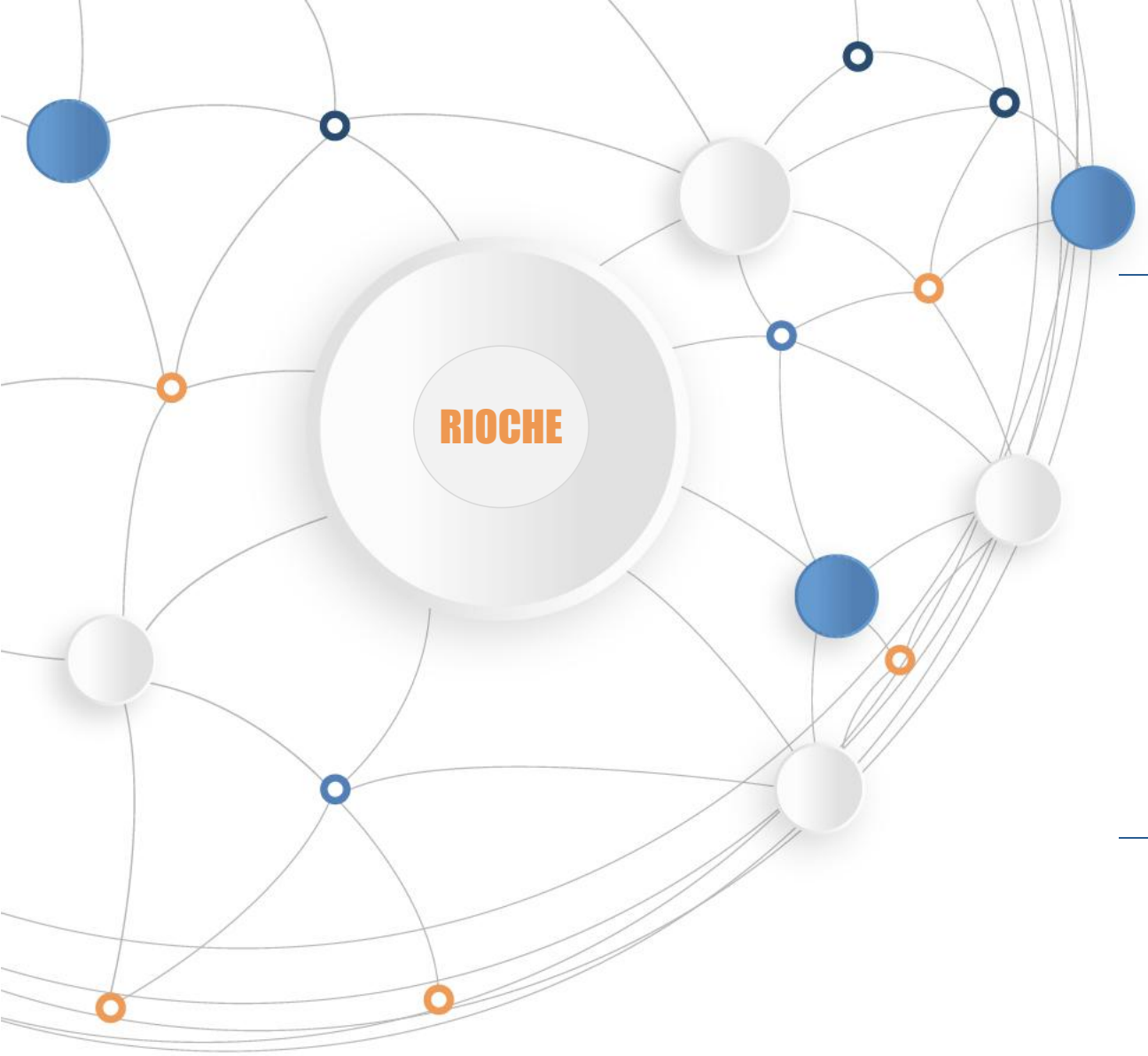


CANoe 基础培训

上海锐勤电子科技有限公司

*本文档部分素材源自Vector

目录

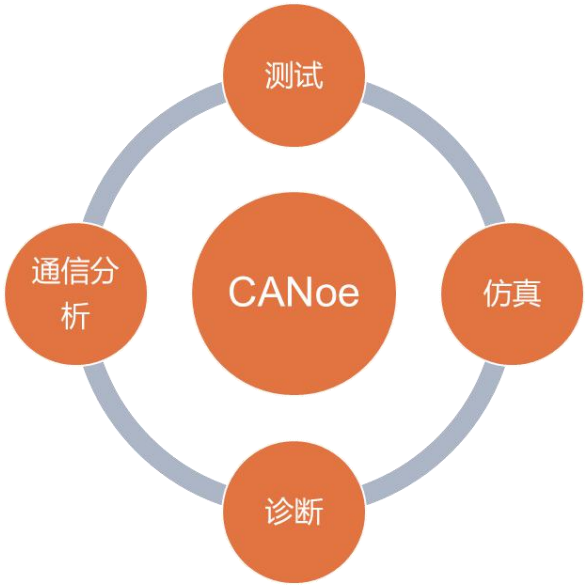
CONTENTS

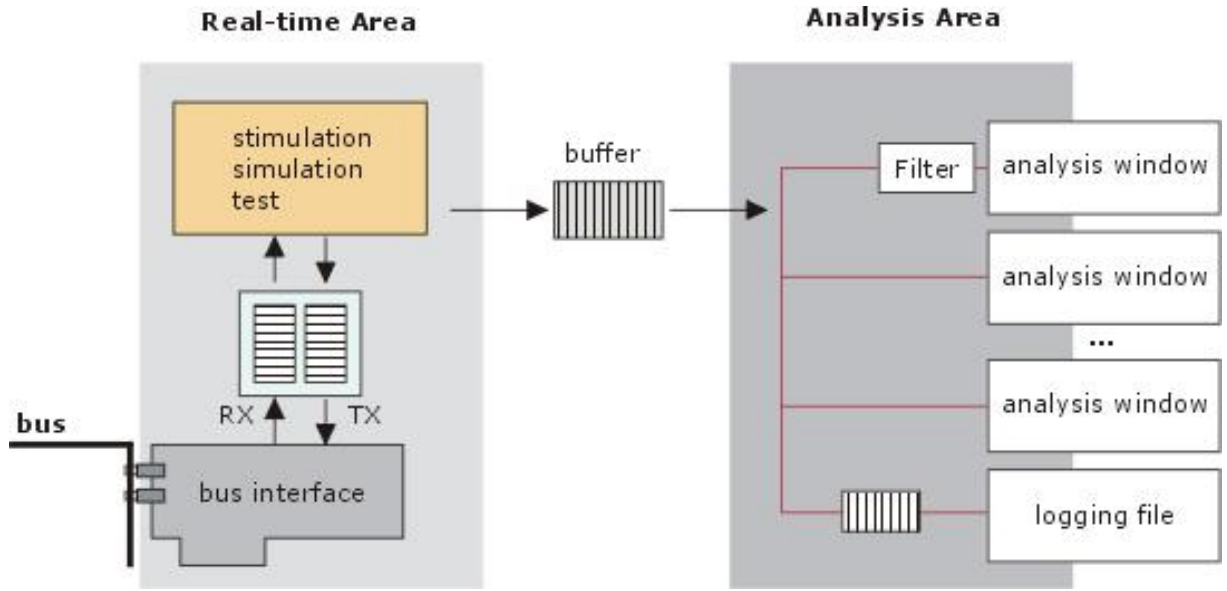
目录

CONTENTS

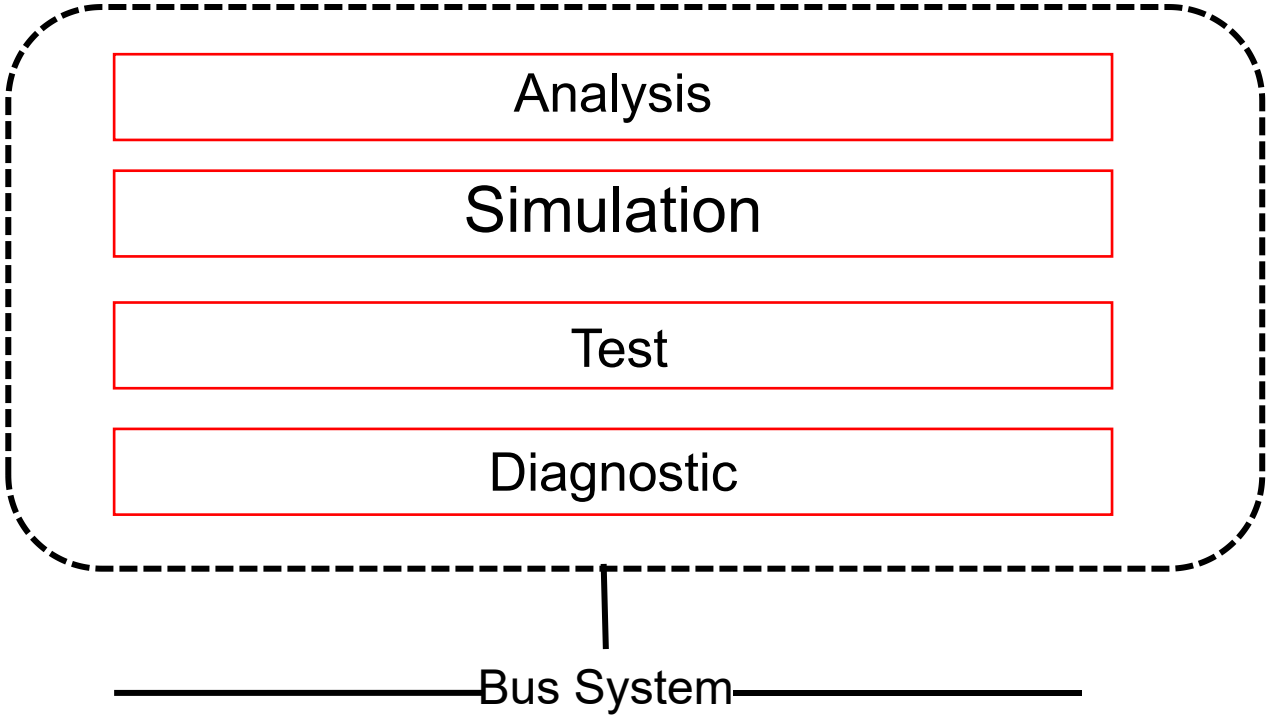
概览

CANoe (CAN 0pen Environment)是Vector公司推出的一款总线开发环境，是进行网络/总线和ECU开发、测试和分析的全面工具，支持总线网络开发从需求分析到系统实现的全过程，CANoe 已被众多整车厂和供应商的系统设计师、开发工程师和测试工程师所广泛使用。

