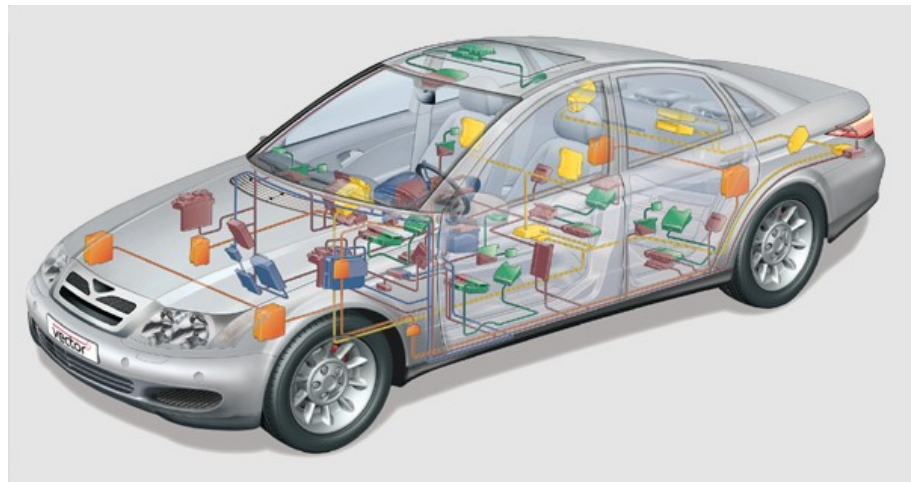


# CANoe 快速入门



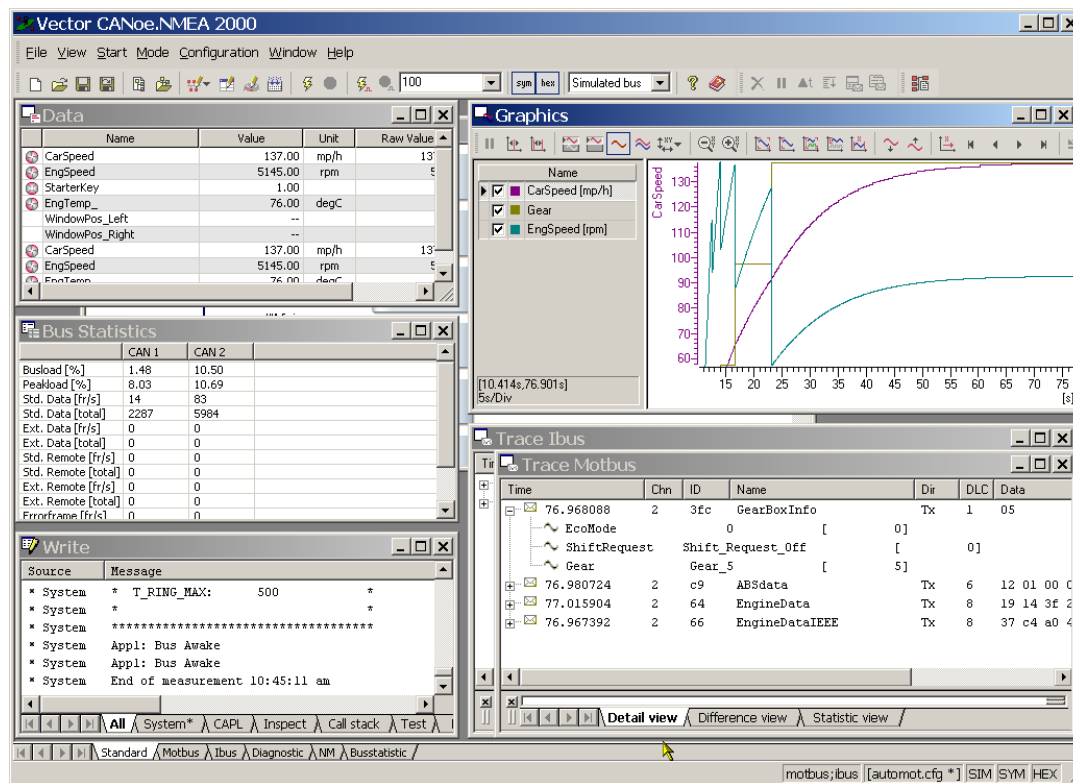
## □ CAN 总线开发工具

□ 测试

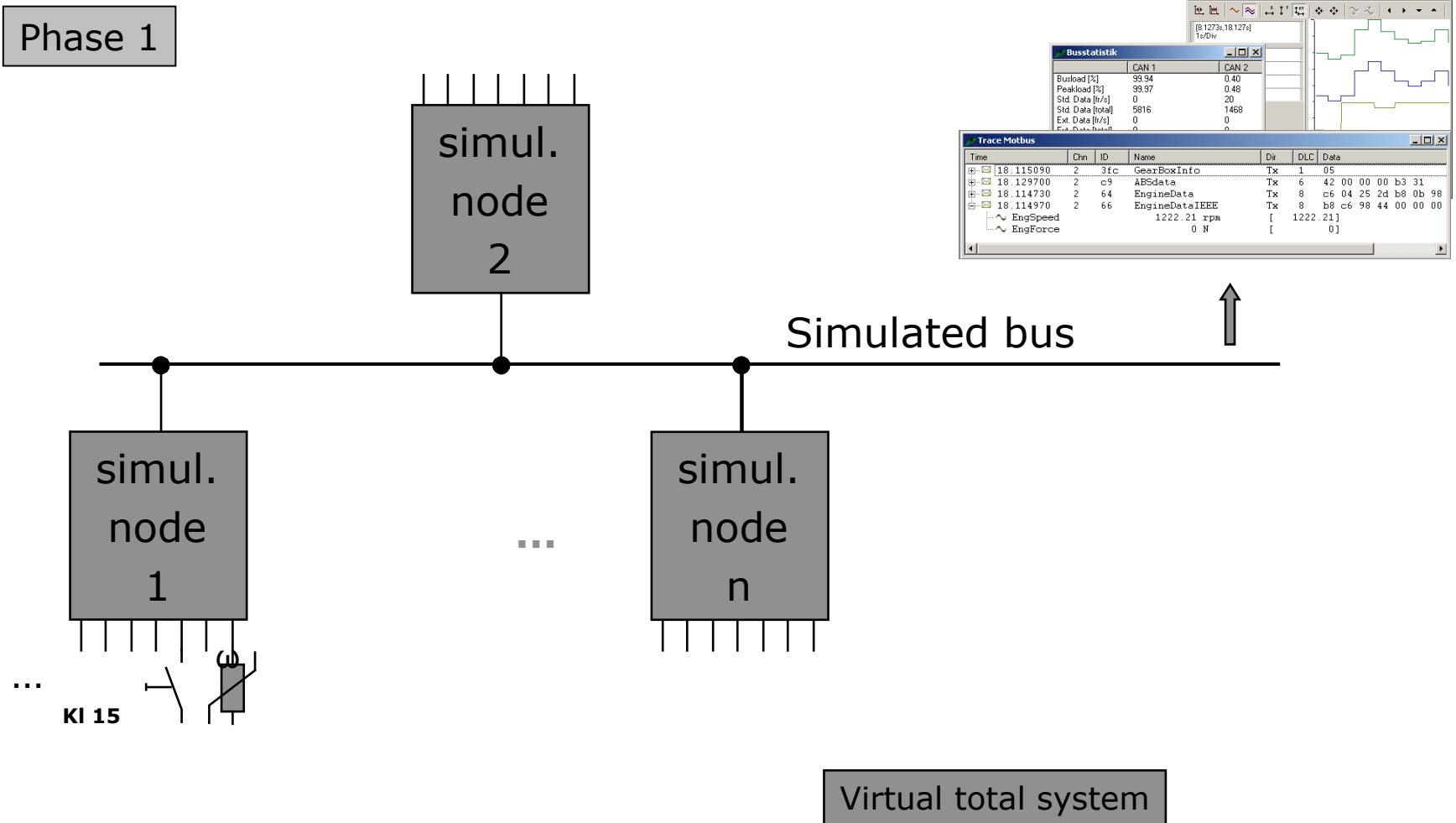
□ 分析

□ 仿真

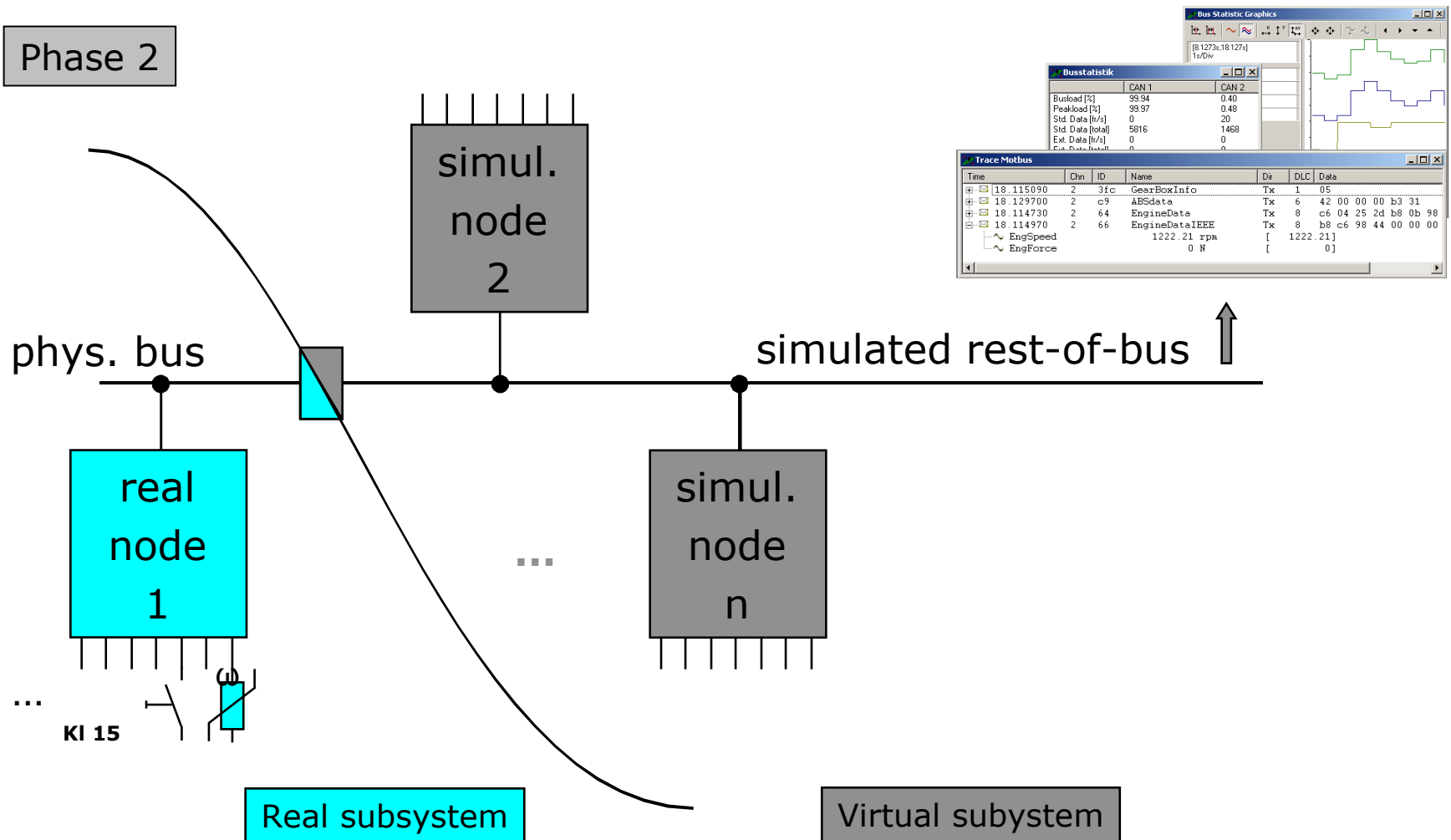
□ 记录



# CANoe 在总线开发中的作用 ( 1 )

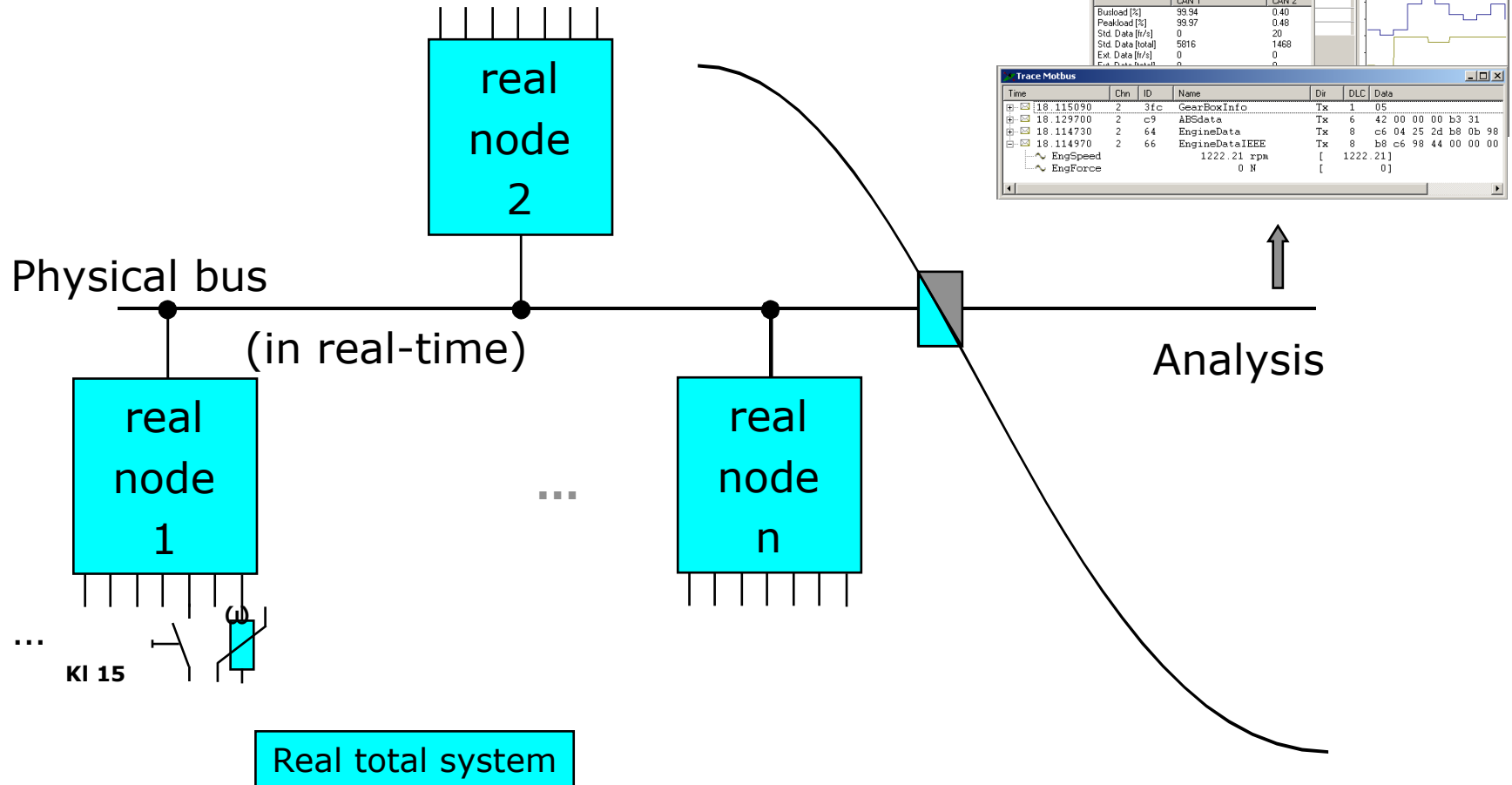


# CANoe 在总线开发中的作用 ( 2 )



# CANoe 在总线开发中的作用 ( 3 )

Phase 3



## □ 硬件接口卡 & “狗”

- CANcardXL

- CANcaseXL

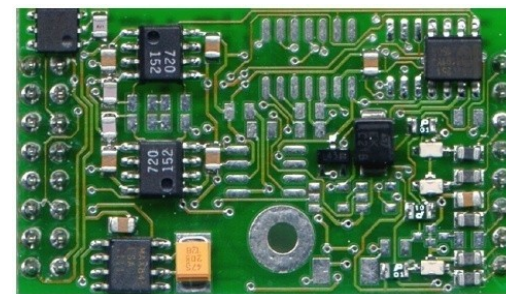
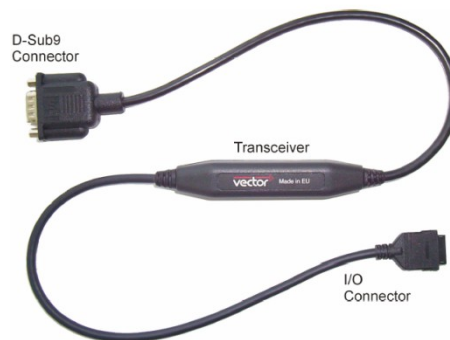
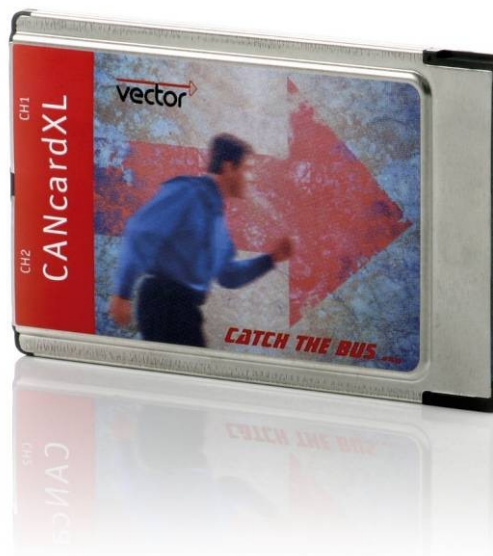
## □ 收发器

- CANcab (CANpiggy)

- 251, 1041, 1054...

- LINcab (LINpiggy)

- 7259



- 功能强大、操作简单

  - CANoe

- 数据库支持

  - CANdb++ Editor

- 可编程

  - CAPL 、 DLL

- 虚拟仪表

  - Panel Editor & Panel Designer

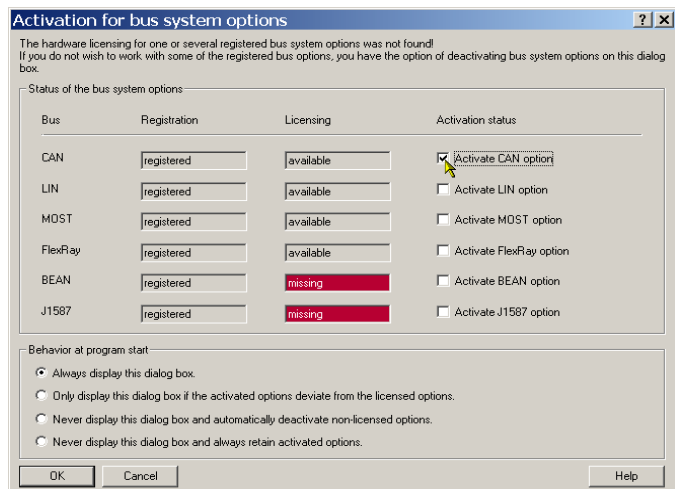
## □ 软件

## □ 硬件

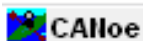
## □ 控制面板

## □ Vector Hardware

### □ License->Overview



#### Details



CANoe Application  
CANoe.CANopen / ProCANopen  
Application DENoe PRO  
CAN for DENoe PRO  
MOST for DENoe PRO  
FLEXRAY for DENoe PRO  
LIN for DENoe PRO



CANape Graph



CANalyzer Application  
Application DENalyzer PRO  
CAN for DENalyzer PRO  
MOST for DENalyzer PRO  
FLEXRAY for DENalyzer PRO  
LIN for DENalyzer PRO



# 使用设置 ( 1 )

## □ 硬件

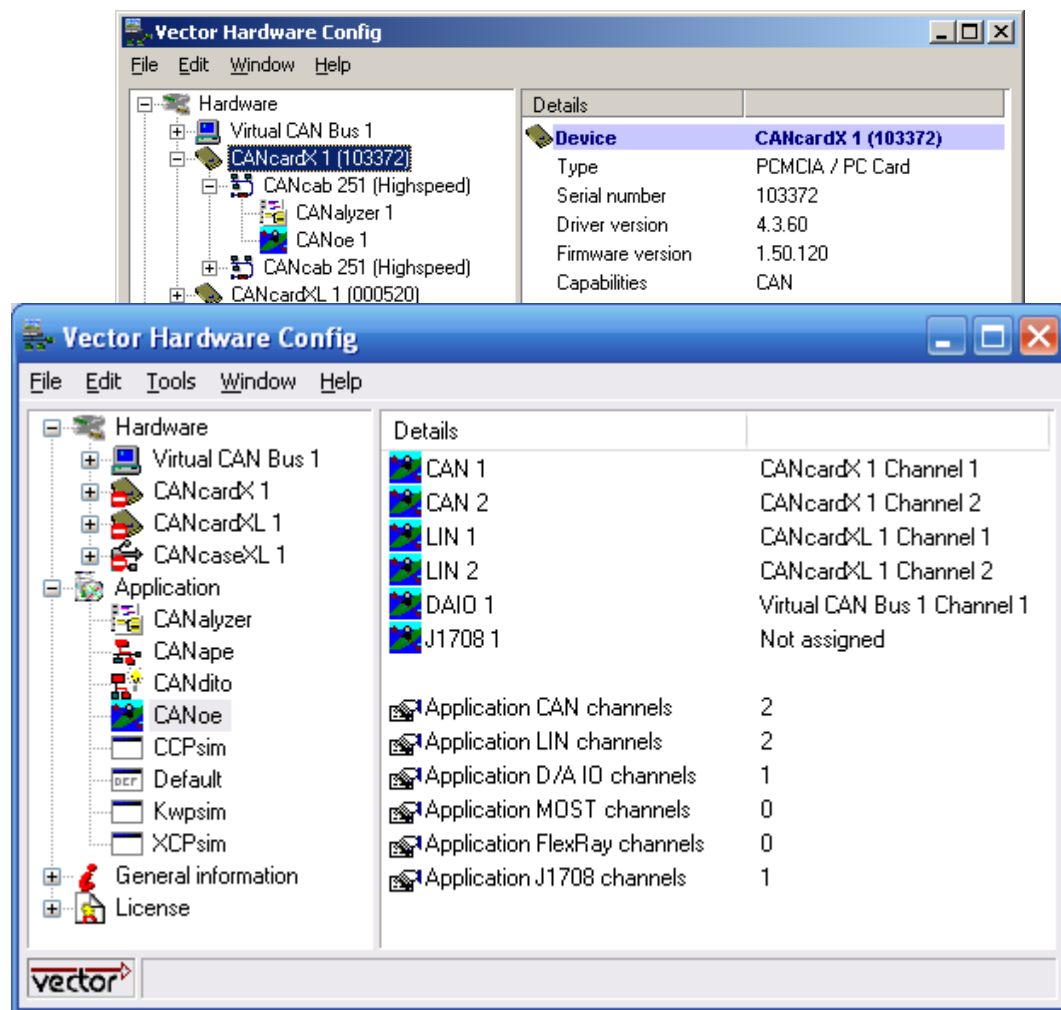
### □ CAN 卡类型 ( 编号 )

### □ 收发器类型

#### □ 应用程序通道

## □ 应用程序

## □ License 信息



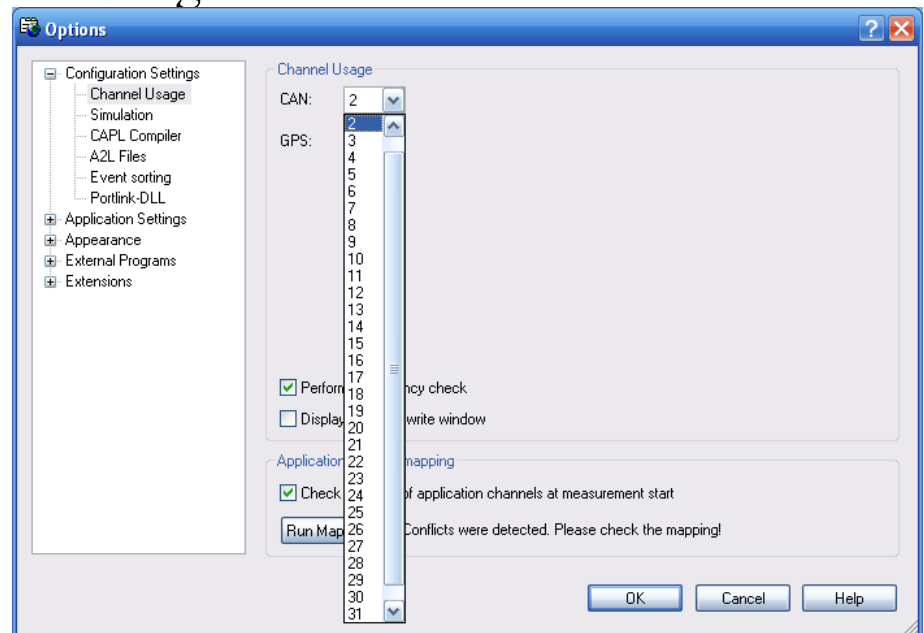
# 使用设置 ( 2 )

## □ CANoe

### □ 通道设置

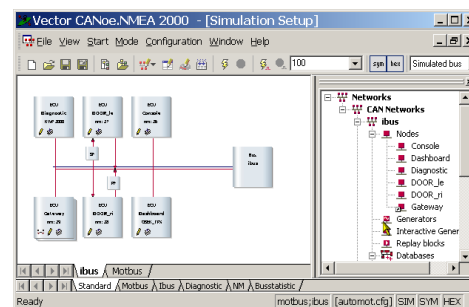
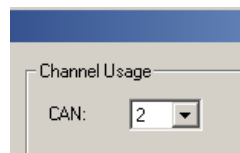
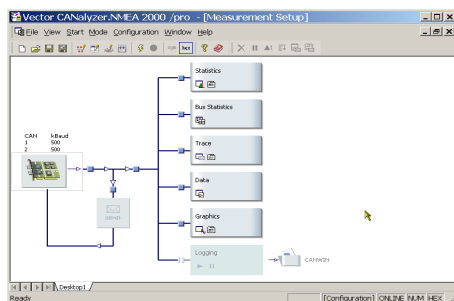
### □ Configuration->Options

### □ Configuration Settings->Channel Usage



# 灵活 = 复杂

Application



App channels

CANalyzer1   CANalyzer2...   CANoe1   CANoe2   CANoe1 for LIN...

