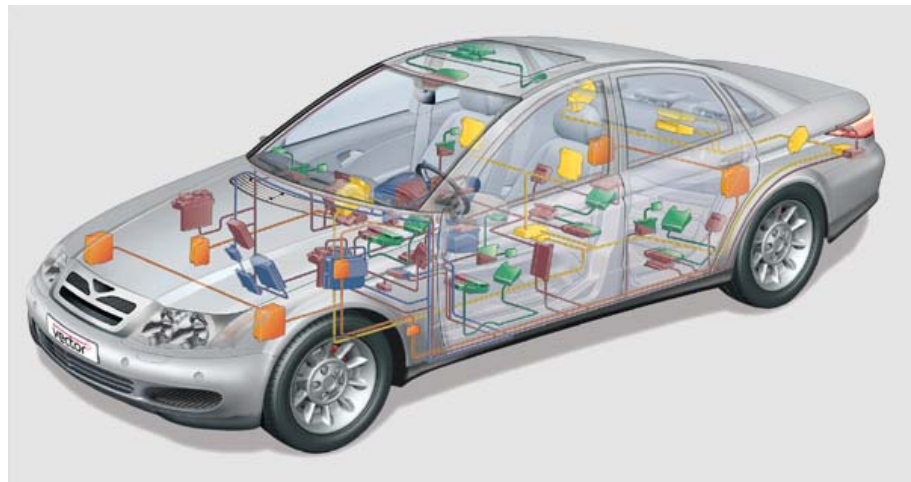


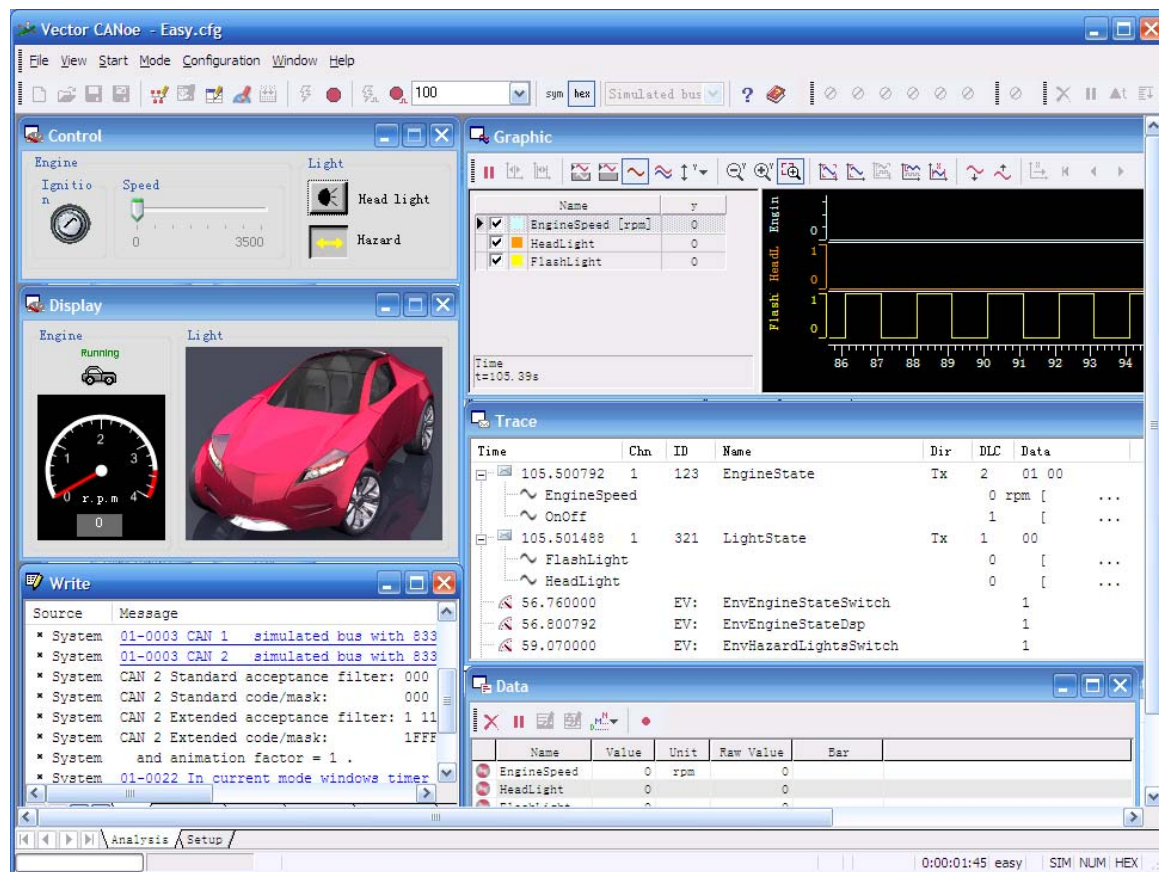
# CANoe快速入门



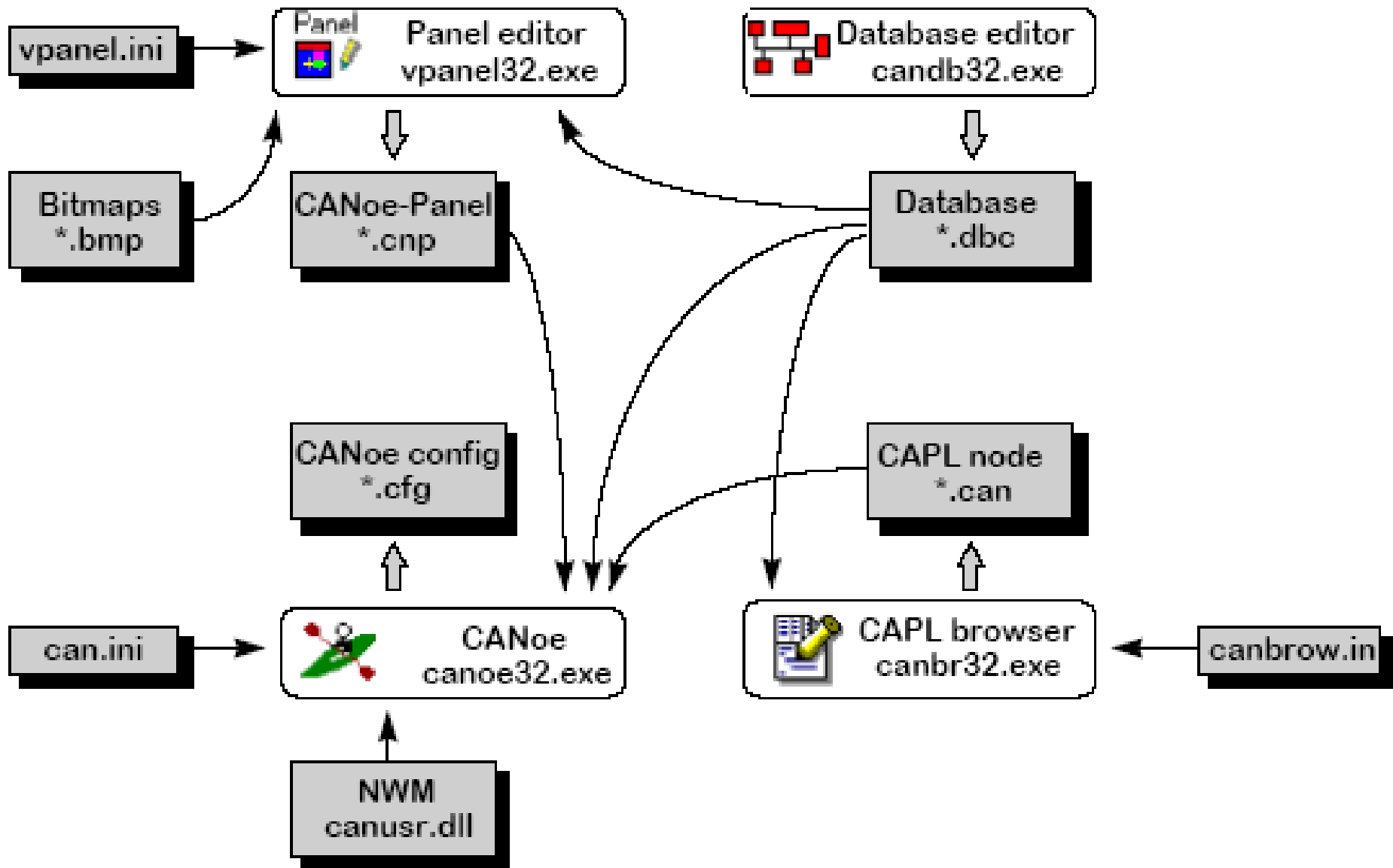
## 为什么叫CANoe? CAN open environment

### □ CAN总线开发工具

- 仿真
- 测试
- 分析
- 记录



# CANoe组成



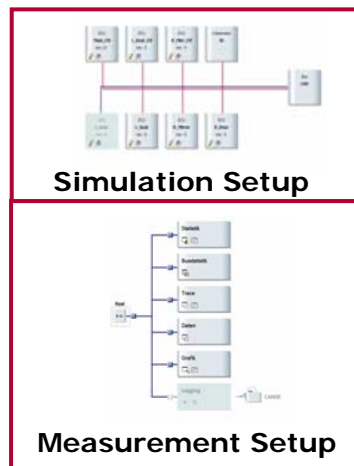
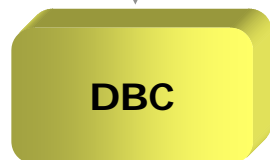
# CANoe在总线开发中的作用 (1)

## 第一阶段 – 网络设计和仿真

数据库  
CANdb++ Editor

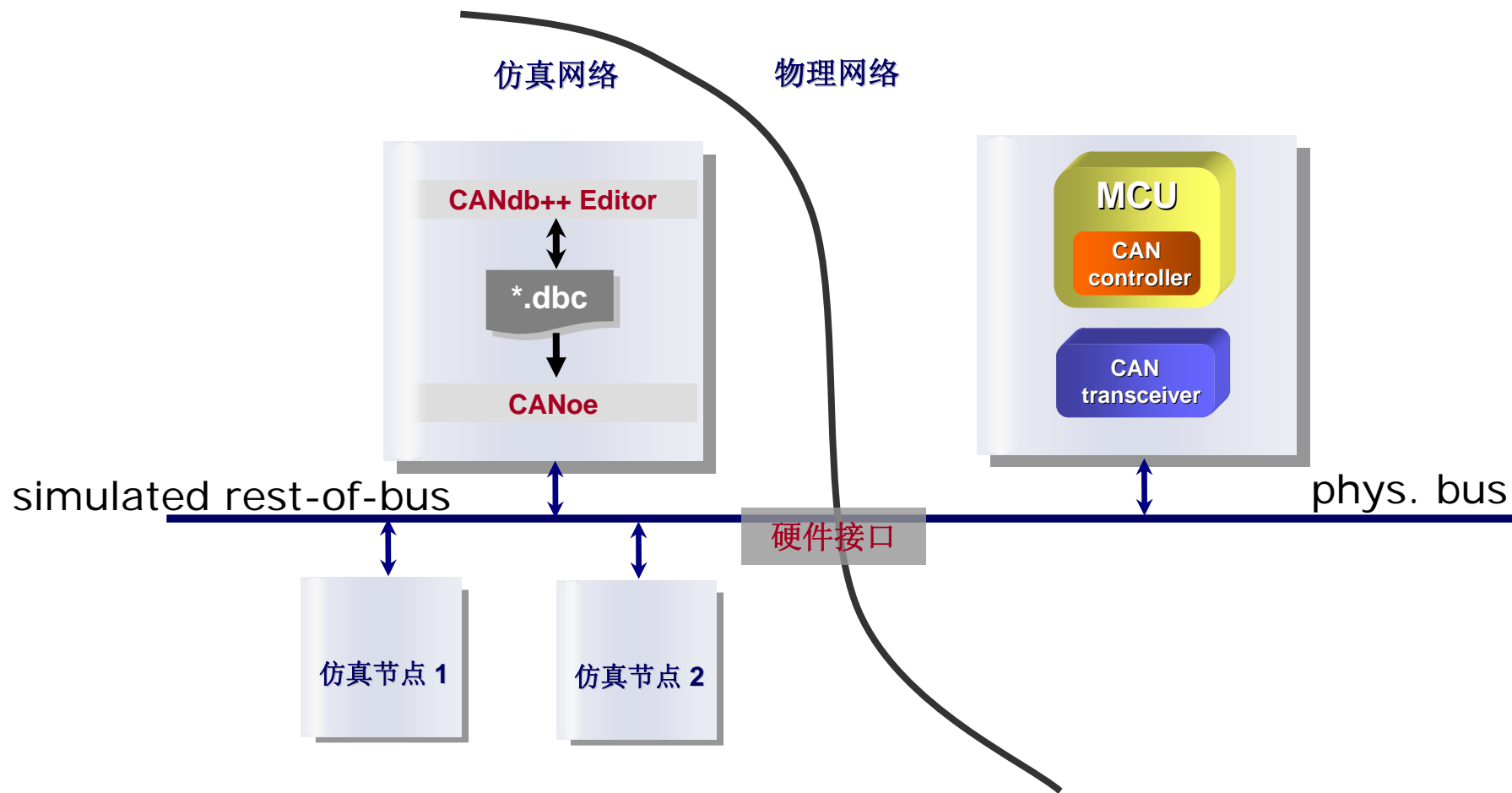
仿真验证  
CANoe

硬件接口卡&“狗”  
CANcaseXL/CANcardXL



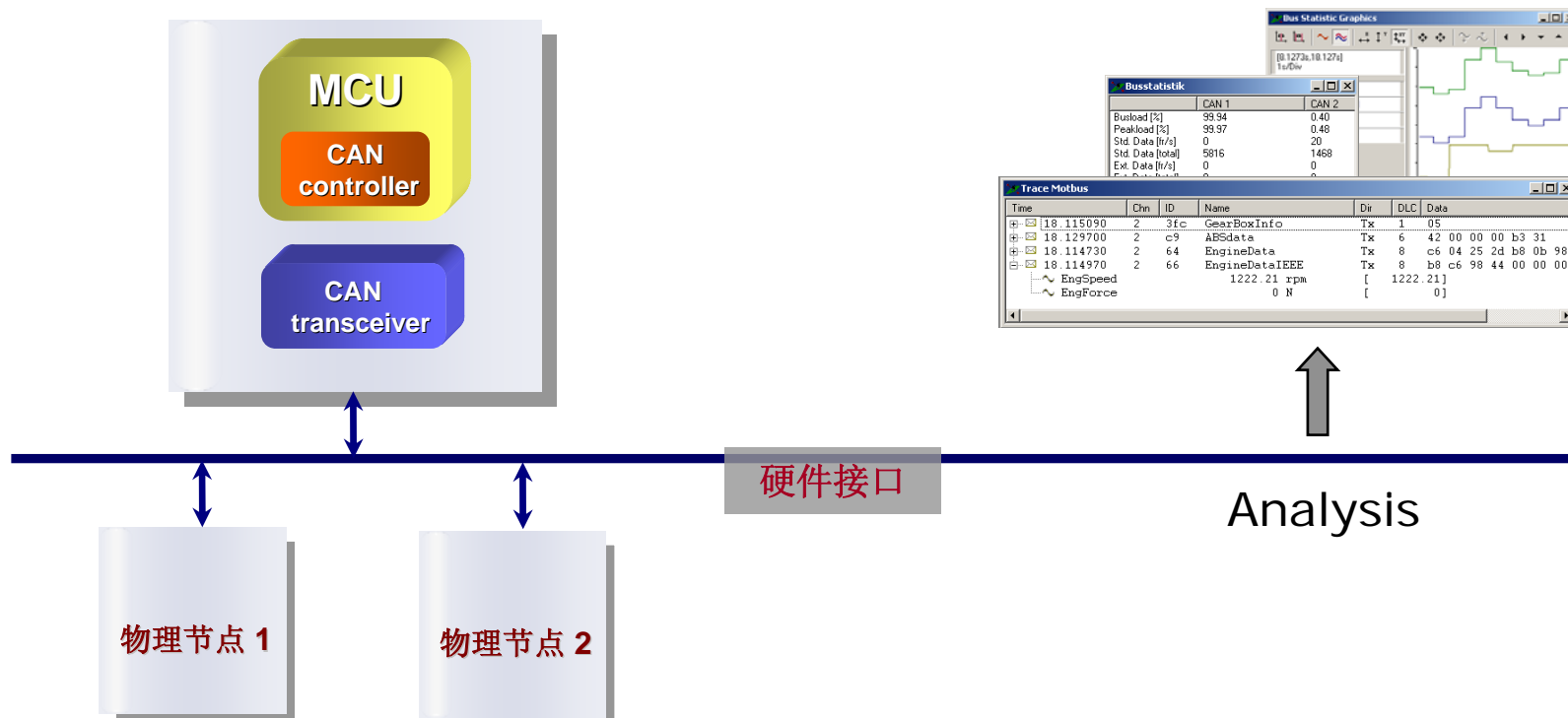
# CANoe在总线开发中的作用 (2)

## 第二阶段 – 节点设计

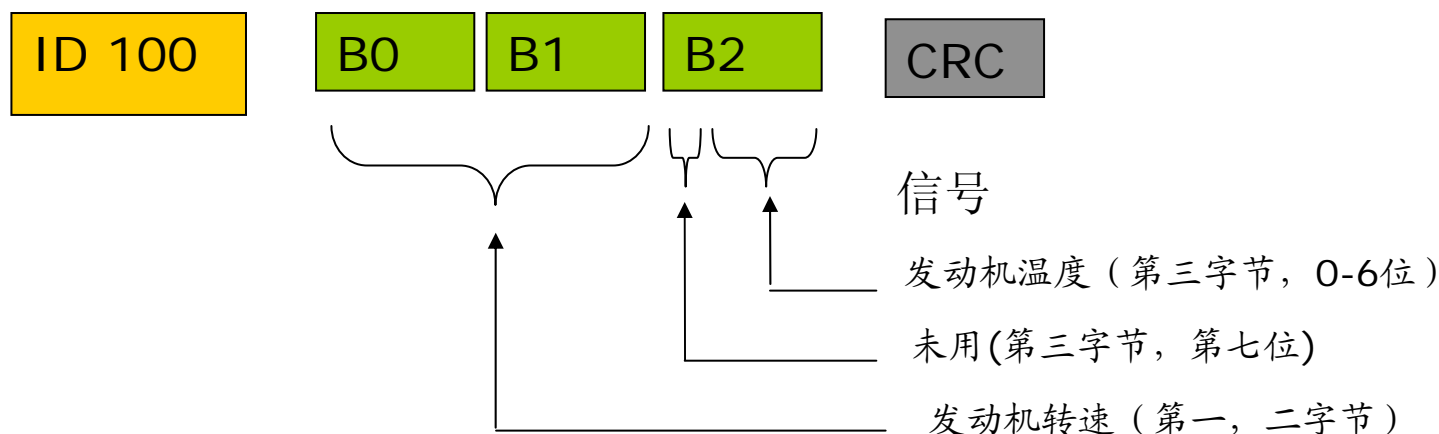


# CANoe在总线开发中的作用 (3)

## 第三阶段 – 系统集成



## 报文：engine data (ID 100)



转换规则

发动机转速 :  $\text{rpm} = 1 * \text{Bit value}$  (0xFF 代表错误)

发动机温度:  $^{\circ}\text{C} = 2 * \text{Bit value} - 50$  (0x7F 代表错误)

## □ 环境变量

- 节点的I/O信号
- 可用于面板或真实I/O

## □ 系统变量

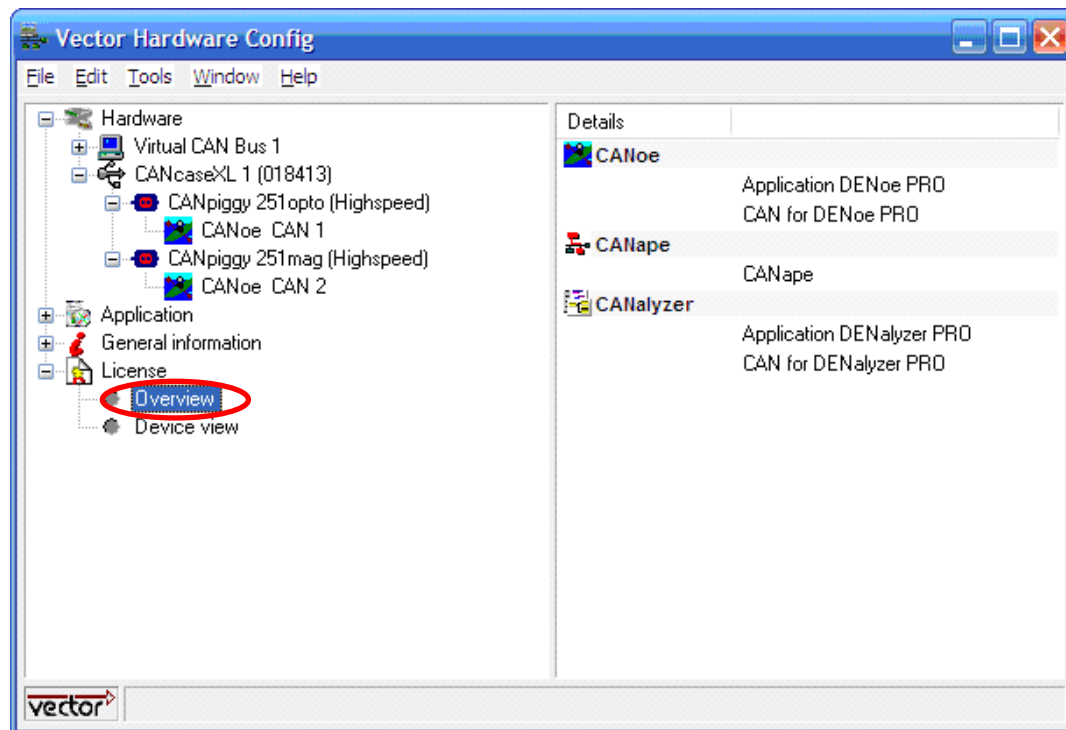
- 节点内部参数
- 或需要观测的某个数值
  - 例如：系统变量1 = 报文1.信号1 - 报文2.信号2