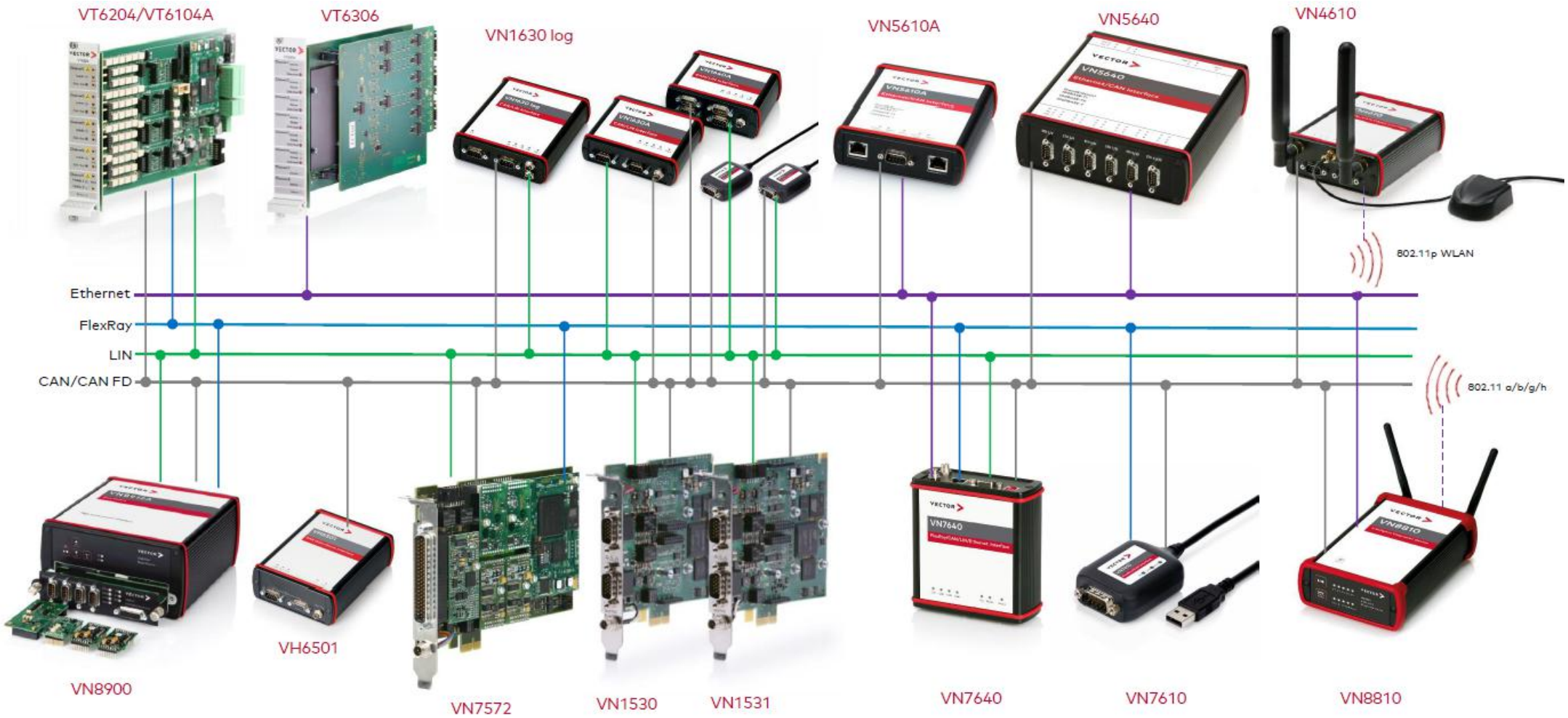




概述

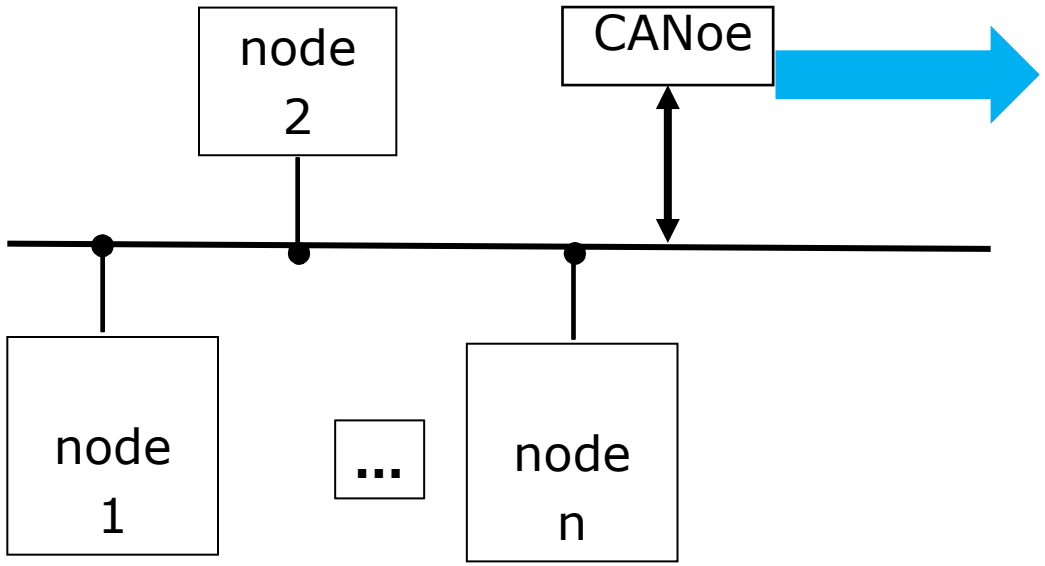


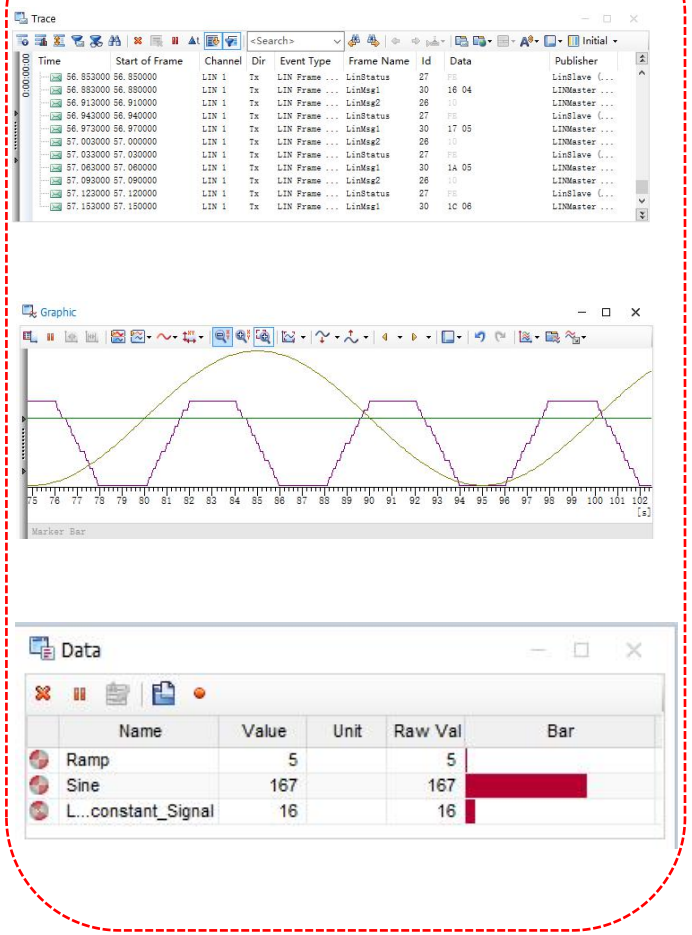
概述


VT6204/VT6104A
VT6306
VN1630 log
VN5610A
VN5640
VN4610
802.11p WLAN
Ethernet
FlexRay
LIN
CAN/CAN FD
802.11 a/b/g/h
VN8900
VH6501
VN7572
VN1530
VN1531
VN7640
VN7610
VN8810