HW channels

Channel1   Channel2   Channel1   Channel2   PiggyBack1   PiggyBack2

Hardware

**CANcardXL #1**



**CANcardXL #2**



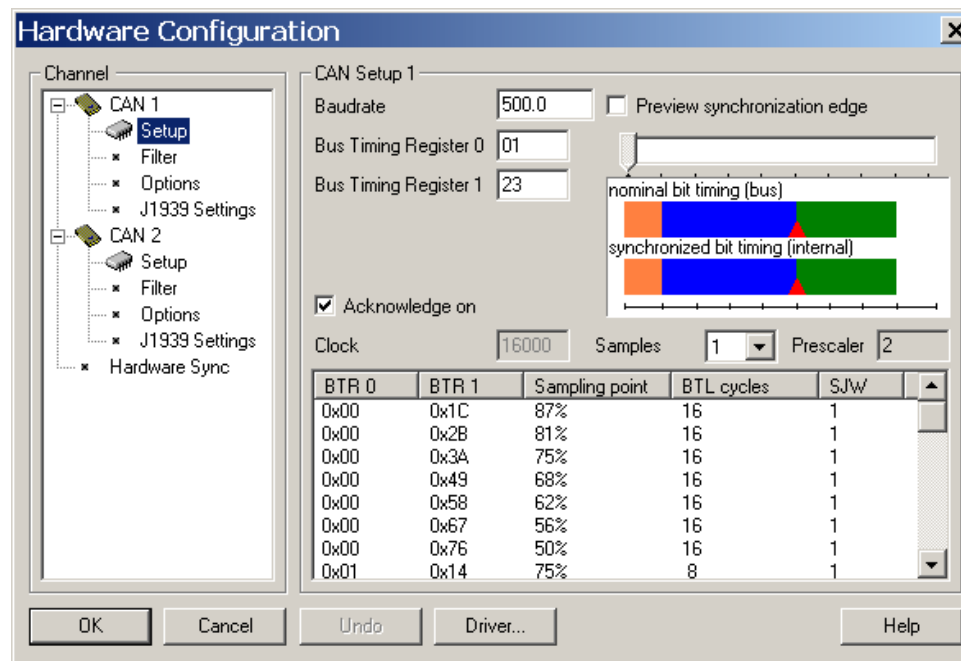
**CANcaseXL**



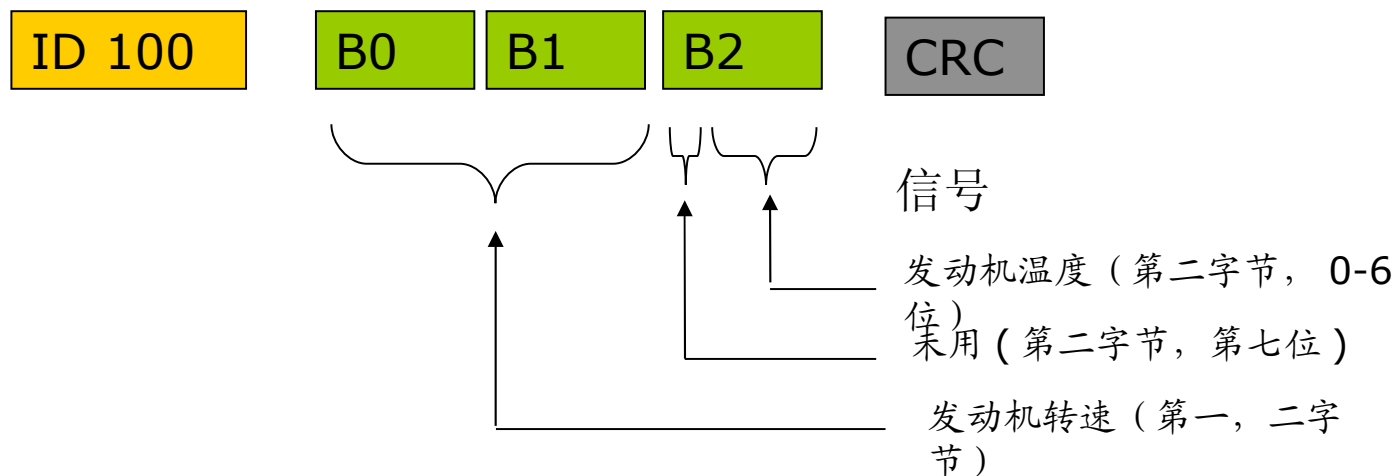
## □ CANoe

### □ 波特率设置

### □ Configuration->Hardware Configuration



## 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

# 欢迎进入 CANoe 的世界

- CANoe
- CANdb++ Editor
- CAPL
- Panel Editor & Panel Designer

# 欢迎进入 CANoe 的世界

## □ CANoe

### □ 8 大窗口

#### □ Trace Window

#### □ Bus Statistics Window

#### □ Statistics Window

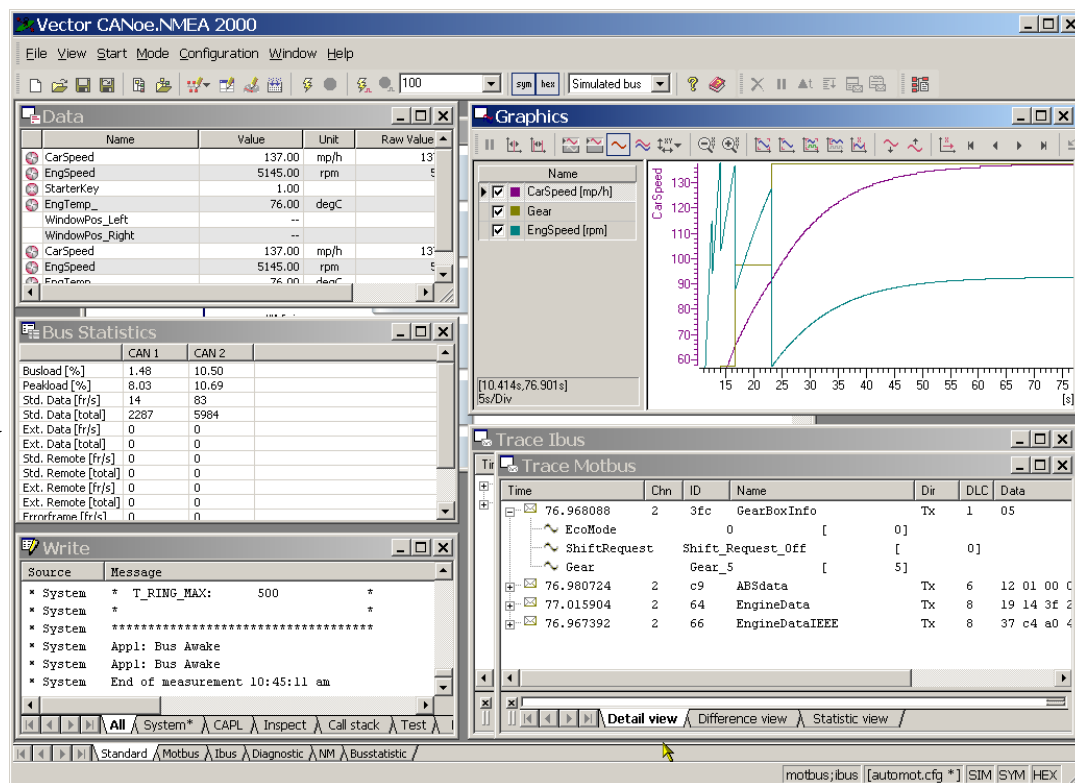
#### □ Data Window

#### □ Graphic Window

#### □ Write Window

#### □ Simulation Setup

#### □ Measurement Setup





# CANoe 窗口介绍 ( 1 )

## Trace Window

□ 报文 ID 和报文名称（数据库）

□ 信号（数据库）

□ 时间（相对值或绝对值）

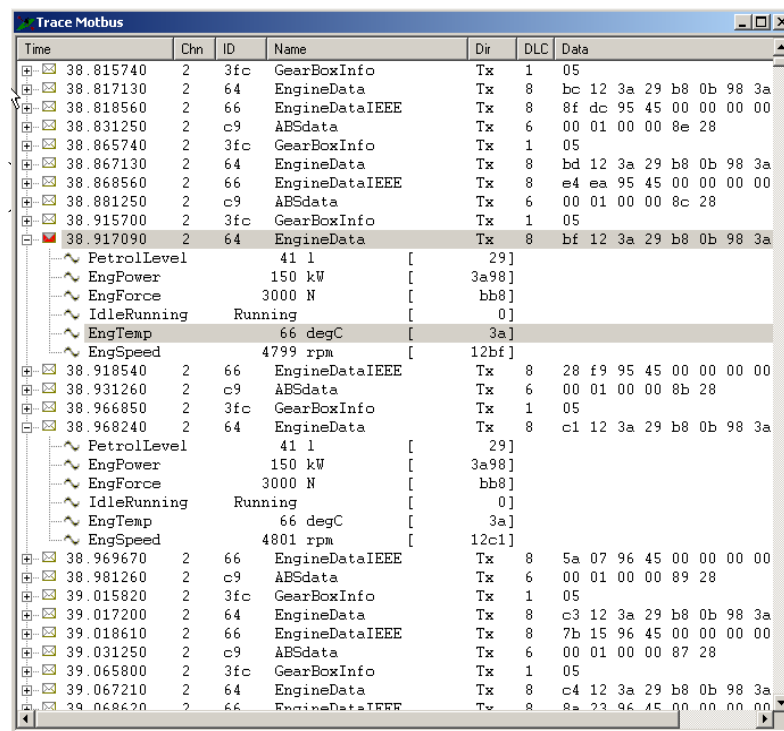
□ 通道

□ DLC

□ Dir （ Tx 或 Rx ）

□ 更多内容见

□ 右键点击窗口空白处 -> Configuration -> Columns



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]						
~ EngTemp 66 degC [ 3a]						
~ EngSpeed 4801 rpm [ 12c1]						
38.969670	2	66	EngineDataIEEE	Tx	8	5a 07 96 45 00 00 00 00
38.981260	2	c9	ABSdata	Tx	6	00 01 00 00 89 28
39.015820	2	3fc	GearBoxInfo	Tx	1	05
39.017200	2	64	EngineData	Tx	8	c3 12 3a 29 b8 0b 98 3a
39.018610	2	66	EngineDataIEEE	Tx	8	7b 15 96 45 00 00 00 00
39.031250	2	c9	ABSdata	Tx	6	00 01 00 00 87 28
39.065800	2	3fc	GearBoxInfo	Tx	1	05
39.067210	2	64	EngineData	Tx	8	c4 12 3a 29 b8 0b 98 3a
39.068620	2	66	EngineDataIEEE	Tx	8	8a 23 96 45 00 00 00 00

# CANoe 窗口介绍 ( 1 )



清空 Trace 窗口

暂停 Trace 窗口

差额（相对）时间显示

滚屏（绝对）时间显示

报文详细信息

Trace 窗口查找  
窗口显示配置

## □ Data Window

□ 数据库

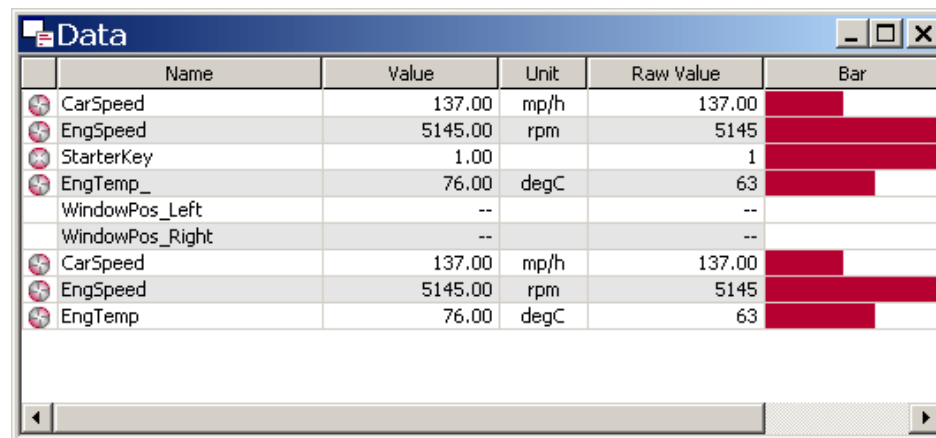
□ 信号名称

□ 信号值 ( Value )

□ 信号单位

□ 原始值 ( Raw Value )

□ Bar 图



	Name	Value	Unit	Raw Value	Bar
⊗	CarSpeed	137.00	mp/h	137.00	
⊗	EngSpeed	5145.00	rpm	5145	
⊗	StarterKey	1.00		1	
⊗	EngTemp_	76.00	degC	63	
	WindowPos_Left	--		--	
	WindowPos_Right	--		--	
⊗	CarSpeed	137.00	mp/h	137.00	
⊗	EngSpeed	5145.00	rpm	5145	
⊗	EngTemp	76.00	degC	63	

# CANoe 窗口介绍 ( 2 )

## □ Data Window

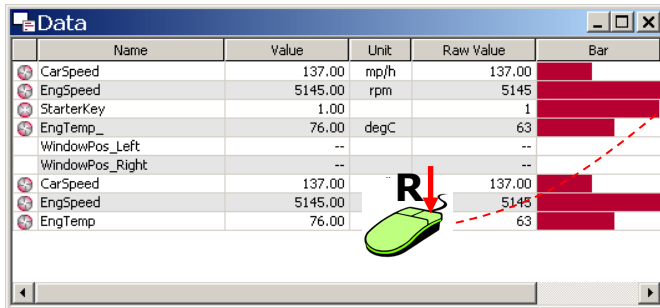
### □ 添加信号

□ 右键单击空白处

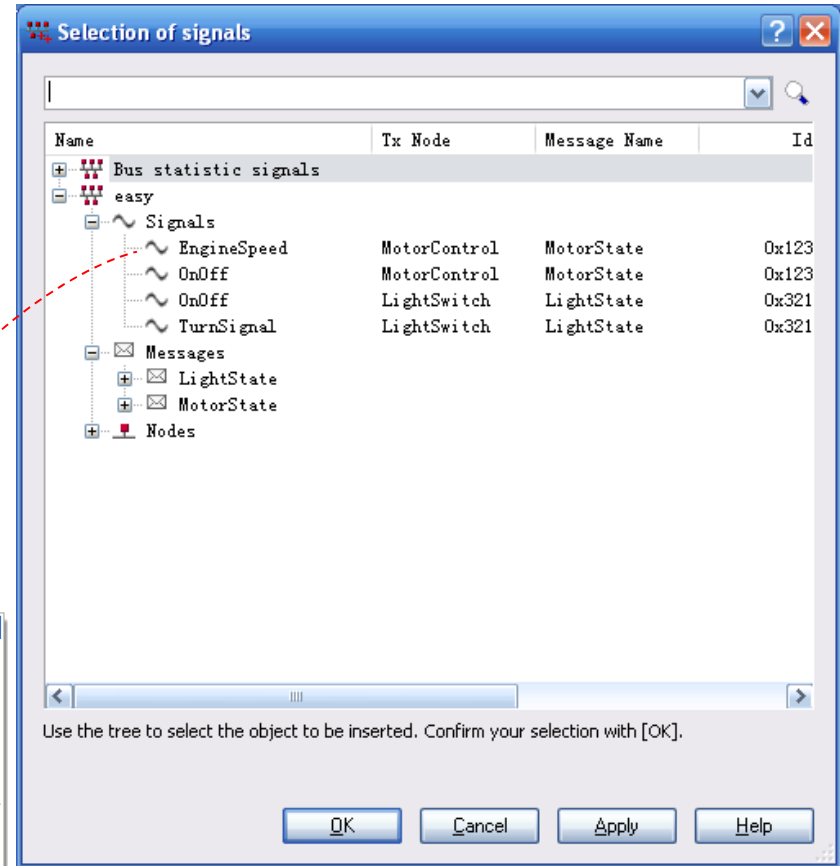
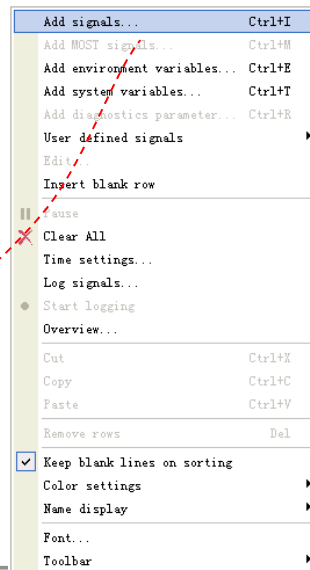
□ Add Signals

□ 选择需要的信号

### □ 支持鼠标拖放



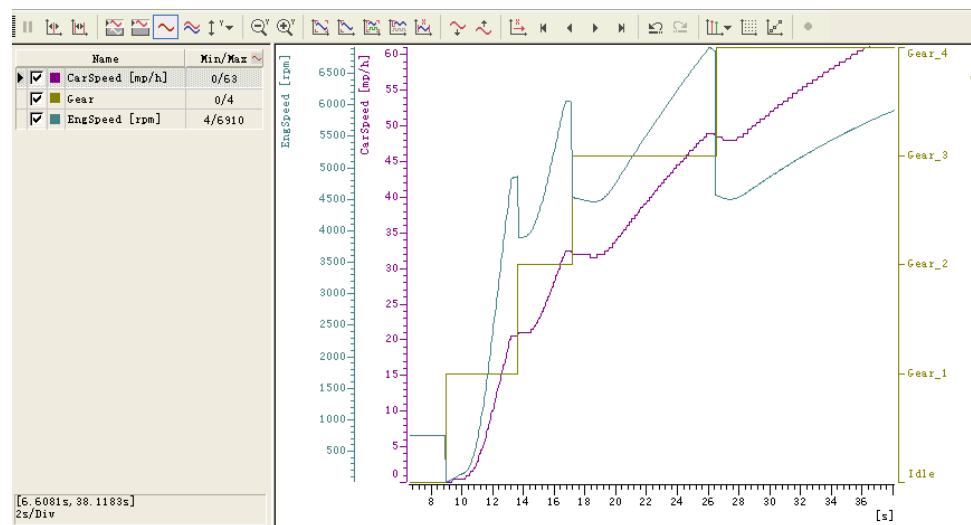
	Name	Value	Unit	Raw Value	Bar
CarSpeed		137.00	mph/h	137.00	
EngSpeed		5145.00	rpm	5145	
StarterKey		1.00		1	
EngTemp_		76.00	degC	63	
WindowPos_Left		--		--	
WindowPos_Right		--		--	
CarSpeed		137.00		137.00	
EngSpeed		5145.00		5145	
EngTemp		76.00		63	



# CANoe 窗口介绍 ( 3 )

## □ Graphics Windows

- 支持数据库
- 显示信号曲线
- 不同的颜色和线形
  - 右键单击空白处
  - 选择 Configuration
- 放大、缩小、平移...



## □ Graphics Windows

### □ 添加信号

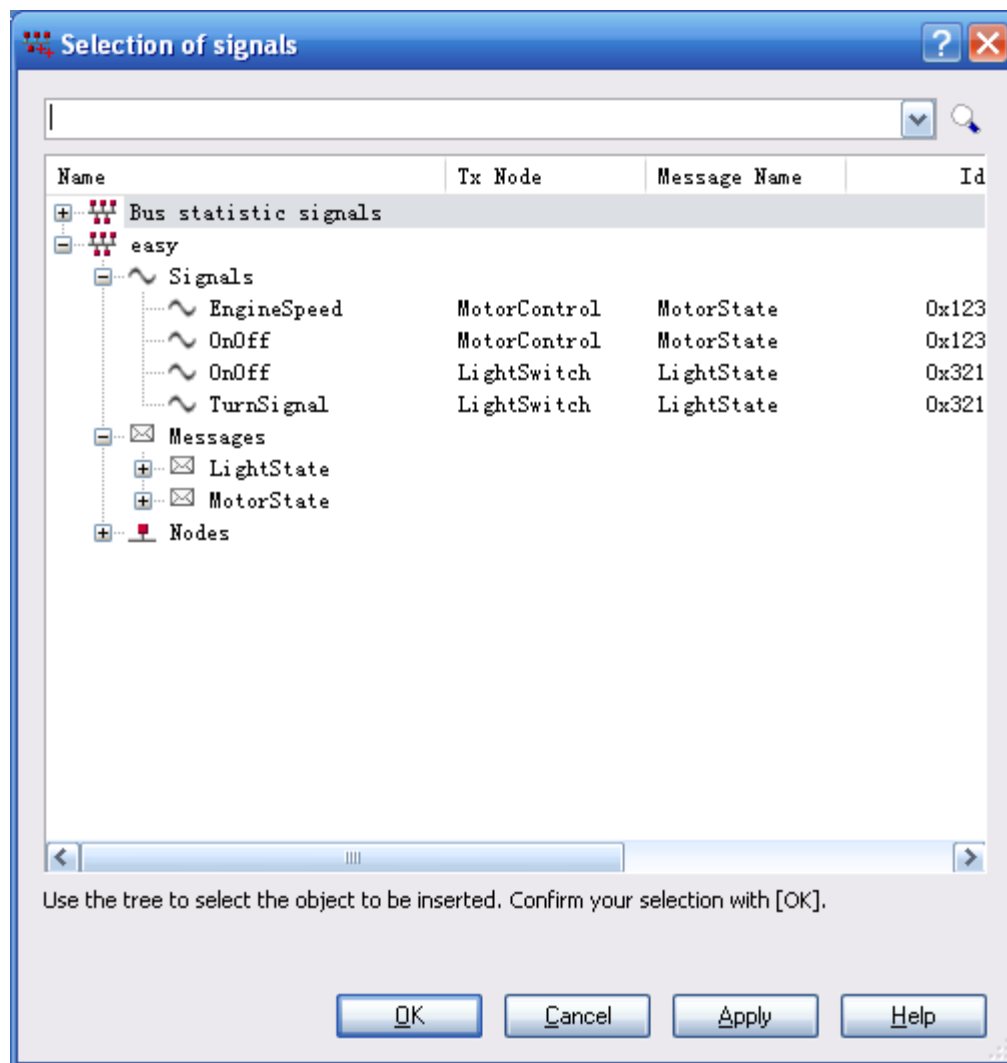
□ 右键单击空白处

□ Add Signals

□ 选择需要的信号

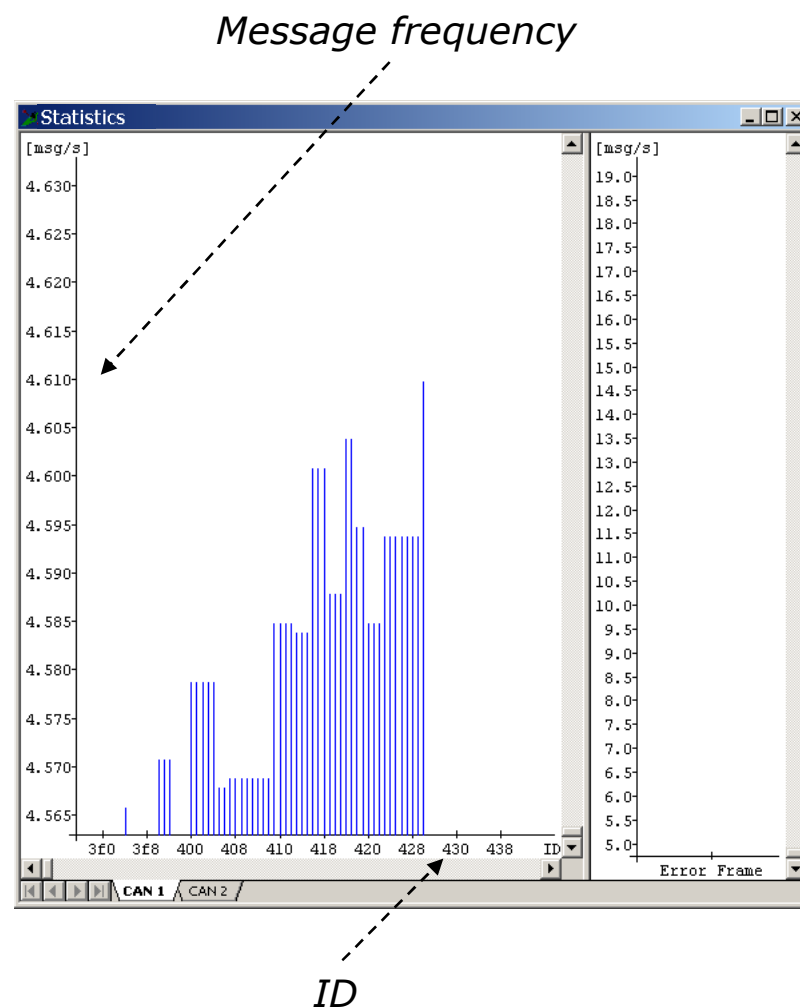
□ 支持鼠标拖放

□ 与Trace窗口同步



## ❑ Statistics Window

- ❑ 显示报文出现频率
- ❑ 显示错误帧出现频率
- ❑ 统计报告
  - ❑ 右键单击空白处
  - ❑ Configuration
  - ❑ Active
  - ❑ 生成统计报告 (Write Window)



## □ Bus Statistics Window

□ 总线负载

□ 数据帧

□ 错误帧

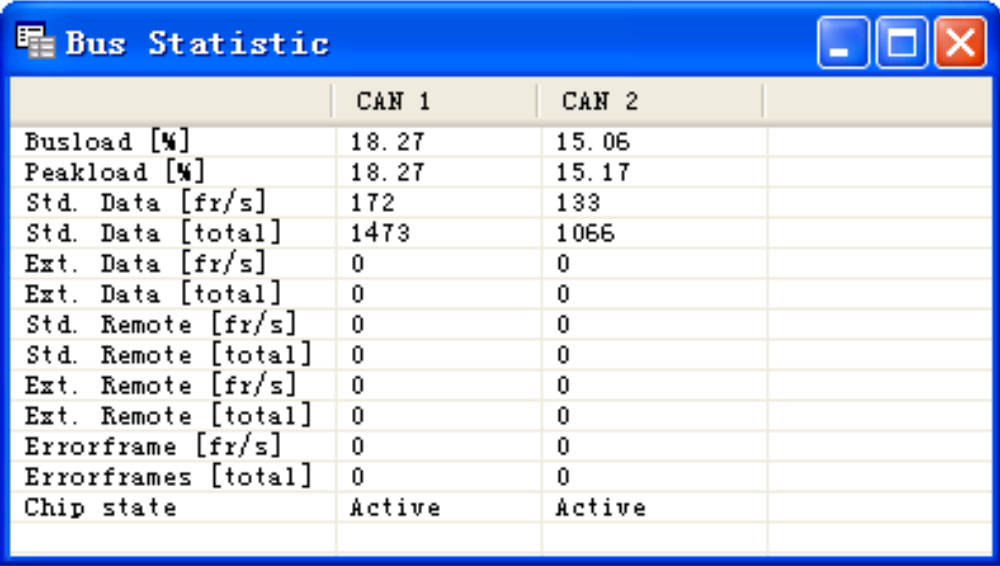
□ CAN 卡控制器状态

□ ACTIVE

□ WARNING

□ PASSIVE

□ OFF



	CAN 1	CAN 2	
Busload [%]	18.27	15.06	
Peakload [%]	18.27	15.17	
Std. Data [fr/s]	172	133	
Std. Data [total]	1473	1066	
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	