## □ CANoe

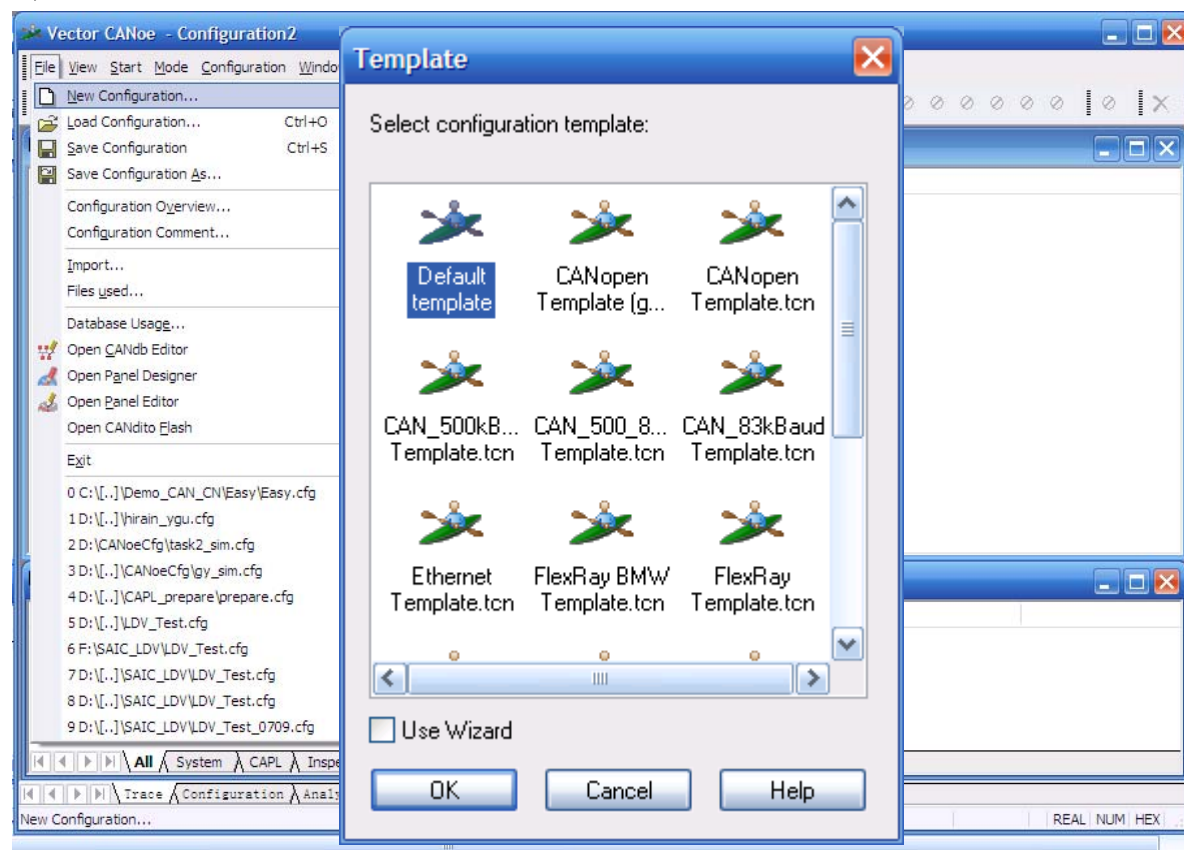
□ 确认CAN卡license信息

□ 控制面板-Vector Hardware



## □ CANoe

### □ 新建配置工程

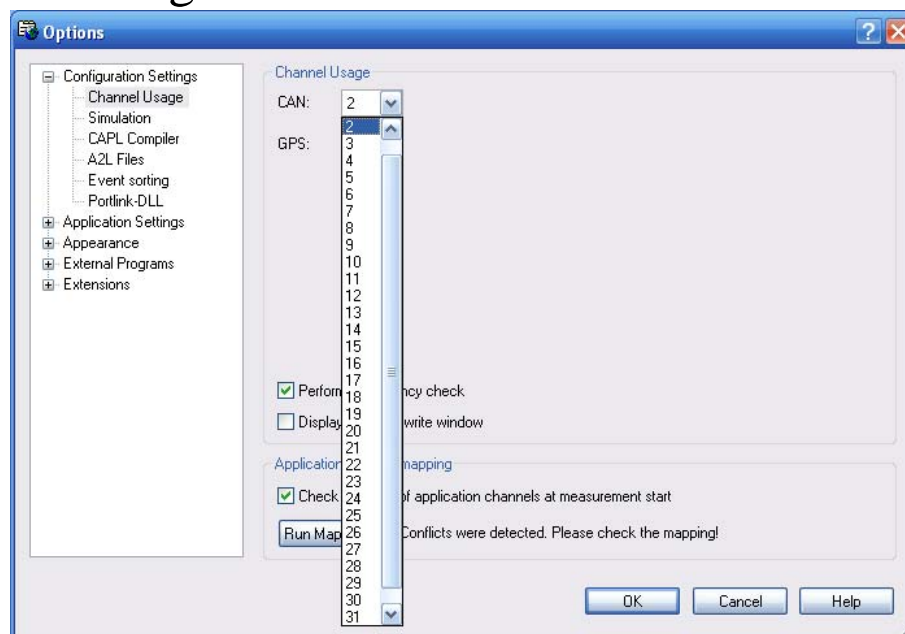


## □ CANoe

### □ 通道数设置

### □ Configuration->Options

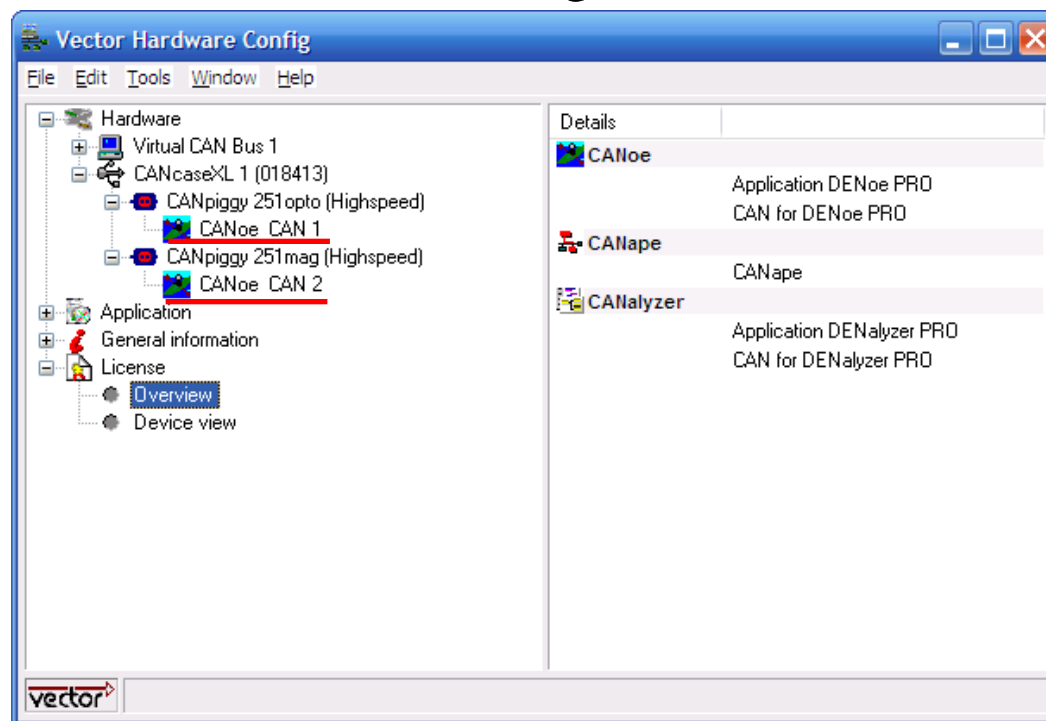
### □ Configuration Settings->Channel Usage



## □ CANoe

### □ 通道配置

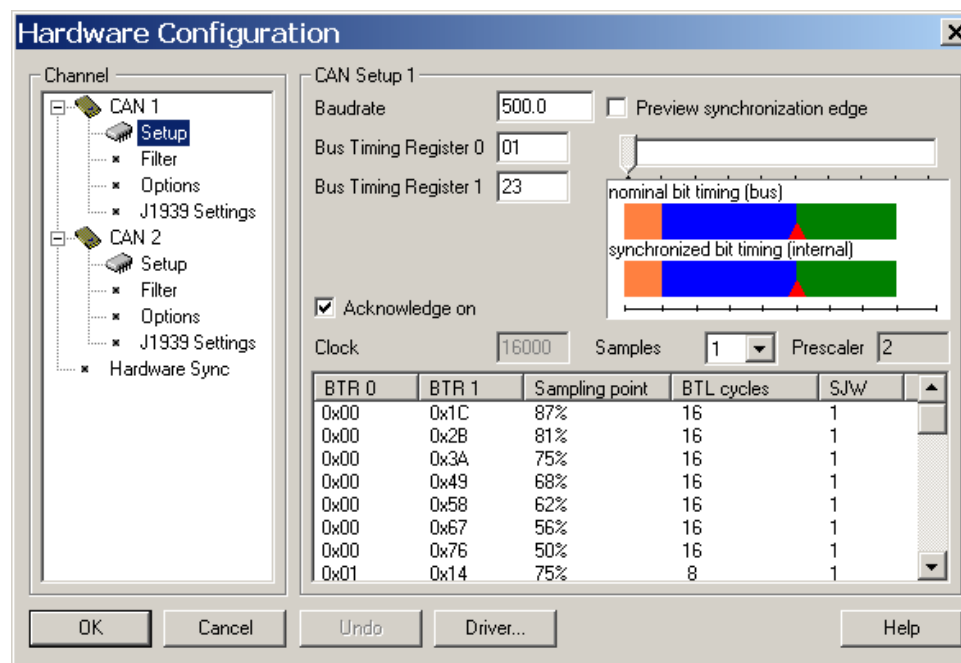
### □ Configuration->Hardware Configuration->Driver



## □ CANoe

### □ 波特率设置

### □ Configuration->Hardware Configuration



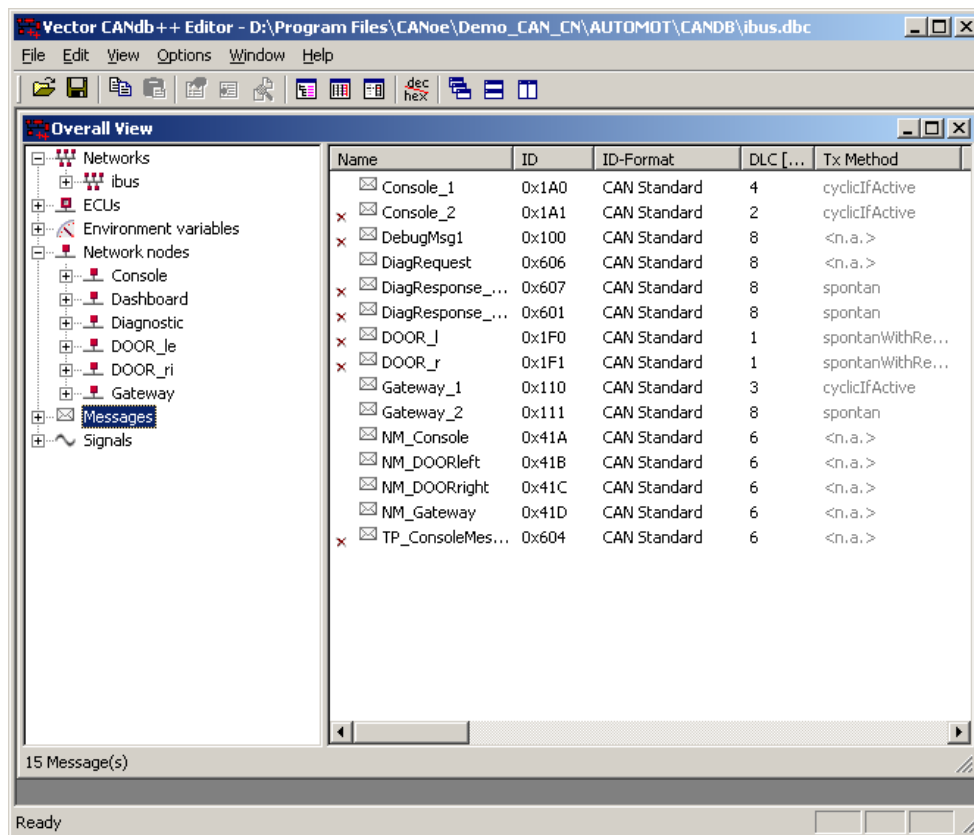
# CAN总线开发Step One: 新建数据库

## □ DBC文件编辑工具

### □ 启动CANoe

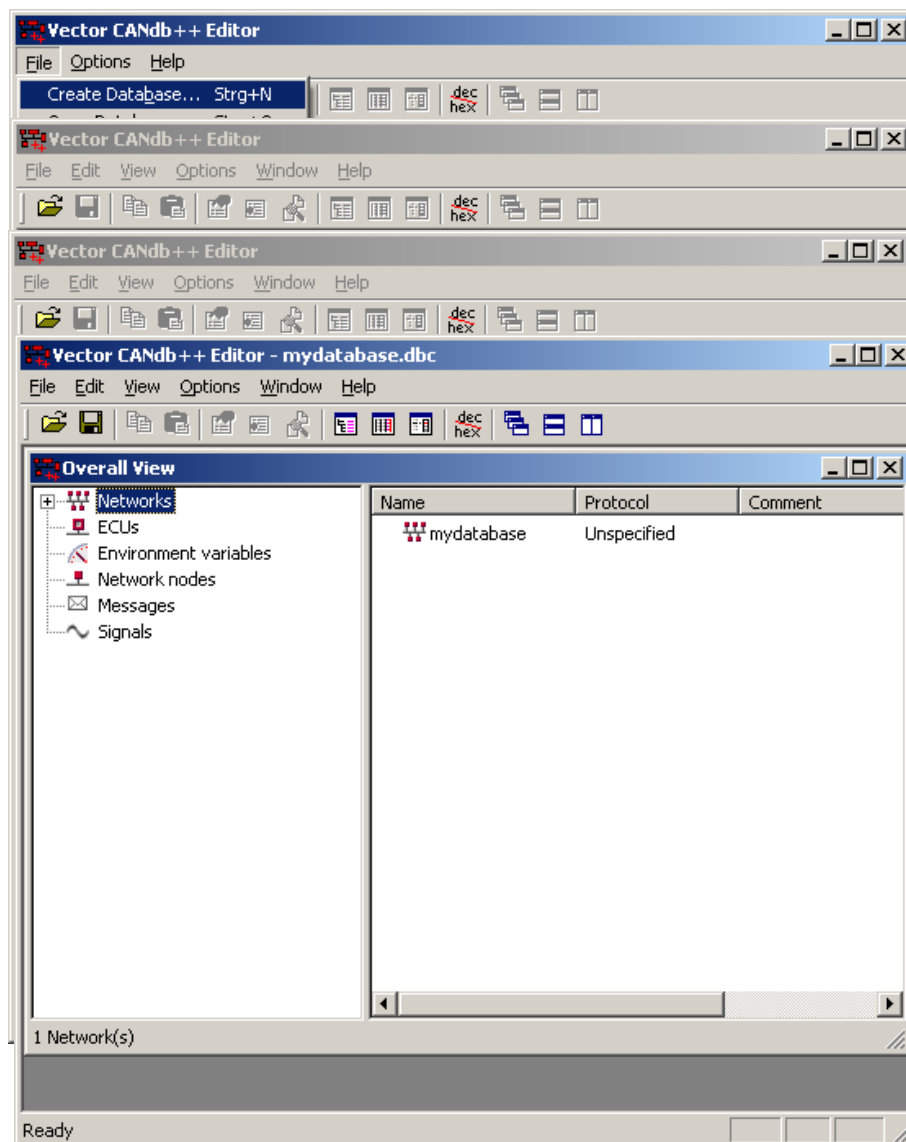
#### □ File->Open CANdb Editor

#### □ 点击



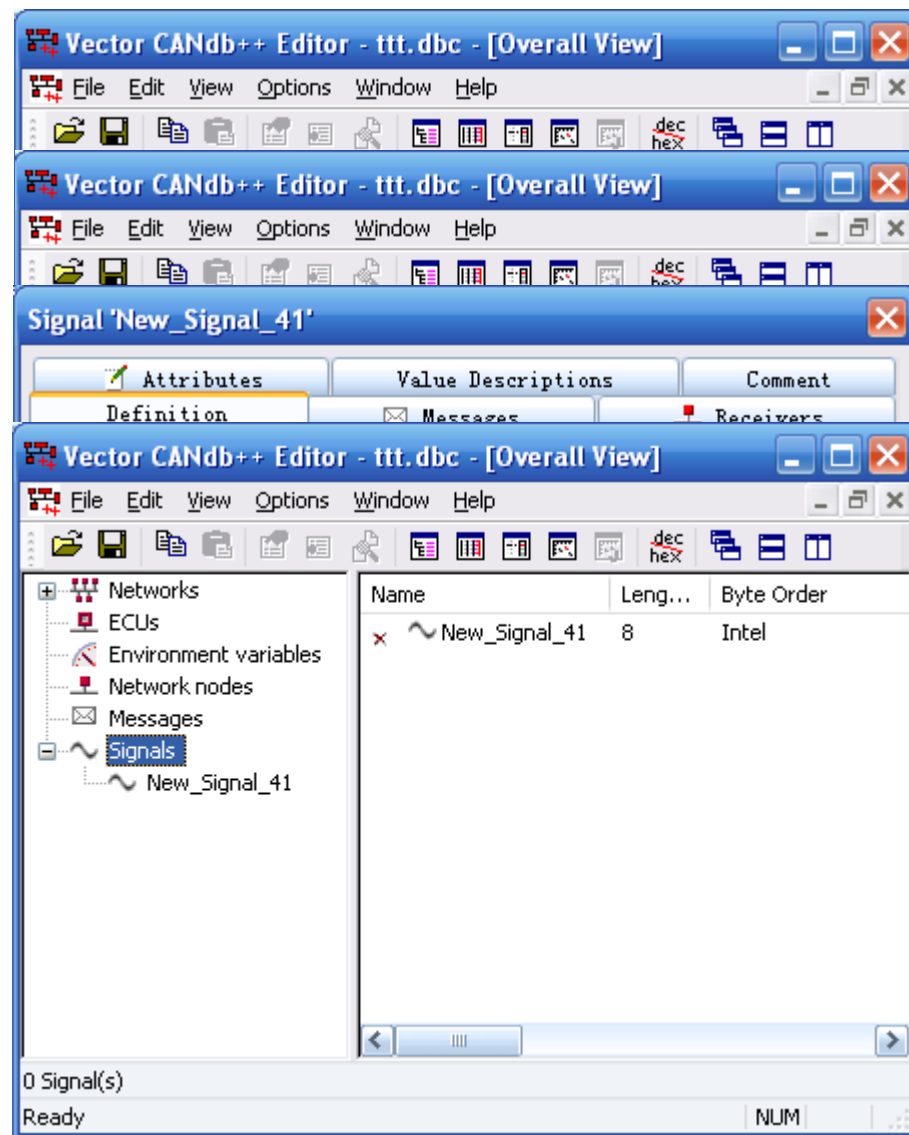
# 创建一个新的CAN数据库

- ❑ File->Create Database...
- ❑ 选择模板，鼠标双击或按 **[OK]**按钮
- ❑ 指定数据库文件类型、文件名及保存目录
- ❑ 按**[Save]**按钮。  
一个新数据库创建完成



# 创建对象（信号、报文、节点、环境变量和ECU）

- ❑ 在Overview窗口左边  
选择所需创建对象的类型
- ❑ 右键点击对象类型，  
在快捷菜单中选择New...
- ❑ 使用配置对话框设置  
所创建对象的系统参数值
- ❑ 点击[确定]按钮，  
一个新对象便创建完毕

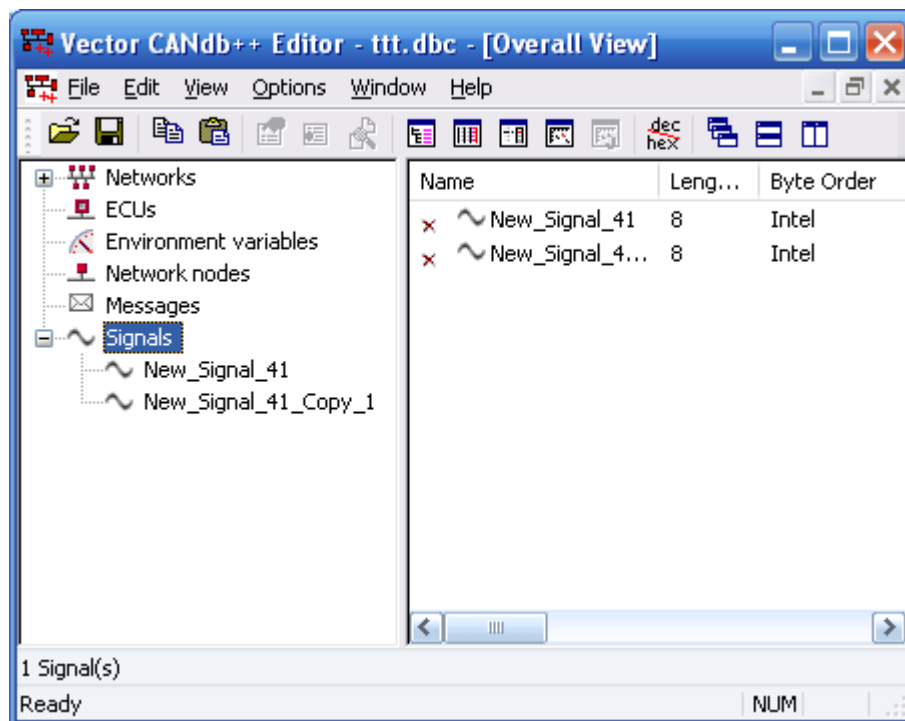




# 复制已有对象

## □ Copy-Paste

- 选择已有对象Ctrl+c
- 选择对象类型Ctrl+v



# 修改/编辑已有对象

## □ 直接双击

Signal 'New\_Signal\_41'

Attributes | Value Descriptions | Comment | Messages | Receivers

Definition

Name: New\_Signal\_41

Length: 8

Byte Order: Intel Unit:

Value Type: Signed Init: 0

Factor: 1 Offset: 0

Minimum: 0 Maximum: 0

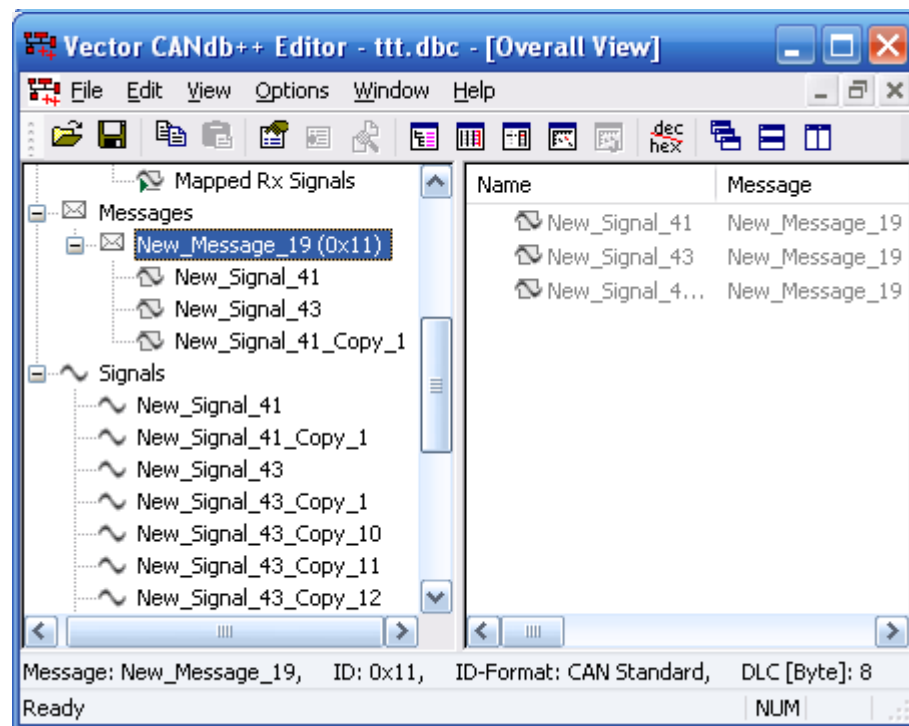
Value: <none>

☒ Automatic min-max calculation

确定 取消 应用(A) 帮助

# 对象链接(1/2)

- 信号与报文之间的连接
- 发送报文与节点之间的连接
  - 鼠标拖拽或Copy-Insert



## □ 接收报文与节点之间的连接

### □ 通过信号间接定义

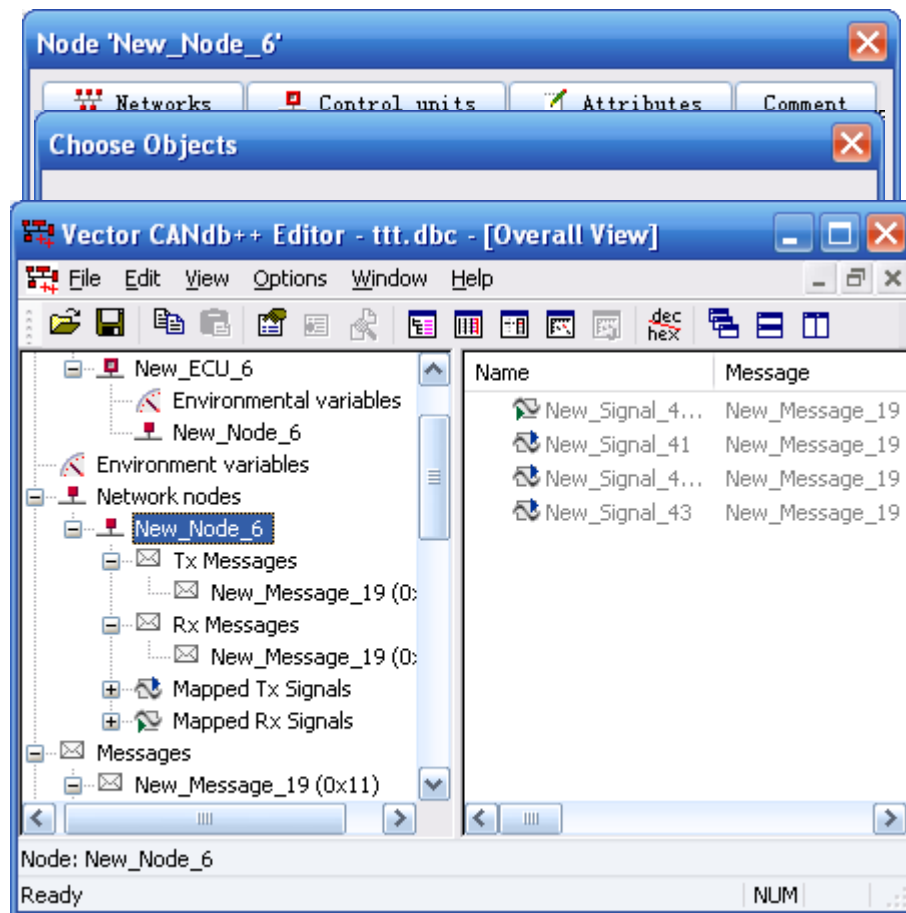
#### □ 双击节点，

选择Mapped Rx Sig.页签

#### □ 点击Add..., 选择接收信号

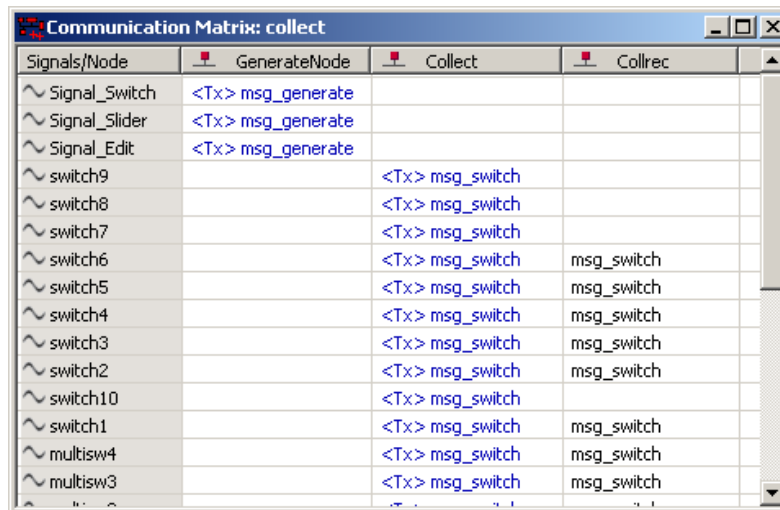
点击OK

#### □ 点击确定



## □ View->Communication Matrix...

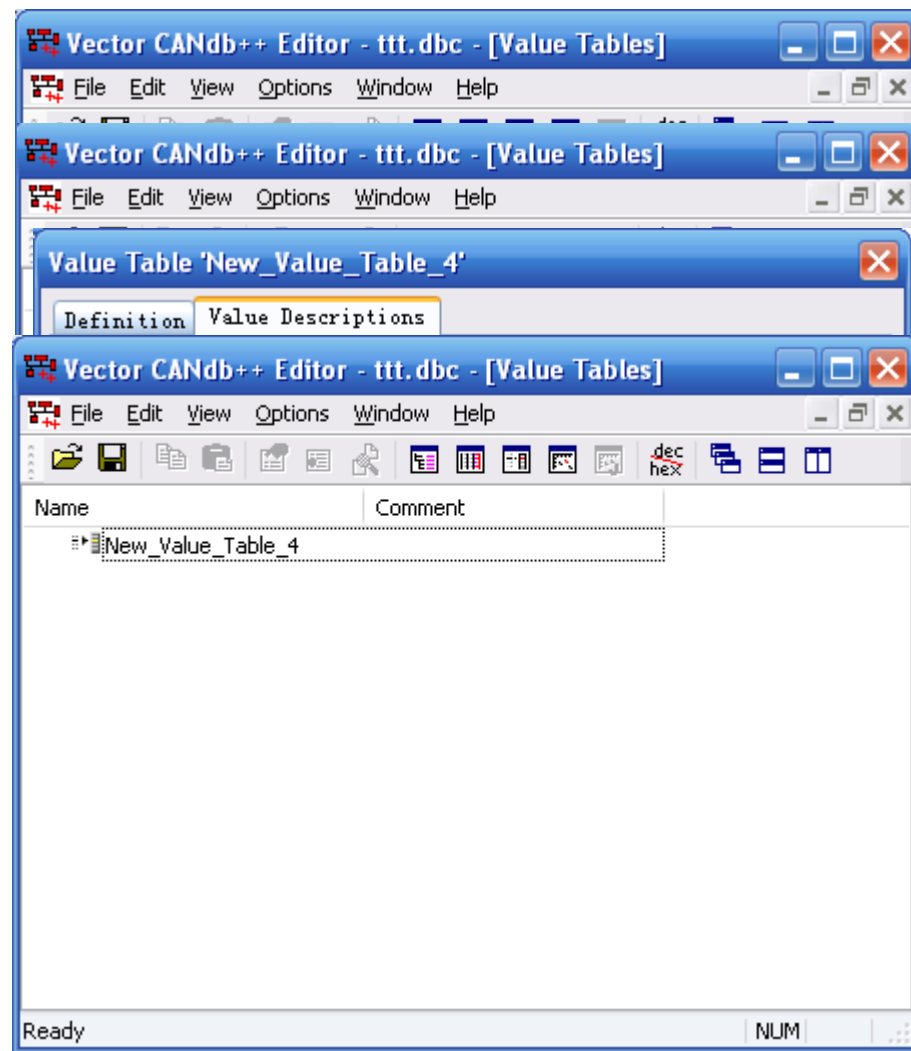
- 显示信号、消息、及网络节点的关系
- 以信号为行，网络节点为列
- 消息名显示于表中，对应了包含的信号与发送/接收的节点



Signals/Node	GenerateNode	Collect	Collrec
~ Signal_Switch	<Tx> msg_generate		
~ Signal_Slider	<Tx> msg_generate		
~ Signal_Edit	<Tx> msg_generate		
~ switch9		<Tx> msg_switch	
~ switch8		<Tx> msg_switch	
~ switch7		<Tx> msg_switch	
~ switch6		<Tx> msg_switch	msg_switch
~ switch5		<Tx> msg_switch	msg_switch
~ switch4		<Tx> msg_switch	msg_switch
~ switch3		<Tx> msg_switch	msg_switch
~ switch2		<Tx> msg_switch	msg_switch
~ switch10		<Tx> msg_switch	
~ switch1		<Tx> msg_switch	msg_switch
~ multislw4		<Tx> msg_switch	msg_switch
~ multislw3		<Tx> msg_switch	msg_switch

# 数值表(1/2)

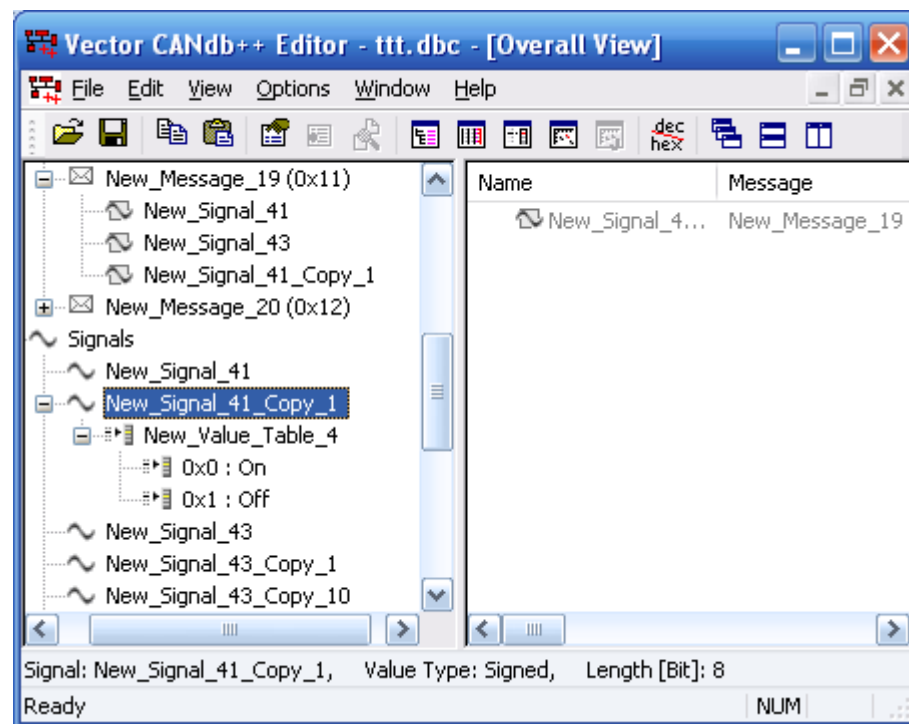
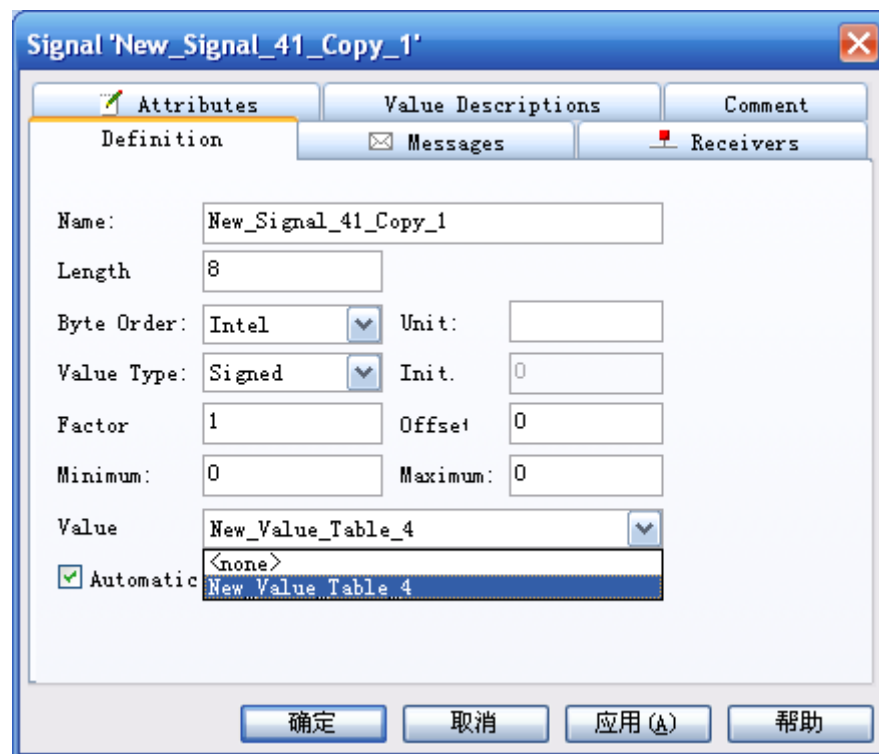
- 新建数值表
  - View->Value Tables
  - 右键点击空白处，  
选择New...
  - 在对话框中输入数值，  
点击确定
  - 新的数值表创建完成