Overview

CANoe 作为分析工具

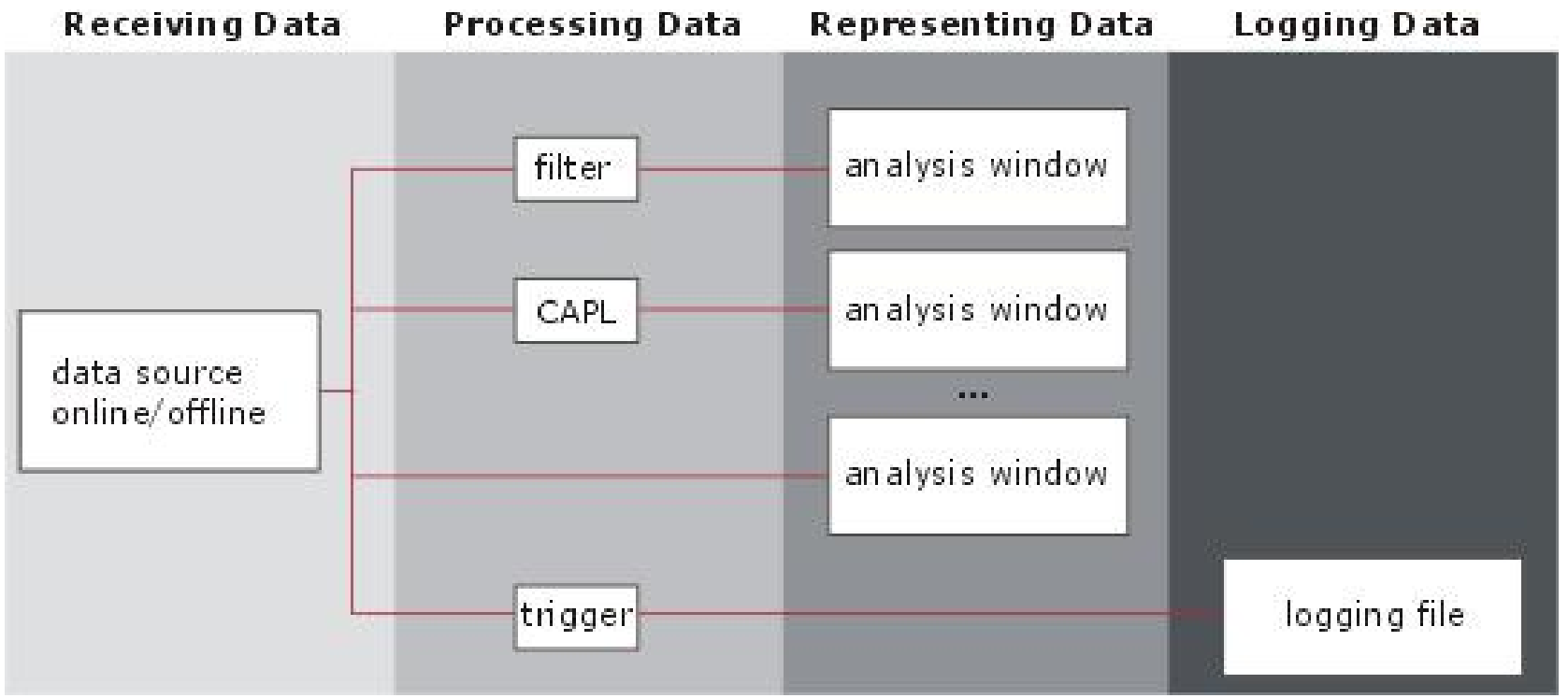




概览

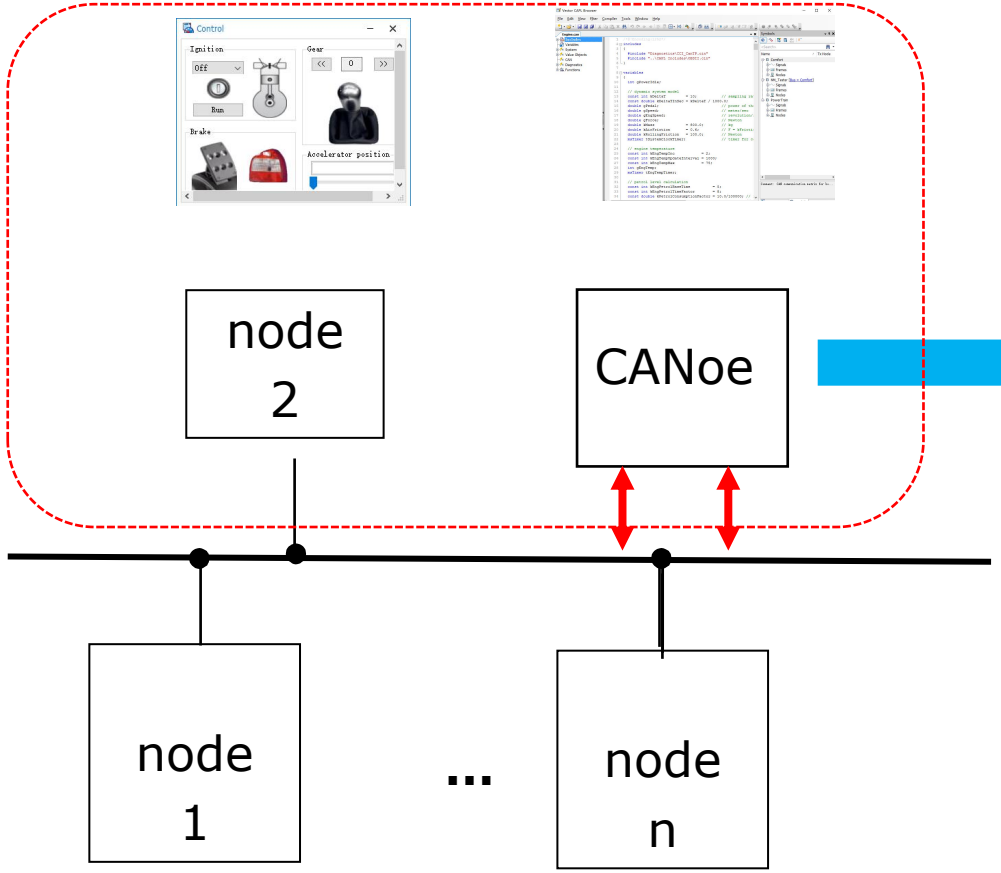
> 分析功能

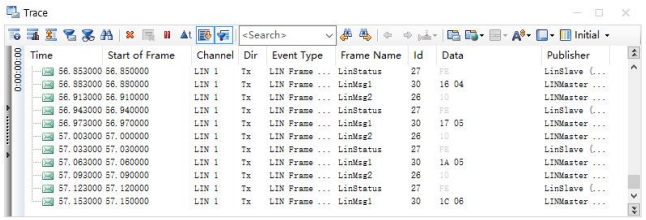
CANoe的分析是基于从数据源到显示或记录的数据流

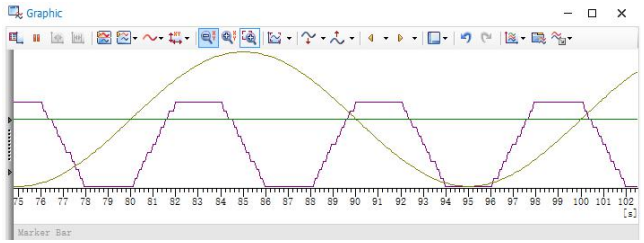


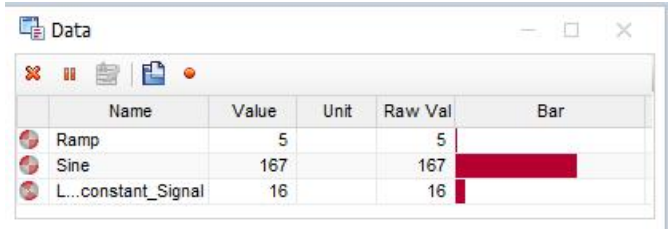
概述

CANoe 作为仿真工具





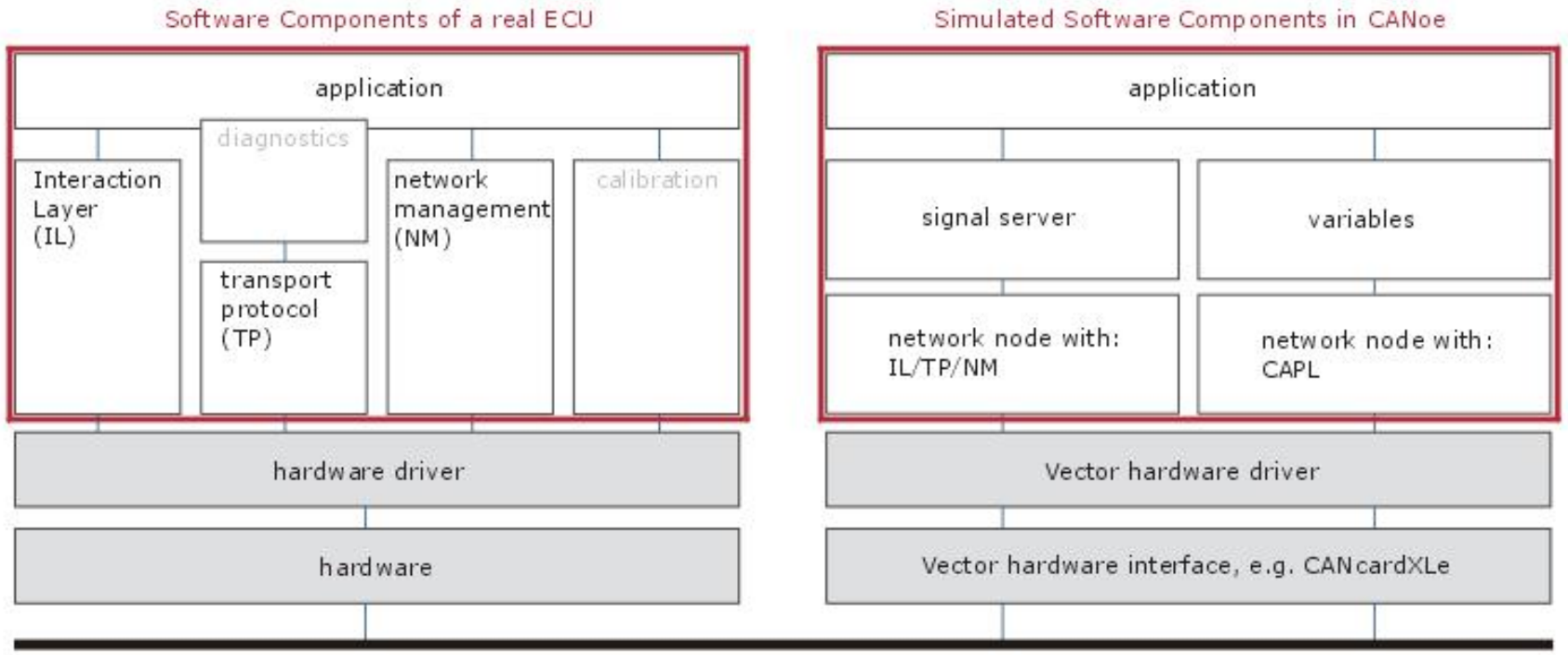




概览

> 仿真功能

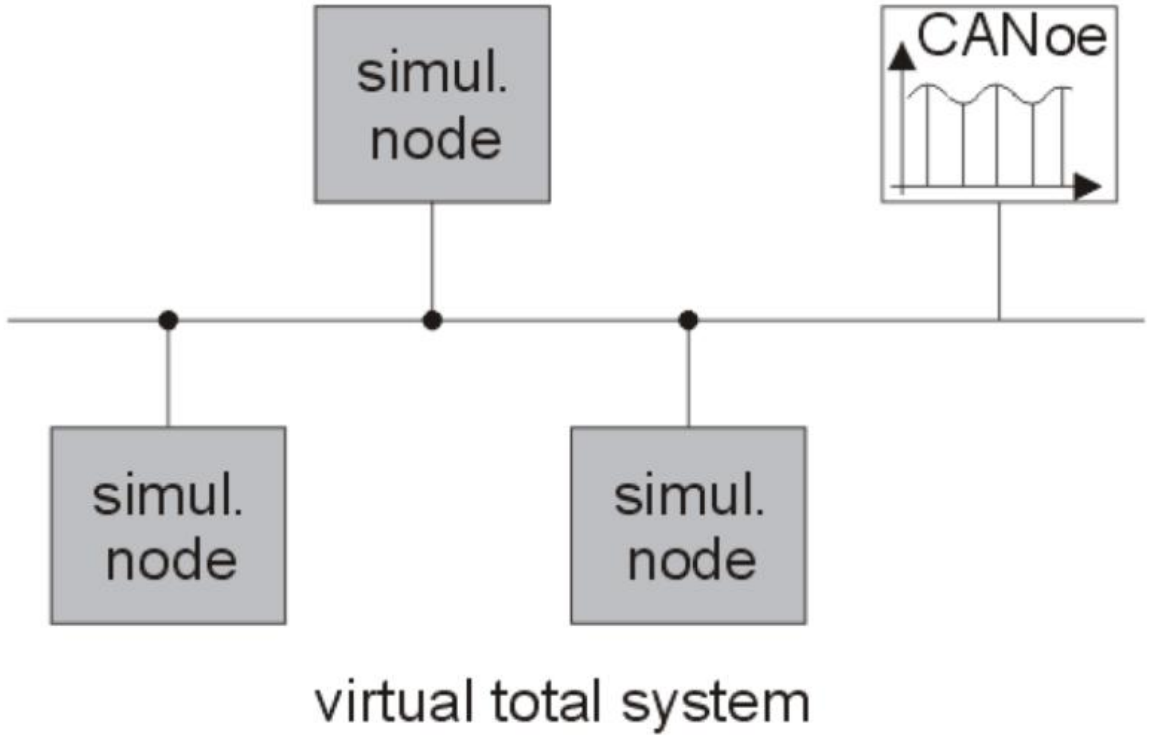
仿真包括在实际控制单元中可能出现的所有软件组件和控制单元的基本传输行为。因此，控制单元的行为被完全仿真。