## □ Write Window

□ 总线负载

□ 数据帧

□ 错误帧

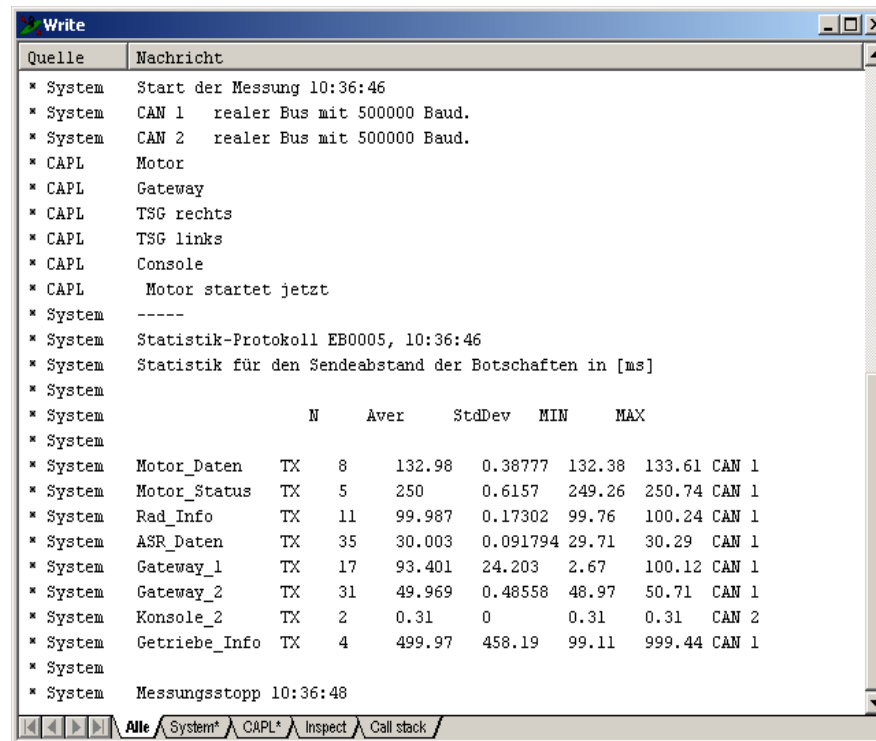
□ CAN 卡控制器状态

□ License 信息

□ 统计报告

□ CAPL 输出窗口

□ Printf = Write

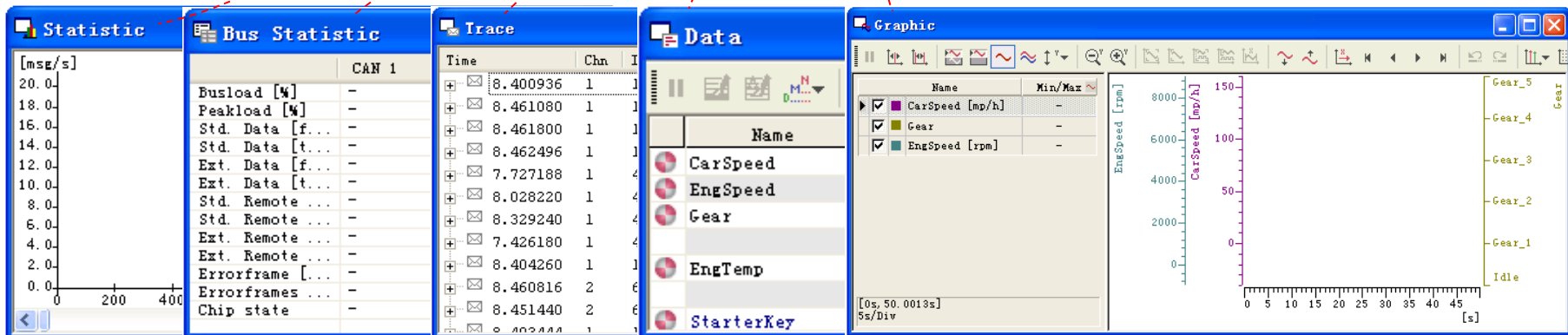
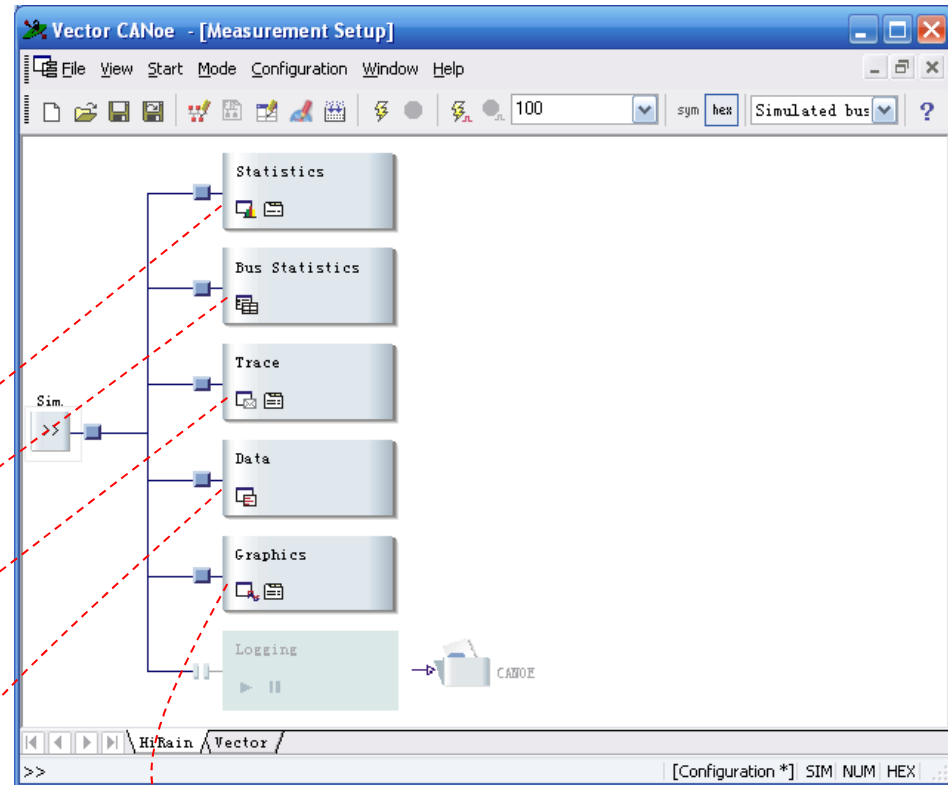


# CANoe 窗口介绍 ( 7 )

## Measurement Setup

### View->Measurement Setup

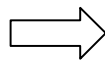
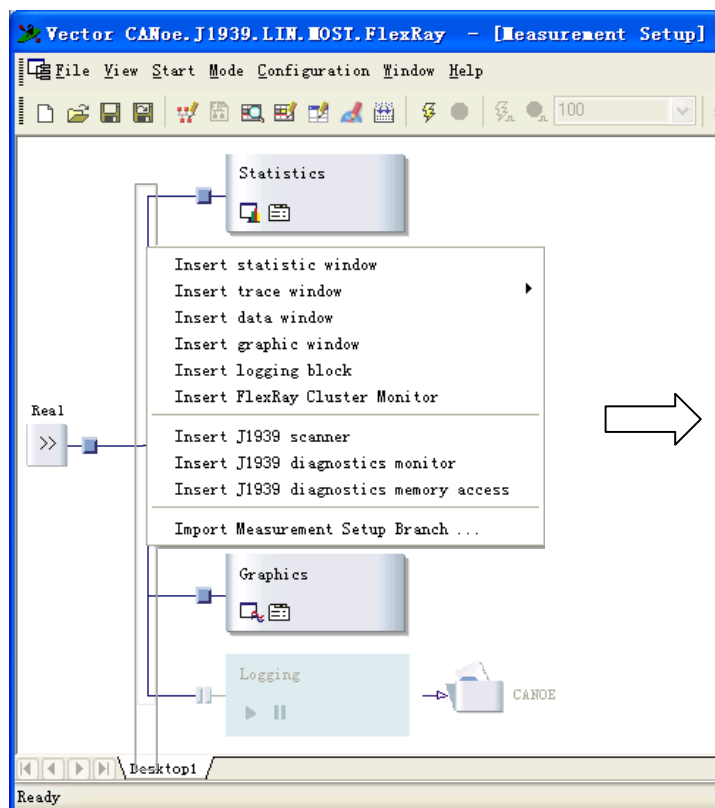
- 每个模块对应一个窗口
- 增加新模块（窗口）
- 插入功能块
- 数据记录



# CANoe 窗口介绍 ( 7 )

## □ 新增模块 (窗口)

### □ 主干线上右键

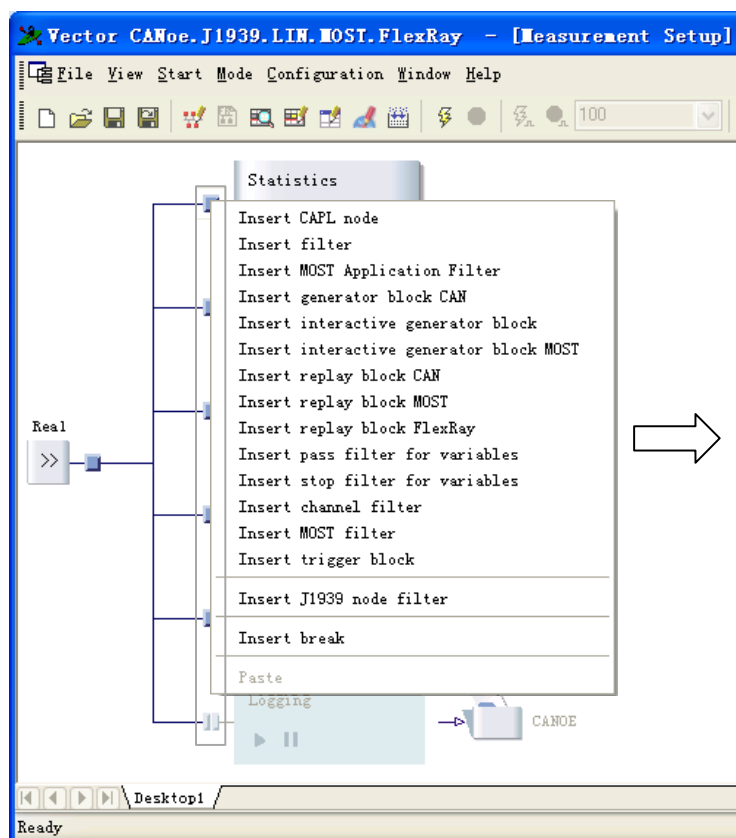


名称	模块	功能
统计模块	 Statistics	按 ID 统计报文频率
跟踪模块	 Trace	显示总线报文细节
数据模块	 Data	以柱状图方式显示数据变化
图形模块	 Graphics	以曲线方式显示历史信号
记录模块	 Logging	根据触发条件记录总线数据
FlexRay 观察模块	 FlexRay Cluster Mon	监控 FlexRay 网络, 快速错误诊断, 检测跟数据库模型不一致的报文, 报文过滤
J1939 扫描模块	 J1939 Scanner	检测激活节点状态, 解析参数组和地址声明报文
DTC 观察模块	 DTC Monitor	诊断故障代码观察器
内存访问模块	 Memory Access	J1939 诊断内存访问 (根据 DM14、DM15)

# CANoe 窗口介绍 ( 7 )

## □ 插入功能块

### □ 支线上节点上右键

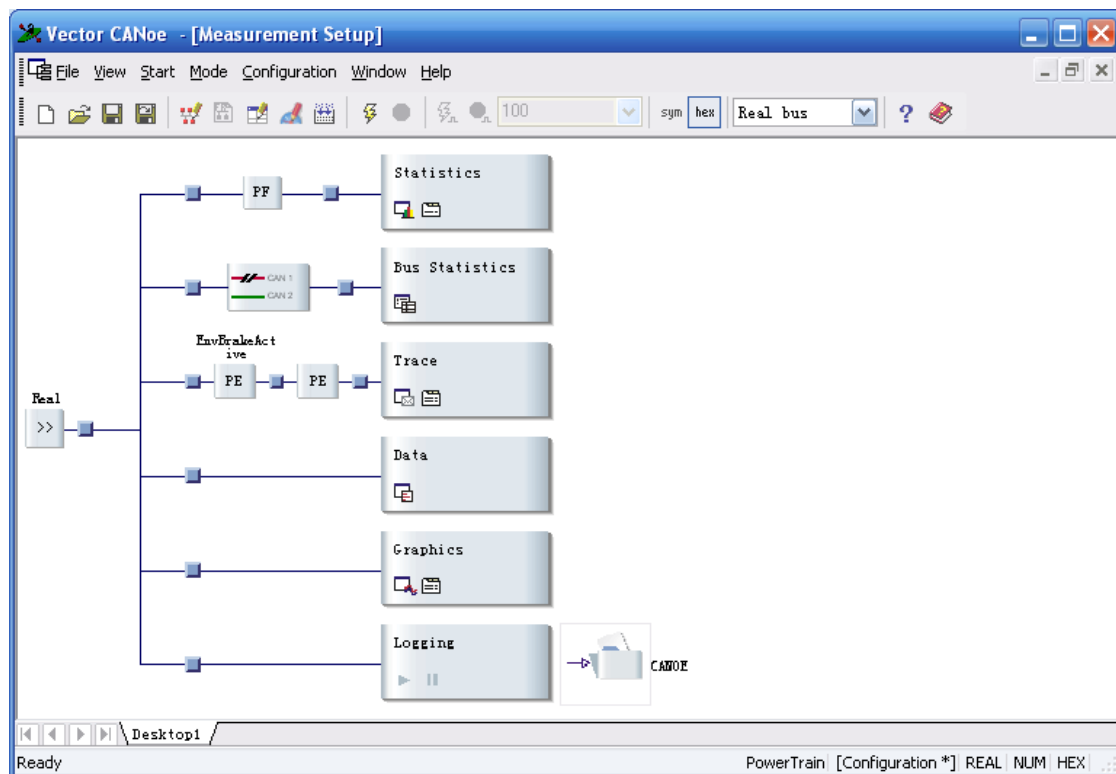


名称	类型	缩写	功能
CAPL 节点	数据源/接收器	P	CAPL 脚本
阻过滤/通过滤	接收器	PF/SF	过滤 CAN、LIN 报文
MOST 过滤	接收器	MAF	过滤 MOST 报文
CAN 发生模块	数据源	G	产生 CAN 报文
交互发生模块	数据源	IG	产生 CAN 信号
MOST 交互发生模块	数据源	MG	产生 MOST 信号
CAN 回放模块	数据源	R	回放 CAN 记录报文
MOST 回放模块	数据源	R	回放 MOST 记录报文
FlexRay 回放模块	数据源	R	回放 FlexRay 记录报文
变量通过滤	接收器	PE	过滤变量
变量阻过滤	接收器	SE	通过变量
通道过滤	接收器	-	过滤总线通道数据
MOST 过滤	接收器	MF	过滤 MOST 报文
触发模块	数据源	T	设置触发条件
J1939 过滤	接收器	-	过滤 J1939 报文

# CANoe 窗口介绍 ( 7 )

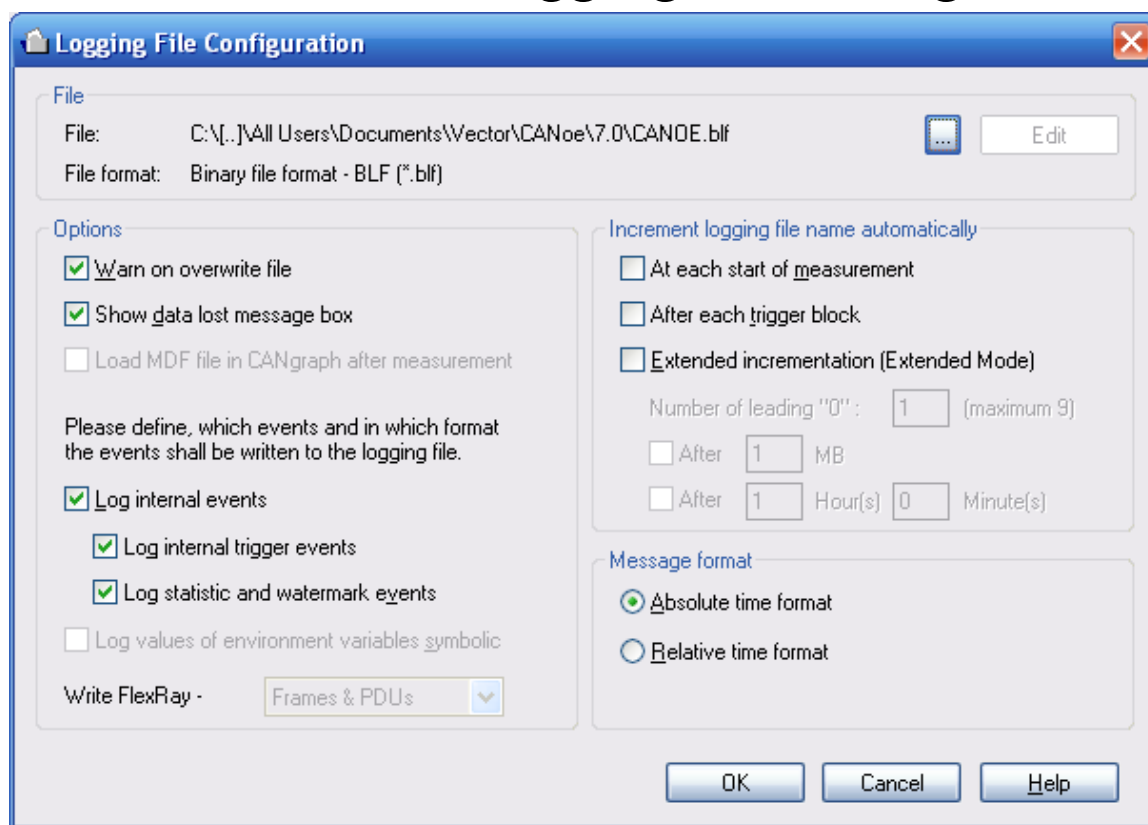
## □ 数据记录

- 默认状态关闭
- 多种记录文件格式
- 多种记录触发配置



## □ 记录文件

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



# CANoe 窗口介绍 ( 7 )

## □ 记录配置方式

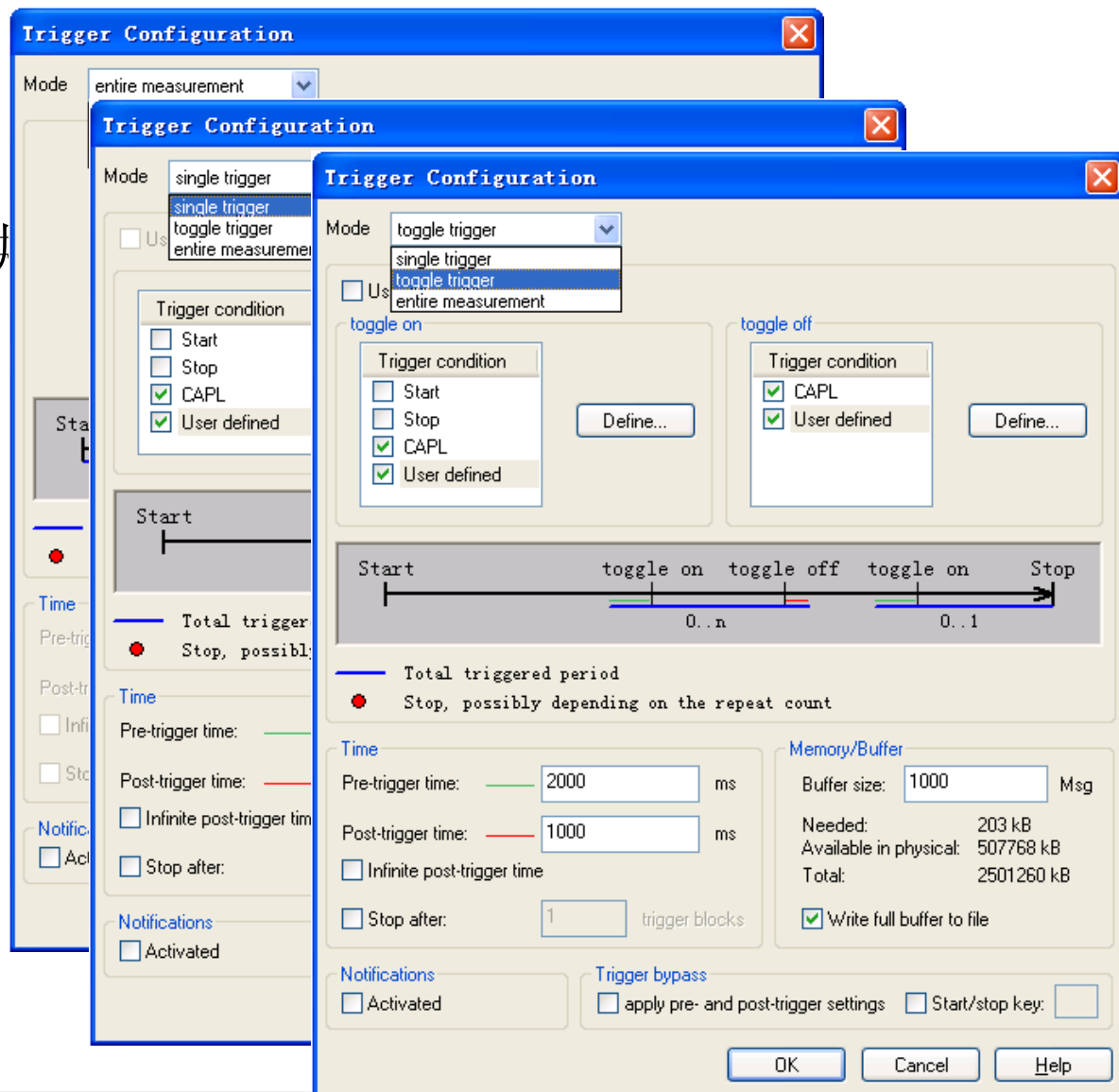
### □ 双击 Logging 模块

#### □ 全部记录

#### □ 单次记录

#### □ 触发记录

### □ 记录数据回放

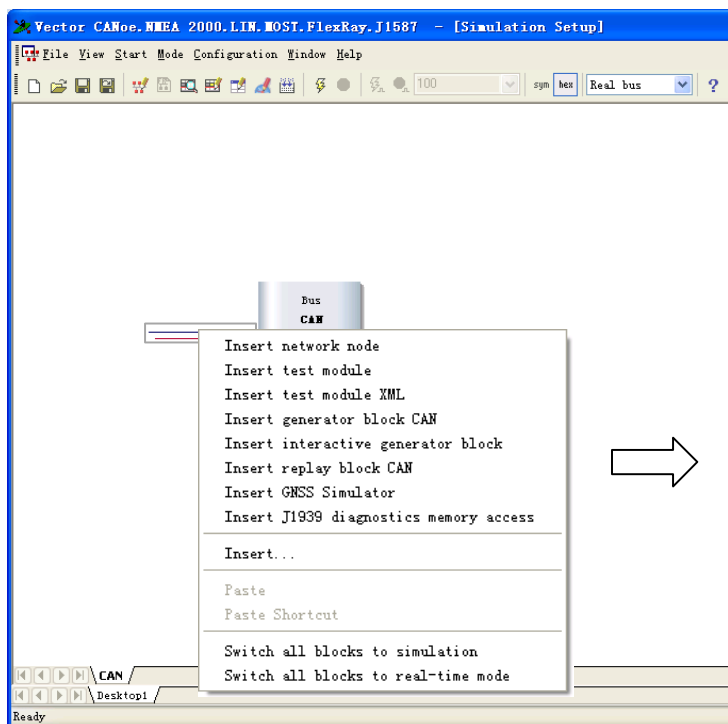


# CANoe 窗口介绍 ( 8 )

## □ Simulation Setup

### □ View->Simulation Setup

#### □ 主干线添加模块



名称	模块	功能
网络节点	ECU ECU 1 Prog	仿真网络节点，可以关联 CAPL 脚本和 DLL 动态链接库。
测试模块	Test Test 2 Prog	测试节点，可以关联 CAN 测试脚本
测试 XML 模块	Test Test 3 Prog	测试节点，可以关联 XML 测试脚本
发生模块 CAN	Generator G [P]	报文发生器，可设置报文序列、触发条件、发送周期
交互发生模块	I-Generator IG	报文、信号发生器，可设置报文序列、信号曲线、触发条件、发送周期、网关转发
回放模块 CAN	Replay ReplayBlock CANOE.blf	总线记录数据回放