# 数值表(2/2)

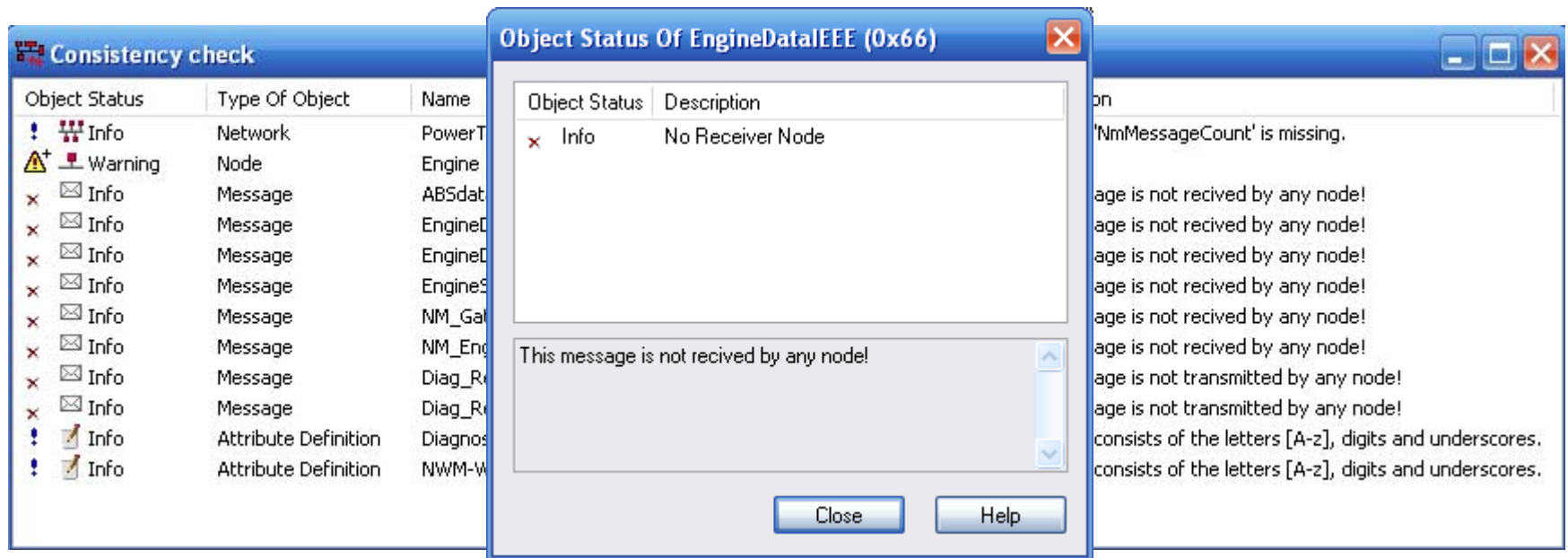
## □ 分配数值表

### □ 数值表可以分配给信号或环境变量



# 一致性检查

## □ File-> Consistency Check

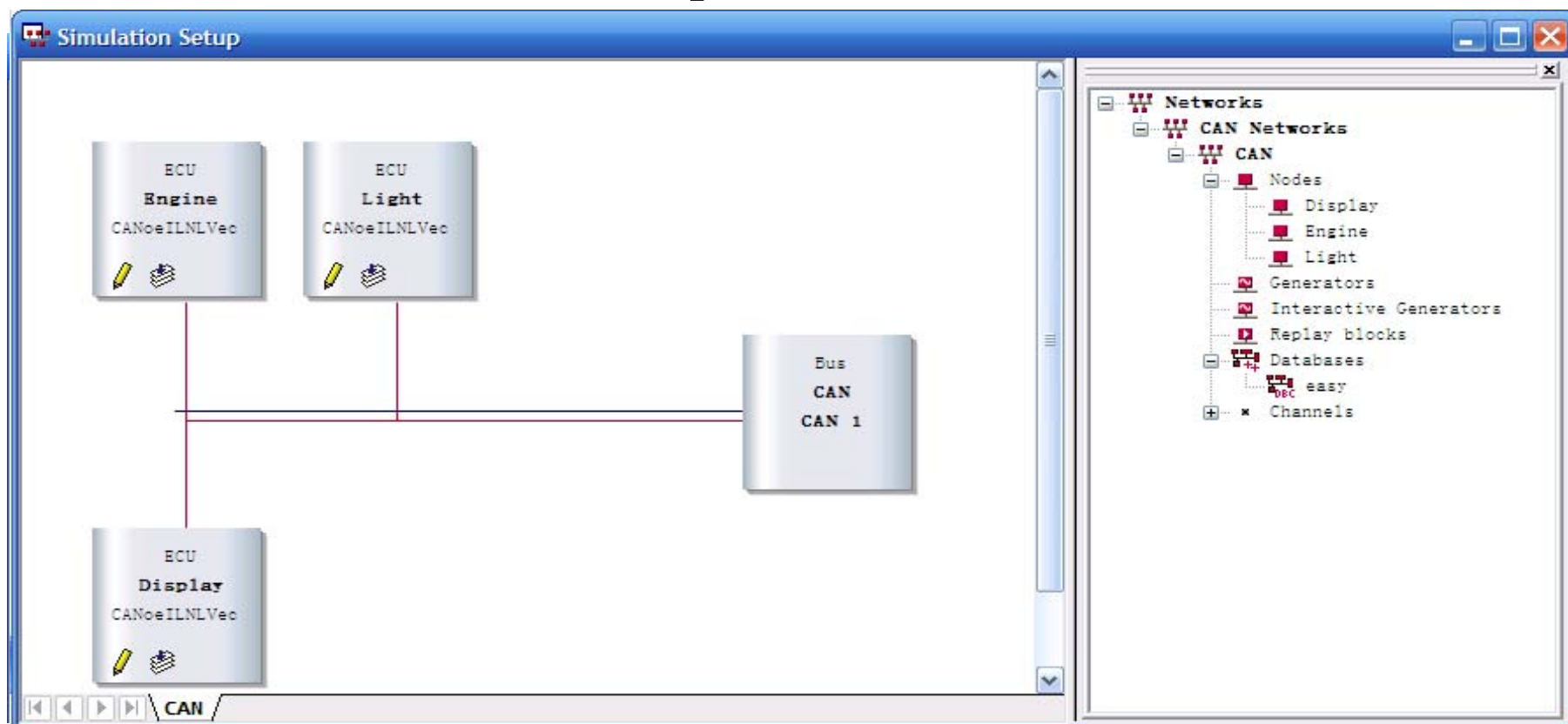




# 添加数据库

□ 在CANoe中添加数据库

□ View->Simulatioin Setup

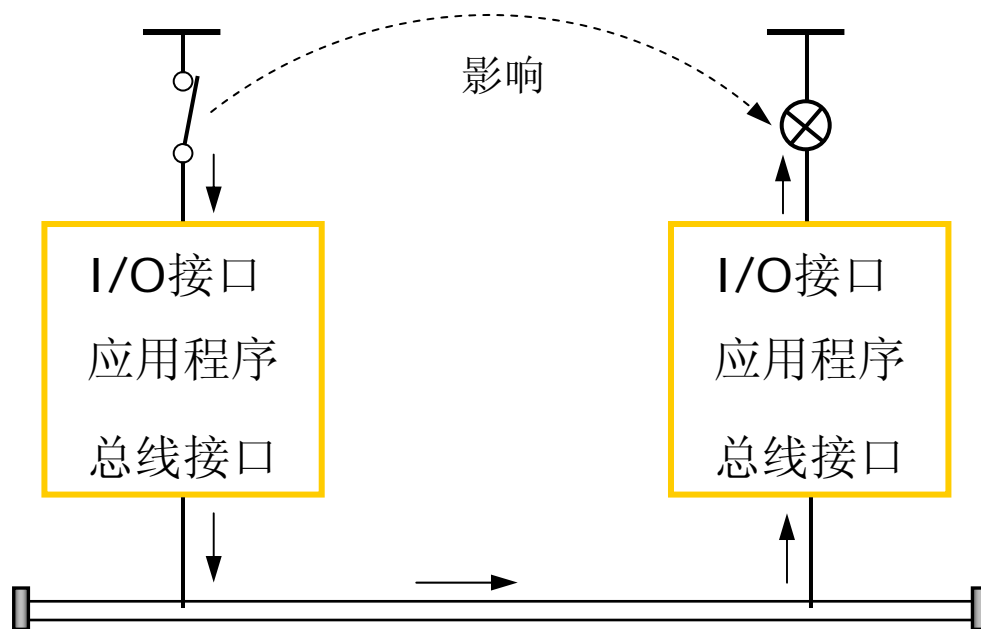


# Excise1

## 1) 分析网络，建立数据库

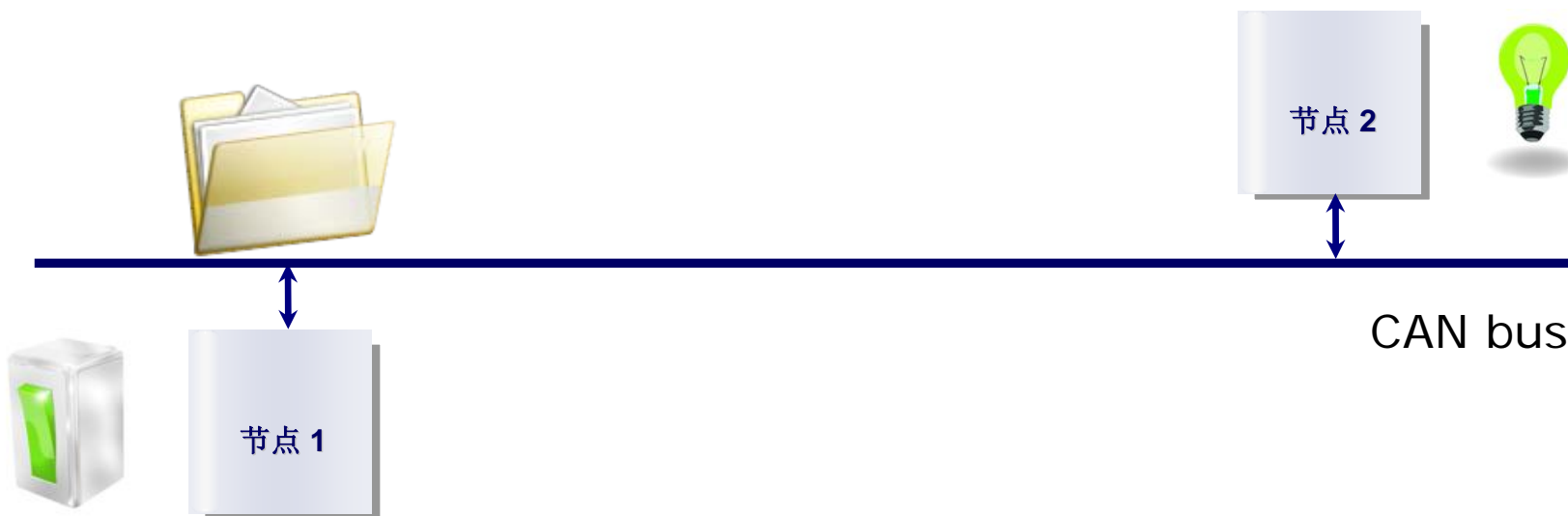
How many nodes?      How many messages?      How many signals?

## 2) 新建CANoe配置工程，并加入数据库文件



# CAN总线开发Step Two: 仿真建模

仿真建模  $\longrightarrow$  节点行为定义, eg: 报文发送和接收



报文发送 { 简单  $\rightarrow$  发生器模块实现  
              复杂  $\rightarrow$  CAPL编程实现

报文接收 CAPL编程实现

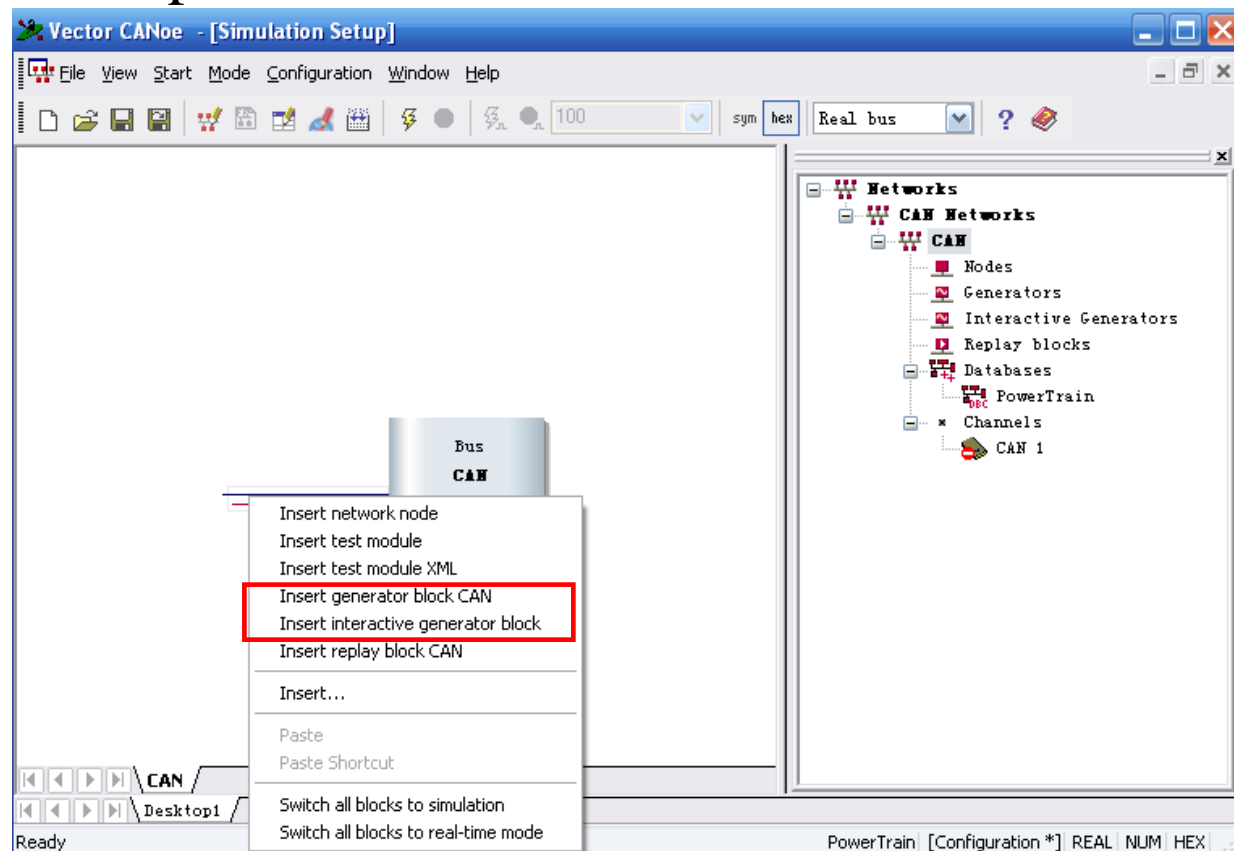
# Easy simulation

## □ Simulation Setup

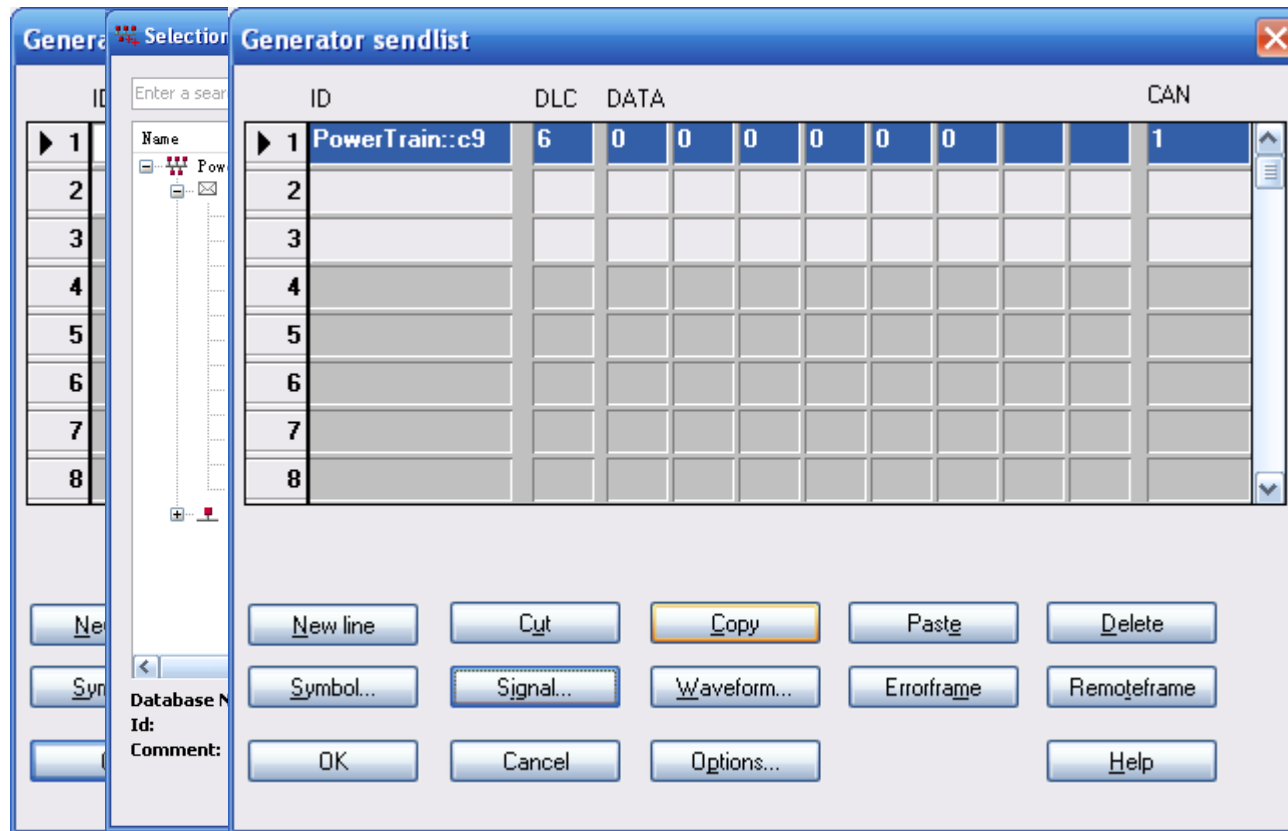
### □ View->Simulation Setup

□ 发生器

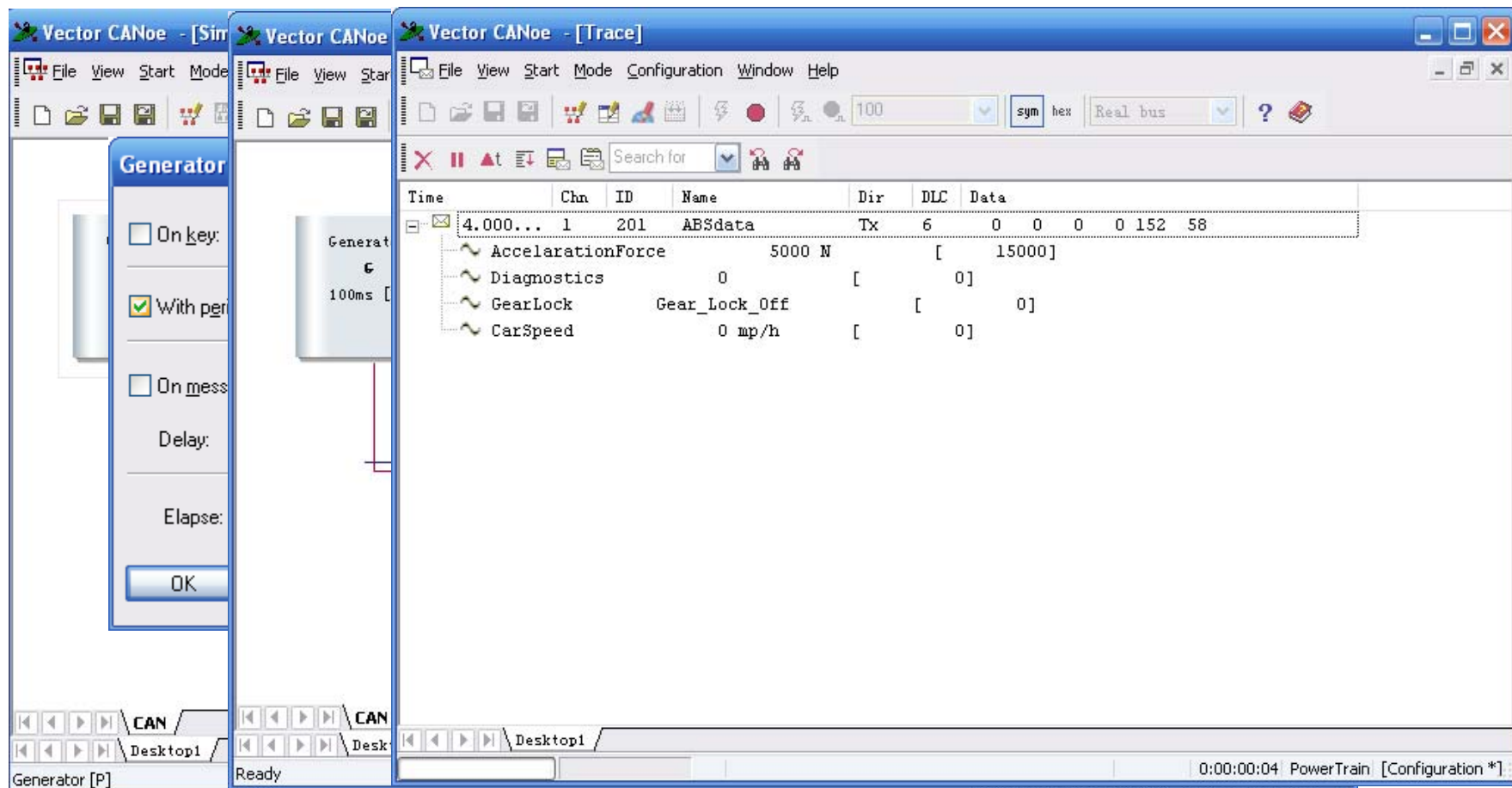
□ 交互式发生器



# 发生器模块



# 发生器模块



# 发生器模块

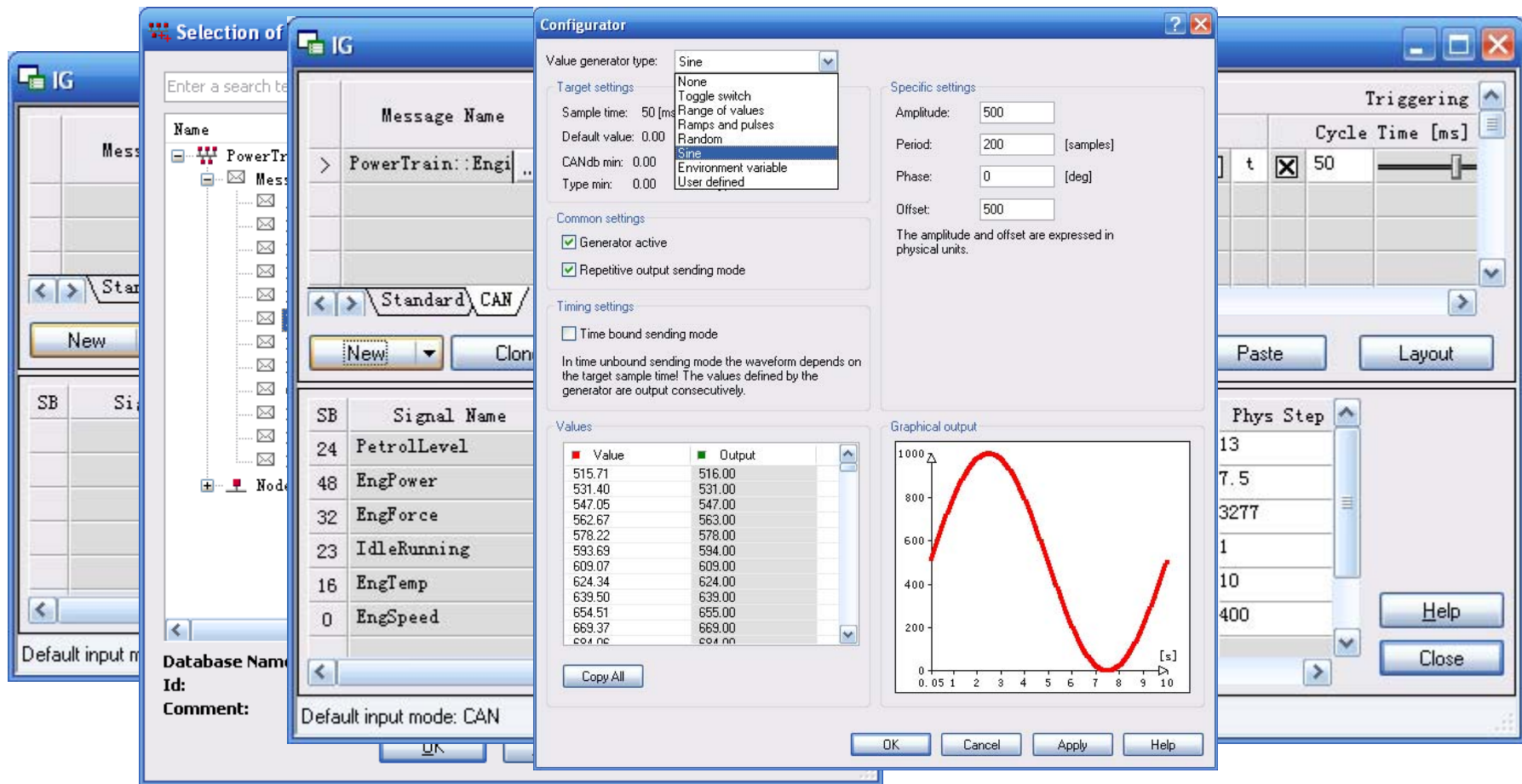
The screenshot displays the Vector CANdb software interface, specifically the 'Generator sendlist' dialog box. The dialog is divided into several sections:

- Generator sendlist (Left):** A table with 8 rows, each representing a CAN message. The first row is selected, showing 'ABSdata' in the ID field.
- Symbolic Selection (Middle):** A list of symbolic names for CAN data, including 'CANdb: PowerTrain::A', 'AccelerationForce', 'CarSpeed', 'Diagnostics', and 'GearLock'. 'CarSpeed' is currently selected.
- Signal function (Right):** A section for defining the signal function. It shows 'ABSdata' as the selected signal. Below it, a 'Preview' window displays a graph of the signal function, showing a ramp from 'n1' to 'n2' over time, with markers 'ta' and 'th'.
- Generator sendlist (Right):** A table showing the generated CAN messages. The first row is selected, showing 'ABSdata' in the ID field. The table includes columns for ID, DLC, and DATA.