概览

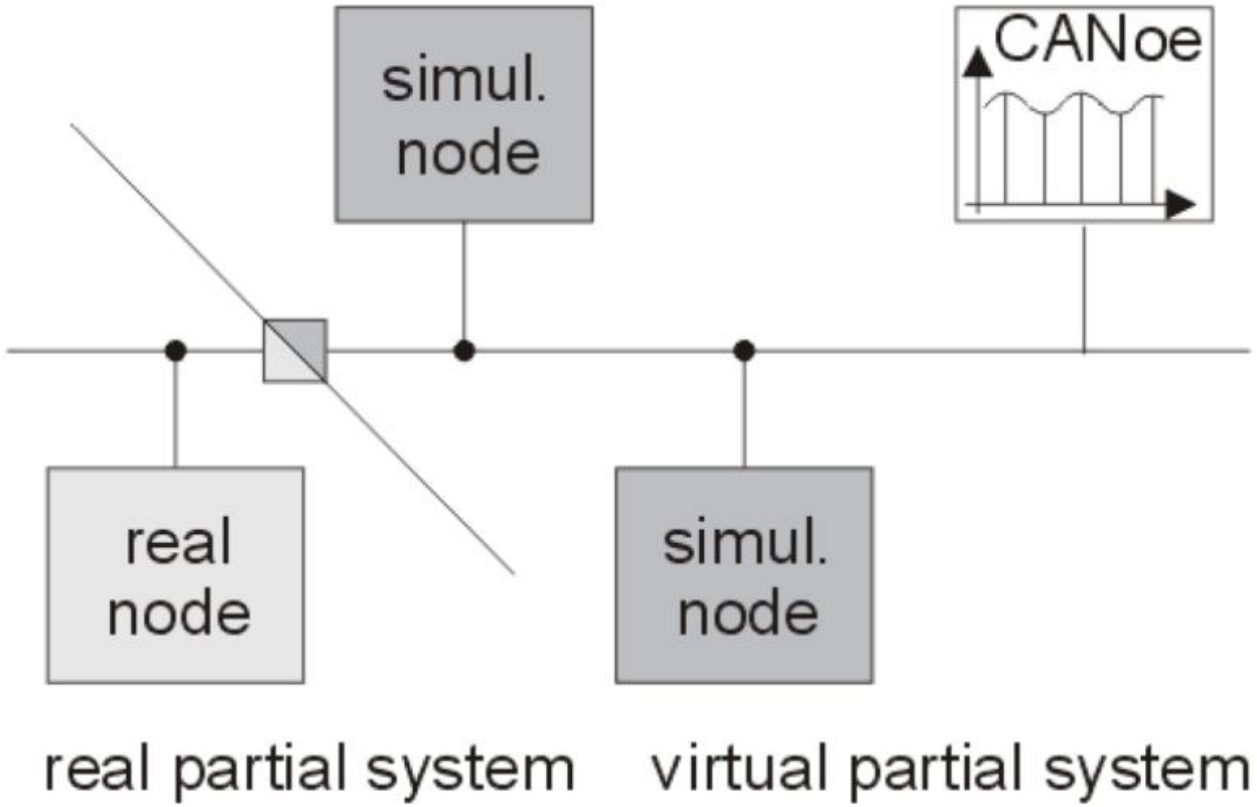
> CANoe在ECU开发中的作用

第一阶段：全仿真网络系统



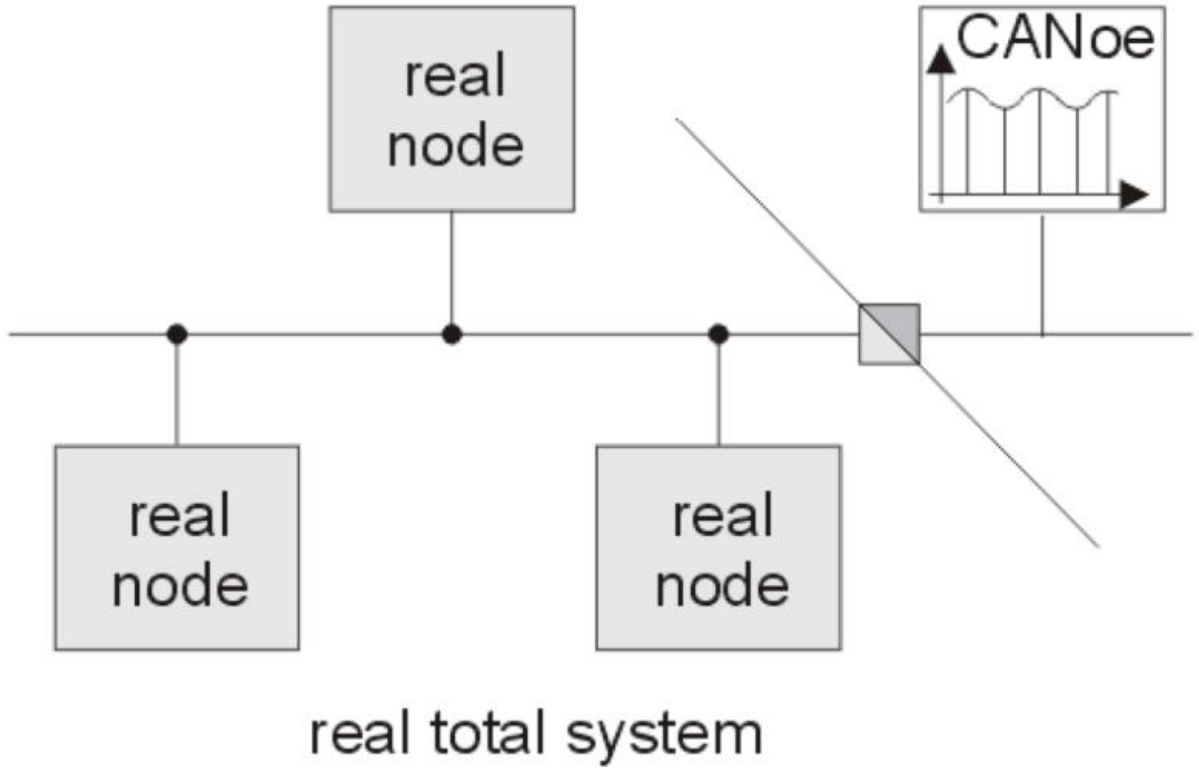
> CANoe在ECU开发中的作用

第二阶段：全真实节点和部分仿真节点共存



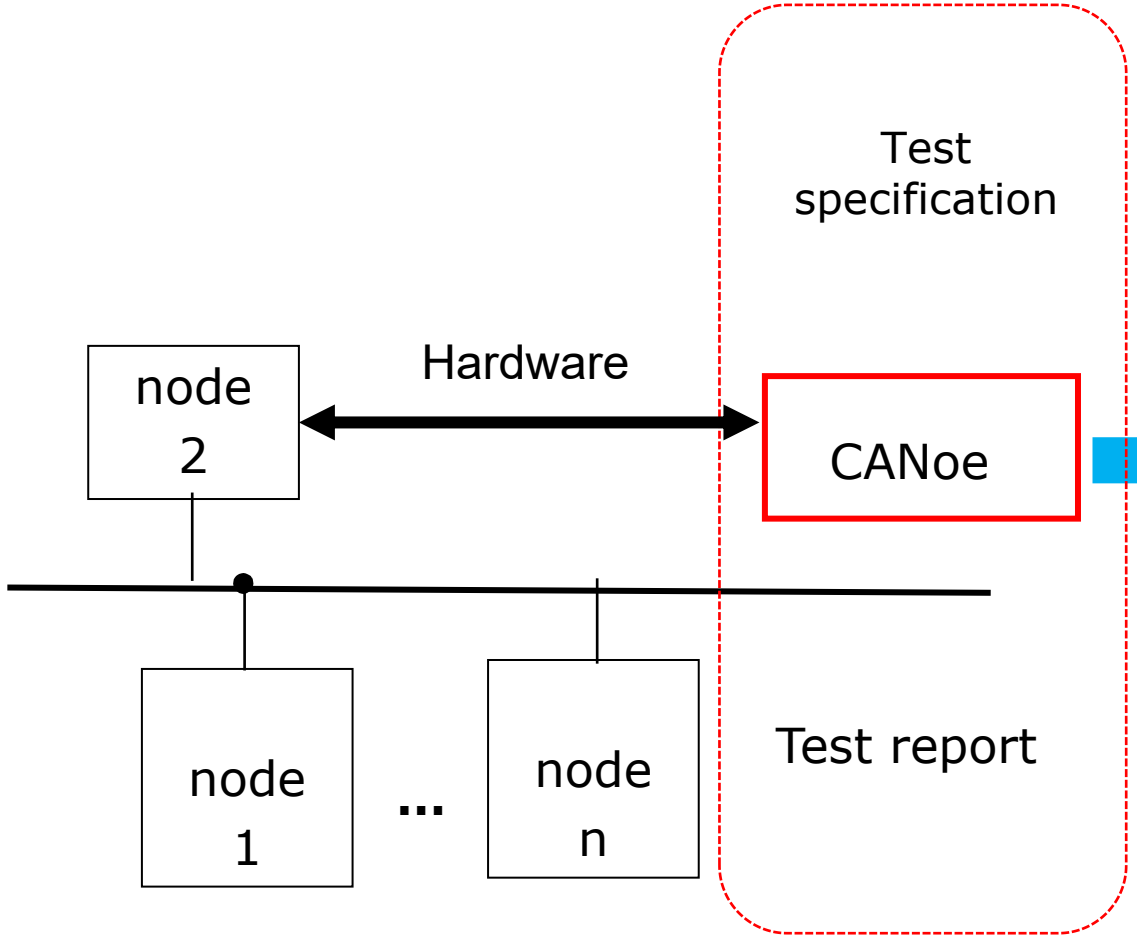
> CANoe在ECU开发中的作用

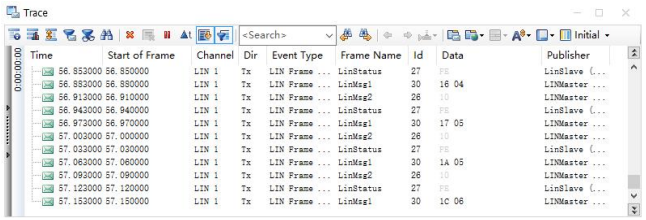
第三阶段：全真实节点的网络系统

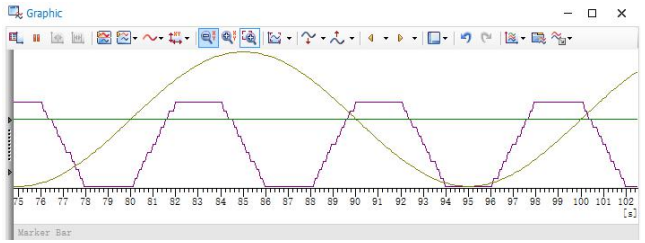


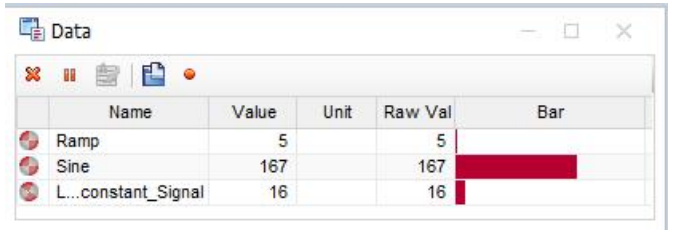
概览

> CANoe 作为测试工具





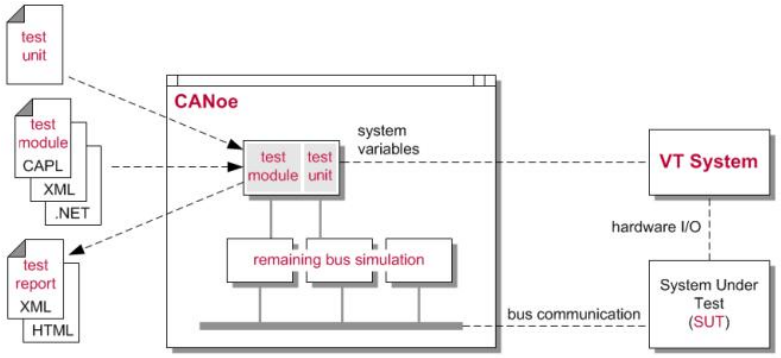