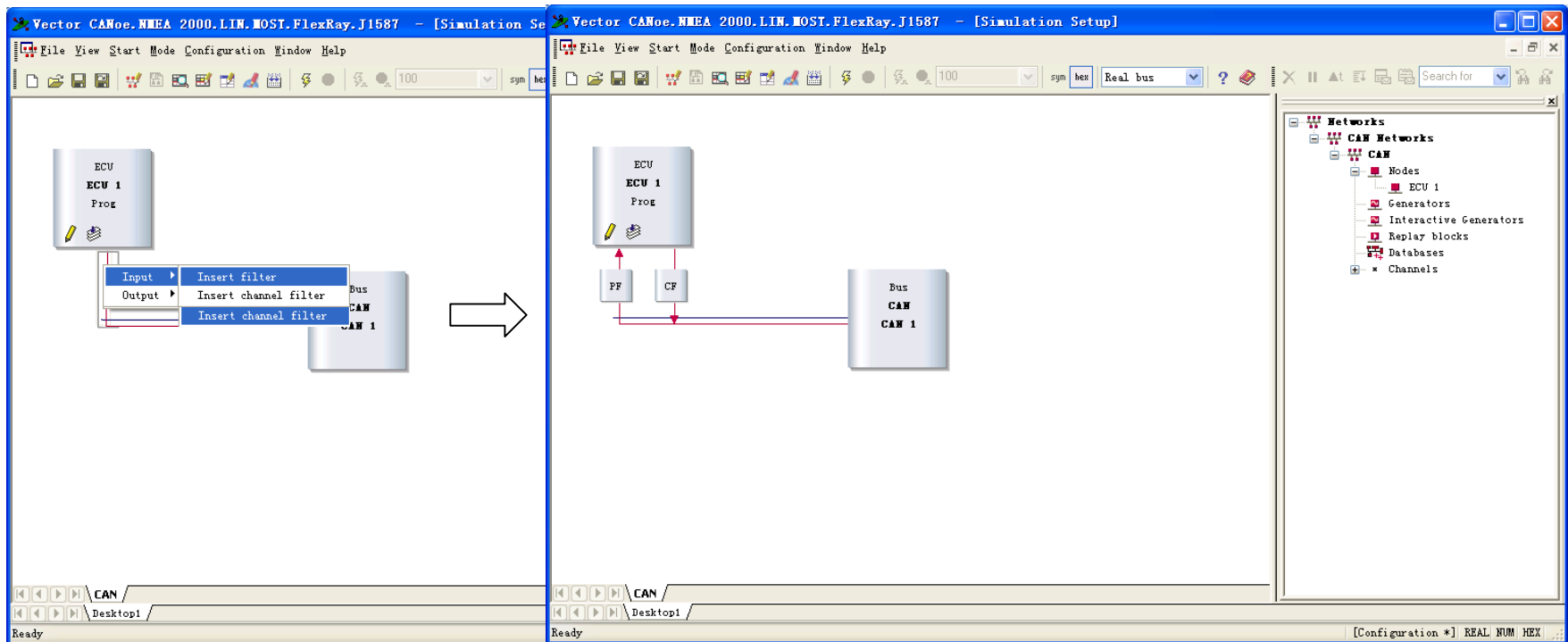


# CANoe 窗口介绍 ( 8 )

## □ Simulation Setup

### □ View->Simulation Setup

#### □ 支线添加模块

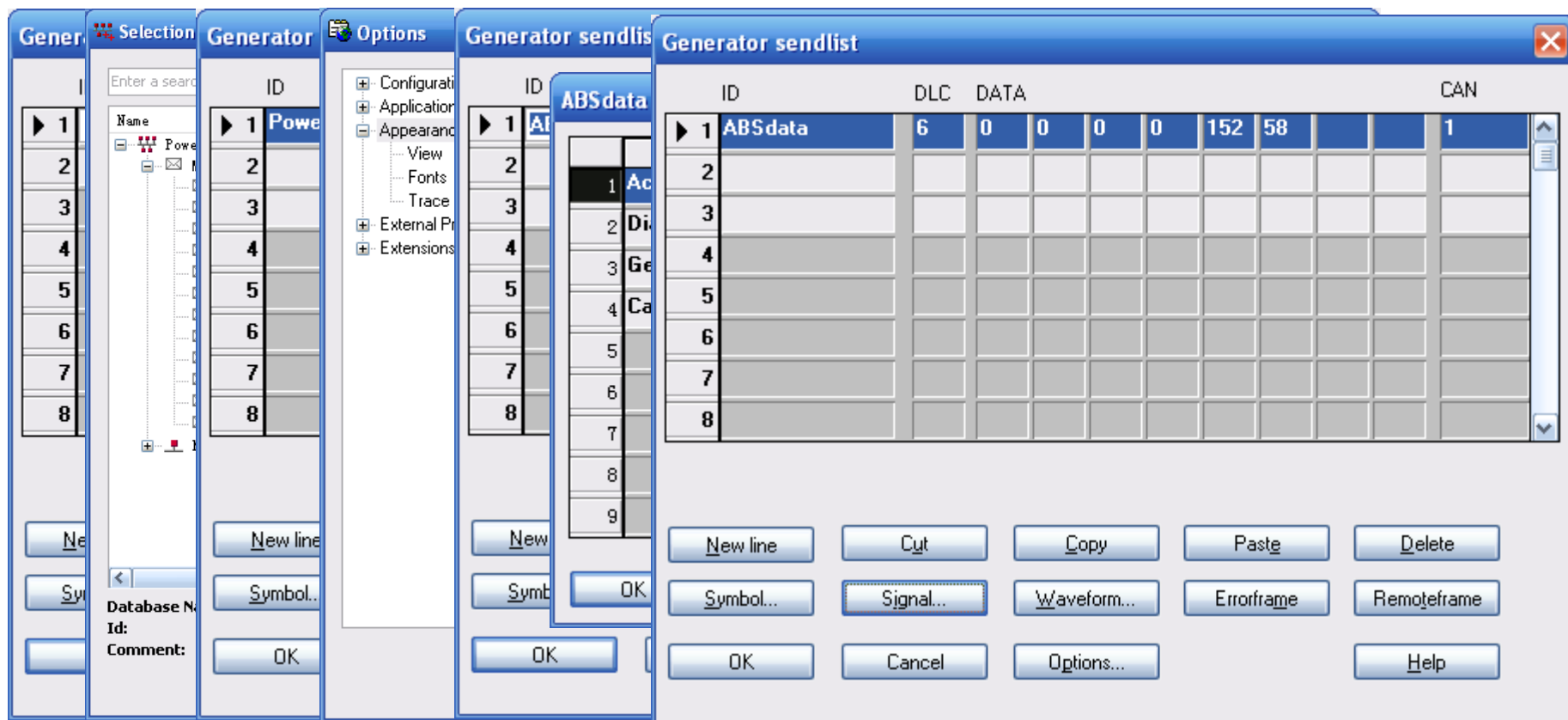


1. 周期报文发送
2. 梯形信号发送
3. 正弦信号发送

# 练习 1

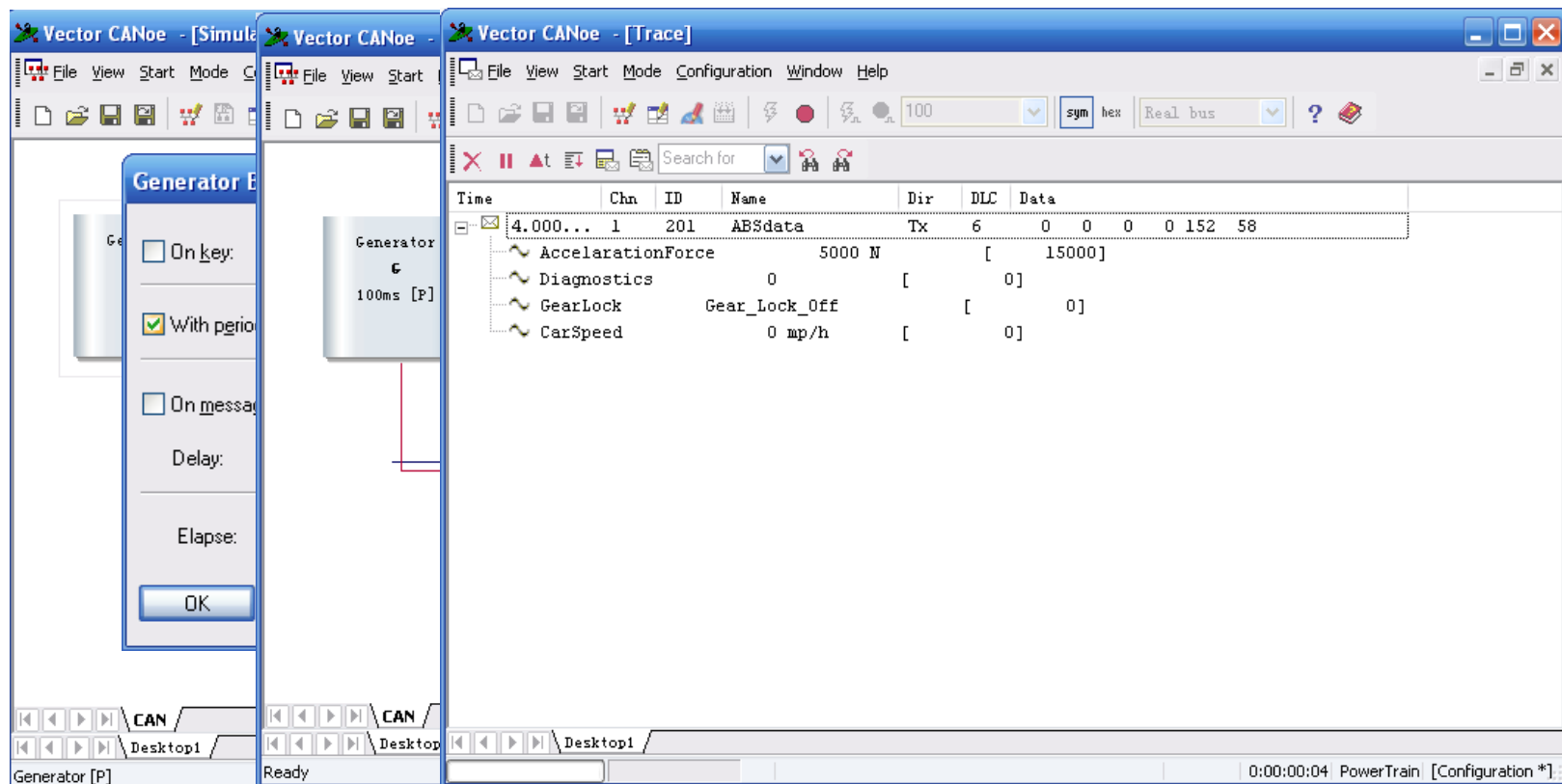
□ 发生器模块发送周期报文

□ 设置发送内容



# 练习 1

- 发生器模块发送周期报文
- 设置发送周期



## □ 发生器模块发送阶梯信号

The screenshot shows the Vector CANdb software interface. The 'Generator sendlist' window is open, displaying a list of 8 CAN messages. The 'Signal function' window is also open, showing a step function graph for 'CarSpeed'.

**Generator sendlist**

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

**Signal function**

Preview

n2

n1

ta

th

## 交互式发生器模块发送正弦信号

The screenshot displays the Vector CANoe software interface with several windows open. The 'Configurator' window is the primary focus, showing the configuration for a sine signal generator. The 'Value generator type' is set to 'Sine'. The 'Specific settings' section includes: Amplitude: 500, Period: 200 [samples], Phase: 0 [deg], and Offset: 500. The 'Common settings' section has 'Generator active' and 'Repetitive output sending mode' checked. The 'Timing settings' section has 'Time bound sending mode' unchecked. The 'Values' table shows a list of input and output values. The 'Graphical output' window shows a sine wave plot. The 'Triggering' window shows the 'Cycle Time [ms]' set to 50. The 'Phys Step' window shows a list of physical steps.

**Configurator**

Value generator type: Sine

Target settings

- Sample time: 50 [ms]
- Default value: 0.00
- CANdb min: 0.00
- Type min: 0.00

Specific settings

- Amplitude: 500
- Period: 200 [samples]
- Phase: 0 [deg]
- Offset: 500

The amplitude and offset are expressed in physical units.

Common settings

- ☒ Generator active
- ☒ Repetitive output sending mode

Timing settings

- ☐ Time bound sending mode

In time unbound sending mode the waveform depends on the target sample time! The values defined by the generator are output consecutively.

Values

Value	Output
515.71	516.00
531.40	531.00
547.05	547.00
562.67	563.00
578.22	578.00
593.69	594.00
609.07	609.00
624.34	624.00
639.50	639.00
654.51	655.00
669.37	669.00
684.00	684.00

Graphical output

Phys Step

Triggering

Cycle Time [ms]

Phys Step

13

7.5

3277

1

10

400

Help

Close

# 欢迎进入 CAPL 的世界

## □ CAPL (CAN Access Programming Language)

### □ 类 C 语言

### □ 仿真

#### □ 单个节点和整个网络

#### □ 外部环境

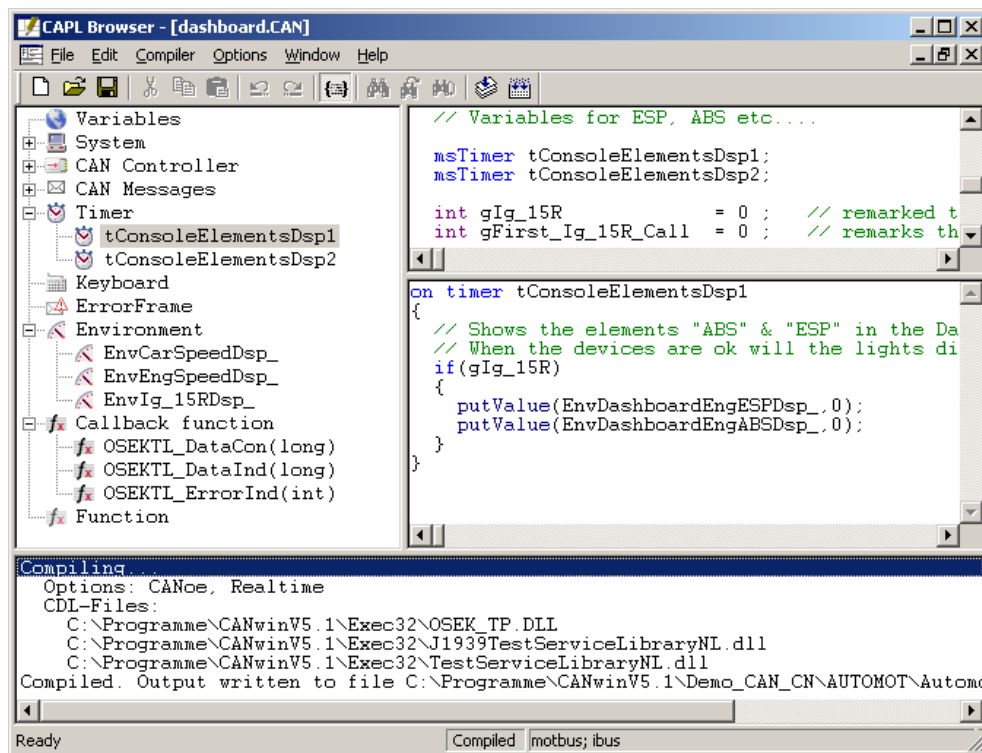
#### □ 测试

### □ 面向事件的编程语言

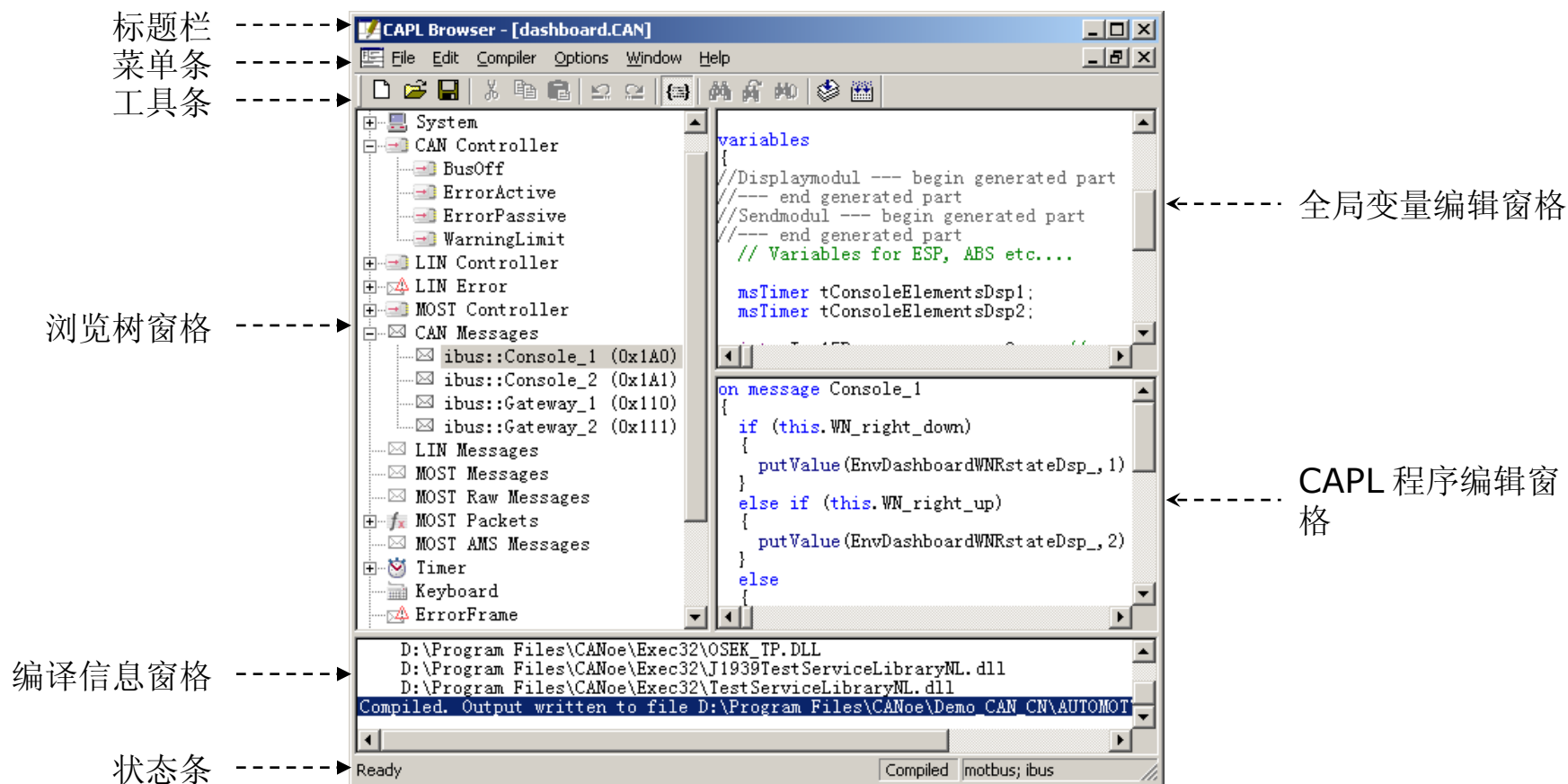
#### □ 总线事件

#### □ 键盘事件

#### □ 时间事件



# CAPL Browser





# CAPL 事件

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</i> <i>Message</i> { ... }
时间事件	自定义	定时时间朝过时执行	<i>on timer</i> <i>Timer</i> { ... }
键盘事件	自定义键值	指定的键被下时执行	<i>on key</i> <i>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</i> <i>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();  
}
```

- ❑ 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

- 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

## □ 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` 设定的时间到响应

## □ 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`

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	11/29 ID
定时器	timer	秒
	msTimer	毫秒
单个字符	char	1 个字节

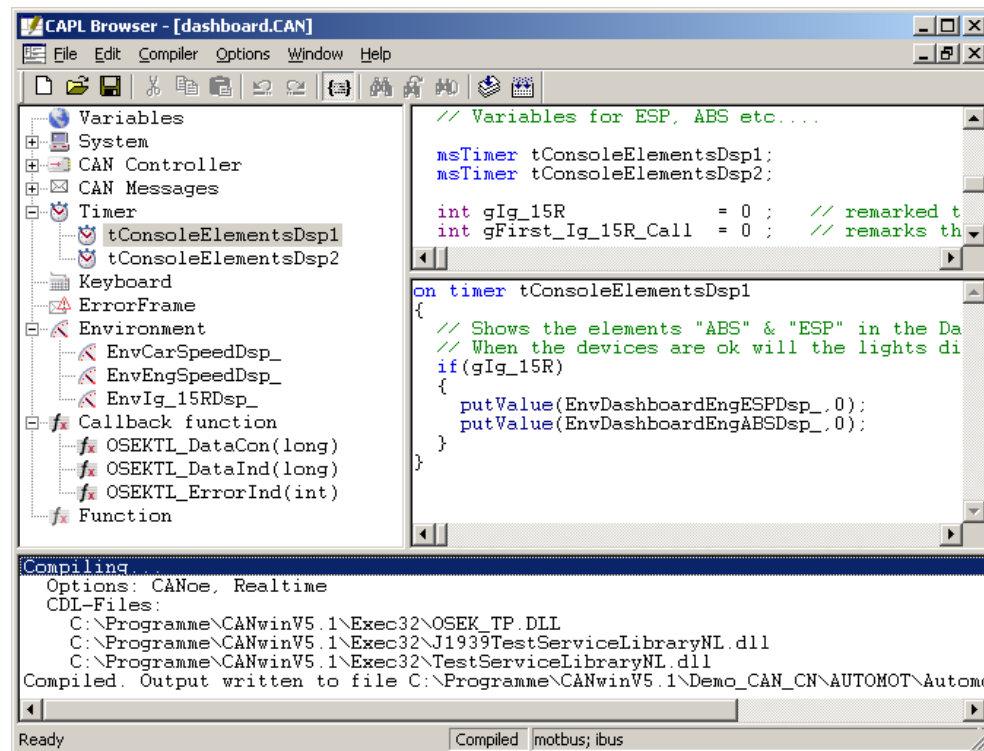
## □ 全局变量和局部变量

## □ 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  
{  
    ...           // 过程指令块  
}  
  
on message xxx  
{  
    ...           // 过程指令块  
}  
  
on key '1'  
{  
    ...           // 过程指令块  
}
```

```
My_function_1(Para_1, Para_2, ...)  
{  
    ...           // 函数体  
}  
  
...  
  
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));
```

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

## □ 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");  
}
```

```
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);  
}
```

# 定时器处理

## 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);  
}
```

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



1. Hello World
2. 报文计数
3. 信号监控

# 练习 1

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

## 练习 2

- 利用发生器模块周期性发送某一报文，例如每隔 200ms 发送一条 EngineData 报文。每当按下 a 键，在 Write Window 窗口输出一句话，例如 “XXX EngineData messages have sent.”
- 注： XXX 为已经发送的 EngineData 报文数量。

## 练习 3

- 不用发生器模块实现 Enginedata 报文的周期性发送。
- 每当按下 a 键时， EngineData 里面 EngSpeed 信号值为 2000 ； 当按下 b 键盘时， EngineData 里面 EngSpeed 信号值为 4000 ；
- 如果 EngineData 里面 EngSpeed 信号为 4000 ， 则发送 ABSData 报文， 同时在 Write Window 输出 “ Warning!”
- 当按下 c 键时， 停止 EngineData 报文发送。