The 'Generator sendlist' table (Right) contains the following data:

ID	DLC	DATA	CAN
1	6	0 0 0 0 0 0	1
2	6	144 1 0 0 0 0	1
3	6	32 3 0 0 0 0	1
4	6	32 3 0 0 0 0	1
5	6	144 1 0 0 0 0	1
6	6	0 0 0 0 0 0	1
7	6	0 0 0 0 0 0	1
8	6	0 0 0 0 0 0	1

# 交互式发生器模块



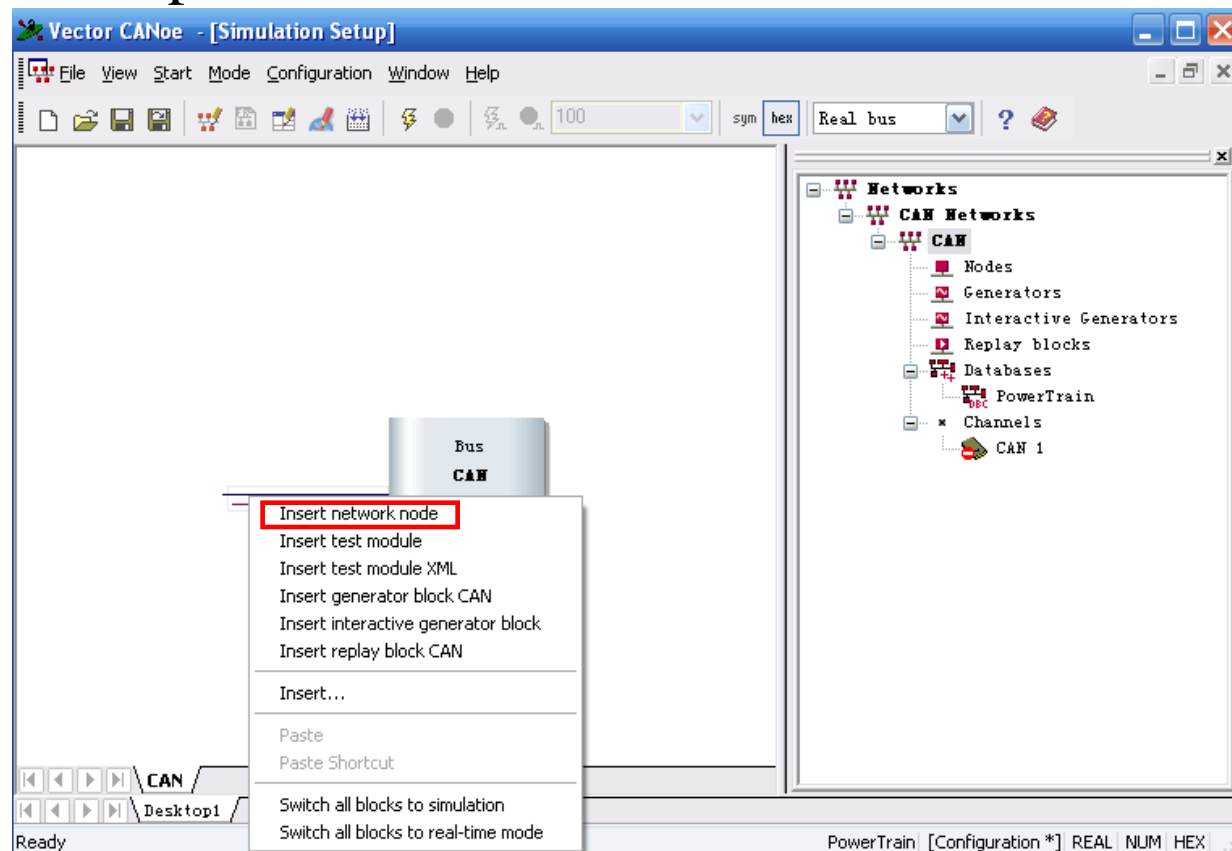


# Complex simulation

## □ Simulation Setup

### □ View->Simulation Setup

#### □ CAPL 节点



## □ CAPL (CAN Access Programming Language)

### □ 类C语言

### □ 仿真

#### □ 单个节点和整个网络

#### □ 外部环境

#### □ 测试

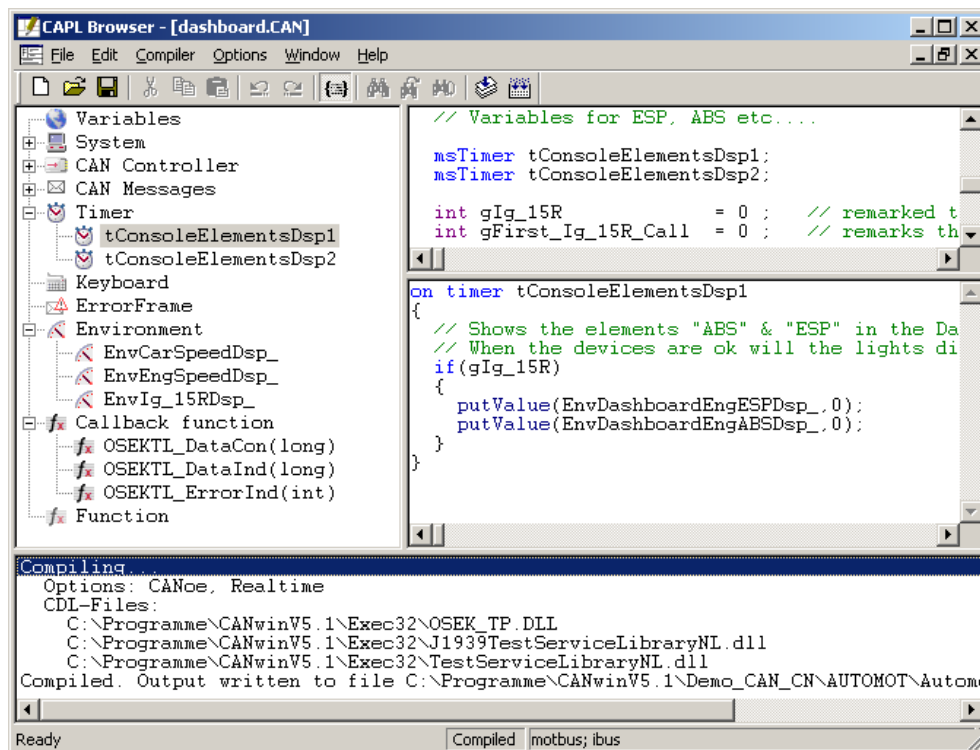
### □ 面向事件的编程语言

#### □ 总线事件

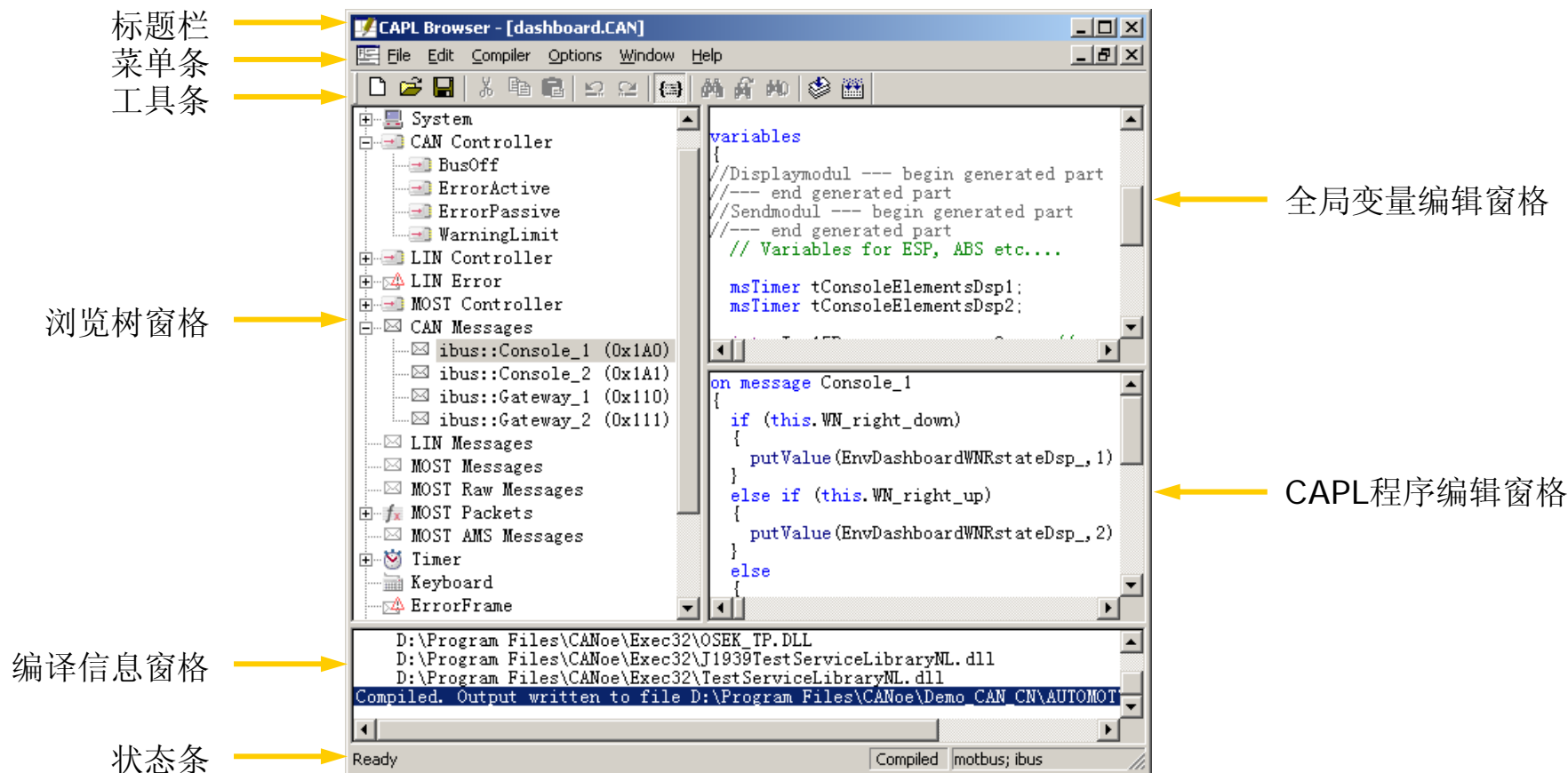
#### □ 键盘事件

#### □ 时间事件

#### □ 环境变量事件



# CAPL Browser



事件类型	事件名	程序执行条件	事件过程语法结构 *
系统事件	<i>PreStart</i>	CANoe初始化时执行	<i>on preStart { ... }</i>
	<i>Start</i>	测量开始时执行	<i>on start { ... }</i>
	<i>StopMeasuremet</i>	测量结束时执行	<i>on stopMeasurement { ... }</i>
CAN控制器事件	<i>BusOff</i>	硬件检测到BusOff时执行	<i>on busOff { ... }</i>
	<i>ErrorActive</i>	硬件检测到ErrorActive时执行	<i>on errorActive { ... }</i>
	<i>ErrorPassive</i>	硬件检测到ErrorPassive时执行	<i>on errorPassive { ... }</i>
	<i>WarningLimit</i>	硬件检测到WarningLimit时执行	<i>on warningLimit { ... }</i>
CAN消息事件	自定义	接收到指定的消息时执行	<i>on message Message { ... }</i>
时间事件	自定义	定时时间朝过时执行	<i>on timer Timer { ... }</i>
键盘事件	自定义键值	指定的键被下时执行	<i>on key Key { ... }</i>
错误帧事件	<i>ErrorFrame</i>	硬件每次检测到错误帧时执行	<i>on errorFrame { ... }</i>
环境变量事件	自定义	指定的环境变量值改变时执行	<i>on envVar EnvVar { ... }</i>

## □ 类C语言，语法与C语言基本相同

### □ 注释

□ // 放置在需要注释的语句之前，注释单行

□ /\* 注释起始符，其后的内容被注释

□ \*/ 注释结束符，结束由‘/\*’开始的注释

□ 分号 程序结束标识

□ 大括号 函数体

```
counter = counter+1;  
if (counter==256)  
{  
    counter=0;  
    stop();  
}
```

# 数据类型

数据类型	名称	注释
无符号整型	byte	1个字节
	word	2个字节
	dword	4个字节
有符号整型	int	2个字节
	long	4个字节
浮点型	float	8个字节
	double	8个字节
CAN报文	message	
定时器	timer	秒
	msTimer	毫秒
单个字符	char	1个字节

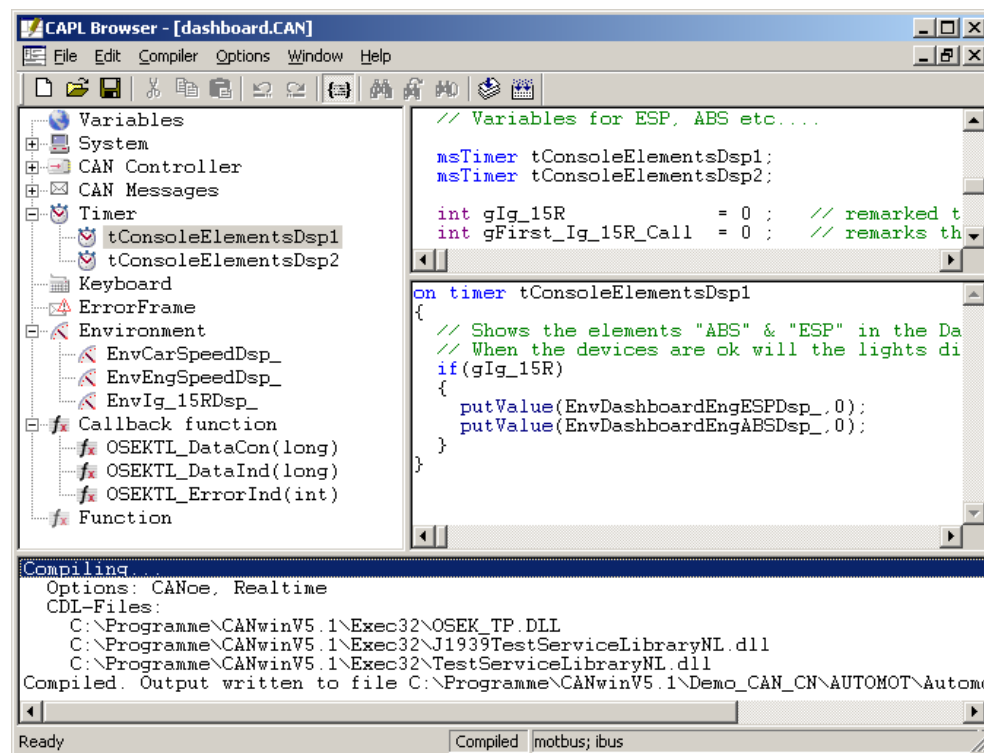
## ▣ 全局变量和局部变量

## ▣ 变量定义

`int i;`

`message 0x123 HiRain;`

`message MotorData Vector;`



# 完整的CAPL程序

## □ 三个部分

□ 变量

□ 各种事件

□ 自定义函数

```
variables  
{  
    ...  
}
```

//申明全局变量

```
on start  
{  
    ...  
}  
  
on message xxx  
{  
    ...  
}  
  
on key '1'  
{  
    ...  
}
```

//过程指令块

```
My_function_1(Para_1, Para_2, ...)  
{  
    ...  
}  
  
...  
  
My_function_n(Para_1, Para_2, ...)  
{  
    ...  
}
```

//函数体



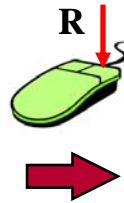
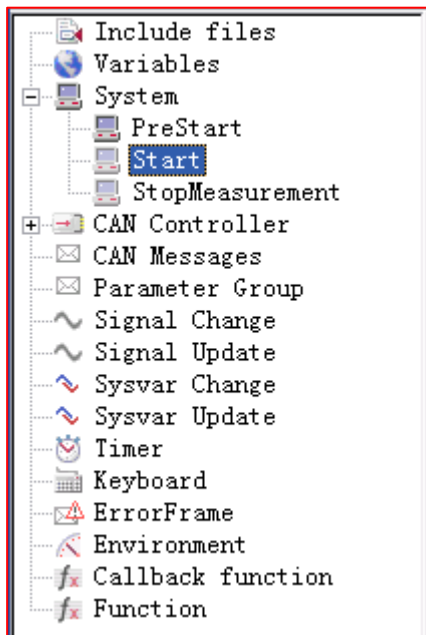
## □ Write Window

### □ write函数

```
int h=100;  
char ch='a';  
char s100[8]="hundred";  
write("Hundred as a number:%d,%x",h,h);  
write("Hundred as a string:%s",s100);  
write("The square root of two is %6.4g",sqrt(2.0));
```

# CAPL练习1

- 当CANoe启动时，向Write Window输出一句话，例如“hello world!”



```
on start
{
    write("hello world!");
}
```

- ❑ on message 123 //对消息123(dec)反应
- ❑ on message 0x123 //对消息123(hex)反应
- ❑ on message MotorData //对消息MotorData(符号名字)反应
- ❑ on message CAN1.123 //对CAN 通道1收到消息123反应
- ❑ on message \* //对所有消息反应
- ❑ on message 100-200 //对100-200间消息反应

## □ this 代表触发事件的对象

```
on message 100 {  
    byte byte_0;  
    byte_0 = this.byte(0);  
    ...  
}
```

```
on envVar Switch {  
    int val;  
    val = getvalue(this);  
    ...  
}
```

```
on message 0x64
```

```
{  
    if(this.byte(2)==0xFF)  
        write("Third byte of the message is invalid");  
}
```

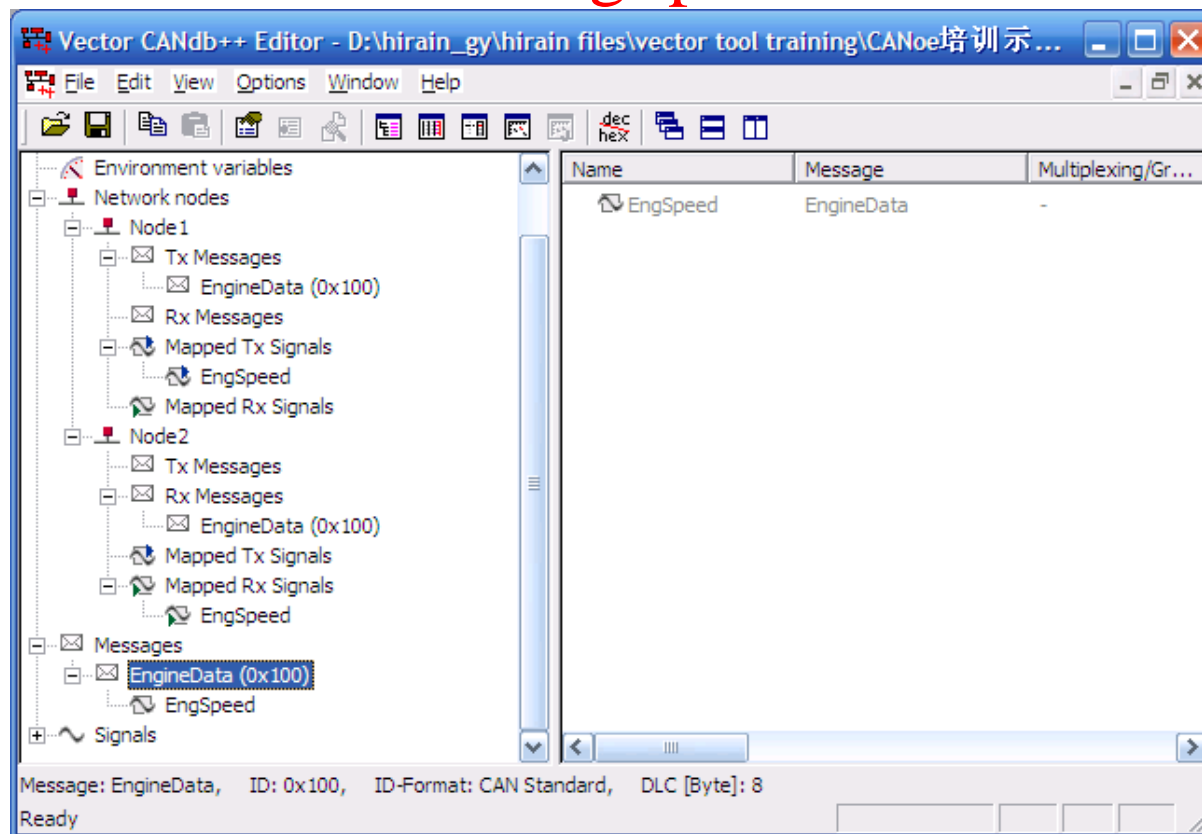
```
on message MotorData
```

```
{  
    if(this.temperature.phys>=150)  
        write("Warning: critical temperature");  
}
```

- `if (this.id==100) { ... }`
- `msg.can=2;`
- `msg.dlc=8;`
- `DWORD t ; t=this.time;`
- `if(this.dir!=RX) {return;}`
- `this.CarSpeed = 200;`

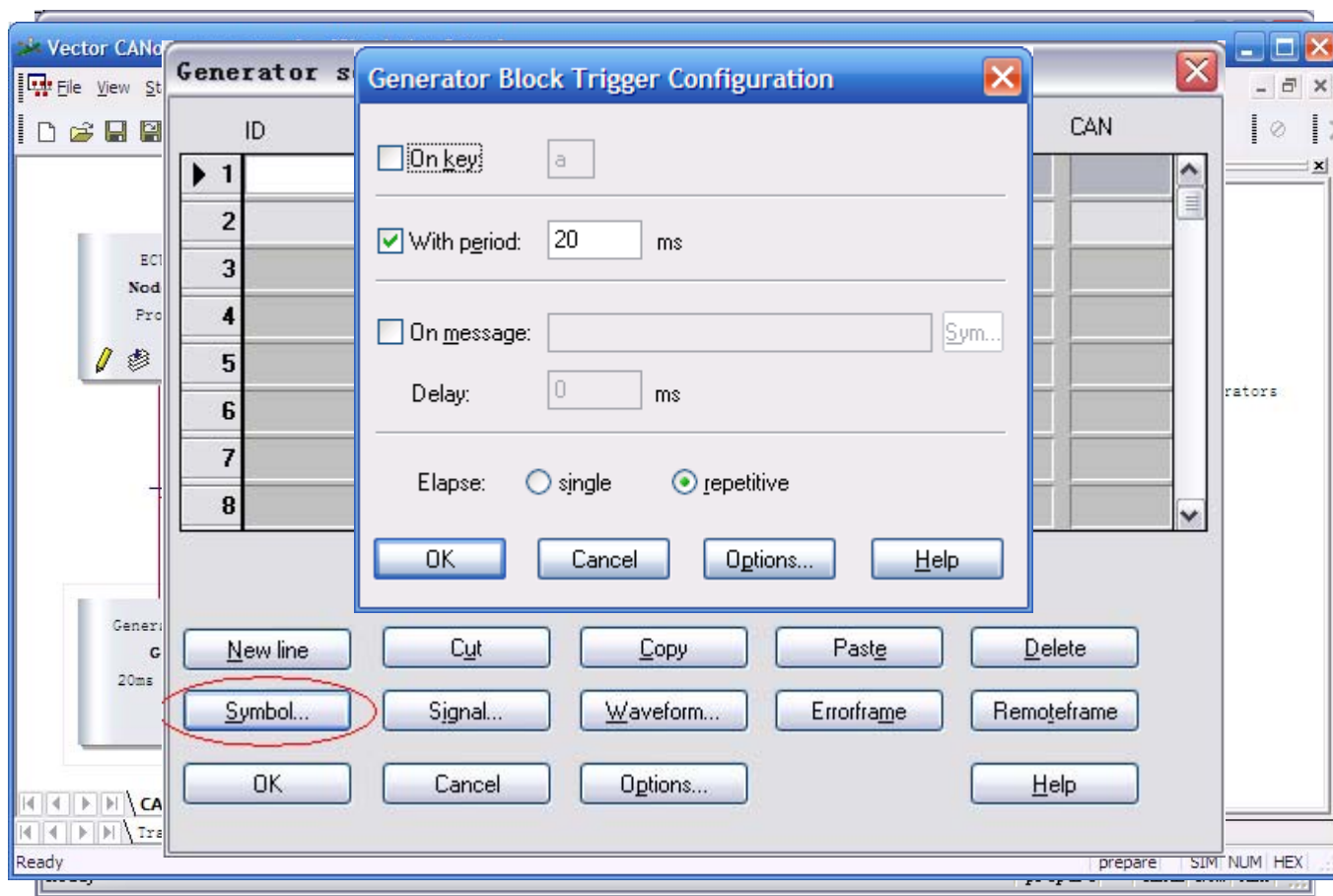
# CAPL练习2 -step1

- 建立一个简单的数据库文件，包括节点**Node1**、**Node2**，添加**Node1**的发送报文**EngineData**（假设ID为0x100），并与16位的信号**EngSpeed**相关联。