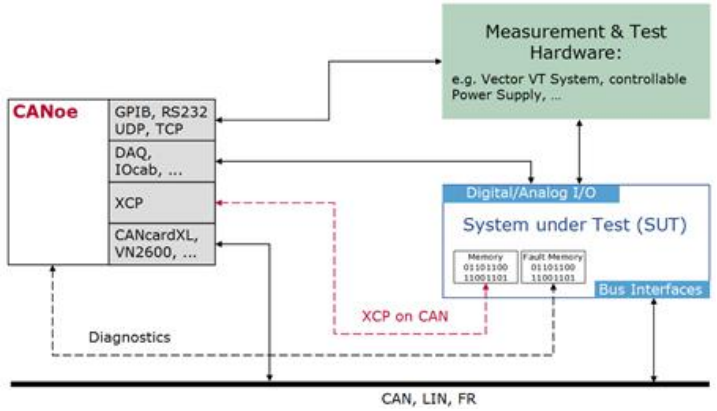


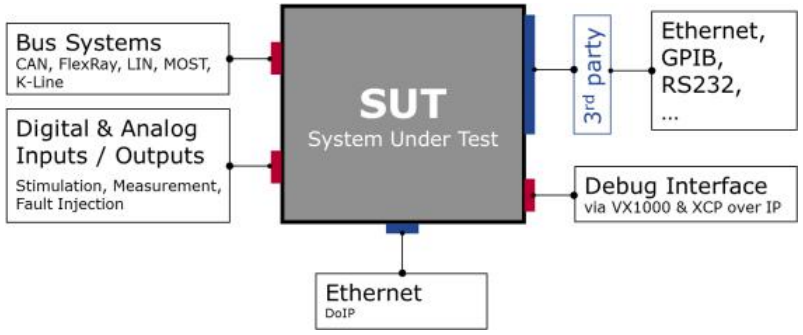
概览

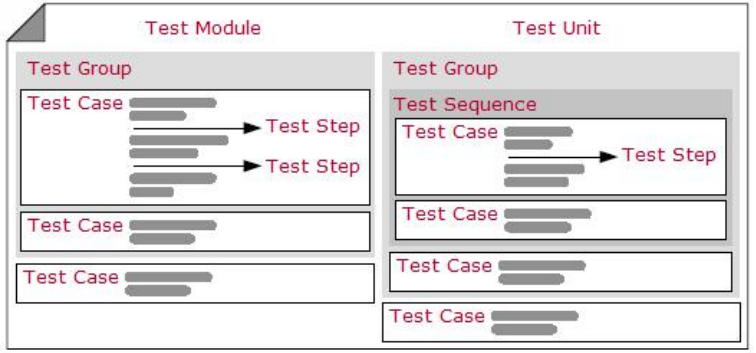
> 测试功能

CANoe 具有测试功能集，用来简化或自动进行测试。运用该功能，可以进行一系列的连续测试，并自动生成测试报告。



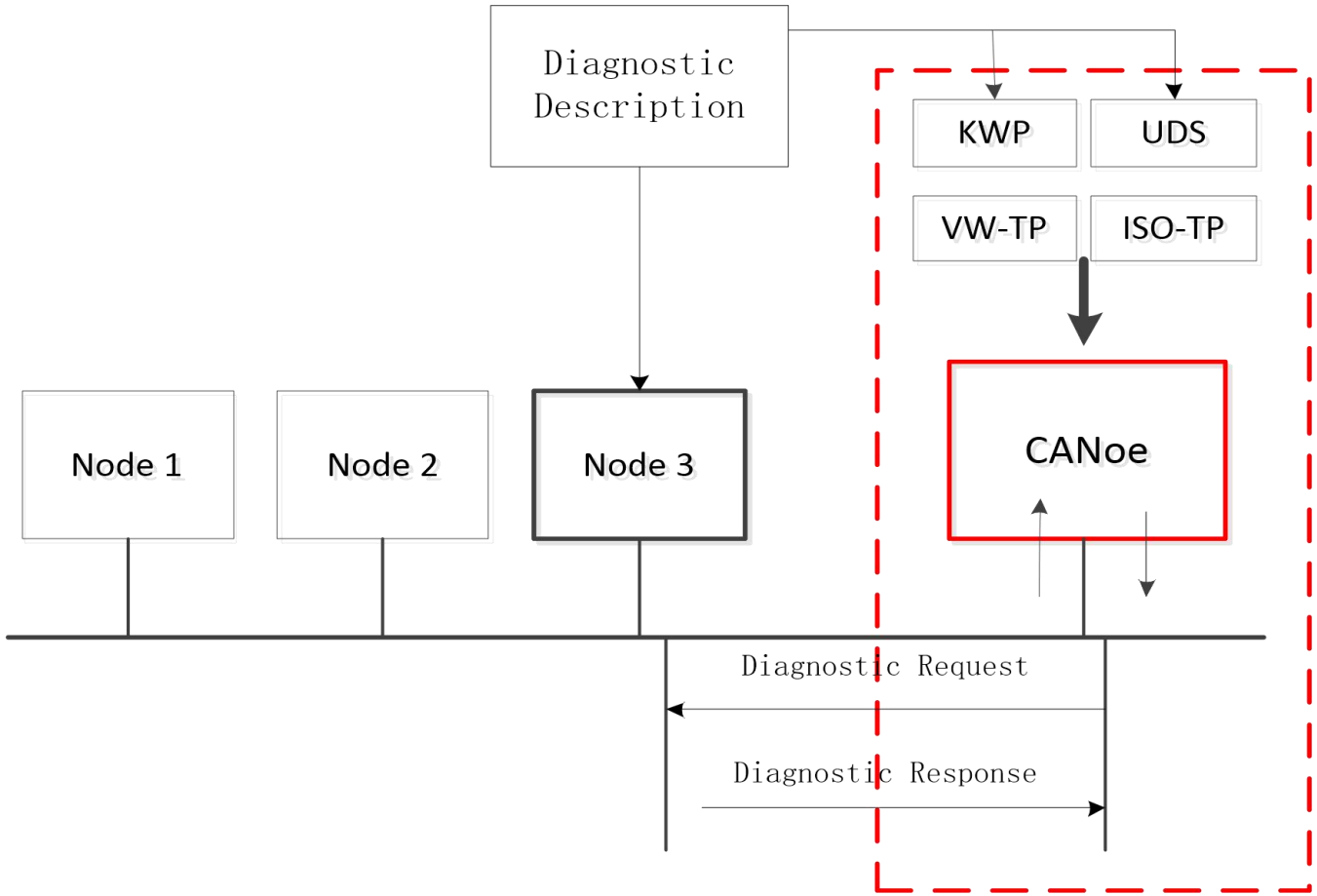






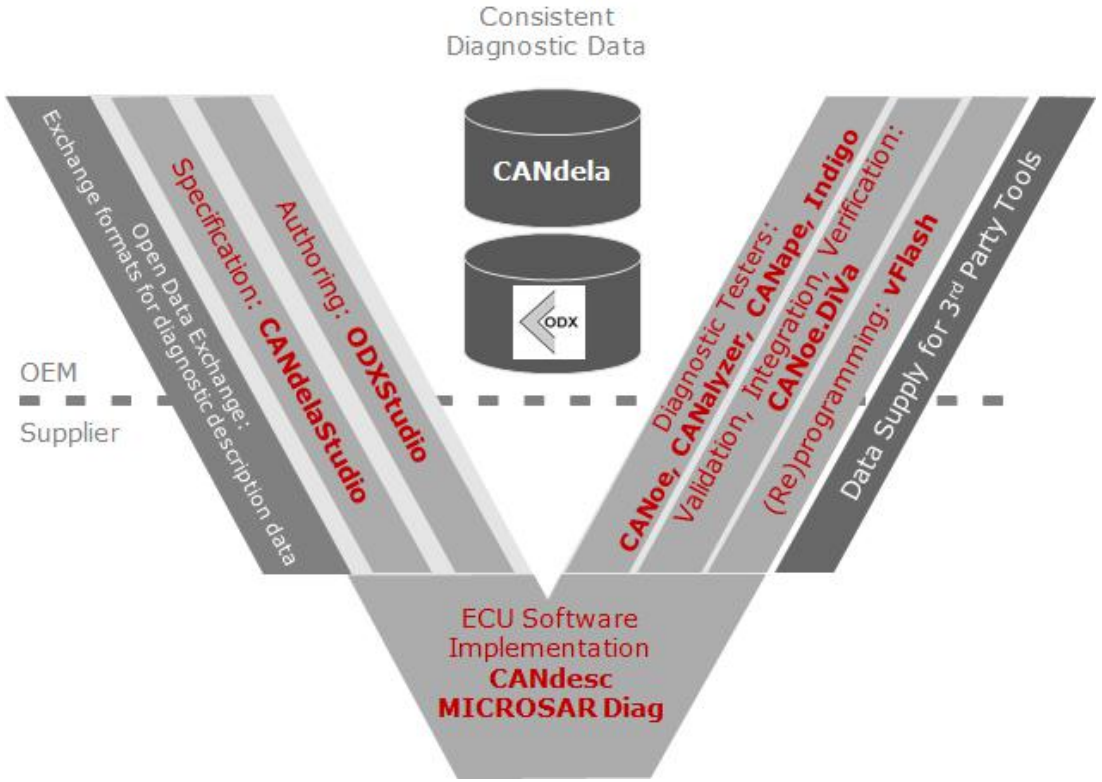
概览

CANoe 作为诊断工具



> 诊断功能

CANoe可以满足ECU开发的各个阶段中对诊断测试的设计和执行的要求。诊断功能集包含通过诊断进行ECU开发、测试和标定所需的各种功能，允许对控制单元的诊断服务进行分析和仿真。