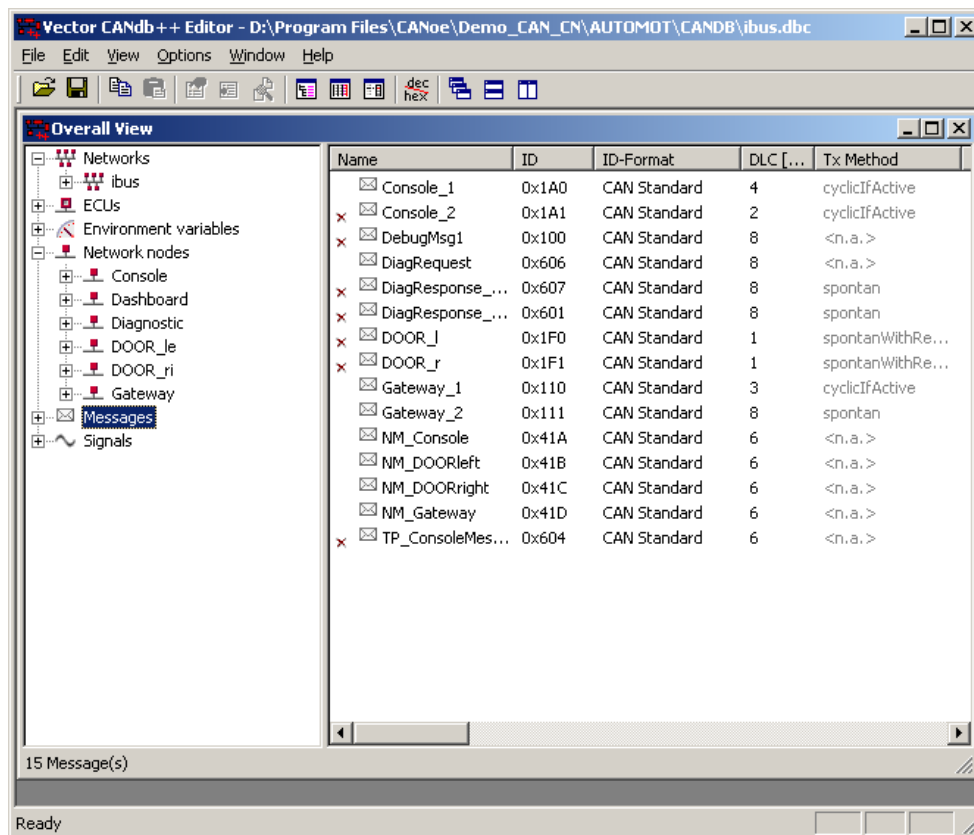
# 欢迎进入 CANdb++ Editor 的世界

## □ DBC 文件编辑工具

### □ 启动 CANoe

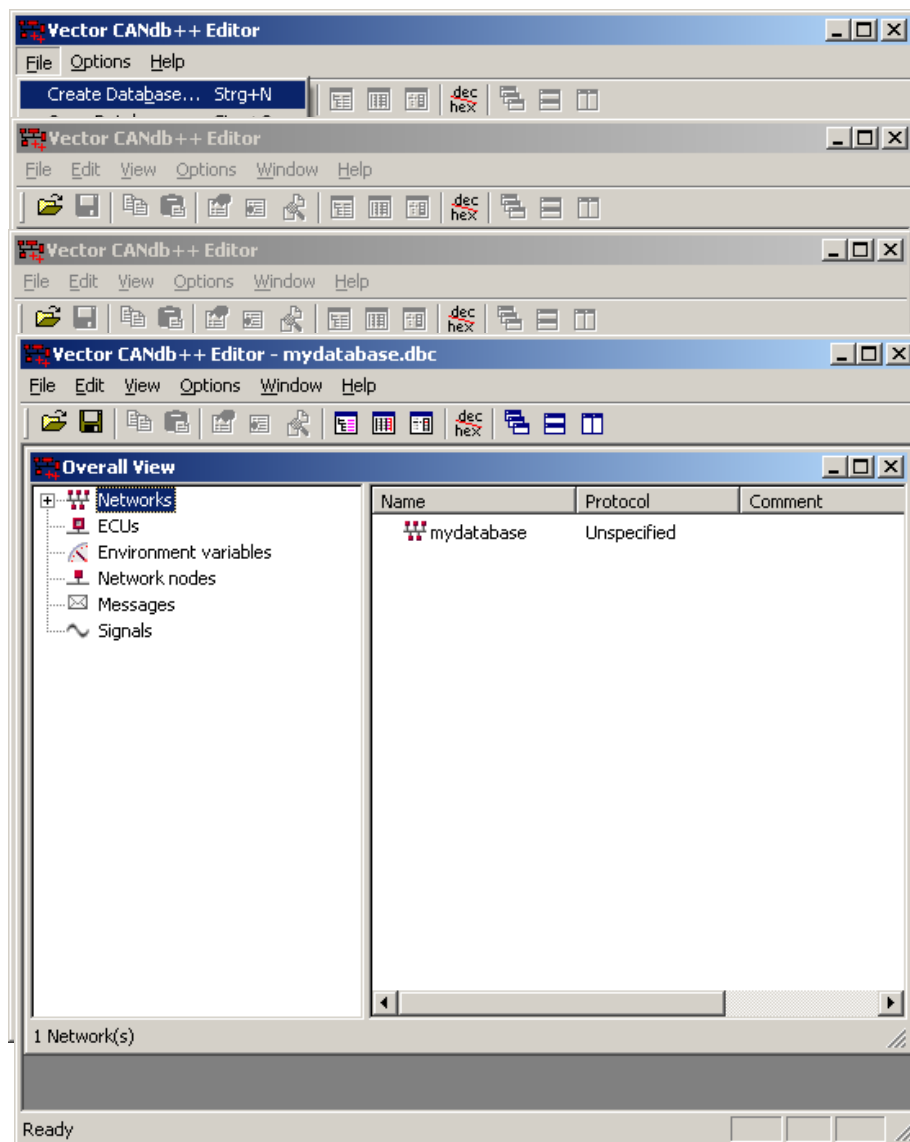
□ File->Open CANdb Editor

□ 点击 



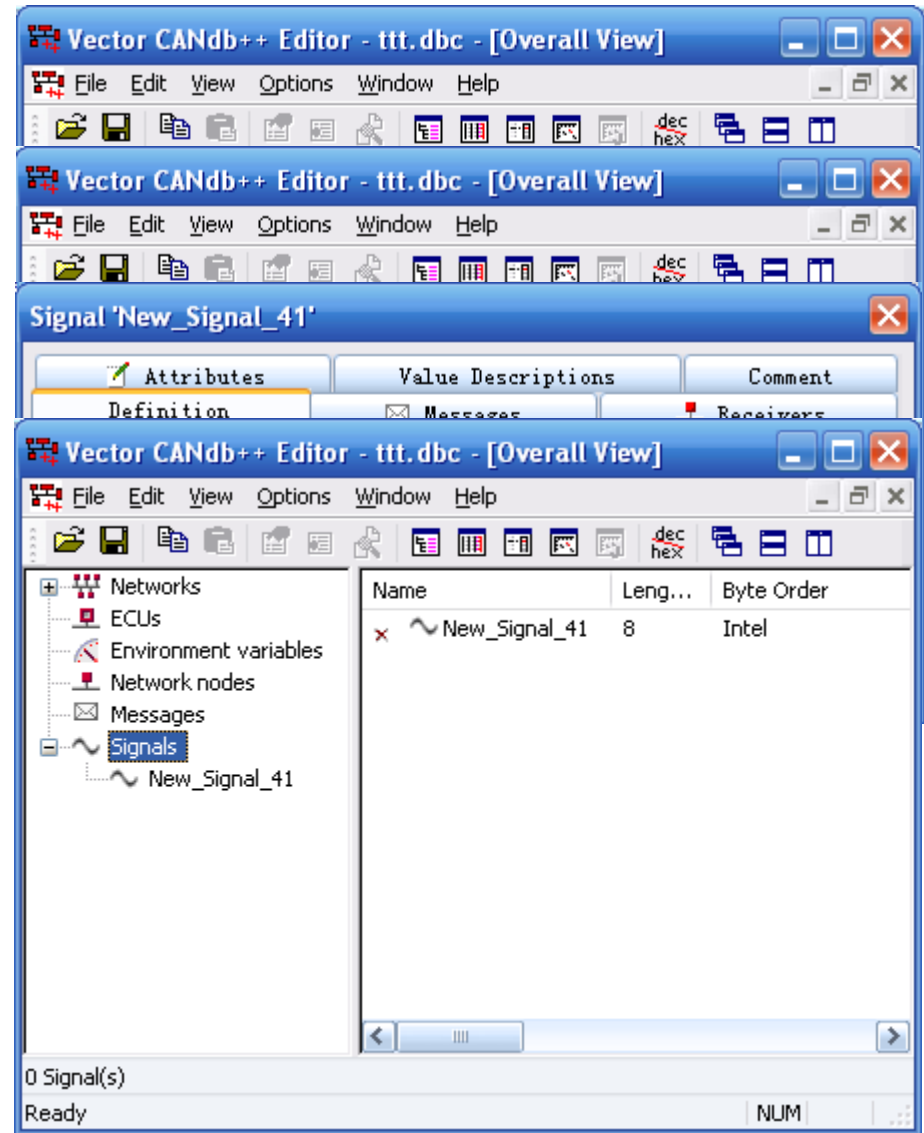
# 创建一个新的 CAN 数据库

- 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）

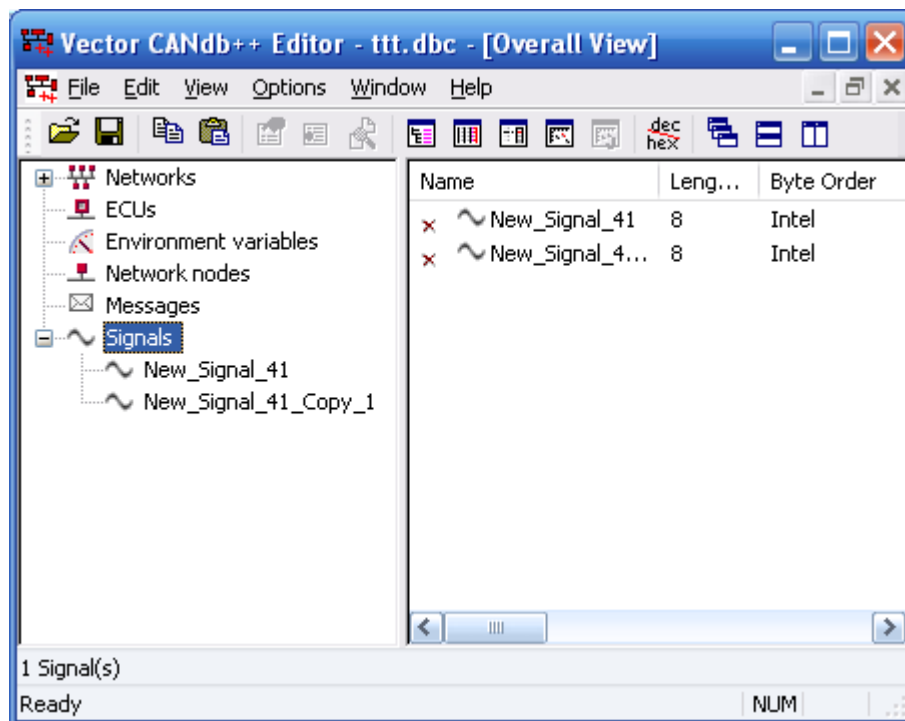
- ❑ 在 Overview 窗口左边  
选择所需创建对象的类型
- ❑ 右键点击对象类型，  
在快捷菜单中选择 New...
- ❑ 使用配置对话框设置  
所创建对象的系统参数值
- ❑ 点击 [ 确定 ] 按钮，  
一个新对象便创建完毕



# 复制已有对象

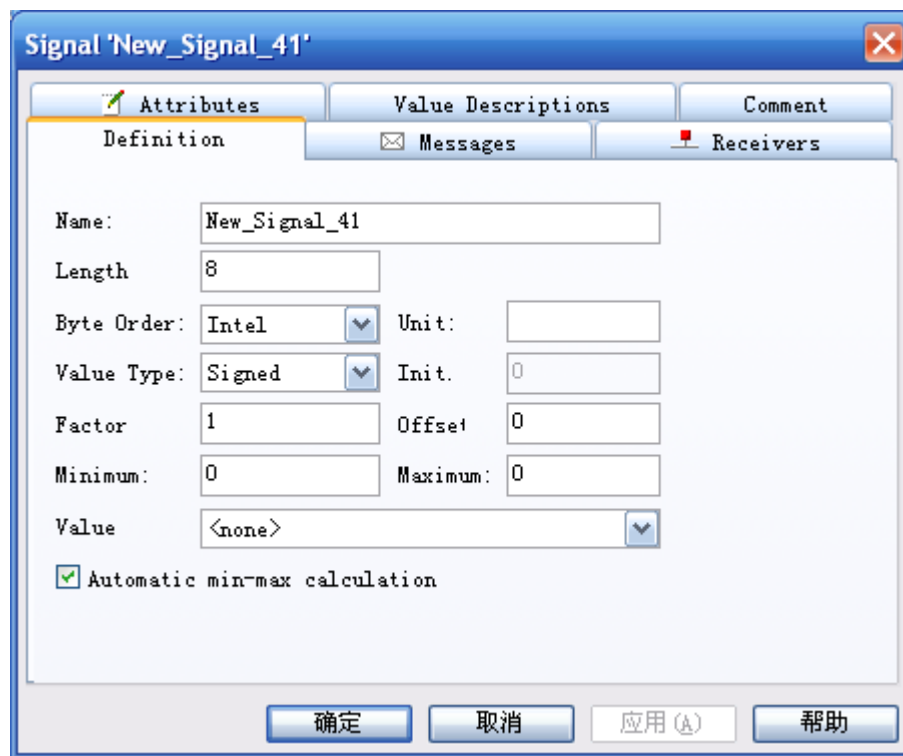
## □ Copy-Paste

- 选择已有对象 Ctrl+c
- 选择对象类型 Ctrl+v





## □ 直接双击



Signal 'New\_Signal\_41'

Attributes Value Descriptions Comment  
Definition Messages Receivers

Name: New\_Signal\_41

Length: 8

Byte Order: Intel Unit:

Value Type: Signed Init. 0

Factor: 1 Offset: 0

Minimum: 0 Maximum: 0

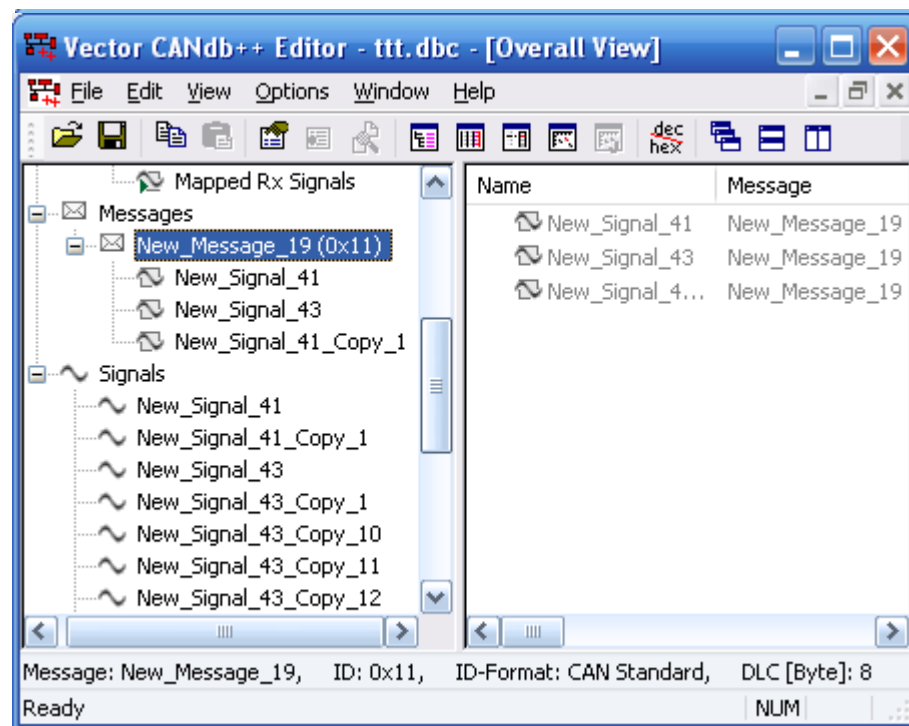
Value: <none>

☒ 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

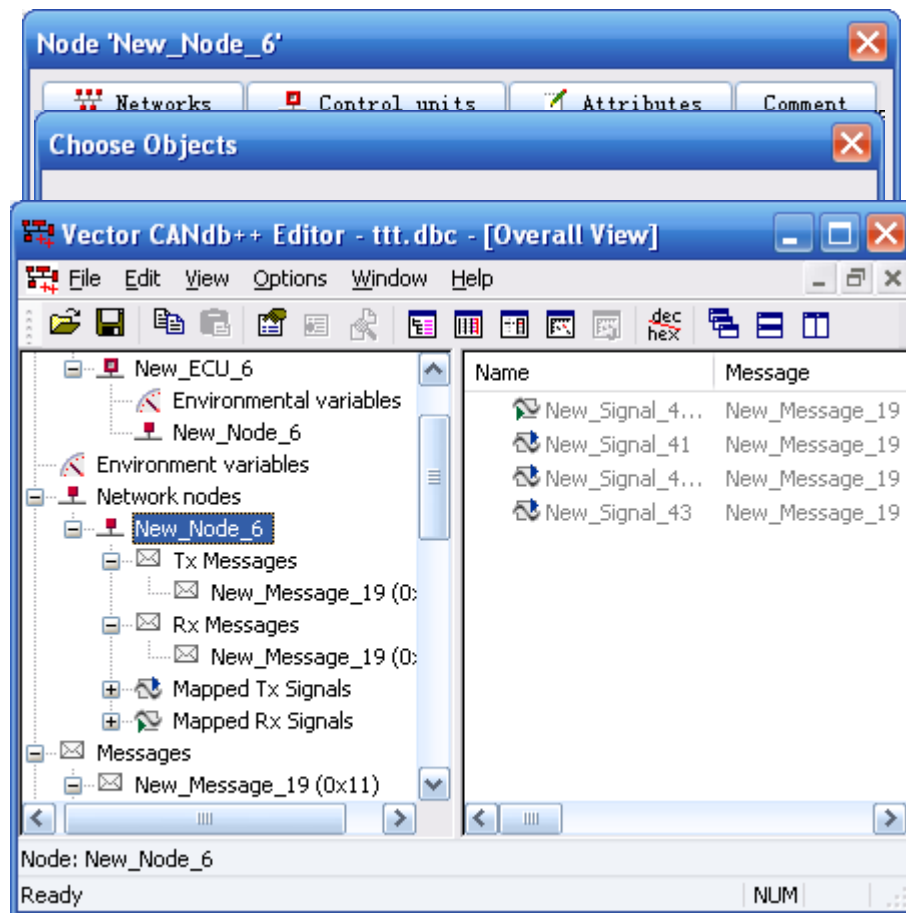
#### □ 双击节点，

选择 Mapped Rx Sig. 页签

#### □ 点击 Add... ， 选择接收信号

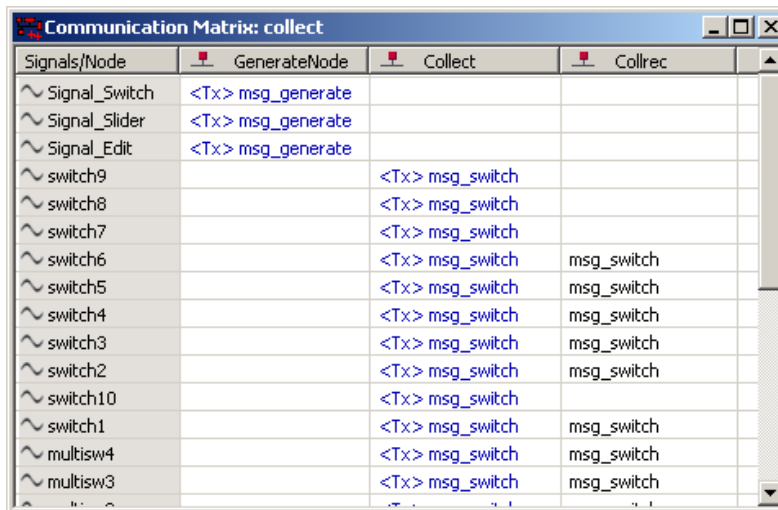
点击 OK

#### □ 点击确定



## □ View->Communication Matrix...

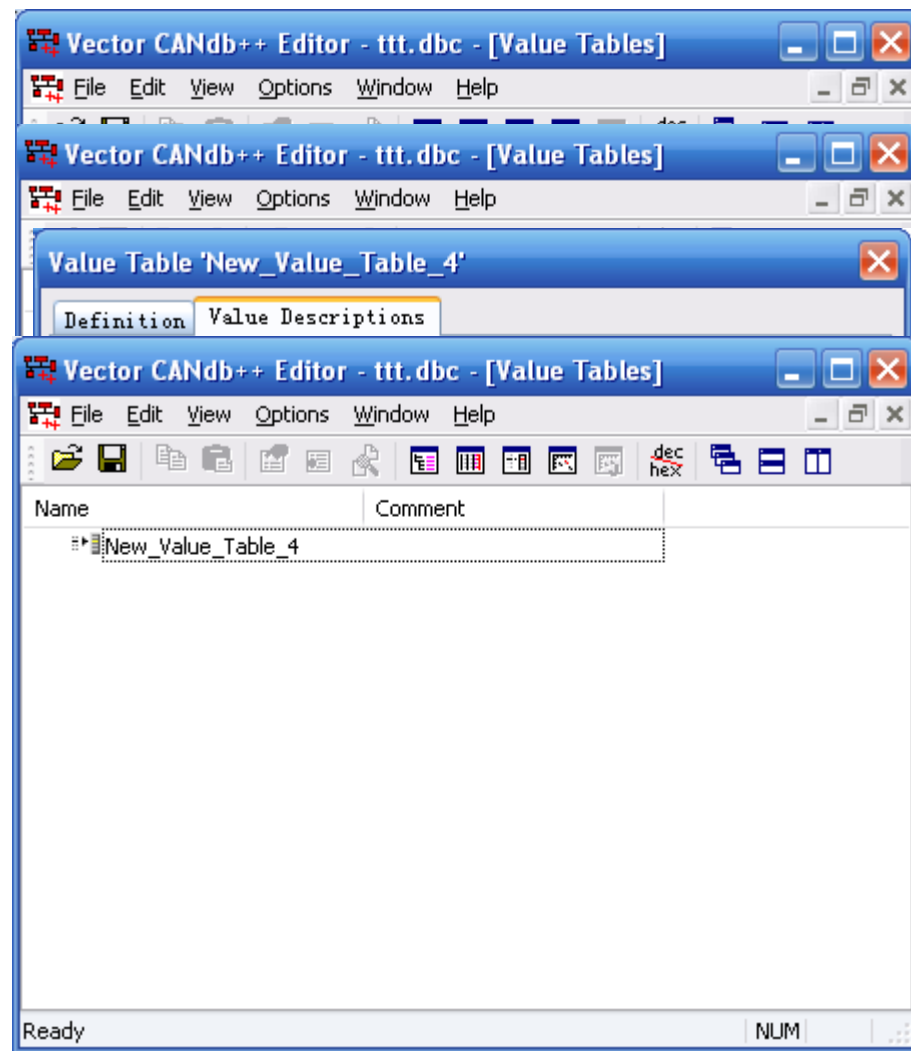
- 显示信号、消息、及网络节点的关系
- 以信号为行，网络节点为列
- 消息名显示于表中，对应了包含的信号与发送 / 接收的节点



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)

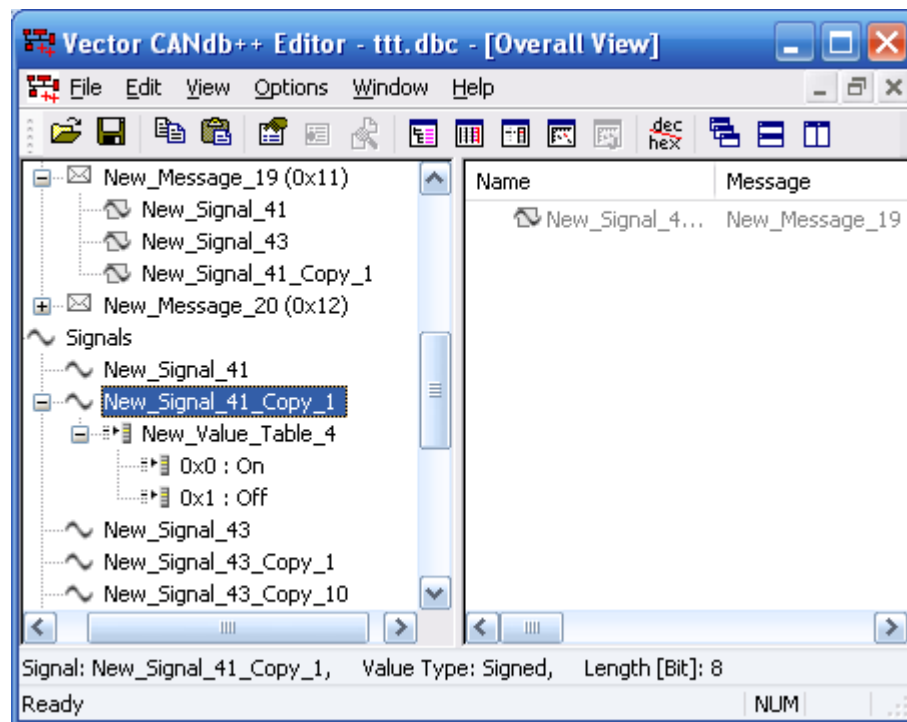
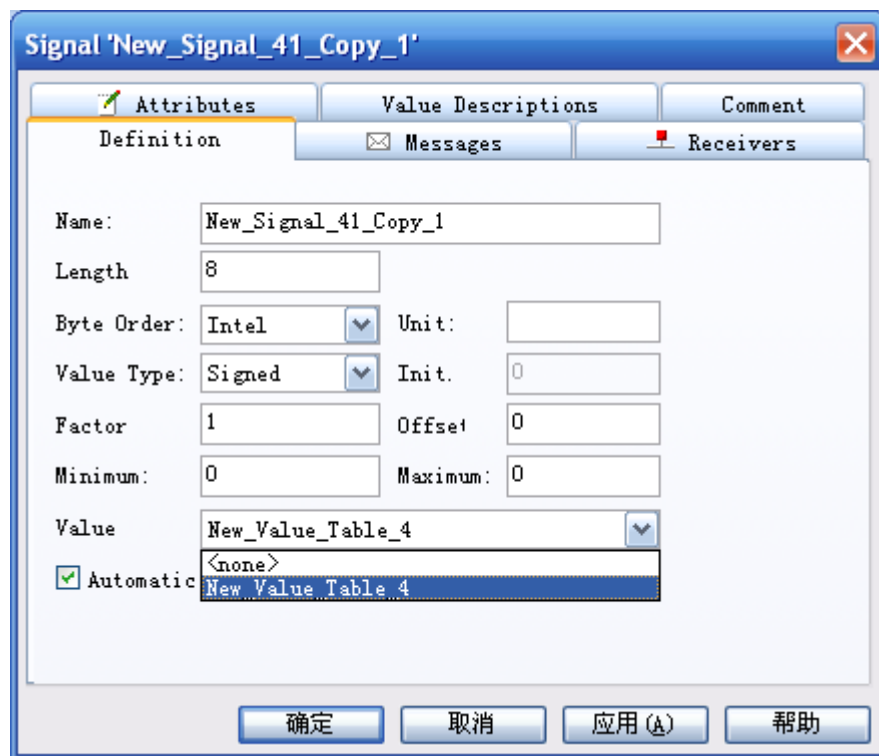
- 新建数值表
  - View->Value Tables
  - 右键点击空白处，  
选择 New...
  - 在对话框中输入数值，  
点击确定
  - 新的数值表创建完成