# CAPL练习2 –step2

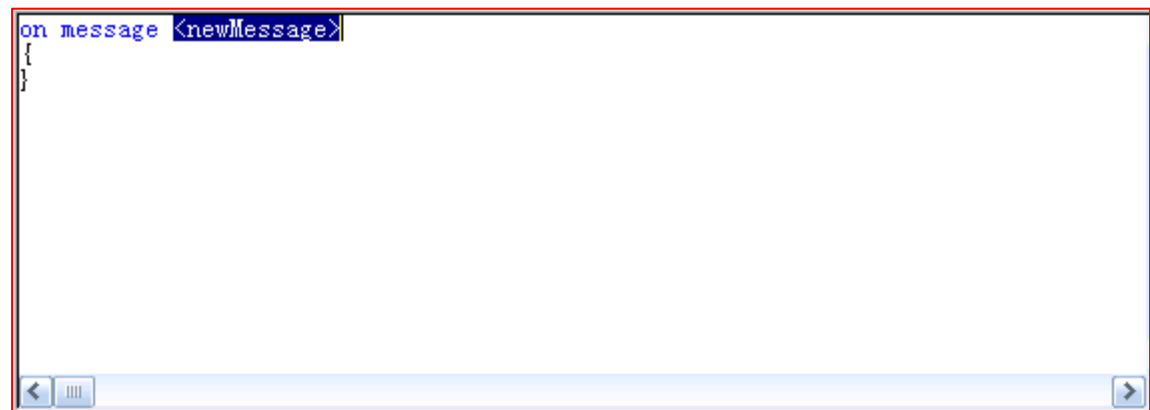
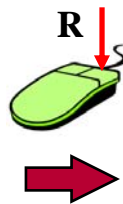
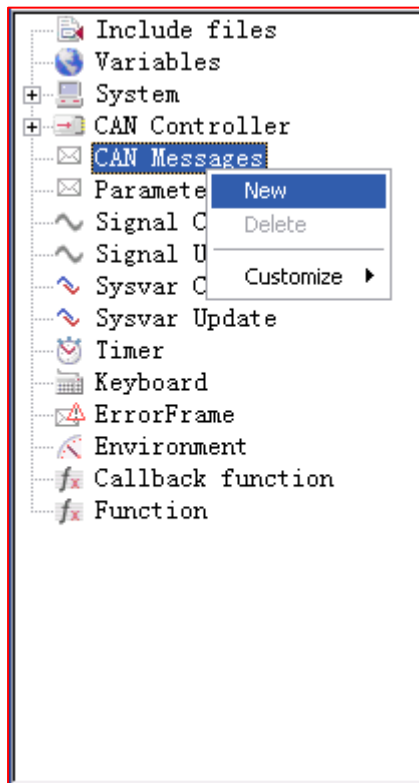
- 利用发生器模块周期性发送EngineData报文，例如每隔20ms发送一条EngineData报文。





# CAPL练习2 –step3

- 当发送5个Enginedata 报文后，在Write Window窗口输出一句话，例如“ The node have sent five EngineData messages.”（提示：定义一整型变量用于计数。）



# 键盘事件处理

❑ on key 'a'

//按'a'键反应

❑ on key ' '

//按空格键反应

❑ on key 0x20

//按空格键反应

❑ on key F1

//按F1键反应

❑ on key Ctrl-F12

//按Ctrl + F12键反应

❑ on key PageUP

//按PageUp键反应

❑ on key Home

//按Home键反应

❑ on key \*

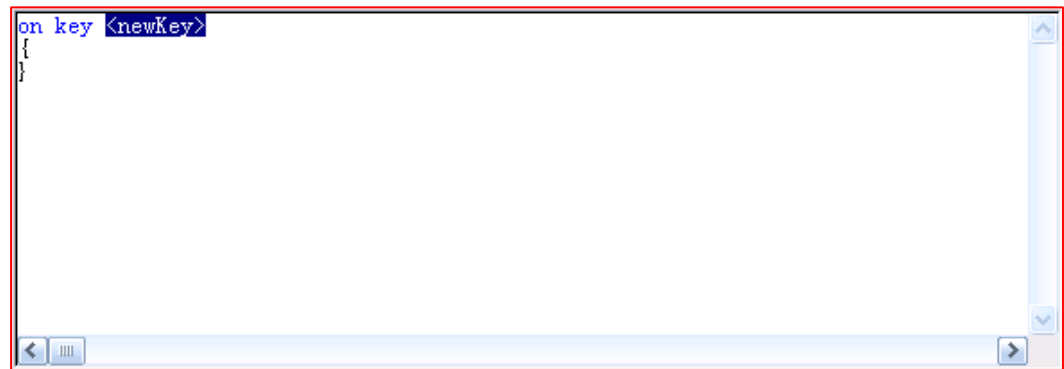
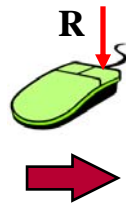
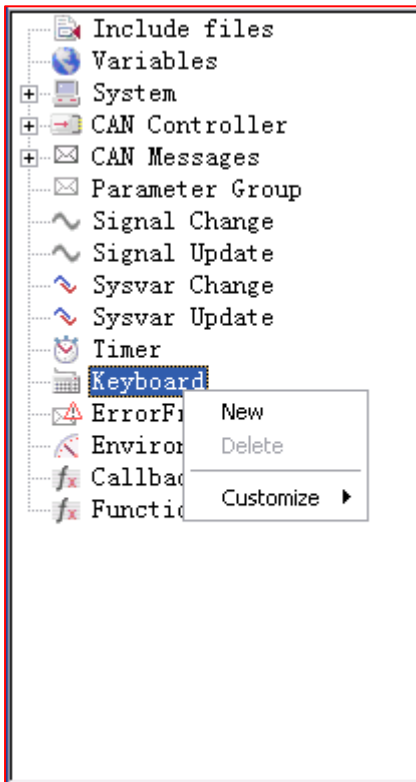
//按所有键反应

# 键盘事件处理

```
on key 'a' {  
    message MotorData mMoDa;  
    mMoDa.temperature.phys=60;  
    mMoDa.speed.phys=4300;  
    output(mMoDa);  
}  
on key 'b' {  
    message 100 m100= {dlc=1};  
    m100.byte(0)=0x0B;  
    output(m100);  
}
```

# CAPL练习3

- 每当按下 **s** 键，在 Write Window 窗口输出一句话，例如“XXX EngineData messages have sent.”
- 提示：XXX为已经发送的EngineData报文数量。



## □ 定时器声明

- `msTimer myTimer;` //将myTimer 申明ms为单位的变量

- `timer myTimer;` //将myTimer 申明s为单位的变量

## □ 定时器函数

- `setTimer(myTimer,20);` //将定时值设定为20ms，并启动

- `cancelTimer(myTimer);` //停止定时器myTimer

## □ 定时器事件

- `on timer myTimer` //对myTimer 设定的时间到反应

# 时间事件处理

## Variables

```
{  
    message 0x555 msg1 = {dlc=1};  
    msTimer timer1;  
}
```

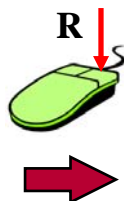
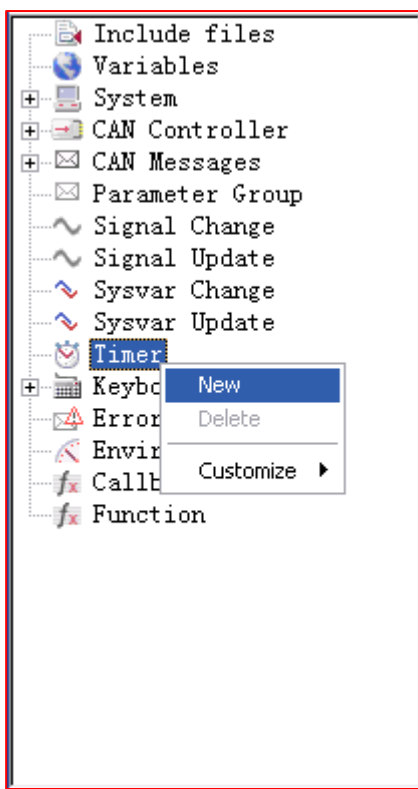
## on start

```
{  
    setTimer(timer1,100);  
}
```

## on timer timer1

```
{  
    setTimer(timer1,100);  
    msg1.byte(0)=msg1.byte(0)+1;  
    output(msg1);  
}
```

- 不用发生器模块实现Enginedata报文的周期性发送。  
(提示：先禁掉发生器模块。)



```
on timer t1
{
    output(msg);
    setTimer(t1,20);
}
```

## □ 环境变量函数

□ `getValue()`           //获取环境变量的值

□ `putValue()`           //设置环境变量的值

## □ 环境变量事件

□ `on envVar XXX`



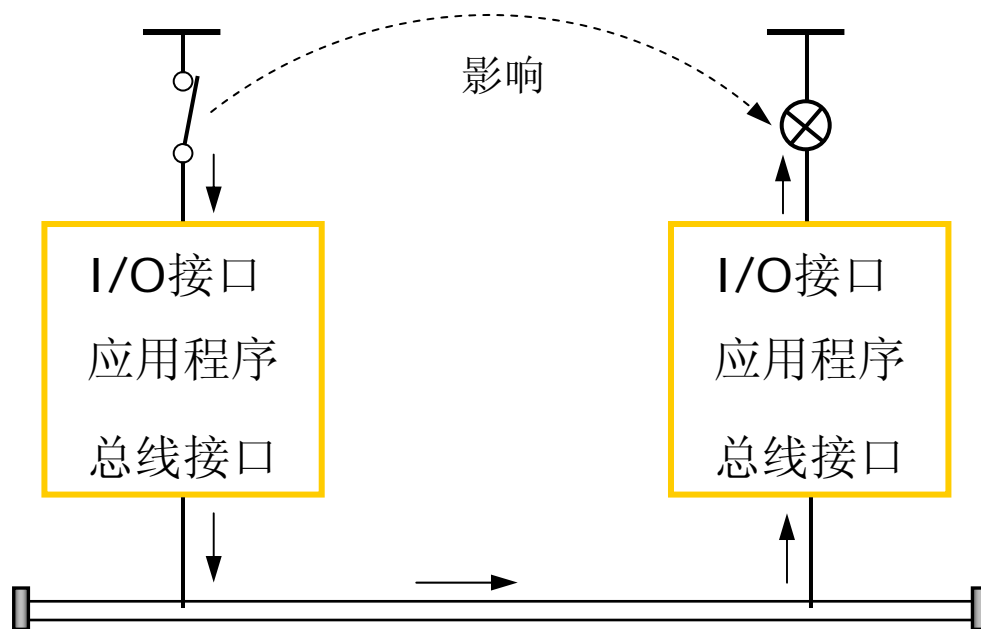
```
on envVar evSwitch
{
    message MotorData msg;
    msg.bsSwitch = getValue(this);
    output(msg);
}
```

# Excise2

## 1) 建立数据库文件

How many nodes?      How many messages?      How many signals?

## 2) 新建CANoe配置工程，并加入数据库文件



## 3) 通过CAPL语言实现仿真节点（报文的发送和接收）

需要人机交互界面?

需要模拟仪表盘?... .

## □ Panel Editor

- 传统的面板编辑器

- File->Open Panel Editor

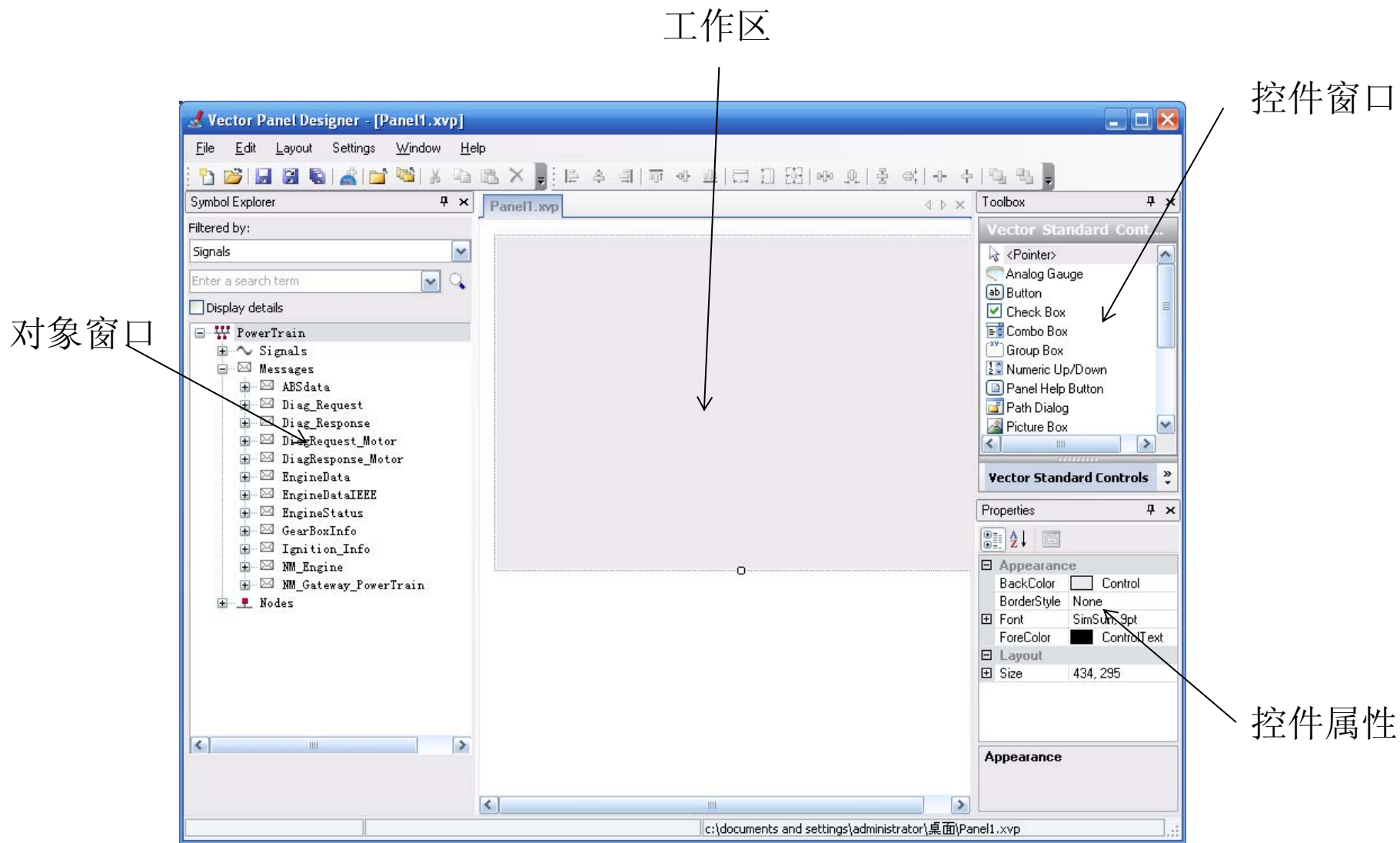
## □ Panel Designer



- 新的面板编辑器

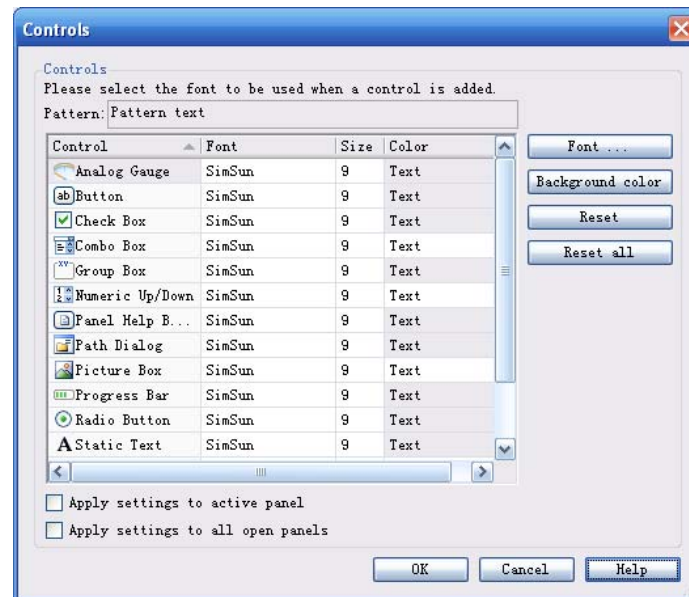
- File->Open Panel Designer

# Panel Designer



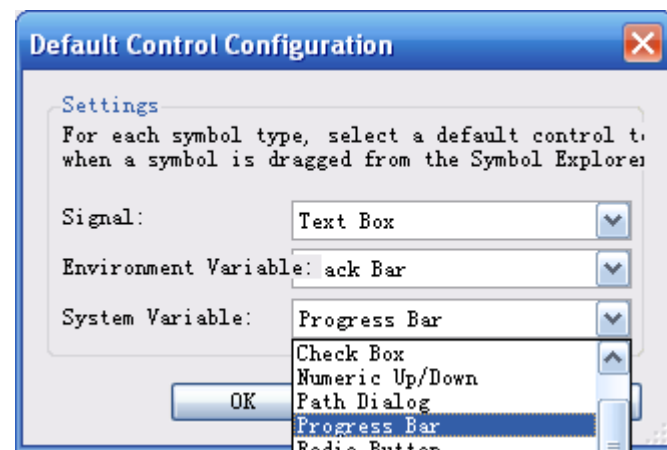
## □ Settings->Controls Properties

□ 设置控件的字体、颜色和字号

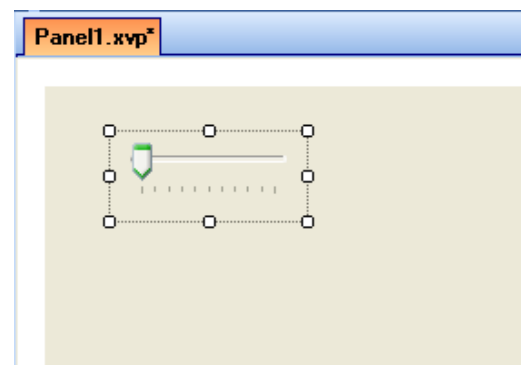
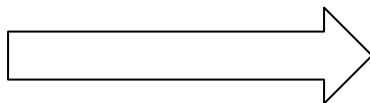
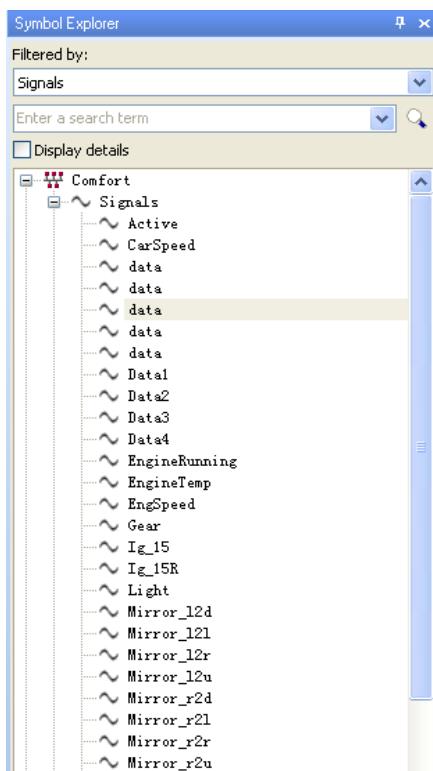


## □ Settings->Symbol Explorer

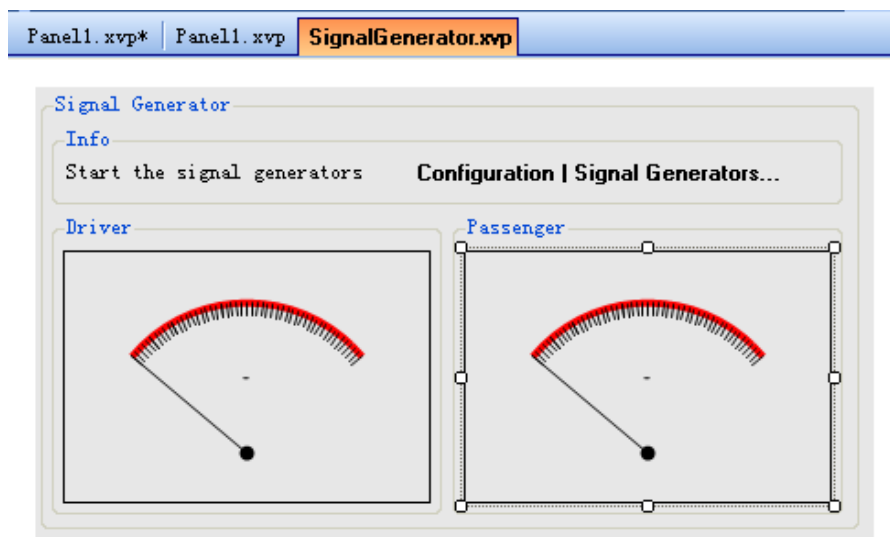
□ 设置信号、环境变量和系统变量  
对应的默认控件



- ▣ 显示信号、环境变量和系统变量
- ▣ 直接拖拽变量到工作区生成控件



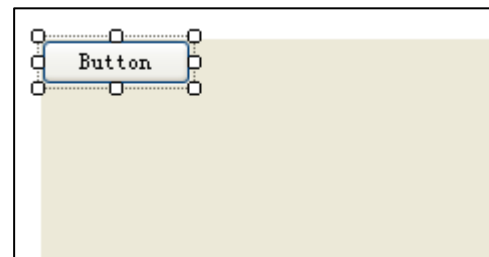
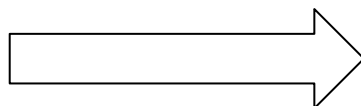
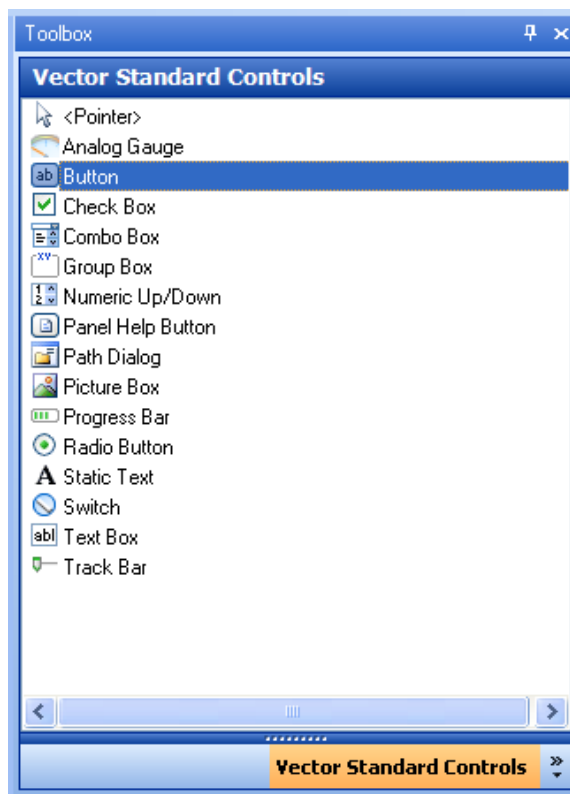
- 创建面板
- 支持同时编辑多个面板