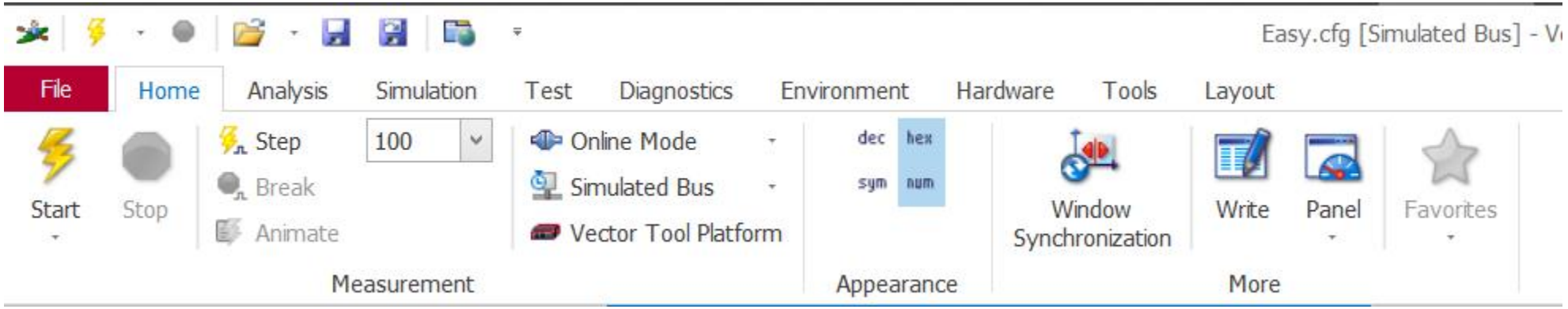
目录

CONTENTS

工程创建

> CANoe菜单栏

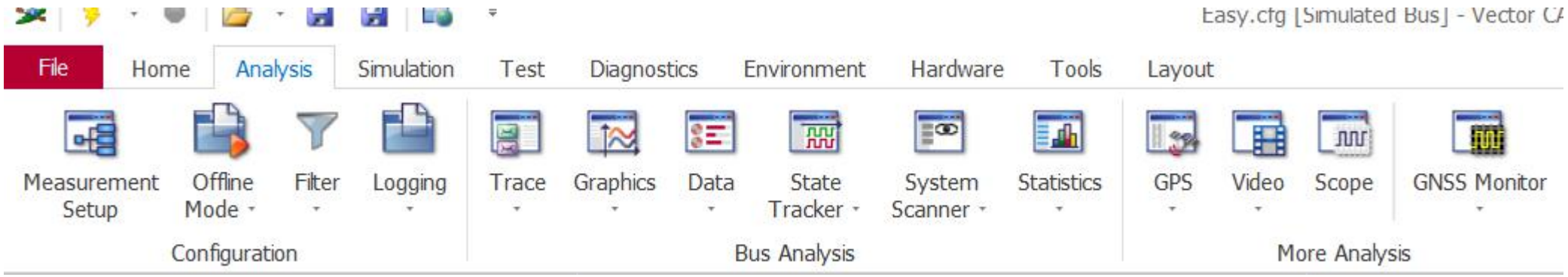
Home | 测量设置，模式切换，数据显示格式等



工程创建

CANoe菜单栏

Analysis | 测试配置，总线分析，其他类型信号分析等



工程创建

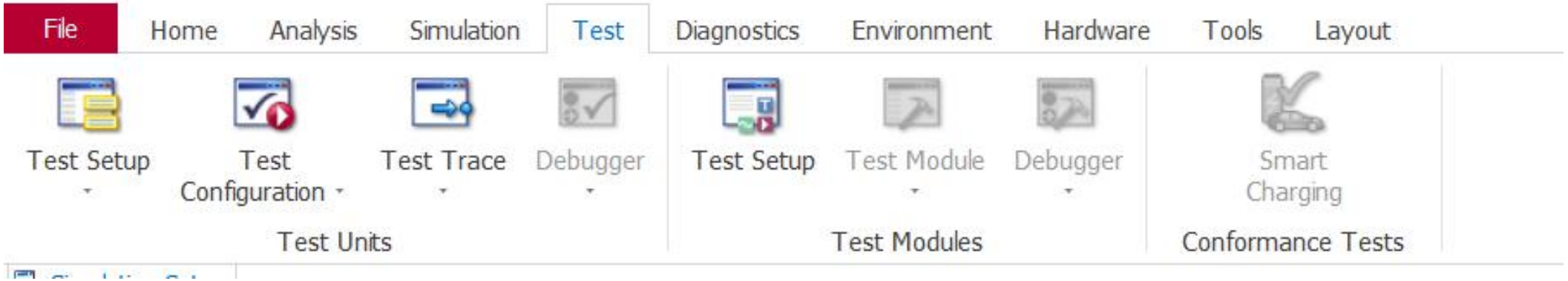
> CANoe菜单栏

Simulation | 仿真设置，信号发生器，IG模块，激励等

工程创建

> CANoe菜单栏

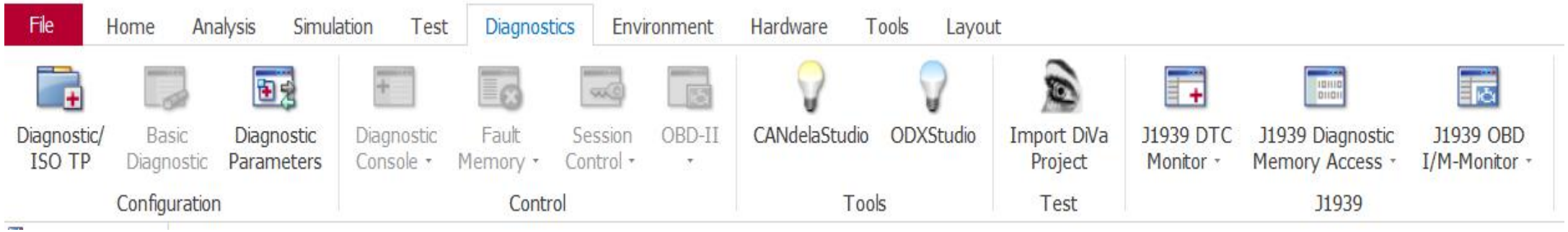
Test | 测试单元设置，测试模块设置等



工程创建

> CANoe菜单栏

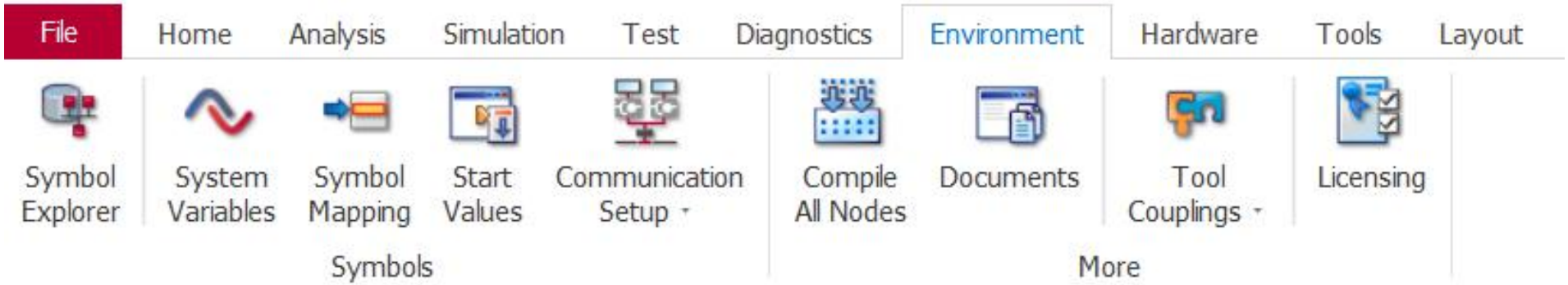
Diagnostics | 诊断功能配置，控制，诊断工具接口，以及诊断测试，J1939相关模块。