# 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



## □ Vector Tool Chain Attributes

### □ General

- Manufacturer

### □ Interaction Layer

- GenMsgCycleTime

### □ Transport Protocol and Diagnostics

- DiagRequest, DiaResponse

### □ Network Management

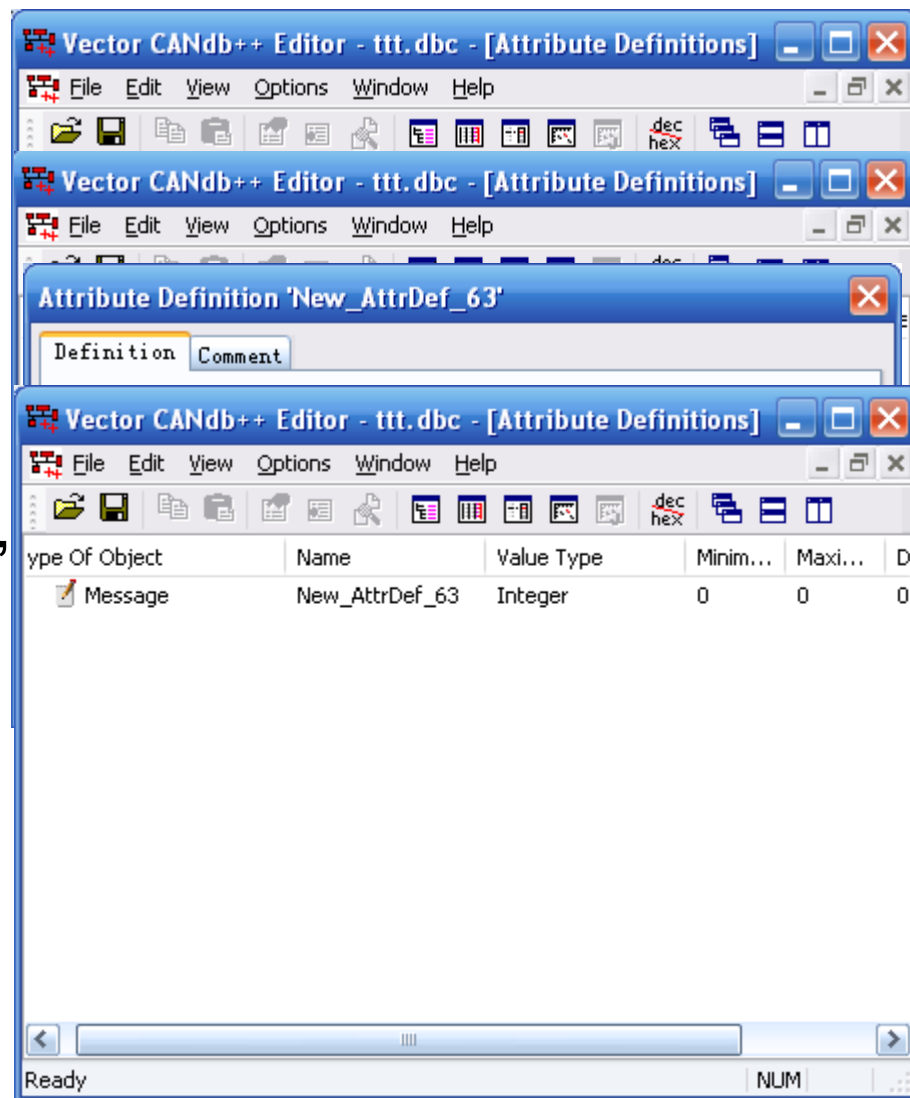
- NmBaseAddress, NmStationAdress

### □ Tool specific

- BusType

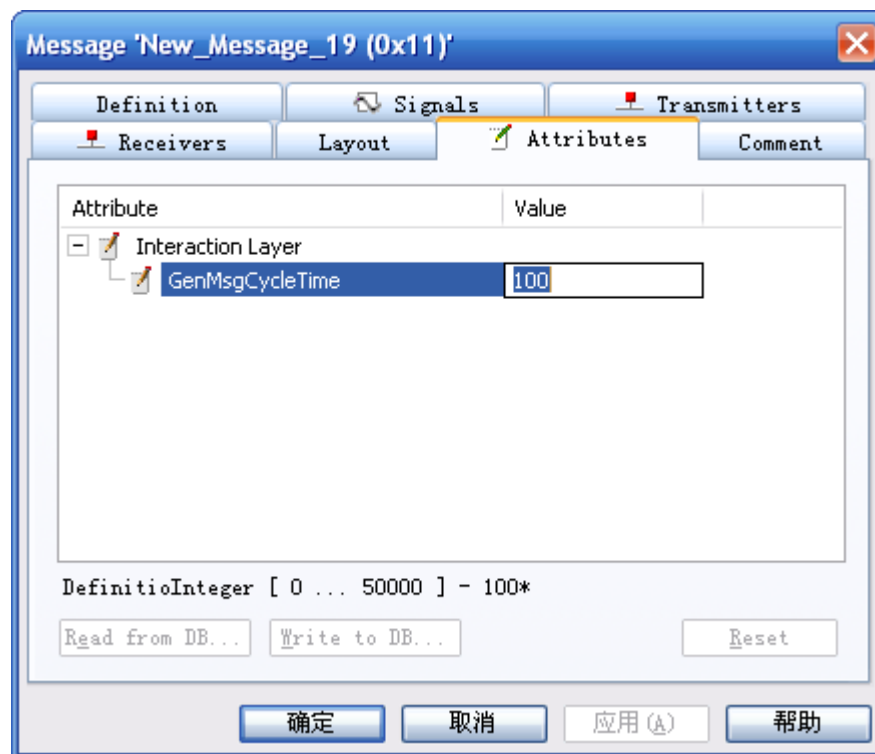
# 新建属性

- View->Attribute Definitions
- 右键点击空白处，  
选择 New...
- 在对话框中输入相关参数，  
点击确定
- 新的属性创建完成

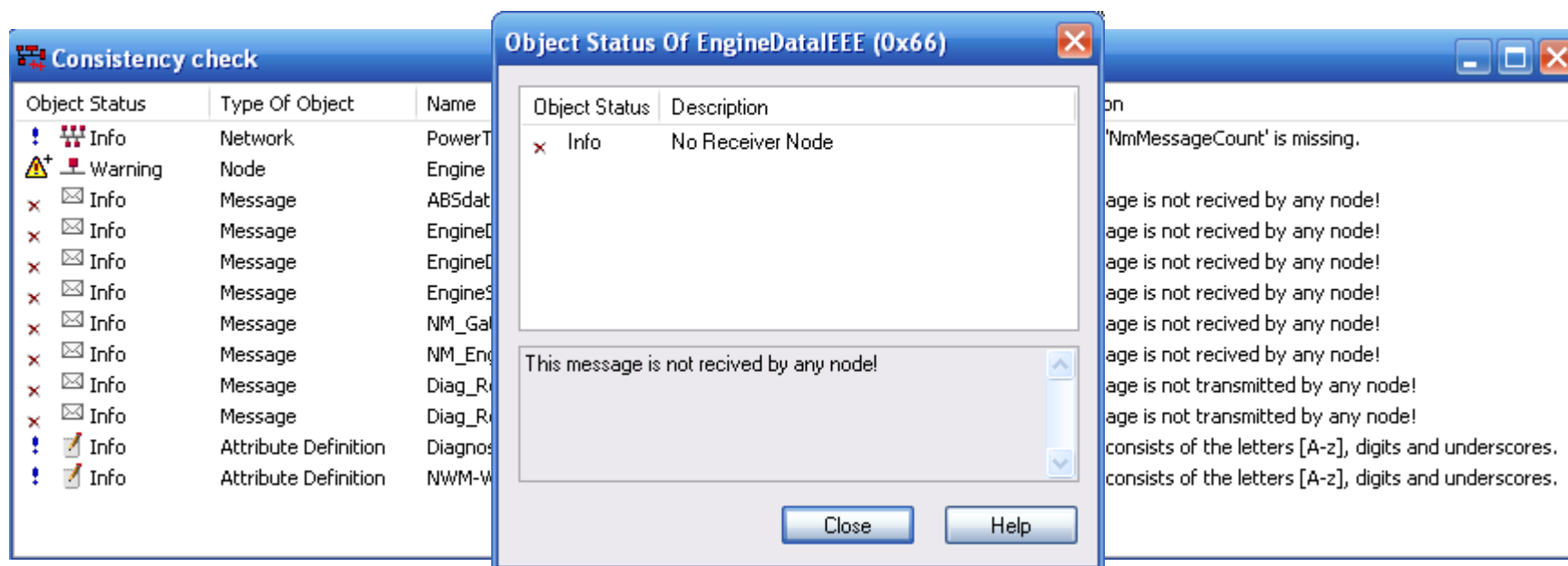




## □ 双击对象



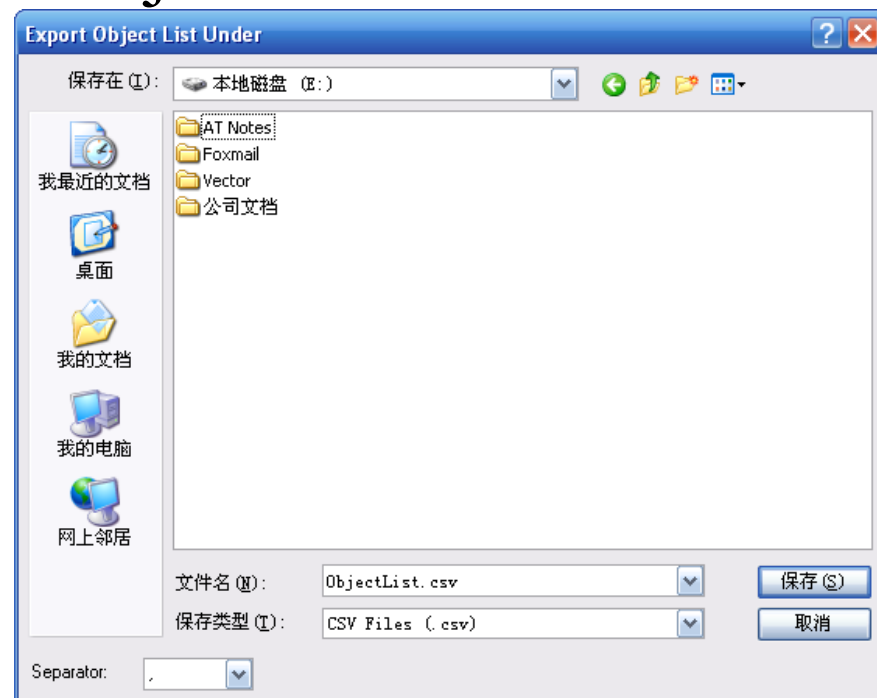
## □ File-> Consistency Check



## □ 选择需要导出的对象

□ 信号，报文，节点， ECU 或网络

## □ File->Export->Export List of Objects



# 欢迎进入 Panel Editor&Panel Designer 的世界

## □ Panel Editor

- 传统的面板编辑器

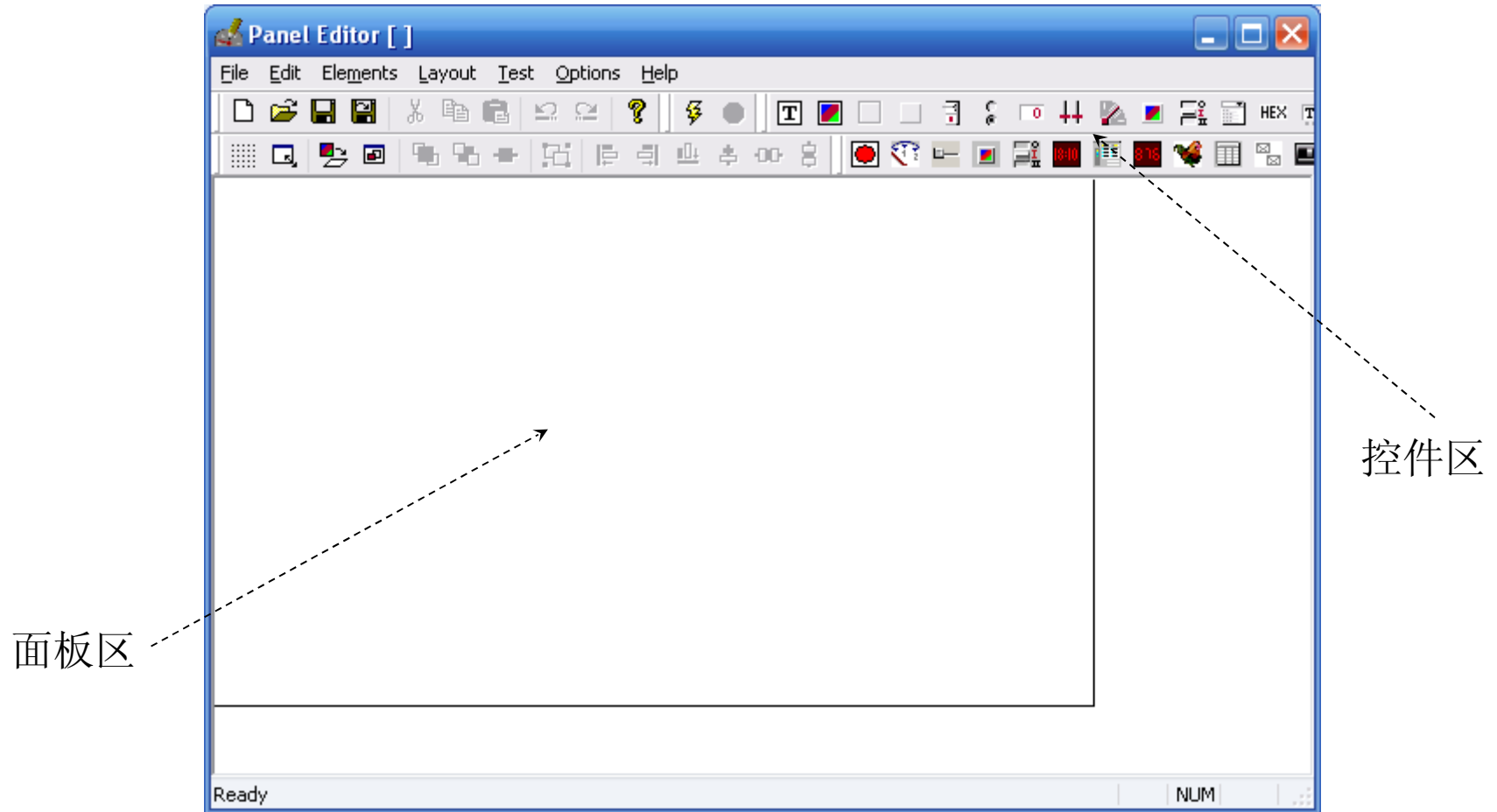
- File->Open Panel Editor

## □ Panel Designer

- 新的面板编辑器

- File->Open Panel Designer

# Panel Editor

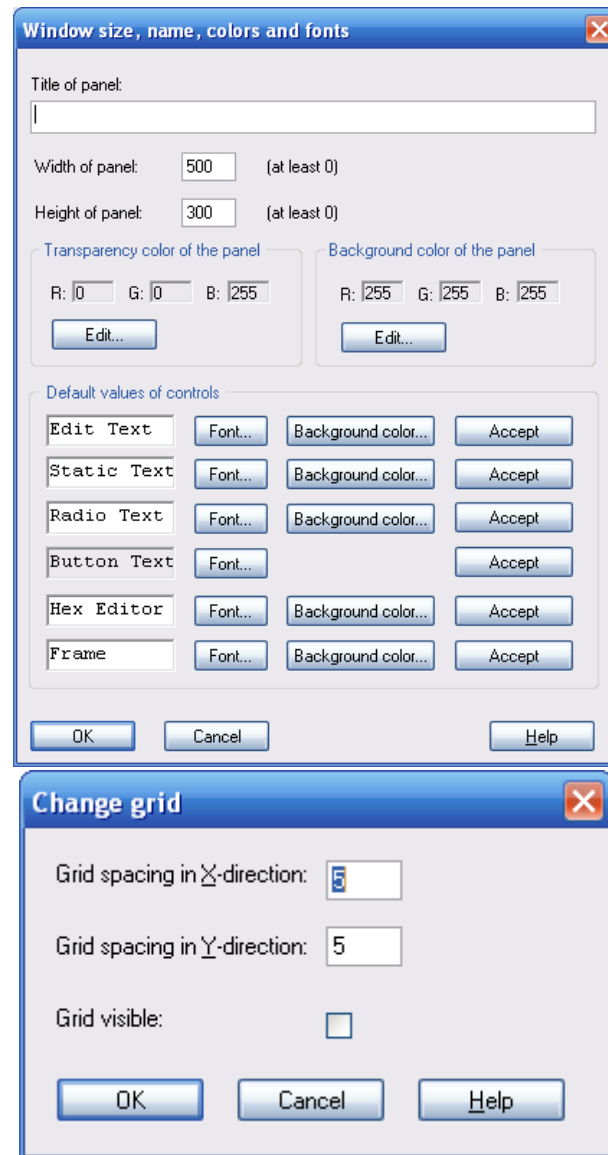


## □ Options->Window setting

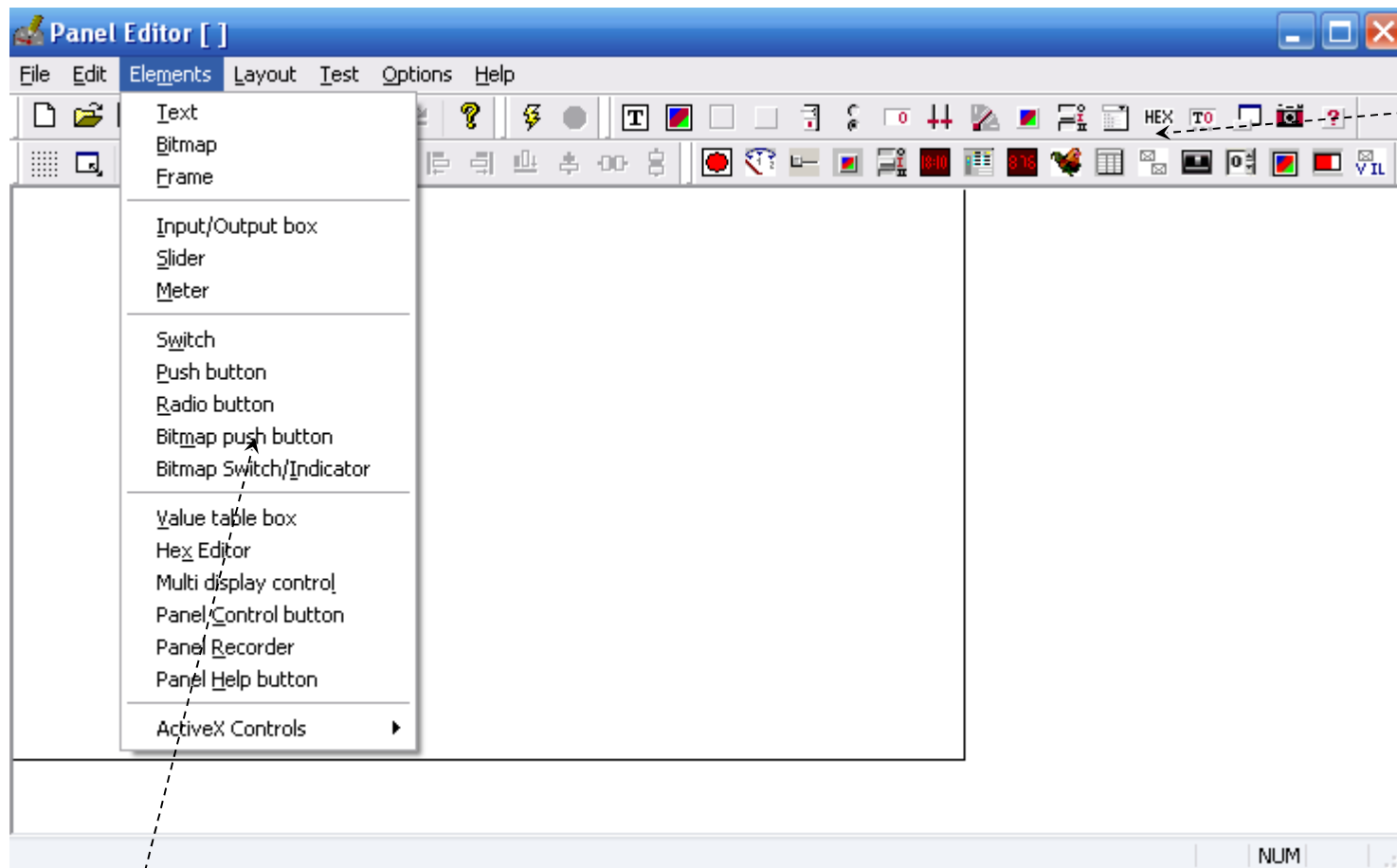
- 定义面板名称
- 面板尺寸
- 背景颜色
- 透明色
- 控件的缺省字体和颜色