# 控件窗口

## □ 显示控件

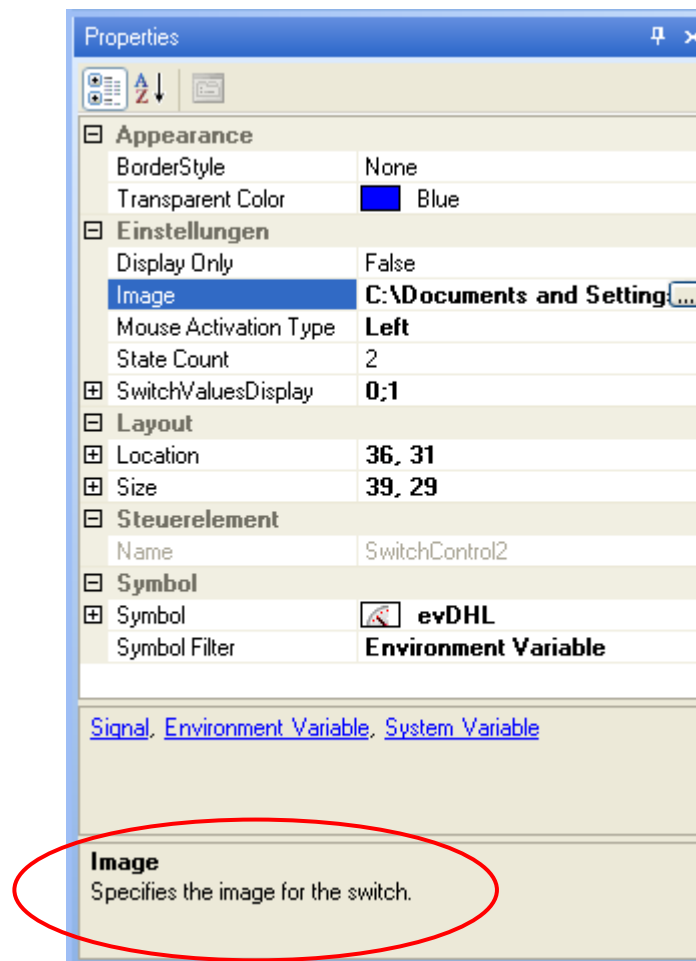
## □ 双击在工作区产生控件





# 控件属性窗口

- 显示选中控件的相关设置
- 点击某项设置后会在下方出现相关说明

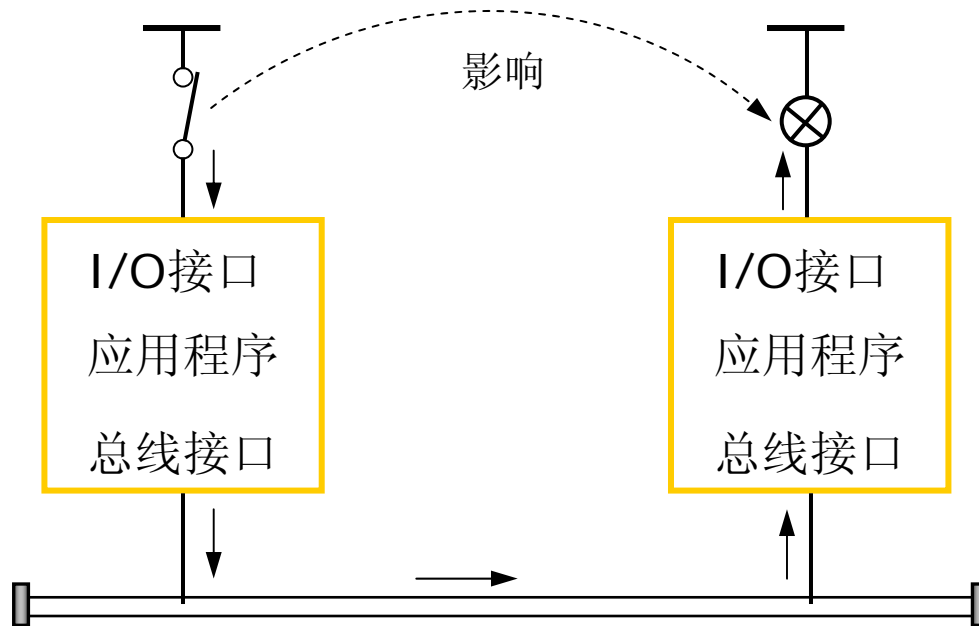


# Excise3

## 1) 建立数据库文件

How many nodes?      How many messages?      How many signals?

## 2) 新建CANoe配置工程，并加入数据库文件



## 3) 通过CAPL语言实现仿真节点（报文的发送和接收）

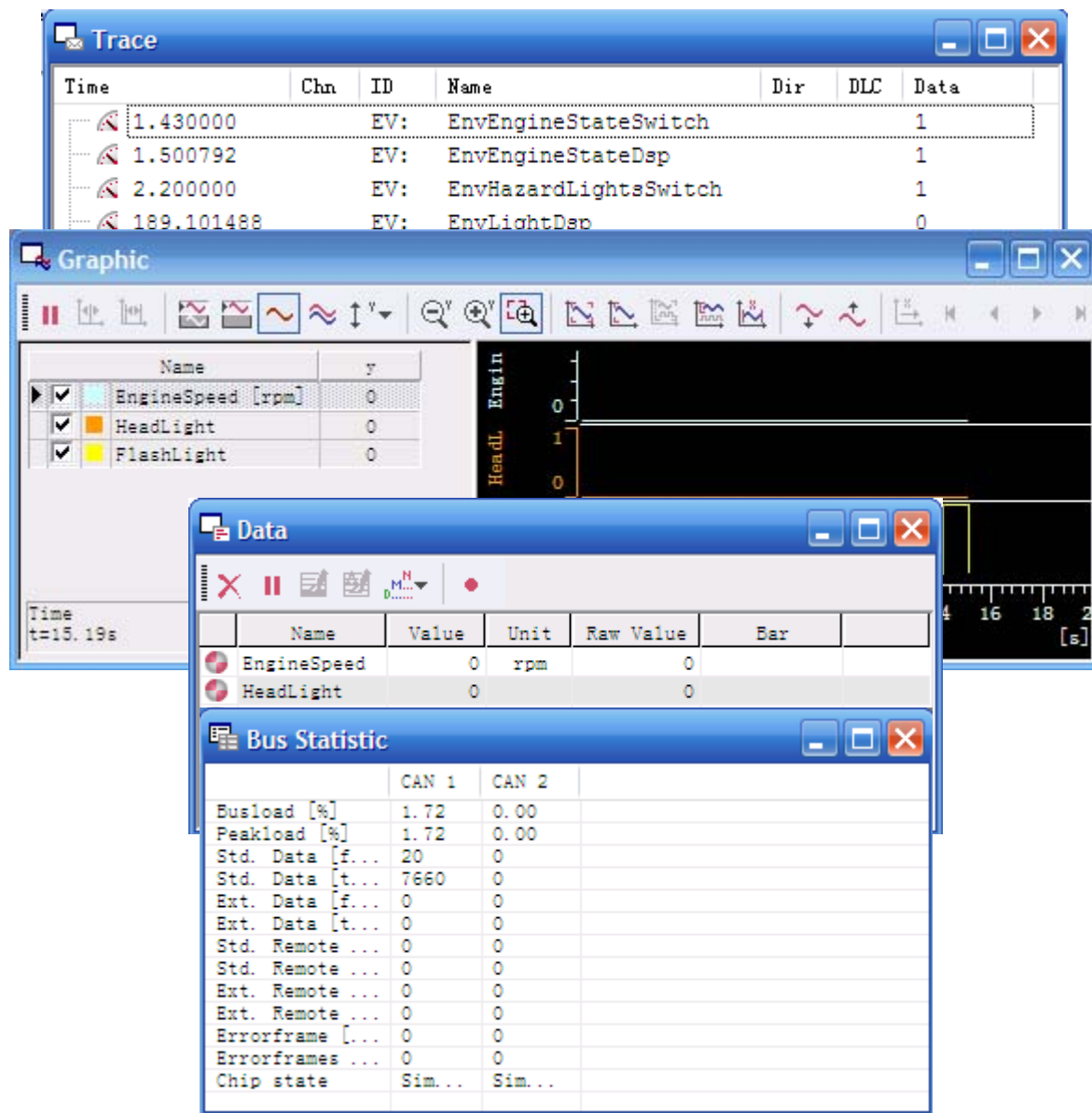
## 4) 建立面板，并与相关信号或环境变量关联

# CAN总线开发Step Three: 测试分析

网络仿真阶段

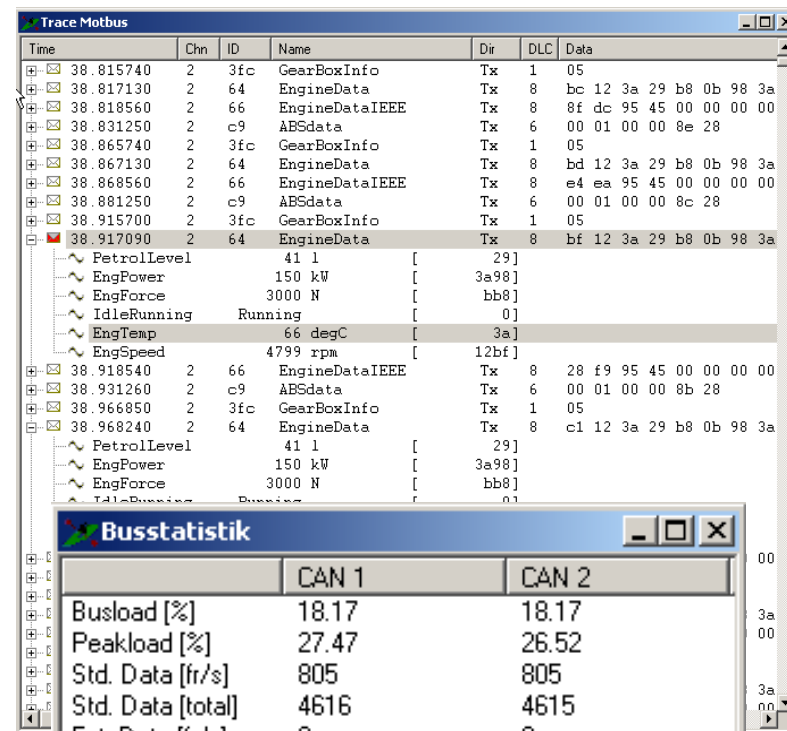
节点开发阶段

系统集成阶段



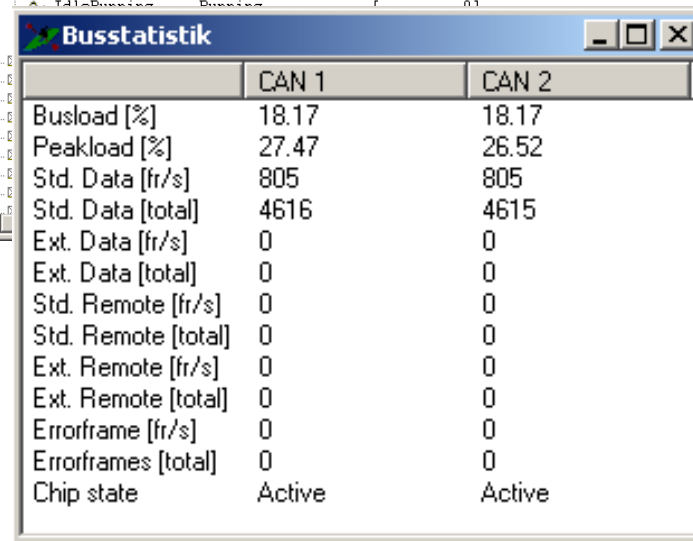
# 主要测试分析窗口 (1)

## Trace Window



Time	Chn	ID	Name	Dir	DLC	Data
38.815740	2	3fc	GearBoxInfo	Tx	1	05
38.817130	2	64	EngineData	Tx	8	bc 12 3a 29 b8 0b 98 3a
38.818560	2	66	EngineDataIEEE	Tx	8	8f dc 95 45 00 00 00 00
38.831250	2	c9	ABSdata	Tx	6	00 01 00 00 8e 28
38.865740	2	3fc	GearBoxInfo	Tx	1	05
38.867130	2	64	EngineData	Tx	8	bd 12 3a 29 b8 0b 98 3a
38.868560	2	66	EngineDataIEEE	Tx	8	e4 ea 95 45 00 00 00 00
38.881250	2	c9	ABSdata	Tx	6	00 01 00 00 8c 28
38.915700	2	3fc	GearBoxInfo	Tx	1	05
38.917090	2	64	EngineData	Tx	8	bf 12 3a 29 b8 0b 98 3a
~ PetrolLevel 41 l [ 29]						
~ EngPower 150 kW [ 3a98]						
~ EngForce 3000 N [ bb8]						
~ IdleRunning Running [ 0]						
~ EngTemp 66 degC [ 3a]						
~ EngSpeed 4799 rpm [ 12bf]						
38.918540	2	66	EngineDataIEEE	Tx	8	28 f9 95 45 00 00 00 00
38.931260	2	c9	ABSdata	Tx	6	00 01 00 00 8b 28
38.966850	2	3fc	GearBoxInfo	Tx	1	05
38.968240	2	64	EngineData	Tx	8	c1 12 3a 29 b8 0b 98 3a
~ PetrolLevel 41 l [ 29]						
~ EngPower 150 kW [ 3a98]						
~ EngForce 3000 N [ bb8]						
~ IdleRunning Running [ 0]						

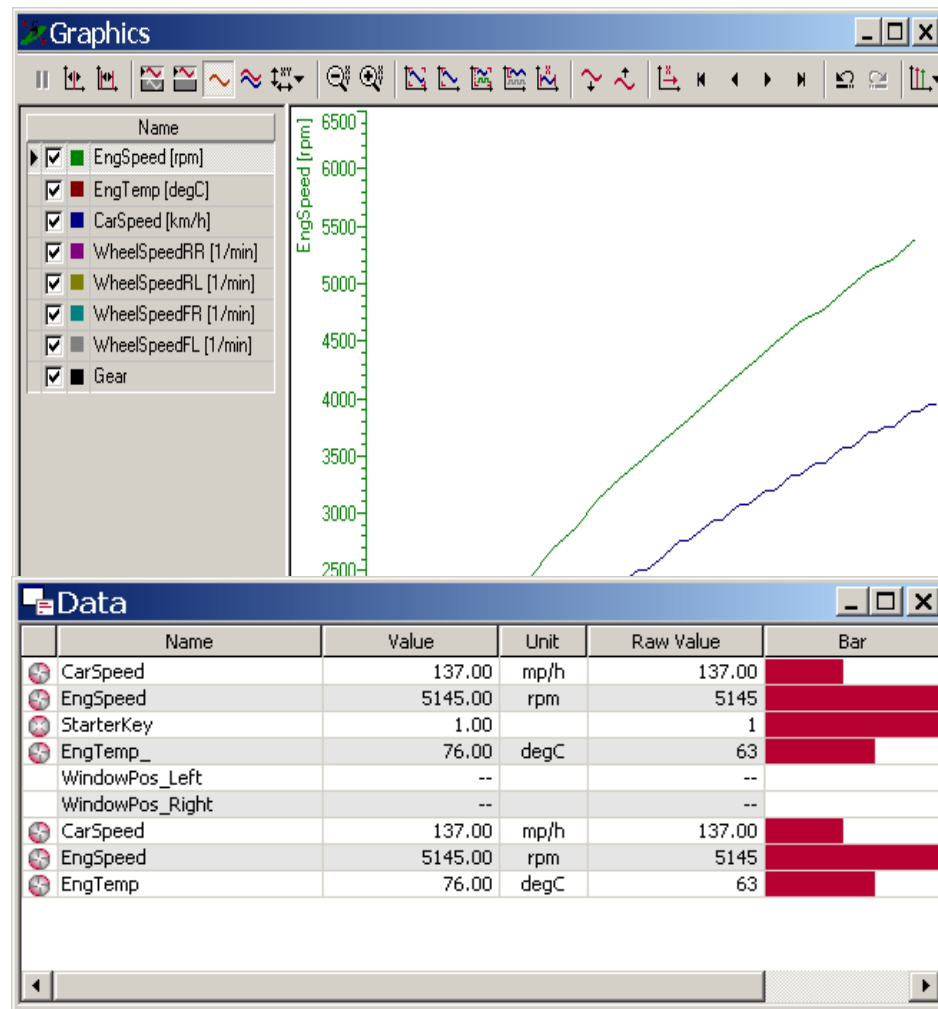
## Bus Statistics Window



	CAN 1	CAN 2
Busload [%]	18.17	18.17
Peakload [%]	27.47	26.52
Std. Data [fr/s]	805	805
Std. Data [total]	4616	4615
Ext. Data [fr/s]	0	0
Ext. Data [total]	0	0
Std. Remote [fr/s]	0	0
Std. Remote [total]	0	0
Ext. Remote [fr/s]	0	0
Ext. Remote [total]	0	0
Errorframe [fr/s]	0	0
Errorframes [total]	0	0
Chip state	Active	Active