工程创建

> CANoe菜单栏

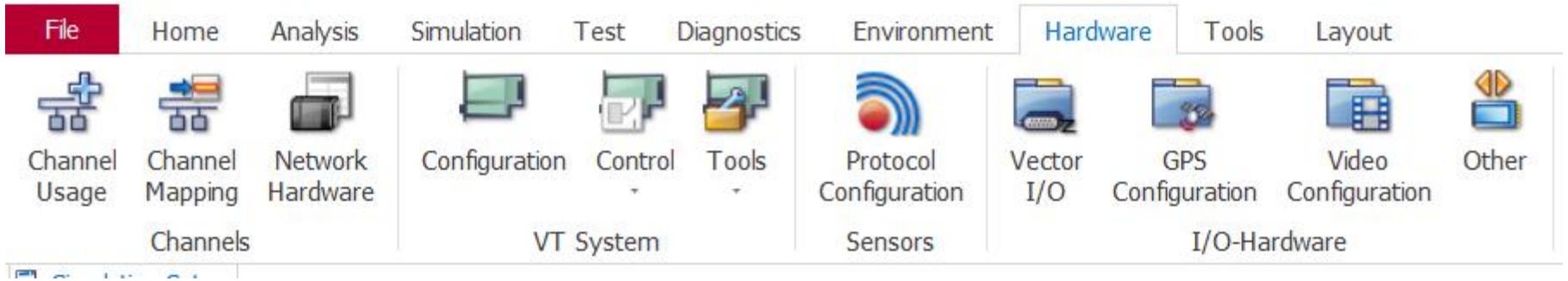
Environment | 对象配置及控制。



工程创建

CANoe菜单栏

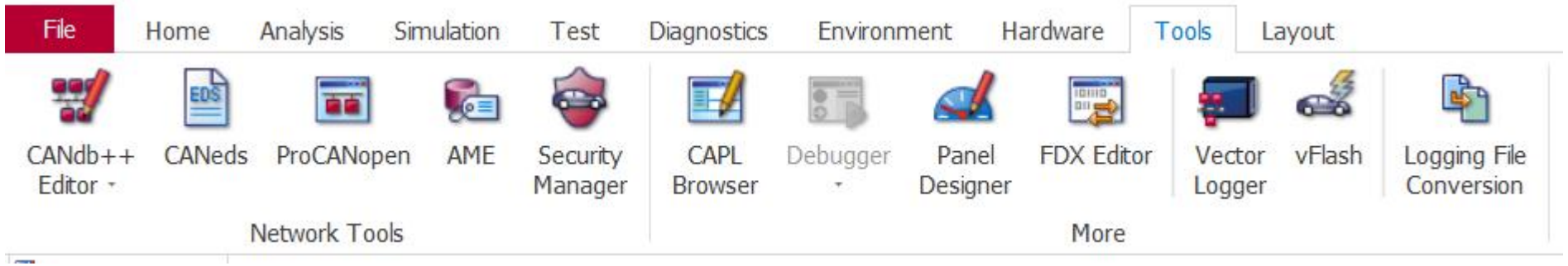
Hardware | 通道设置，VT配置及控制，传感器及I/O硬件控制



工程创建

CANoe菜单栏

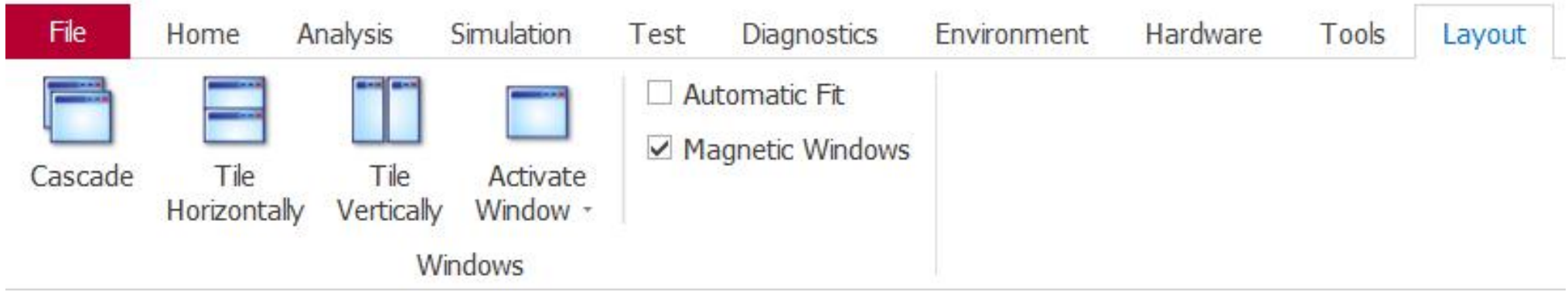
Tools | 数据库编辑器，CAPL浏览器，Panel面板，文件转换等功能。



工程创建

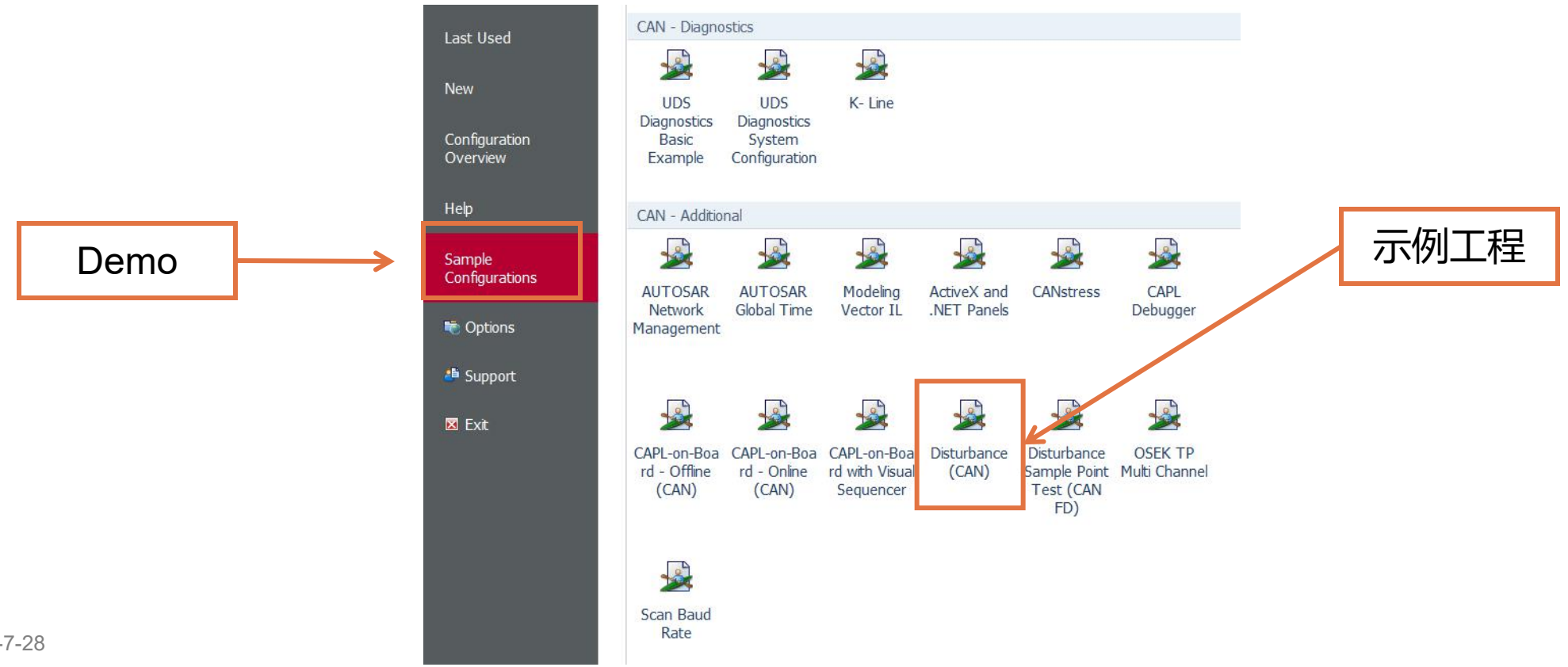
> CANoe菜单栏

Layout | CANoe界面中窗口的排列方式切换。



Sample

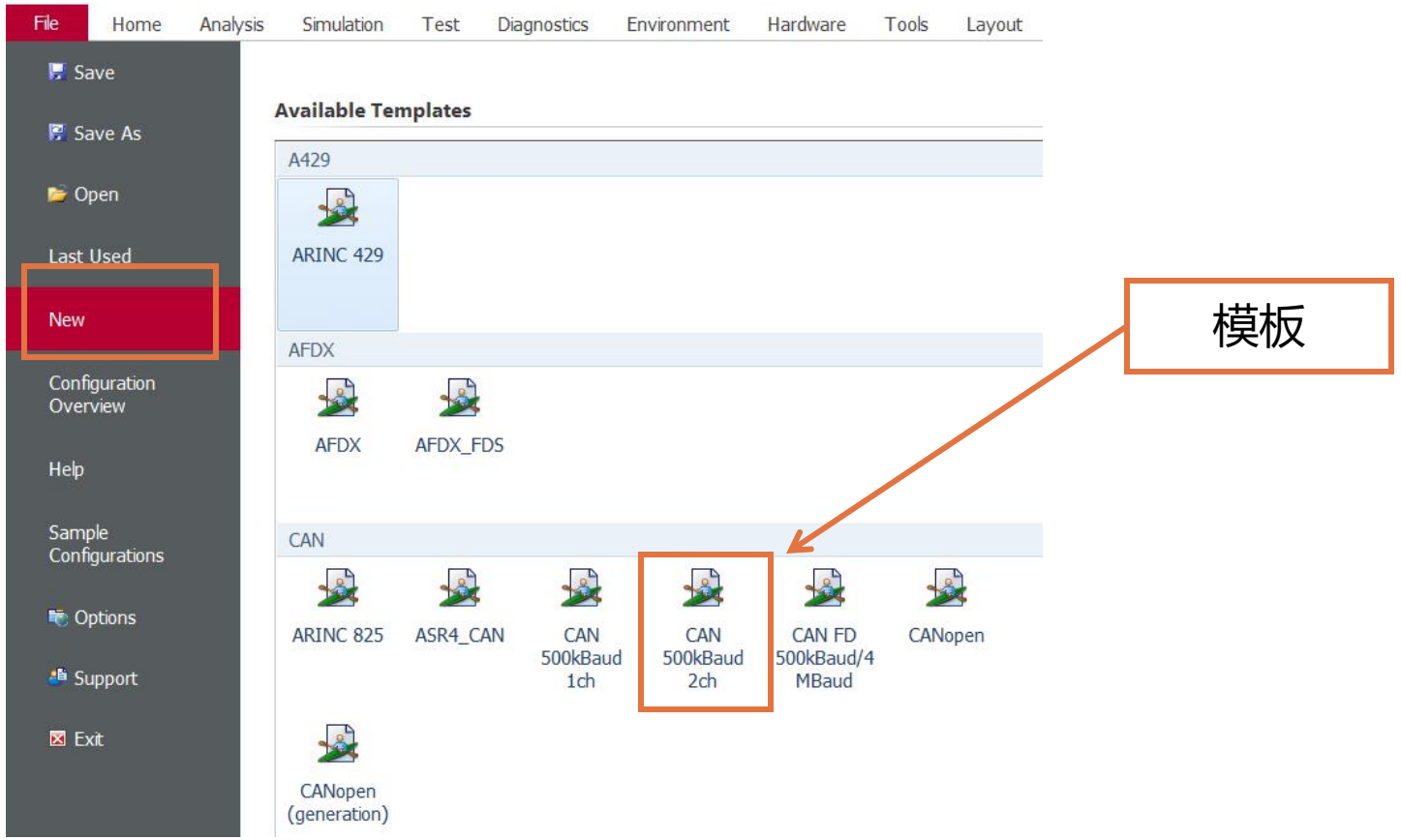
> Sample Configurations



创建配置

> 创建新的配置

新建配置，选择对应的模板 File | New | CAN……



工程创建

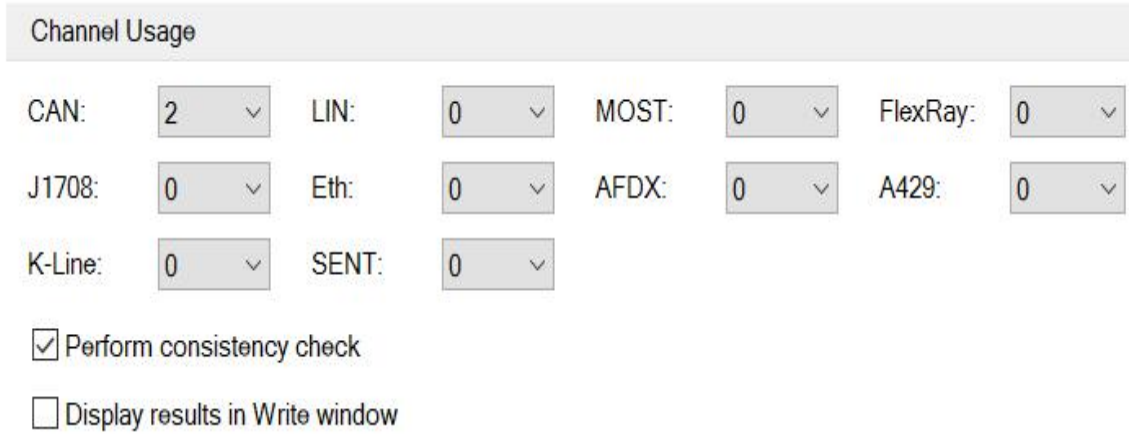
> 创建新的配置

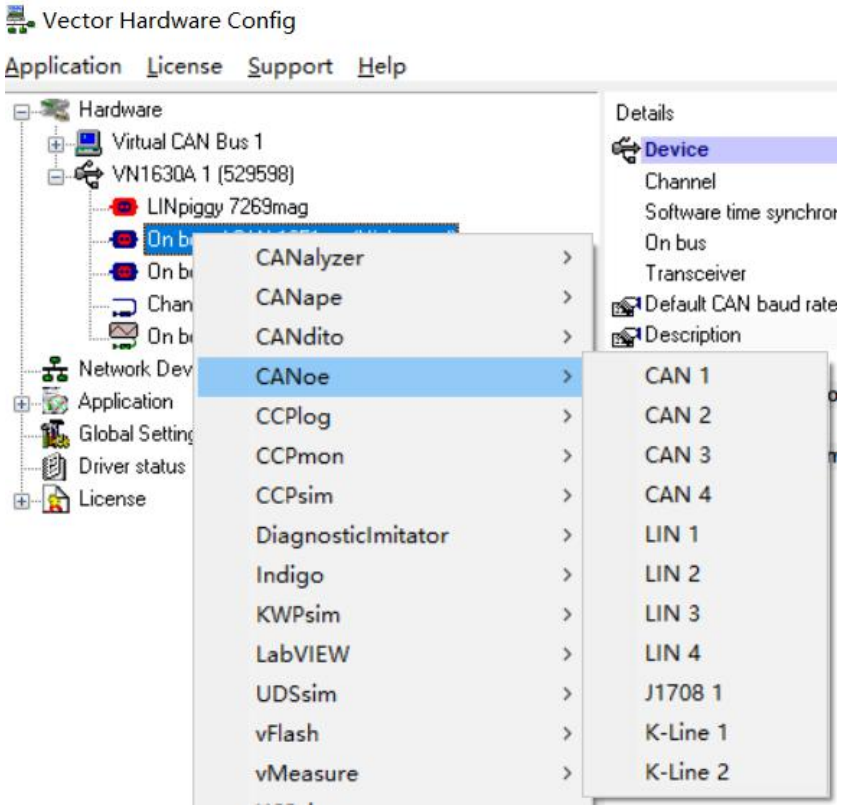
进行通道的配置，考虑软件与硬件两个方面，

Hardware | Channel Usage ..... 软件通道

Hardware | Network Hardware..... 硬件通道

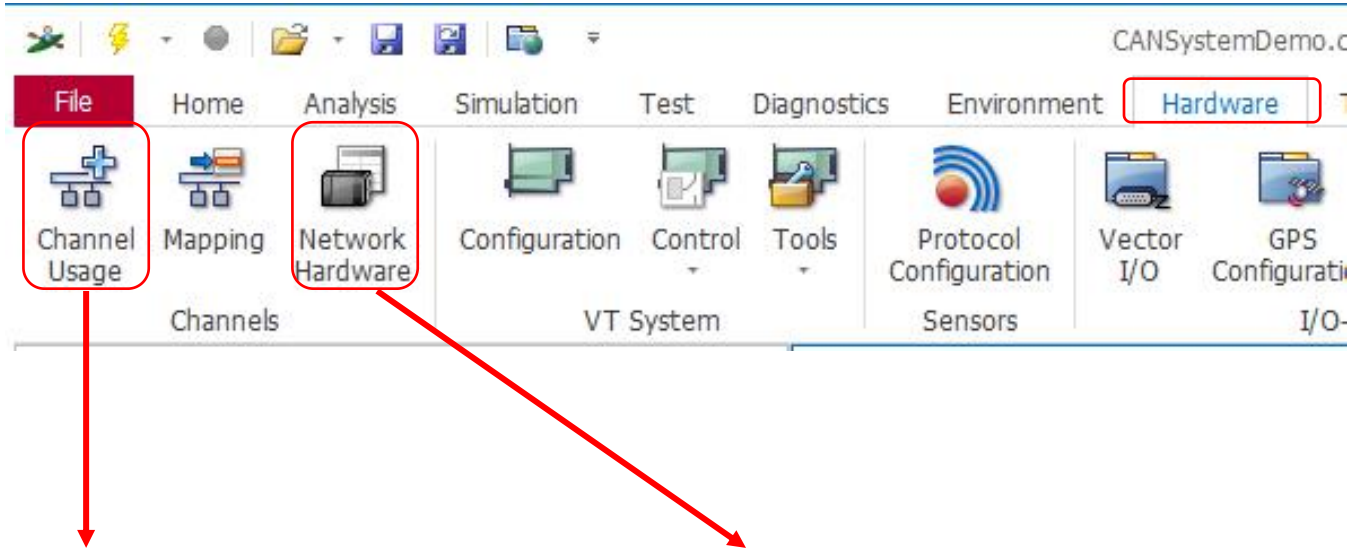
Hardware | Channel Mapping..... 通道映射





通道配置

> Configuration-Hardware



- Channel Usage
  - 设置通道数量
  - 通道全局设置

- Network Hardware
  - 波特率
  - Acknowledge
  - Acceptance Filter