## □ Options->Change grid

- 网格大小
- 网格可视



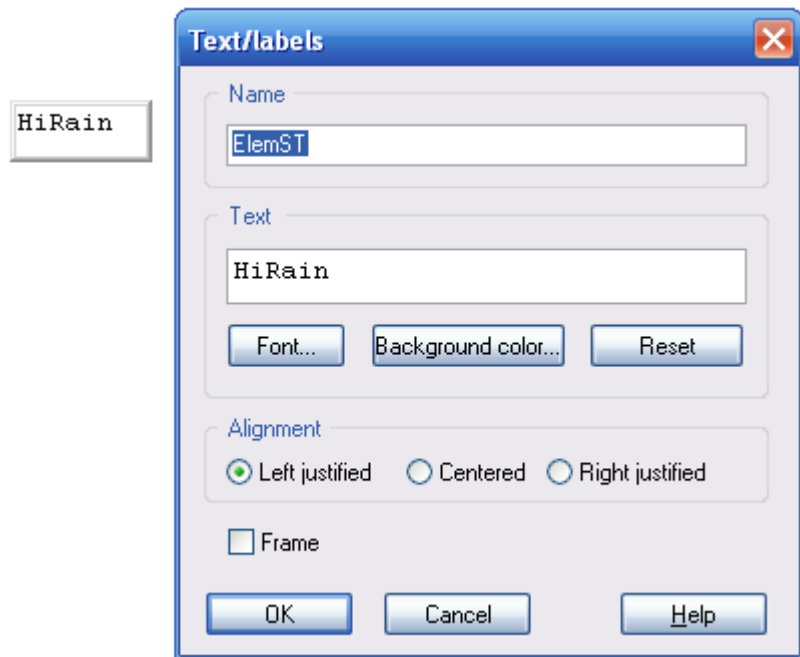
# 控件列表



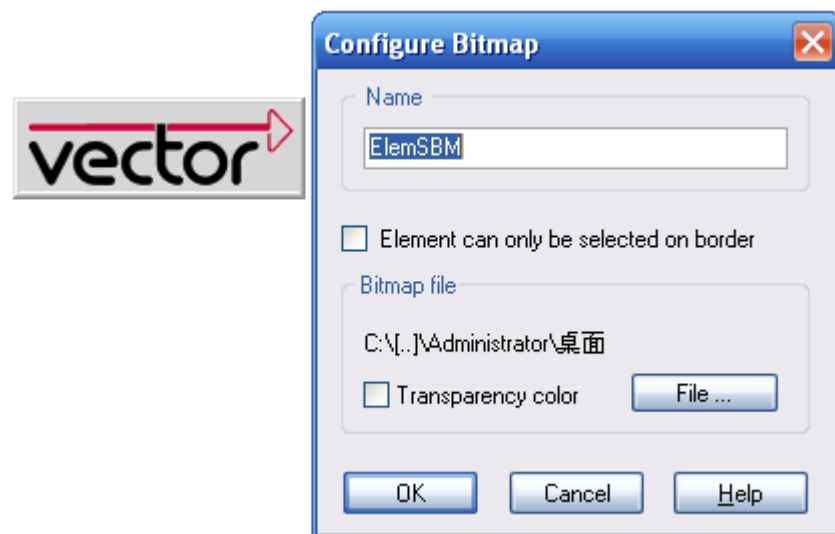
图标

名称

## □ 文本



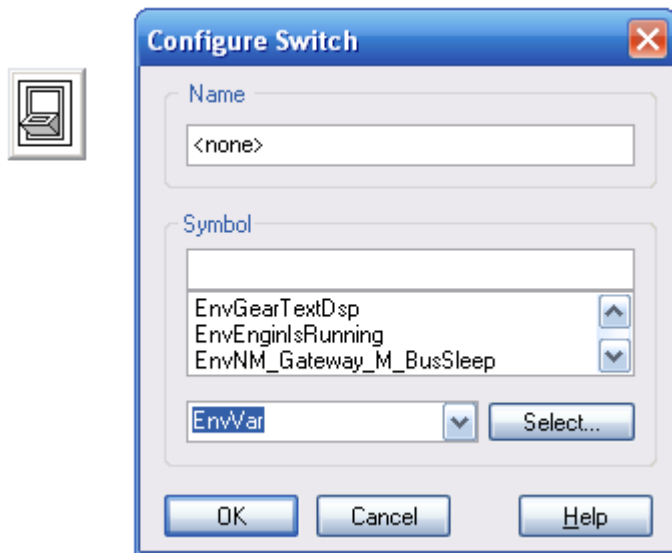
## □ 位图



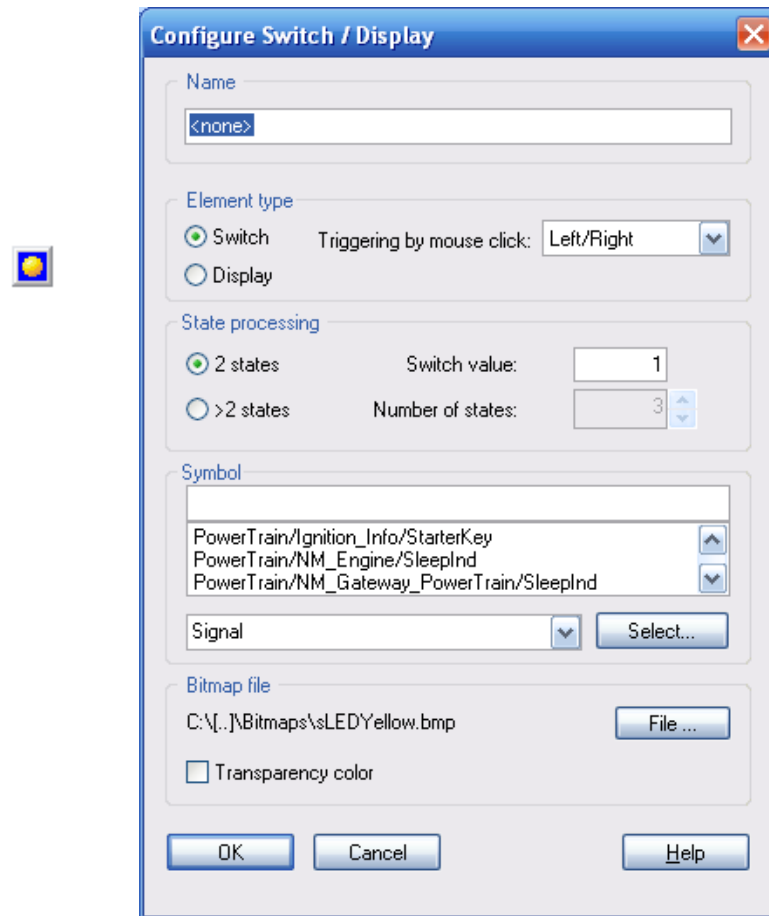


# 开关与多态位图

## □ 开关

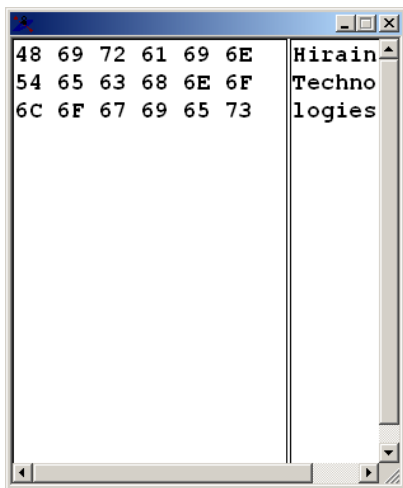


## □ 多态位图



# 其它常见的控件

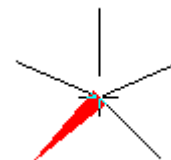
## □ 十六进制



## □ 滑动条



## □ 仪表



## □ 输入 / 输出显示

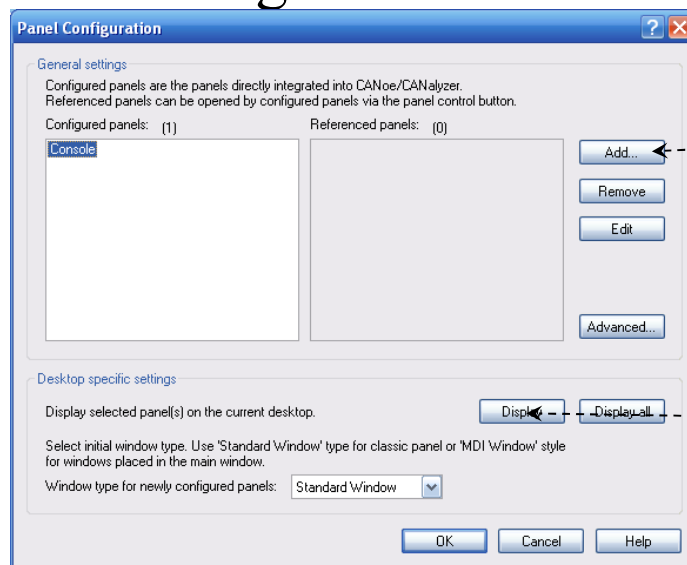


## □ 保存面板

□ File->Save

## □ 使用面板（CANoe）

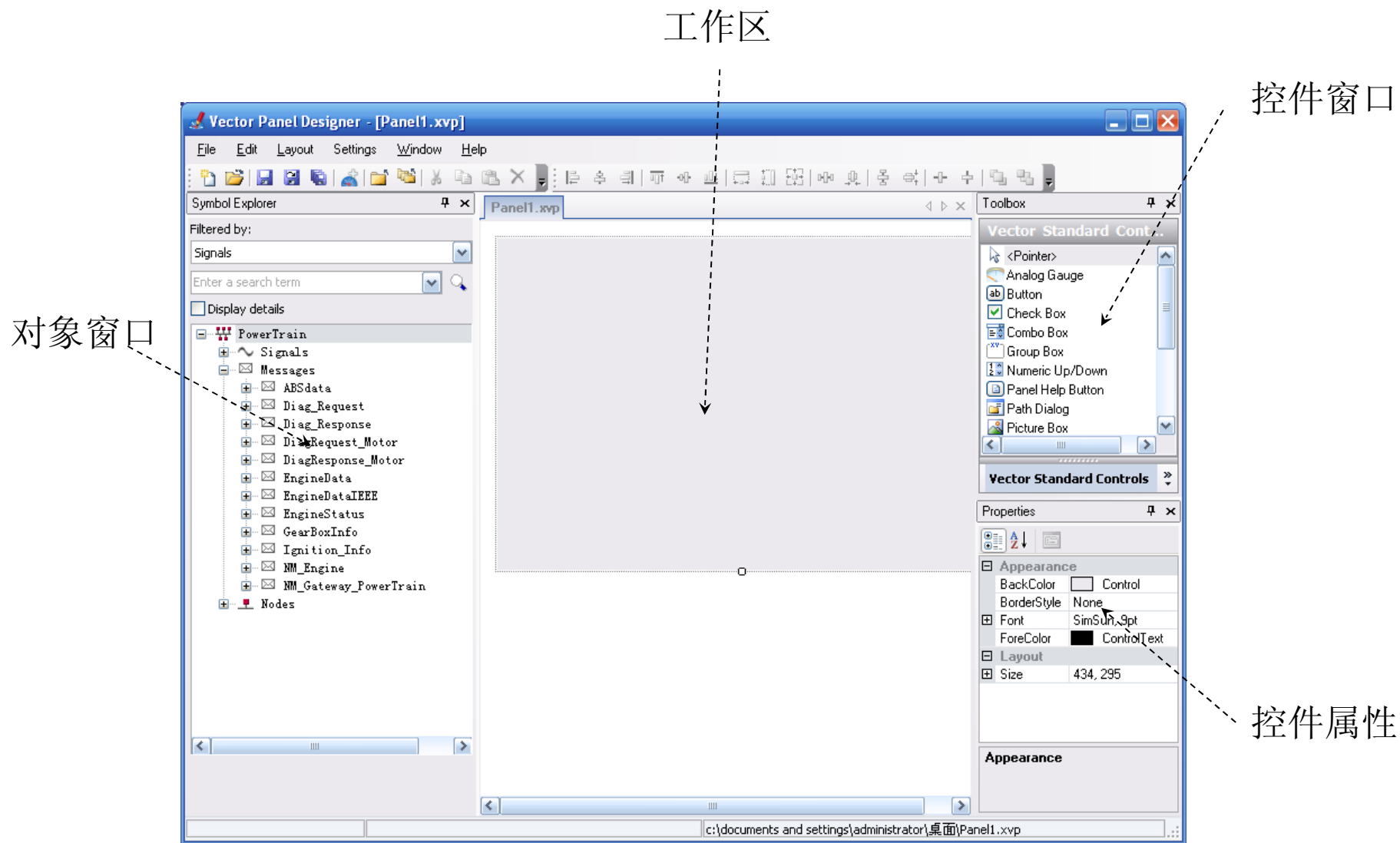
□ Configuration->Panel Configuration



添加面板

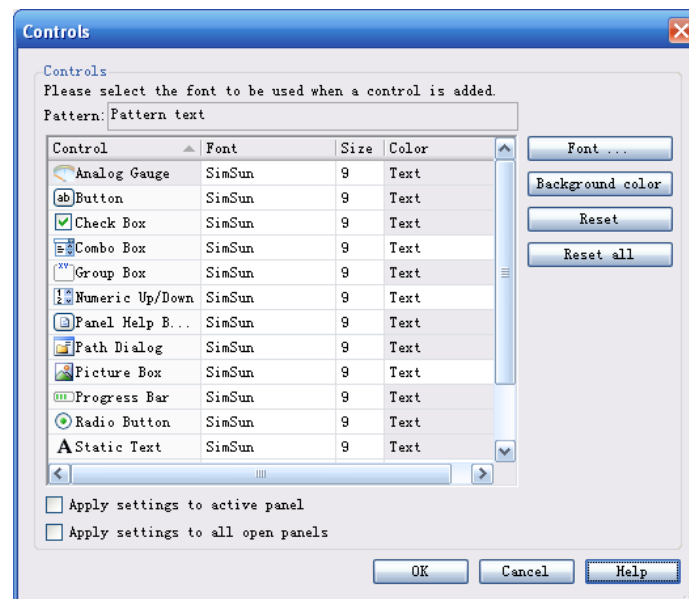
显示面板

# Panel Designer



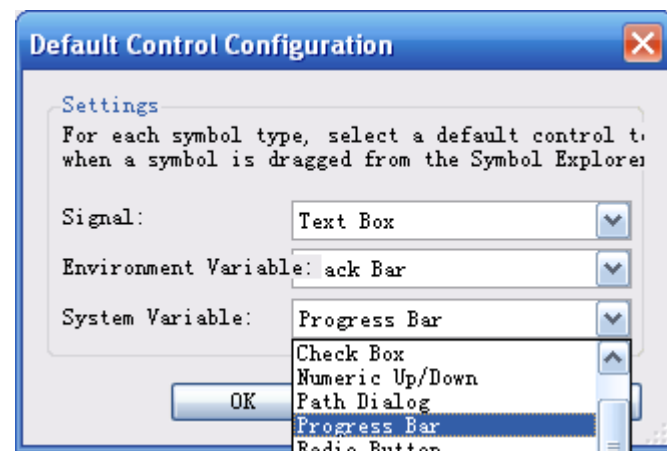
## □ Settings->Controls Properties

□ 设置控件的字体、颜色和字号

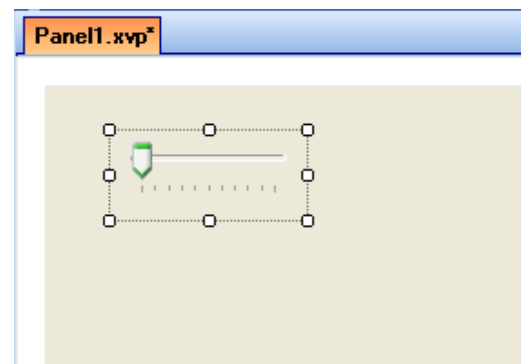
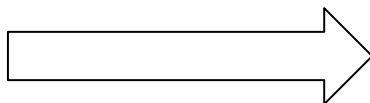
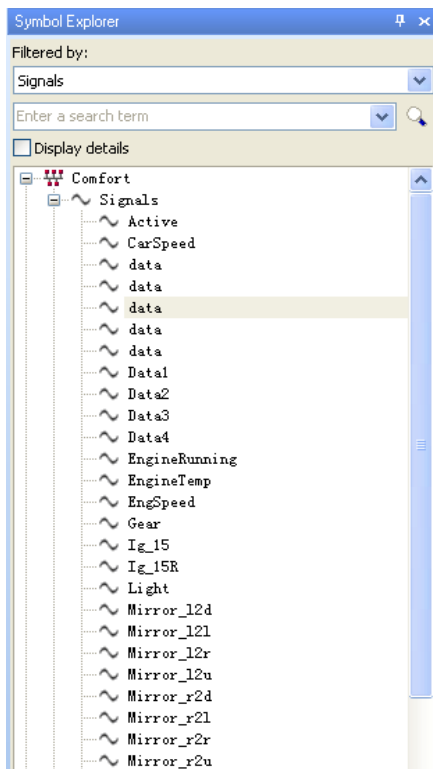


## □ Settings->Symbol Explorer

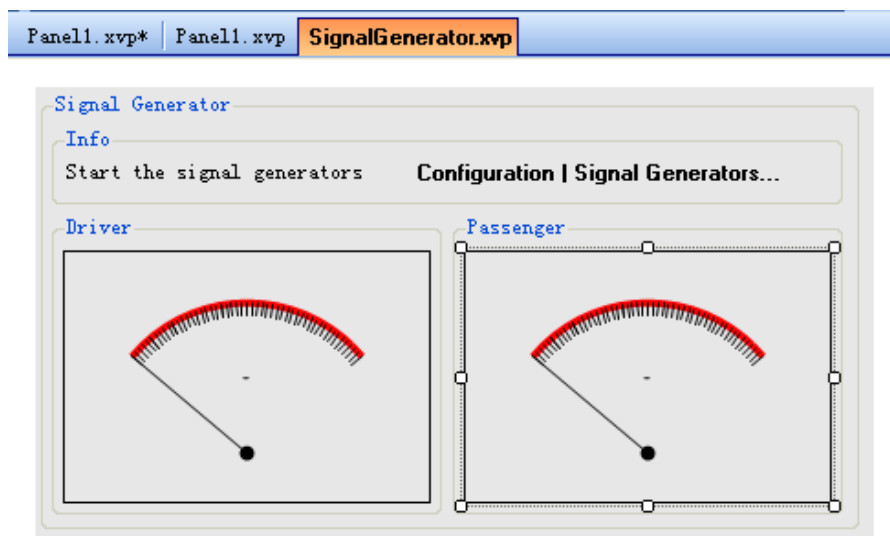
□ 设置信号、环境变量和系统变量  
对应的默认控件



- 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

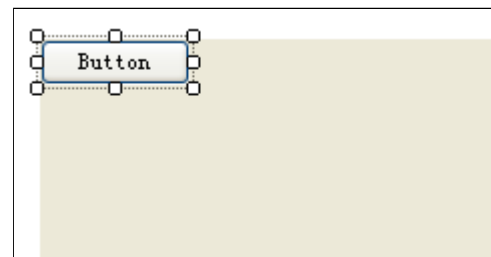
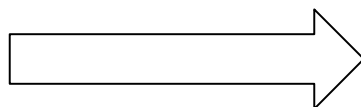
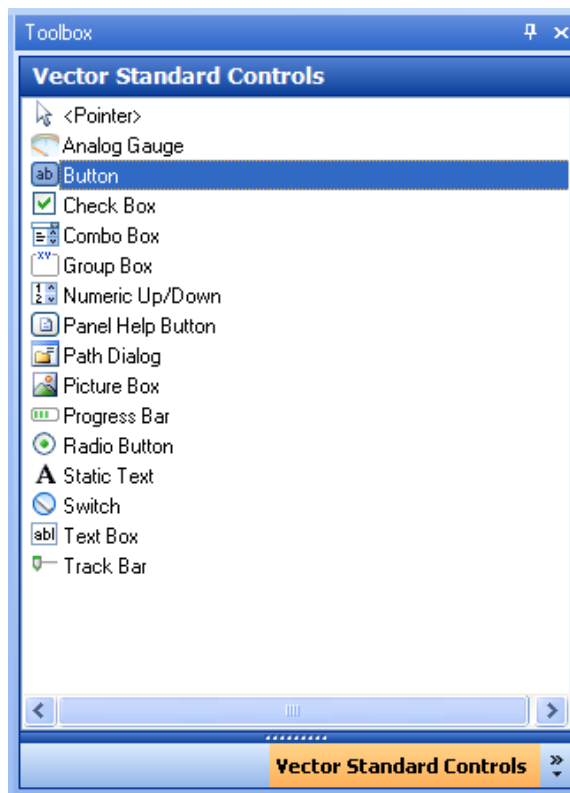


- 创建面板
- 支持同时编辑多个面板



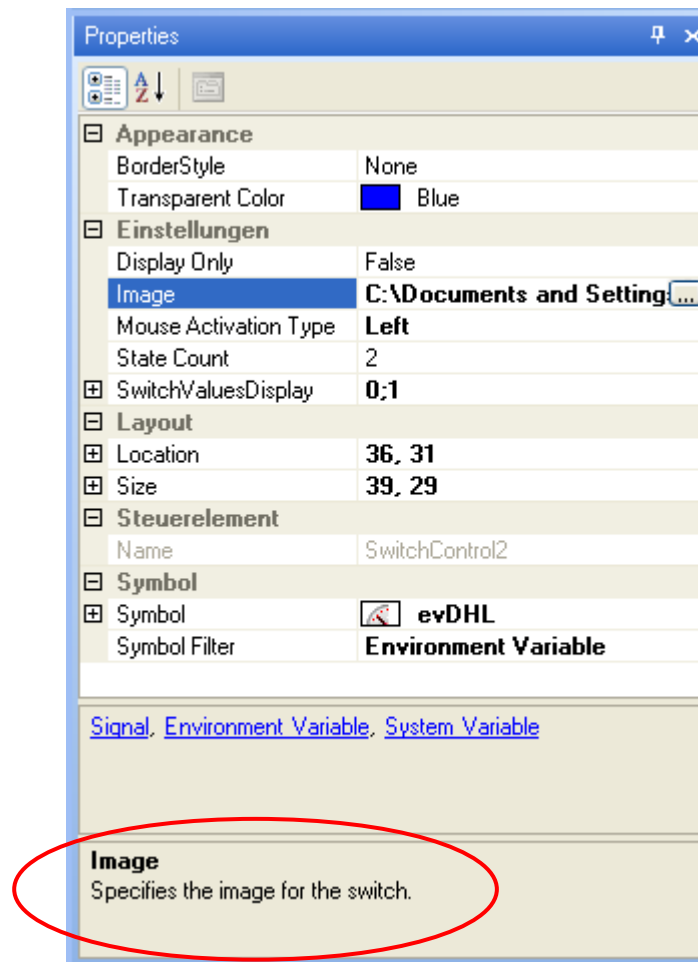
# 控件窗口

- 显示控件
- 双击在工作区产生控件

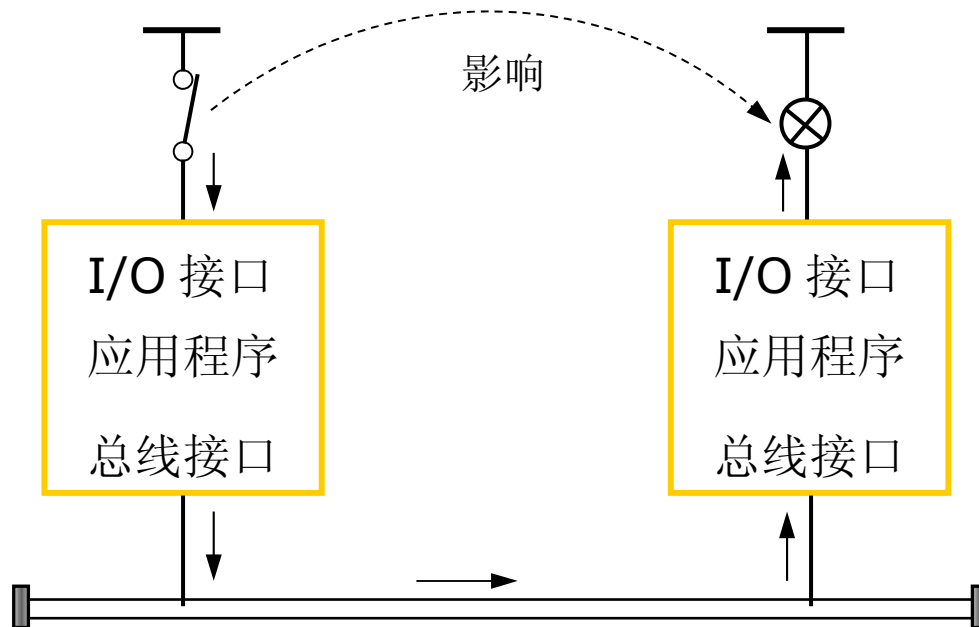




- 显示选中控件的相关设置
- 点击某项设置后会在下方出现相关说明



# 练习 1

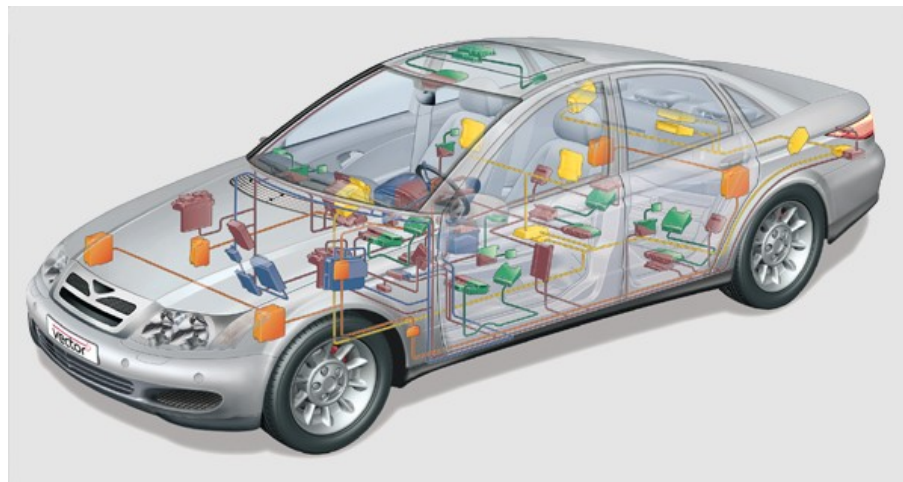


# 练习 2



- ❑ 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

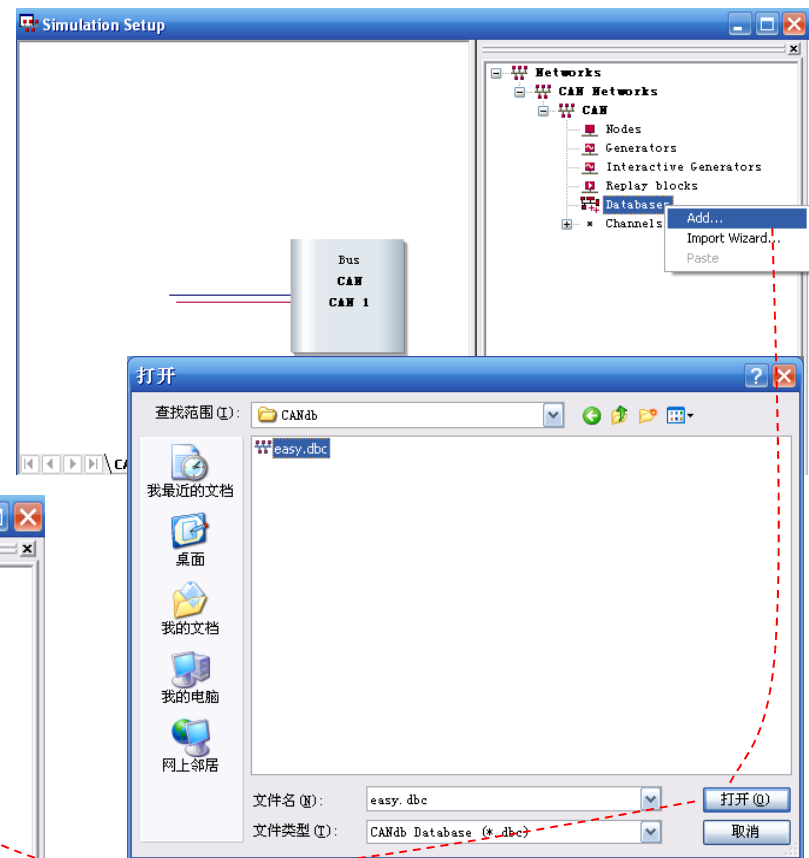
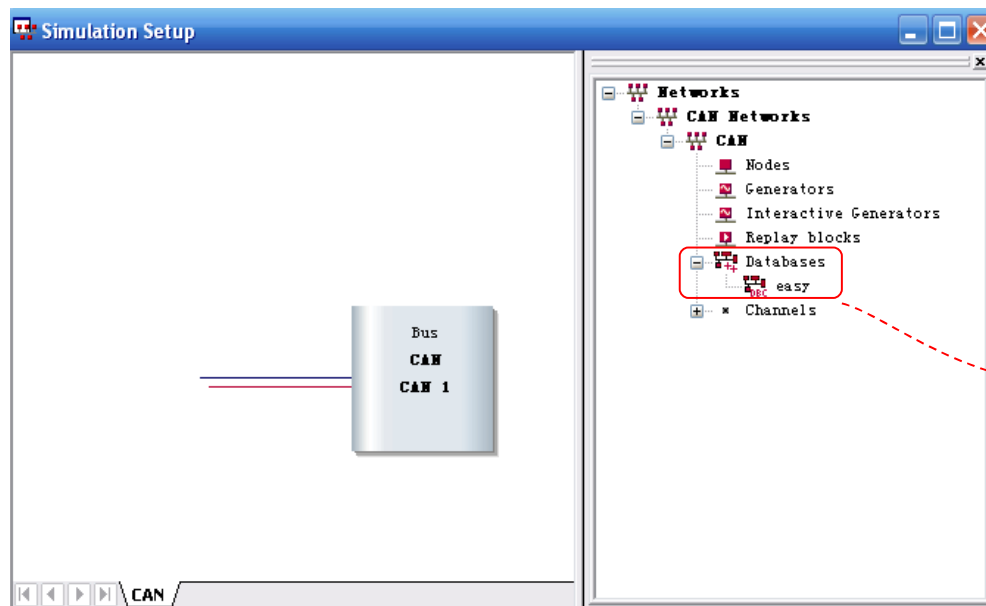
# 谢谢！