# 主要测试分析窗口（2）

## □ Graphics Windows



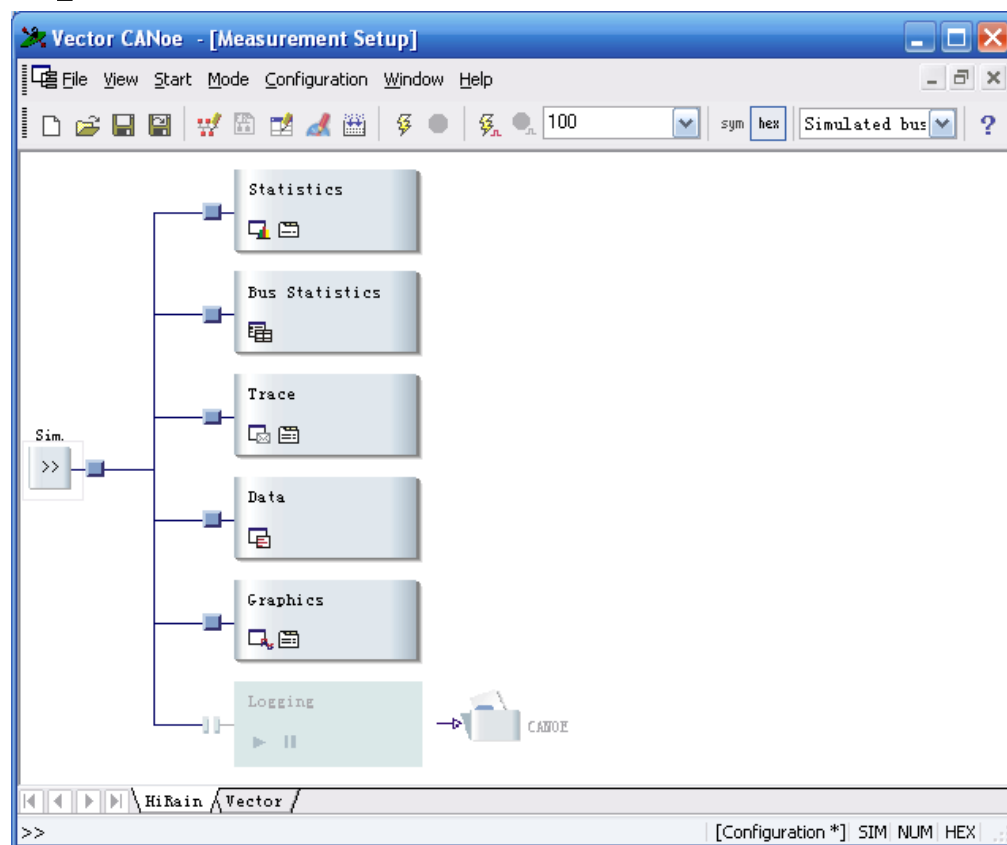
## □ Data Window

# 主要测试分析窗口（3）

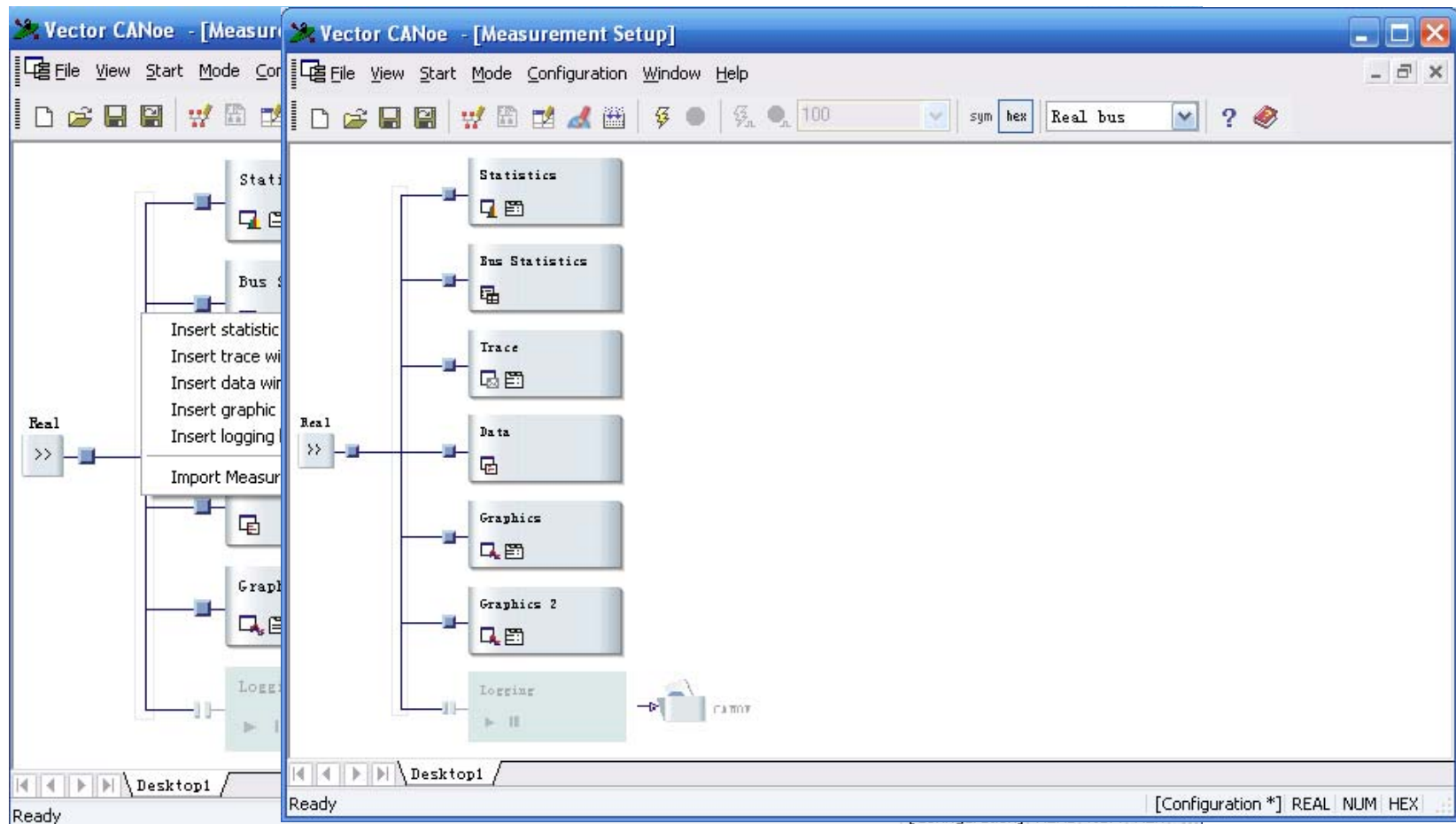
## □ Measurement Setup

### □ View->Measurement Setup

- 每个模块对应一个窗口
- 增加新模块（窗口）
- 插入功能块
- 数据记录



# 新增模块（窗口）



# 插入功能块 (1/2)

## □ 插入功能块

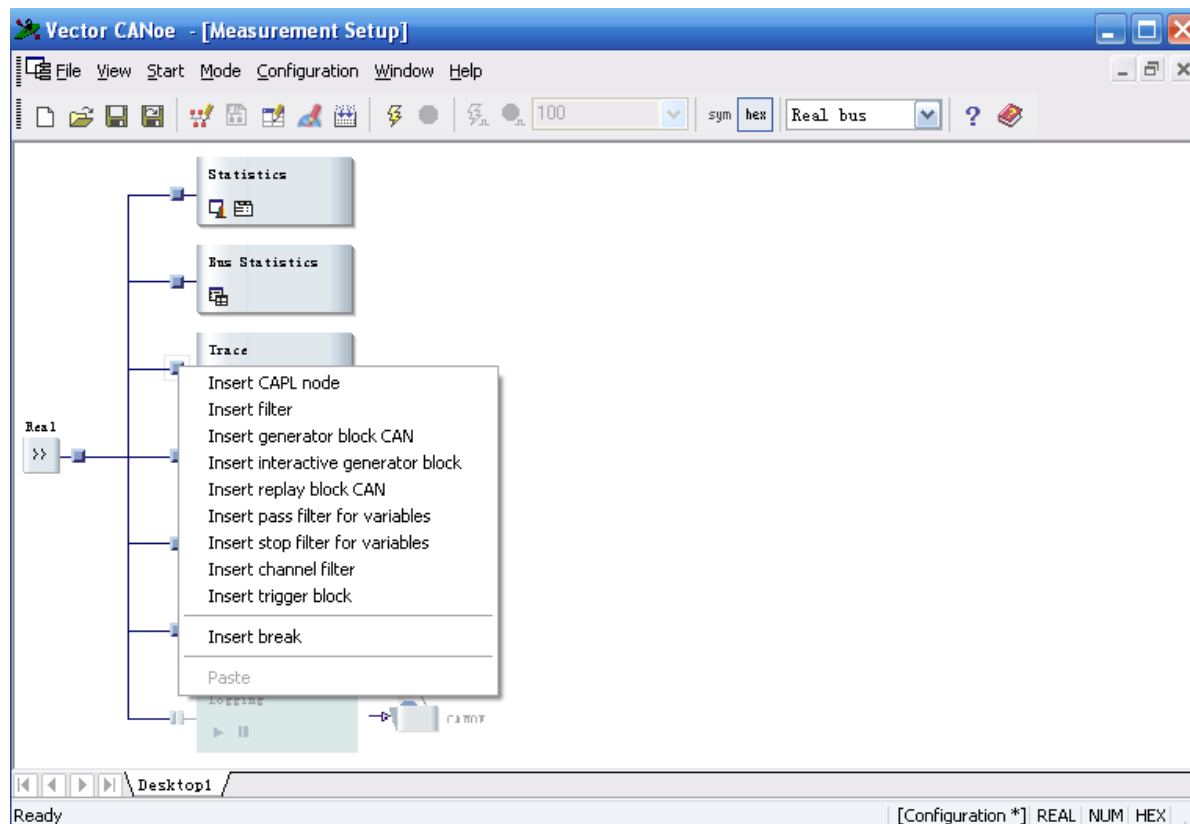
### □ CAPL节点

### □ 发生器模块

### □ 回放模块

### □ 触发模块

### □ 过滤器模块





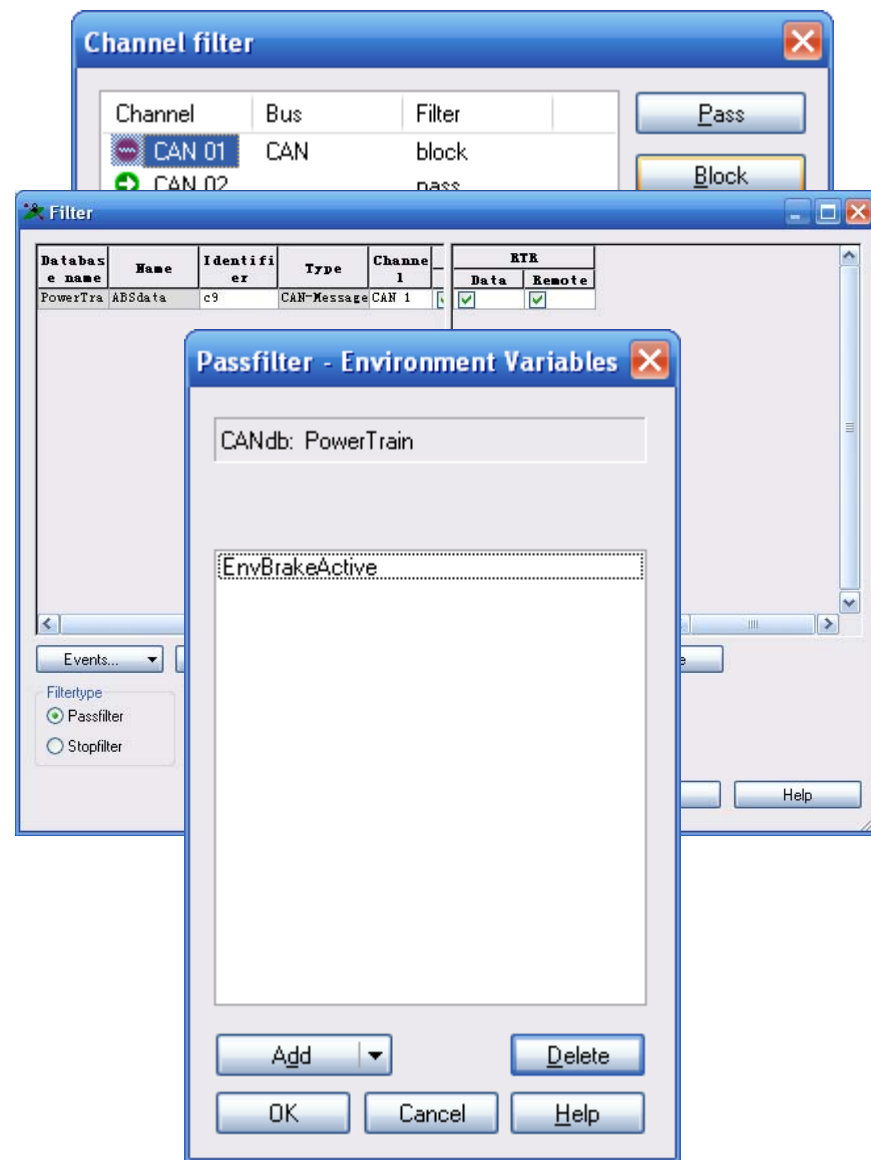
# 插入功能块（2/2）

## □ 过滤器模块

□ 通道过滤（Channel Filter）

□ 报文过滤（Filter）

□ 变量过滤（Variables）



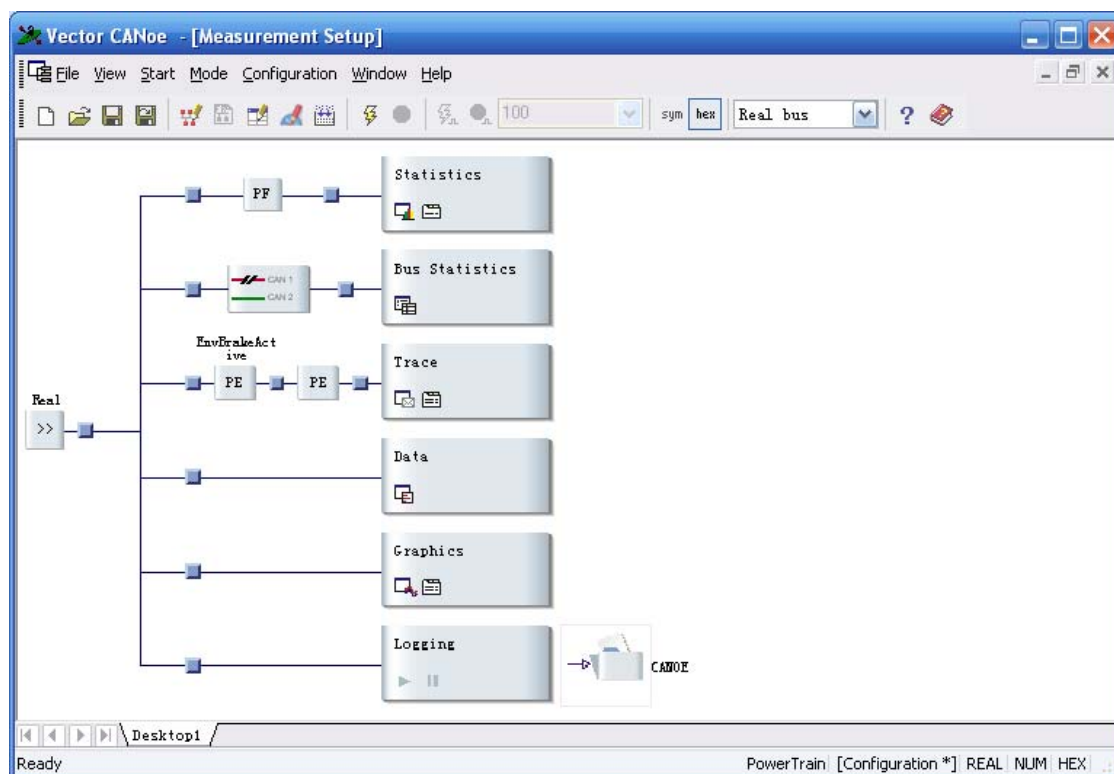
# 数据记录（1/4）

## □ 数据记录

- 默认状态关闭

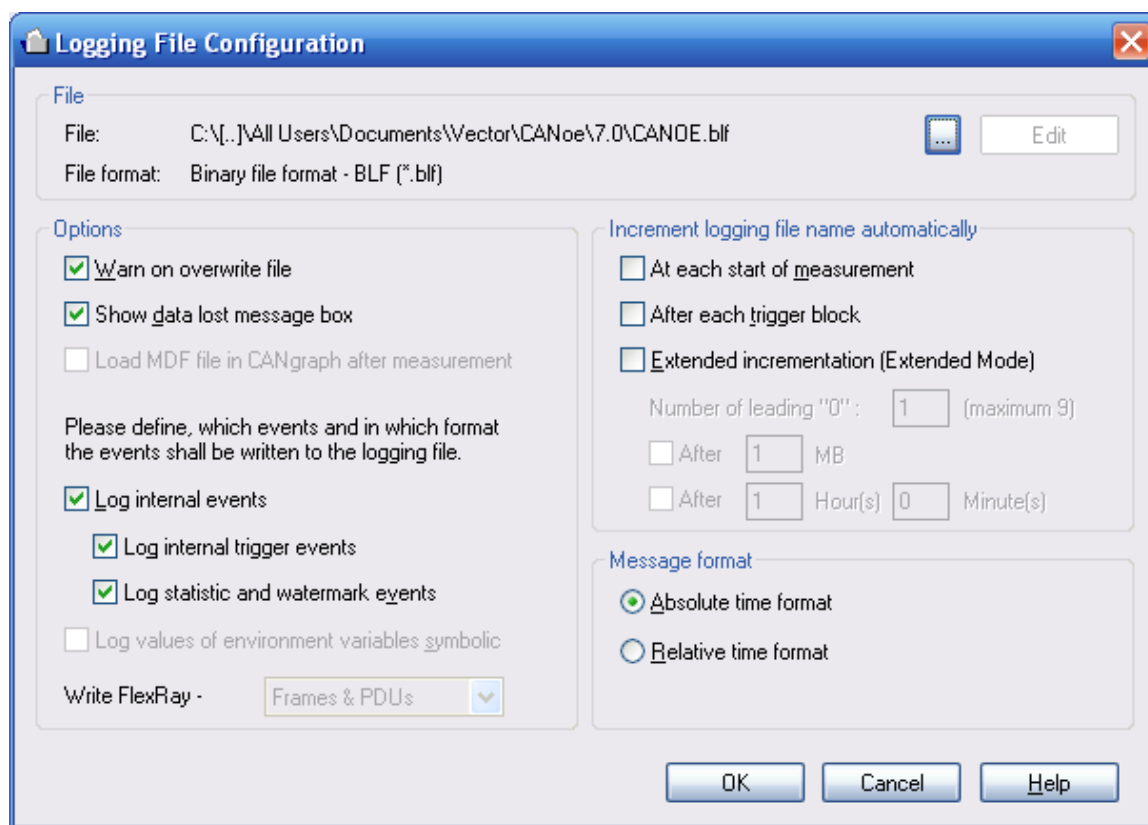
- 多种记录文件类型

- 多种记录配置方式



## □ 记录文件

### □ 右键点击文件图标->Logging file configuration



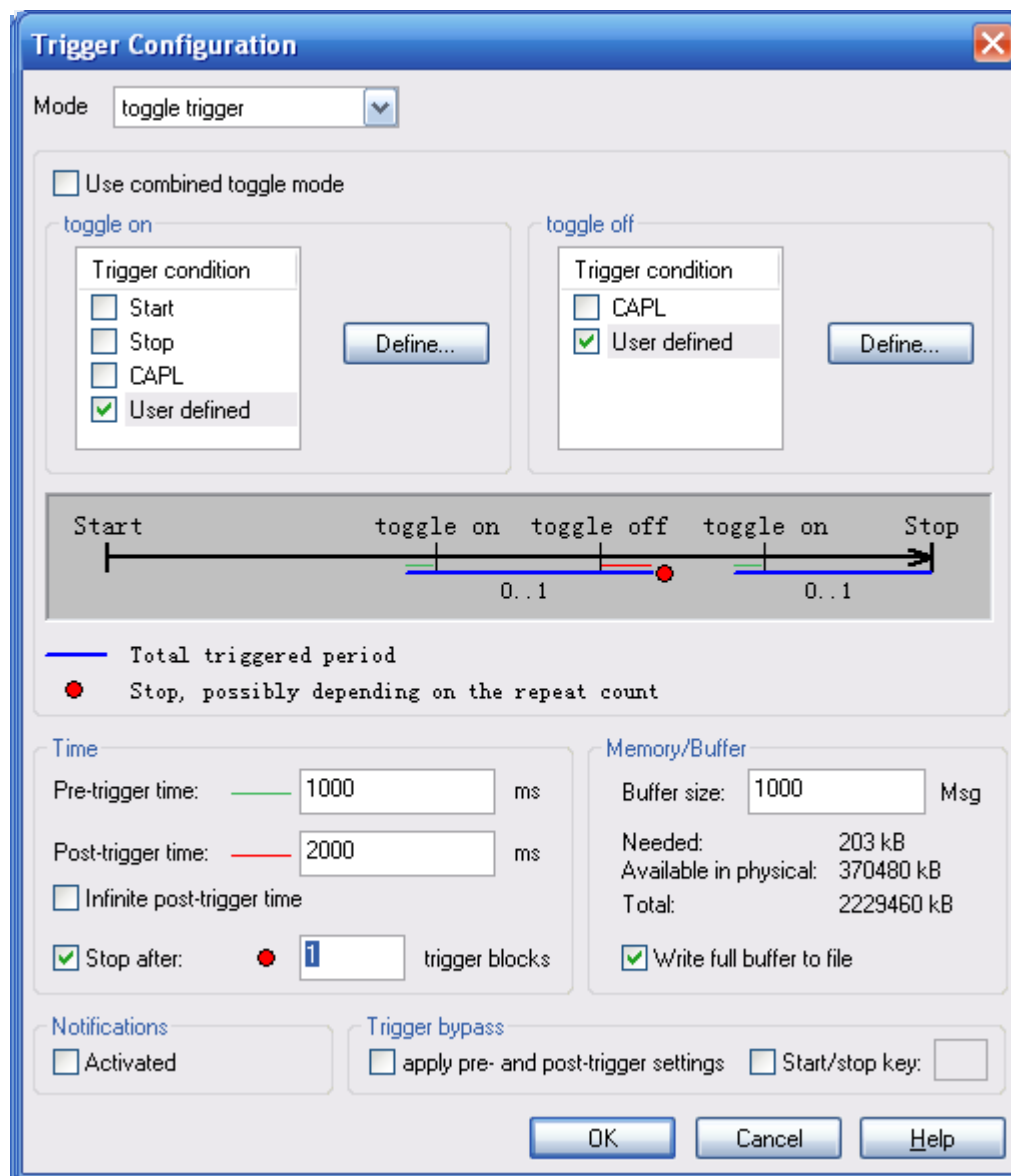
## □ 记录配置方式

### □ 双击Logging模块

□ 全部记录

□ 单次记录

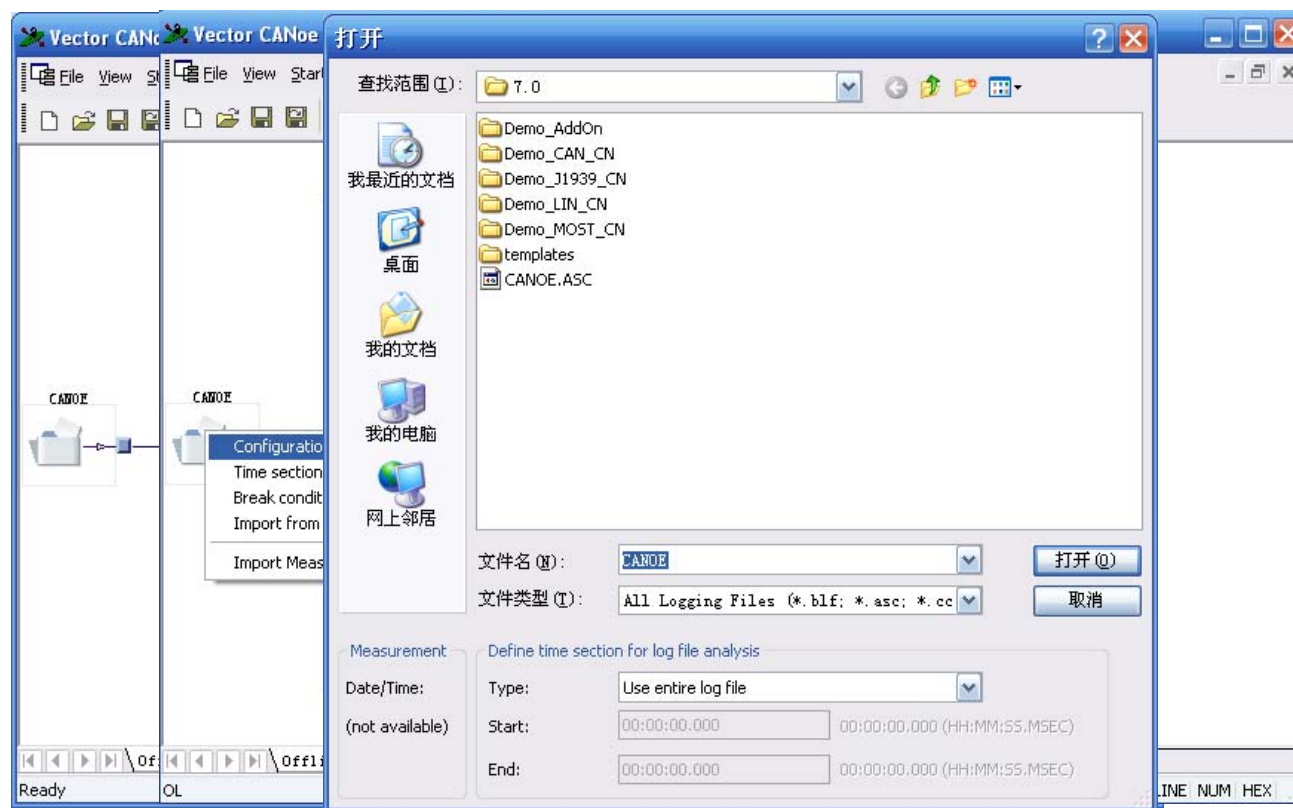
□ 触发记录



# 数据记录 (4/4)

□ 数据记录的目的是为了离线分析

□ Mode->To Offline



- ▣ Measurement Setup窗口和Simulation Setup窗口是CANoe的主要窗口，进行数据流规划
- ▣ 几乎窗口中的所有对象均可通过点击鼠标右键来访问交互菜单
- ▣ 所有数据传输到评估模块时，均会在对应窗口以各自的方式进行显示，记录模块除外
- ▣ 配置文件可以保存CANoe中的所有设置；可以使用已有的配置文件作为新任务的基础，进行简单的修改形成新的配置，提高效率