# 蒙太奇 ( 1 )

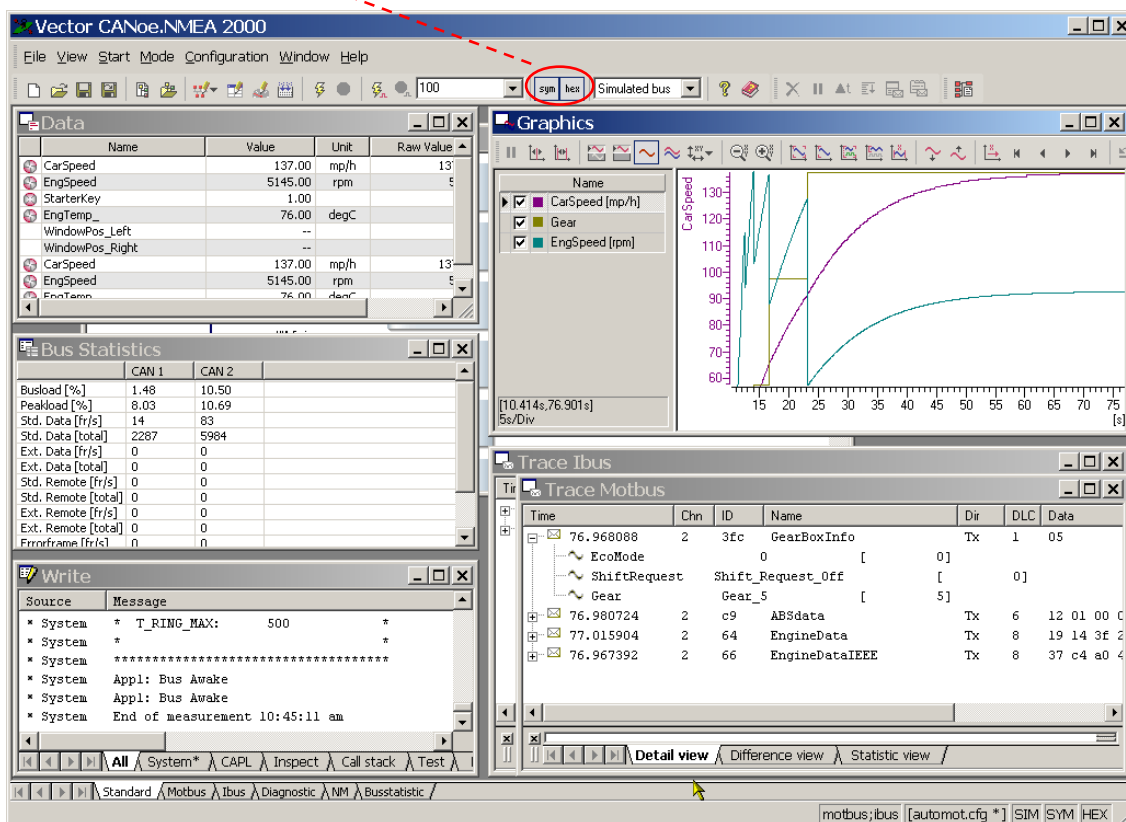
□ 在 CANoe 中添加数据库

□ View->Simulation Setup

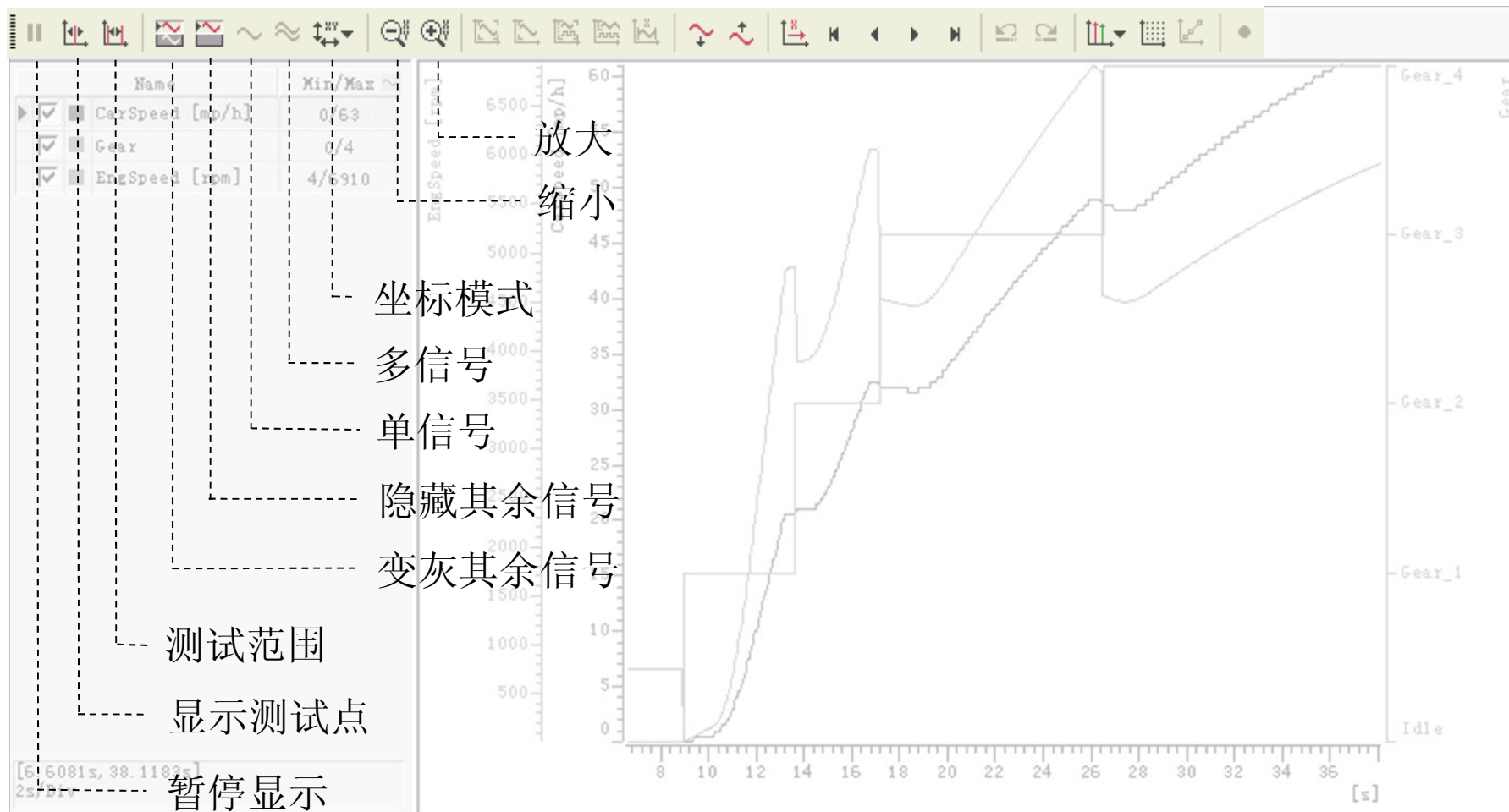


## □ 符号化显示与十进制 / 十六进制切换

*Global switches: Hex/Dec and Numeric/Symbolic toggles*

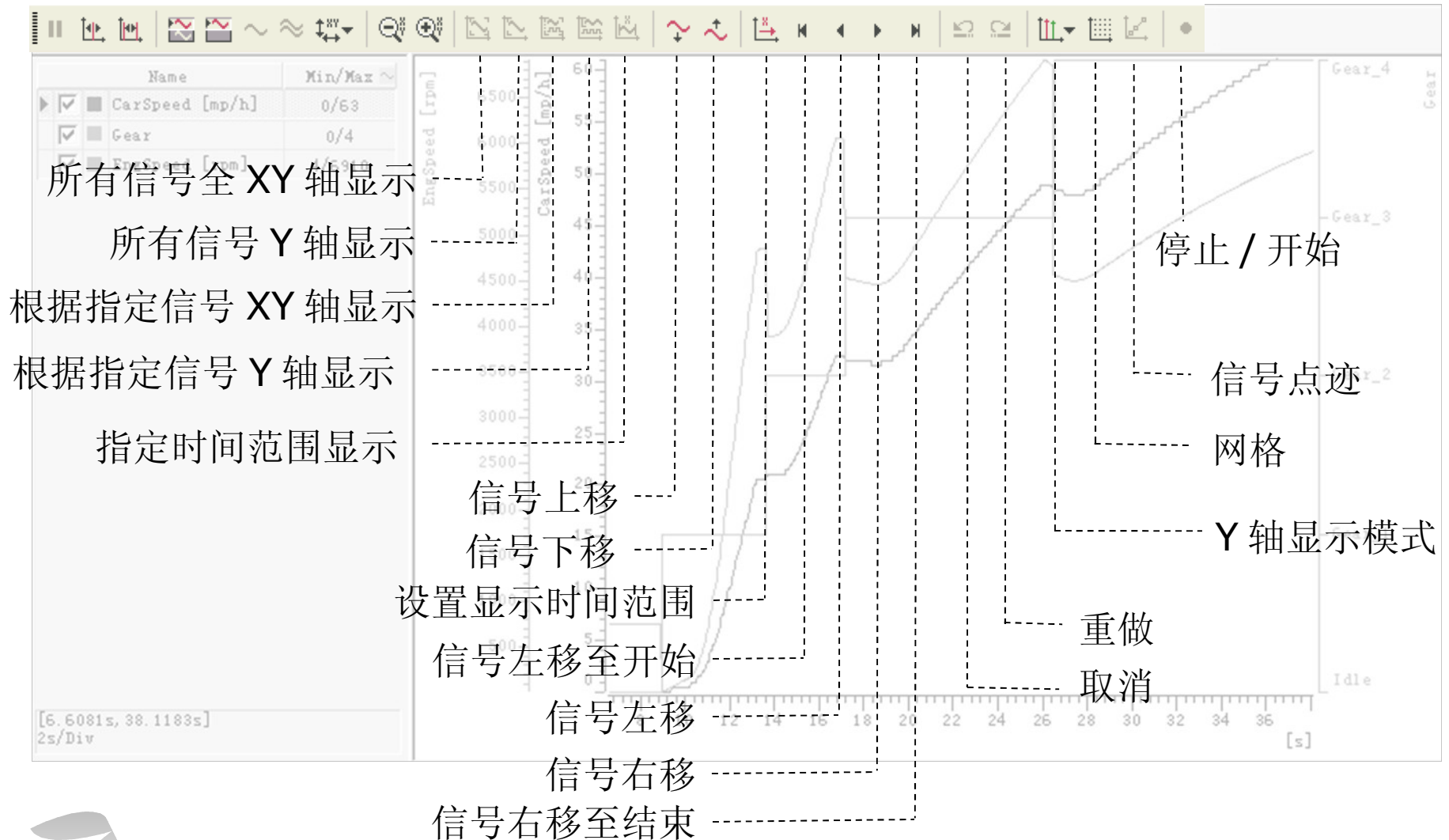


# 蒙太奇 ( 3 )





# 蒙太奇 ( 3 )



# 蒙太奇 ( 4 )

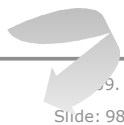
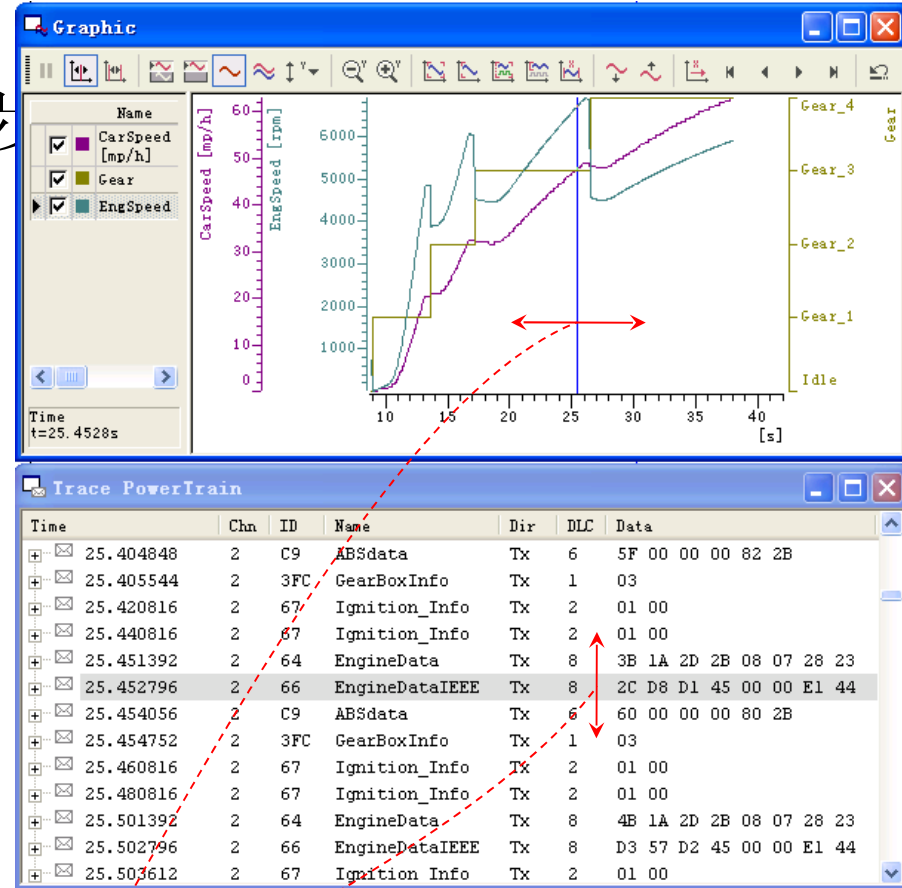
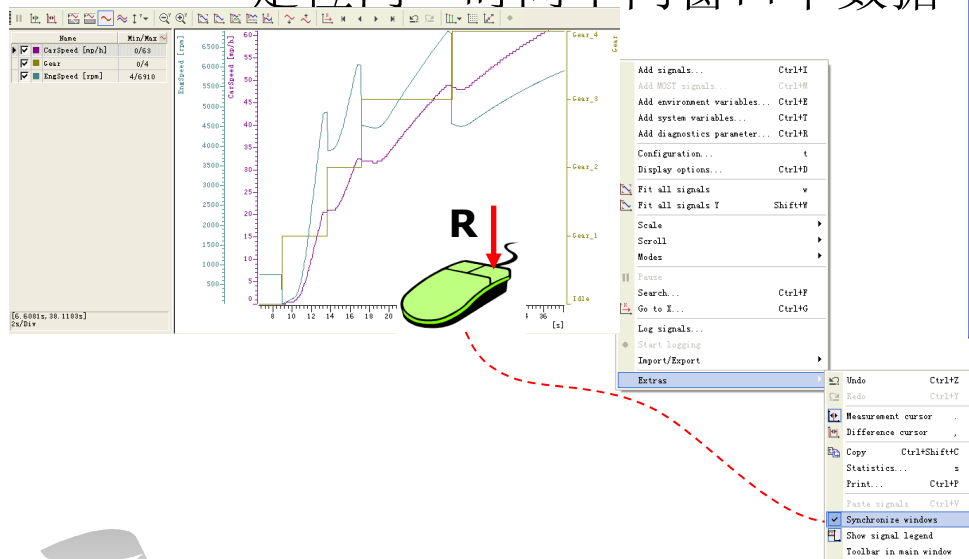
## □ Graphics 与 Trace 窗口同步

### □ 图形窗口

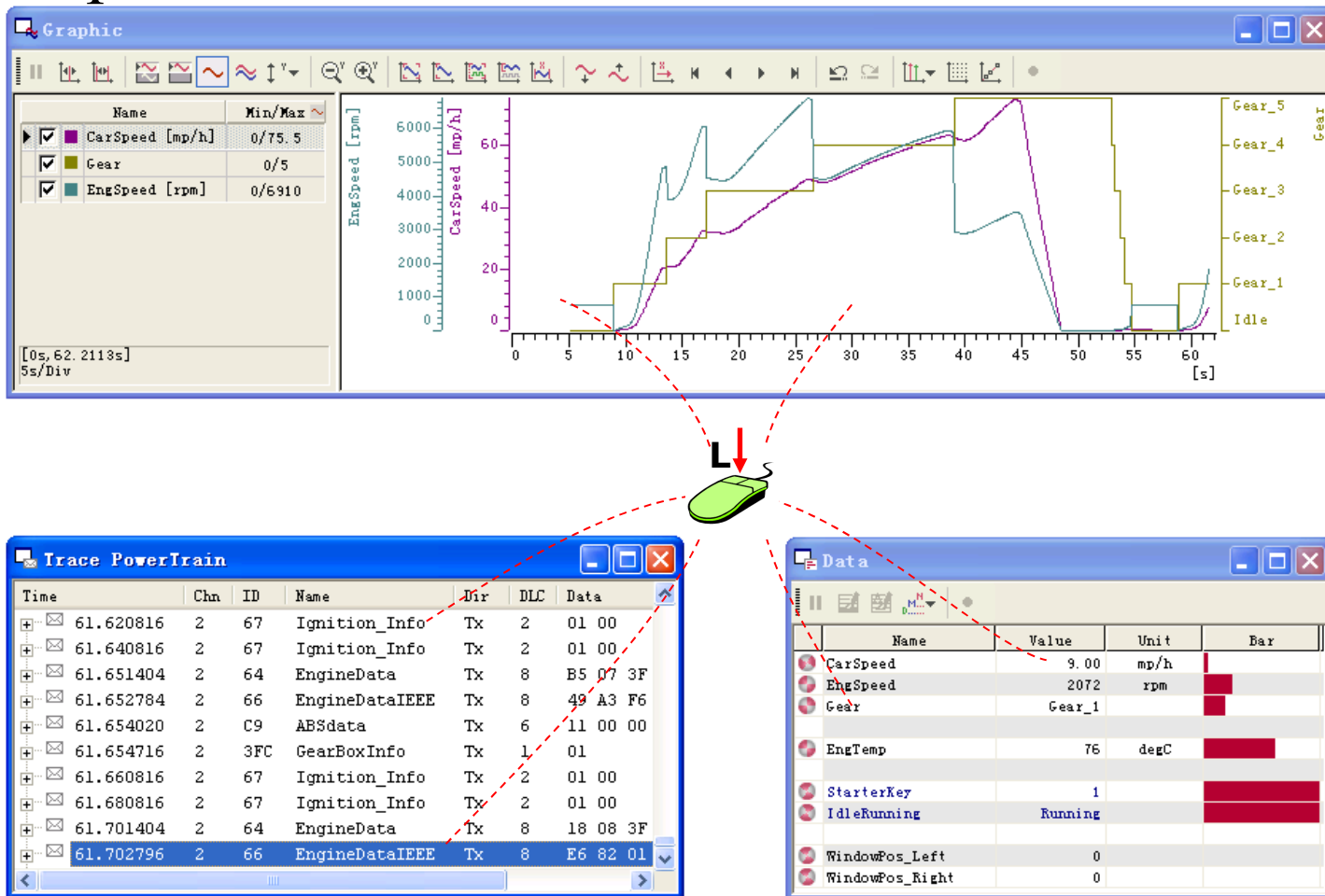
□ 右键单击空白处

□ Extras->Synchronize Windows

□ 定位同一时间不同窗口中数据



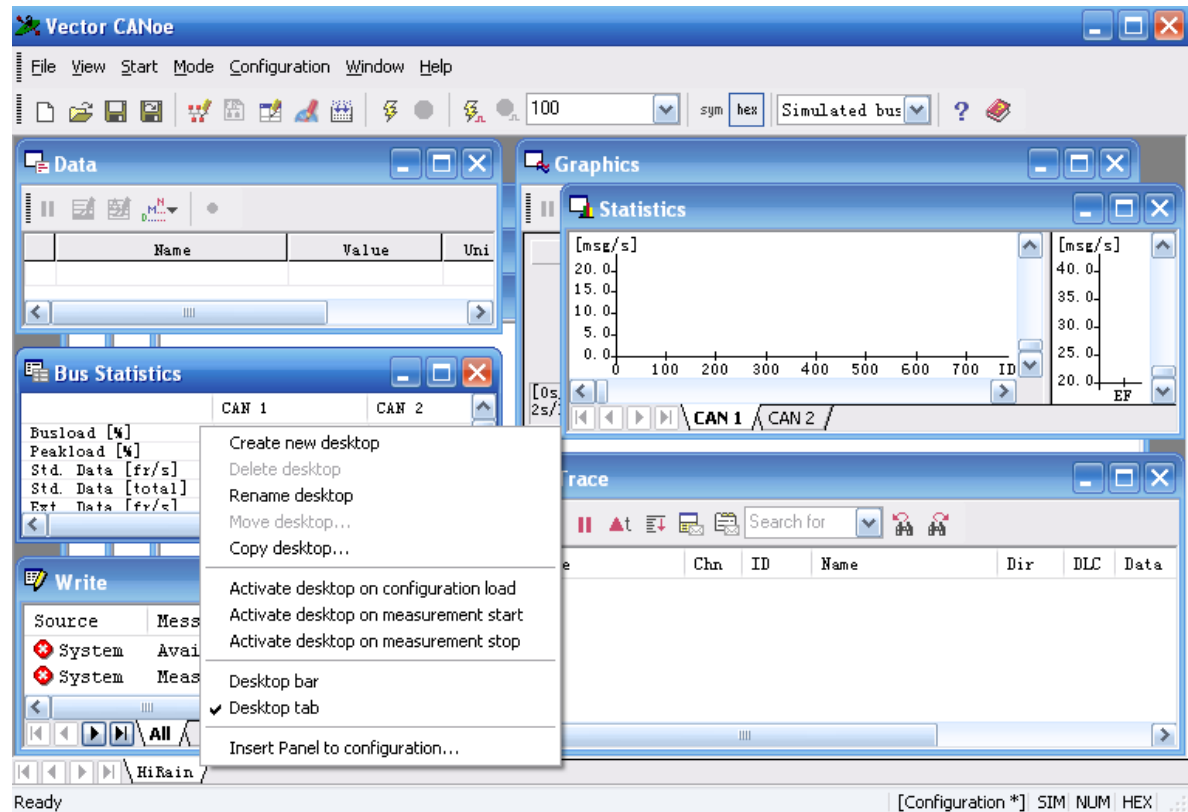
## □ Graphics 、 Trace 、 Data 窗口间鼠标拖放数据



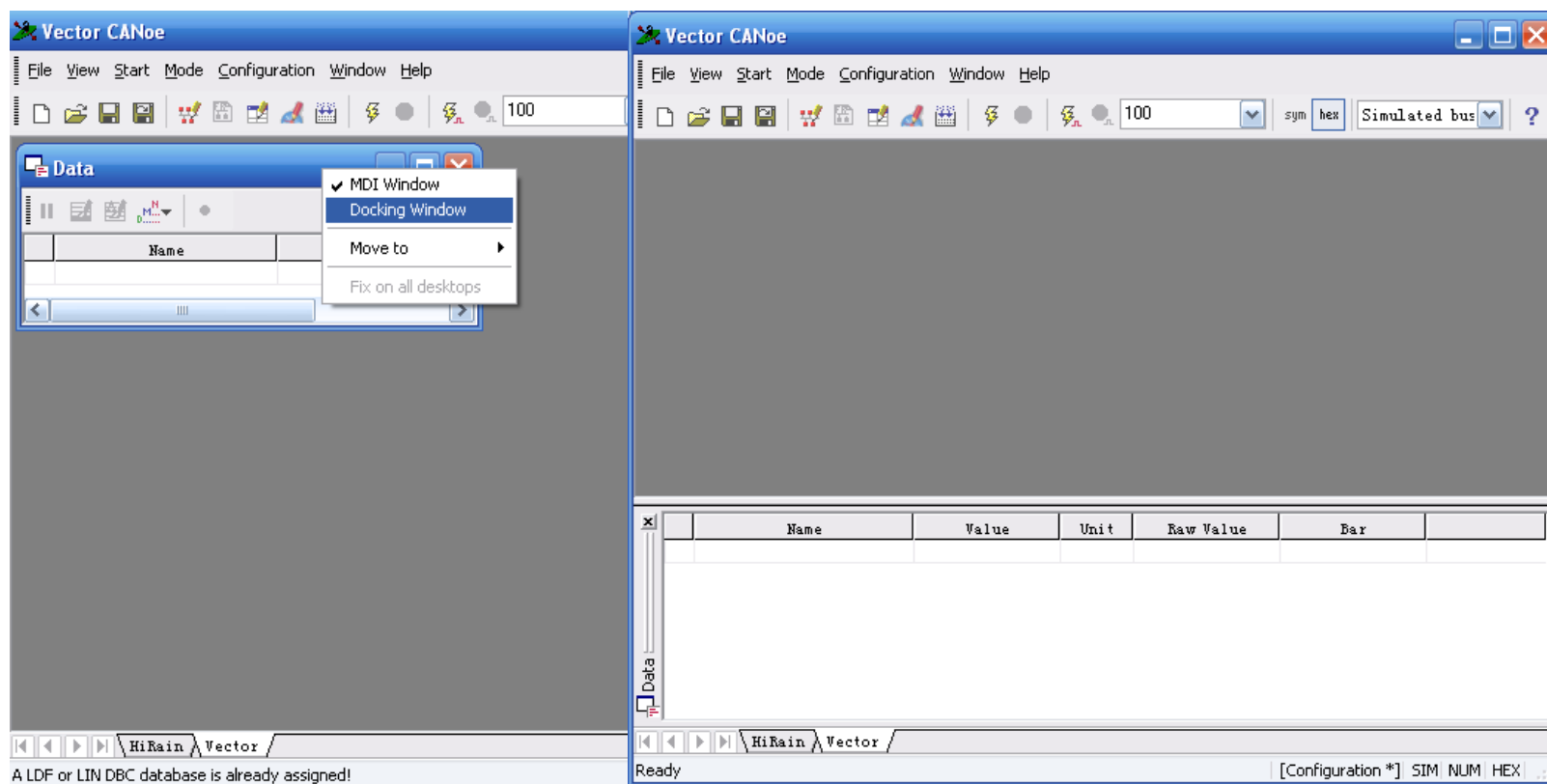
## □ Desktop

### □ Create New Desktop

### □ Rename Desktop

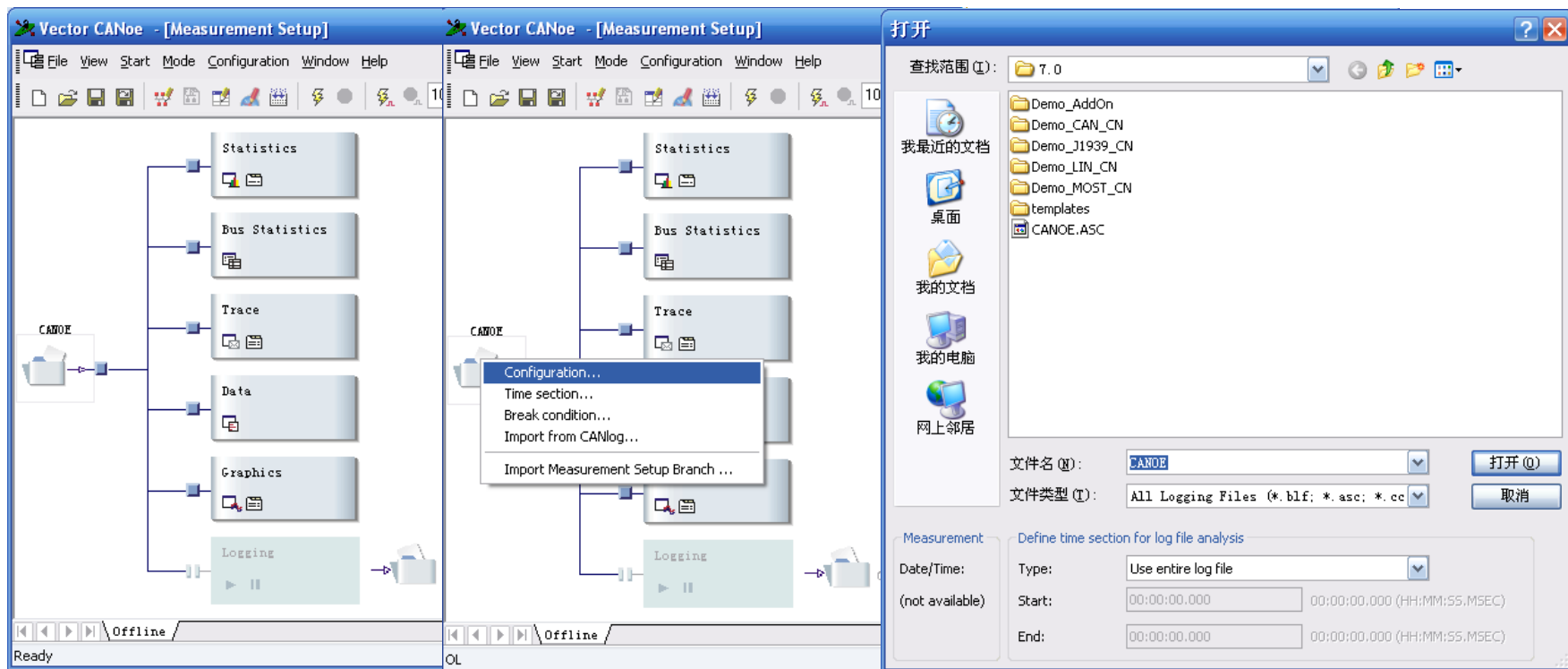


## □ 固定窗口



## □ 离线分析 - 回放记录文件

### □ Mode->To Offline



# 版本记录

版本	更改描述	更改日期	更改人
1.0	初始版本	2008-06-18	高路
1.1	修改蒙太奇链接	2008-07-06	王晗
1.2	增加模块表格	2008-08-06	王晗