  - 设备管理

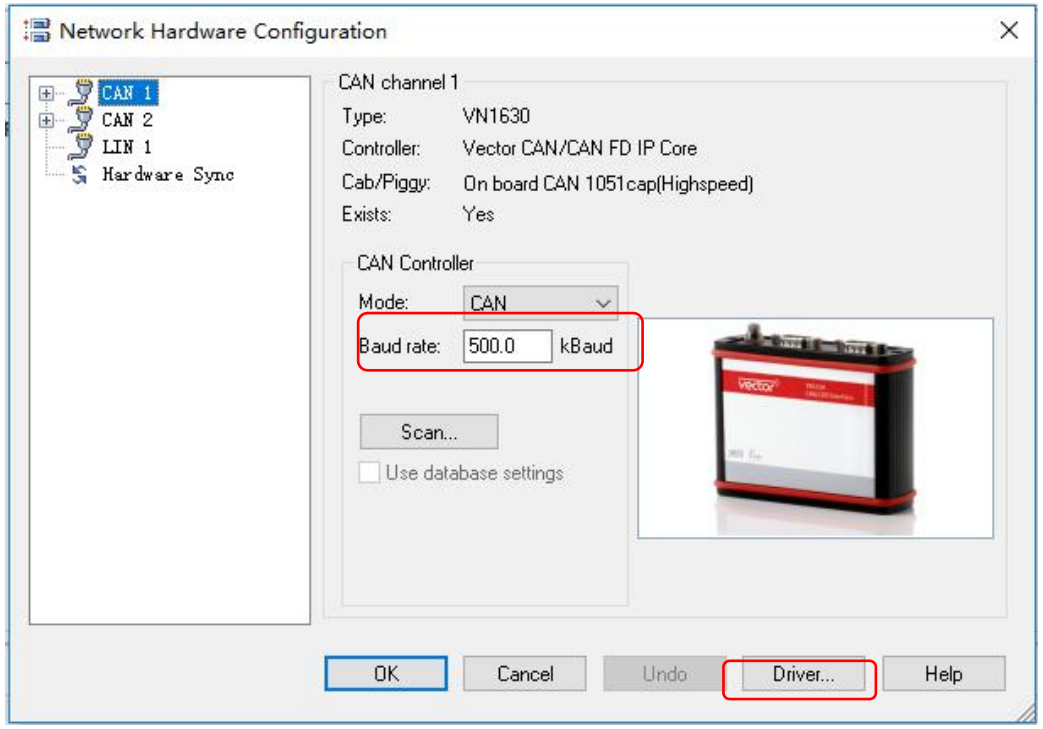
通道设置

> Channel usage—设置通道数量

- 修改总线通道数量
- 根据总线类型选择通道数量

CAN 网络硬件配置

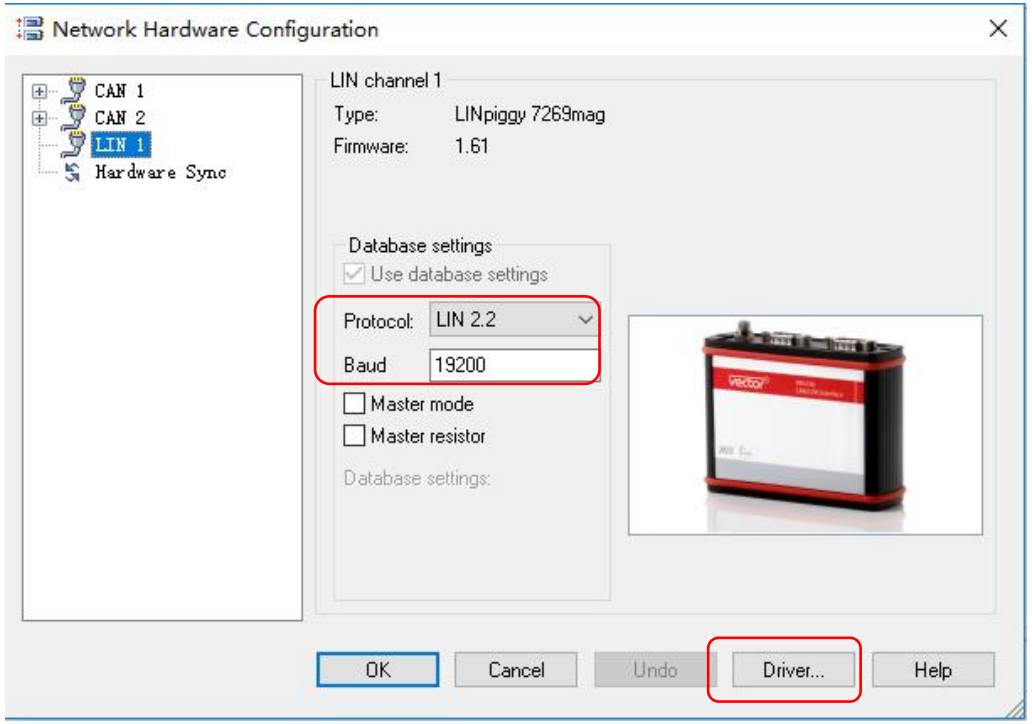
> Network Hardware—通道设置



- 每个通道设置独立
- 只对软件中关联的硬件通道设置有效
- 总线通道数量取决于Channel usage设置
- 通过 Driver 打开 Vector Hardware Configuration

LIN 网络硬件配置

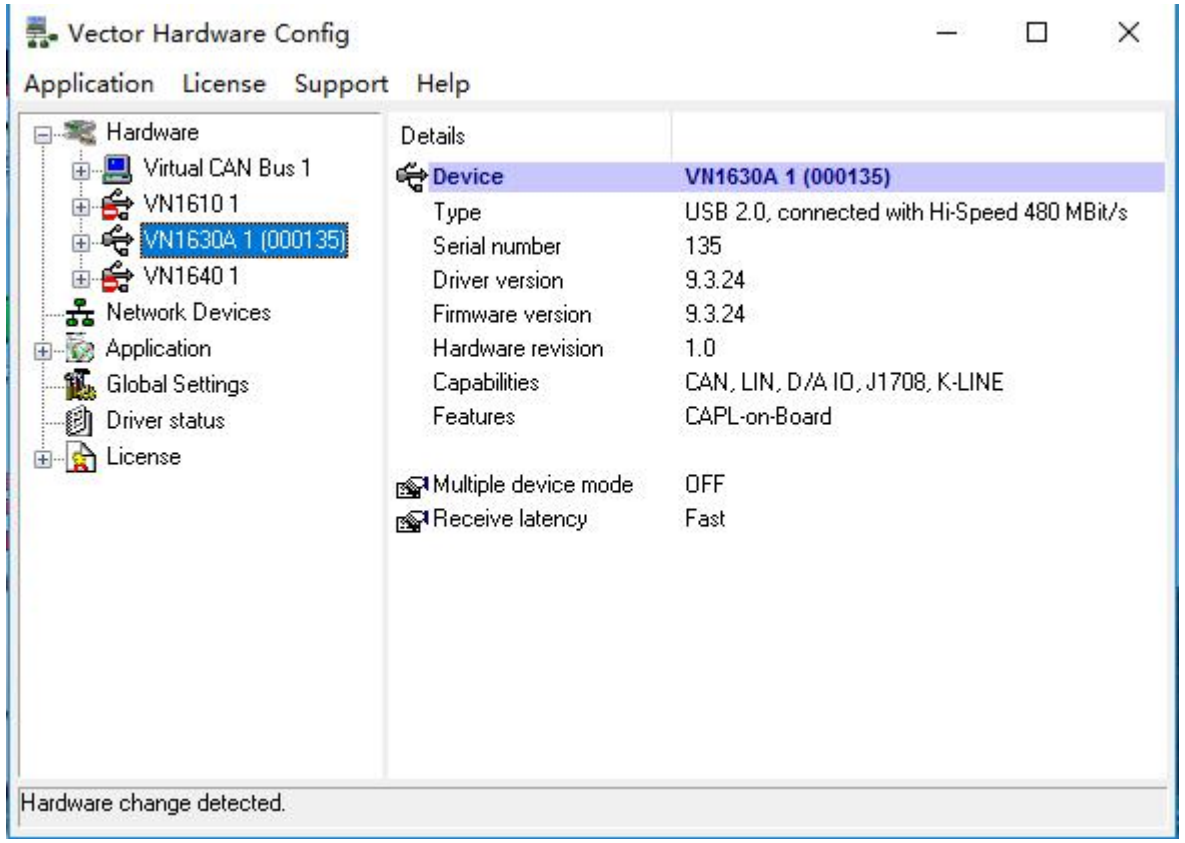
> Network Hardware—通道设置



- 每个通道设置独立
- 只对软件中关联的硬件通道设置有效
- 总线通道数量取决于Channel usage设置
- 通过 Driver 打开 Vector Hardware Configuration

Vector Hardware Configuration

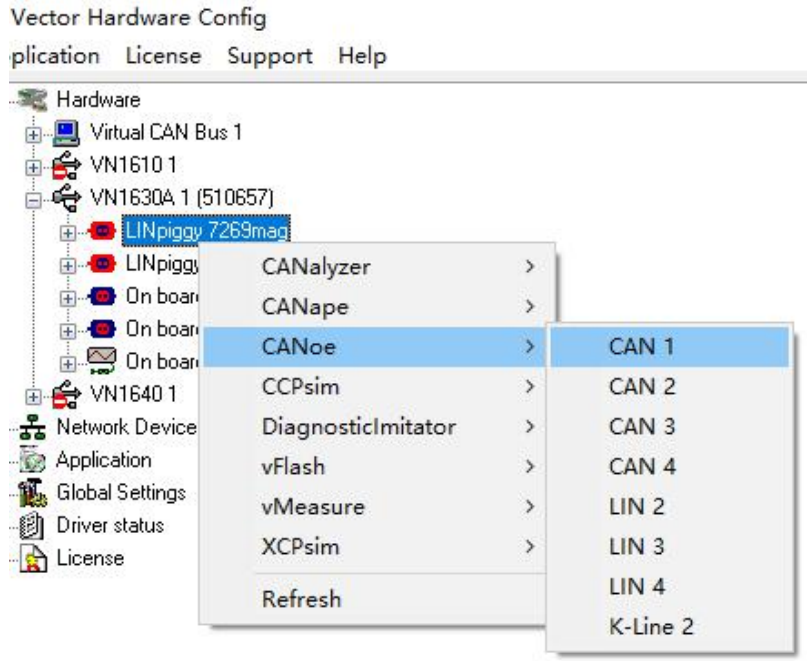
> Vector Hardware Configuration



- **Hardware**
  - 硬件接口通道和收发器
  - 应用通道信息
- **Application**
  - 列出已安装工具
  - 硬件通道应用情况
- **General Information**
  - 硬件基本信息
  - 设备状态
- **License**
  - License 信息（License旧模式）

通道映射

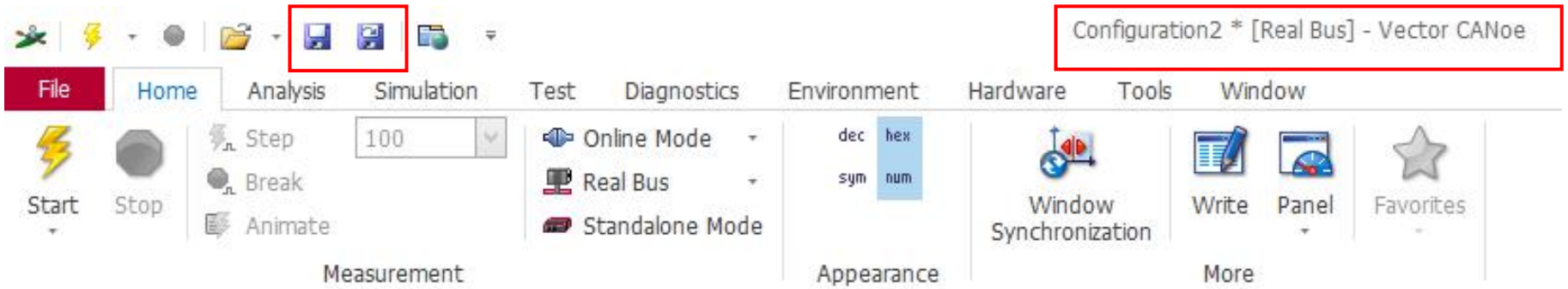
>通道映射



1. 选择 piggy
2. 选择应用软件
3. 选择应用软件的通道

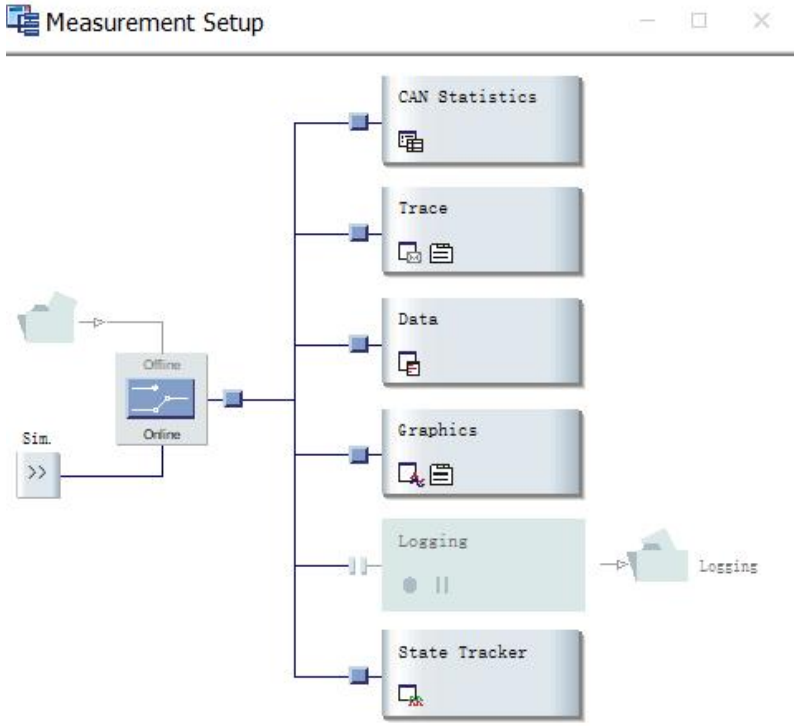
保存工程

- 保存工程配置
- CANoe工程保存文件格式（*.cfg)

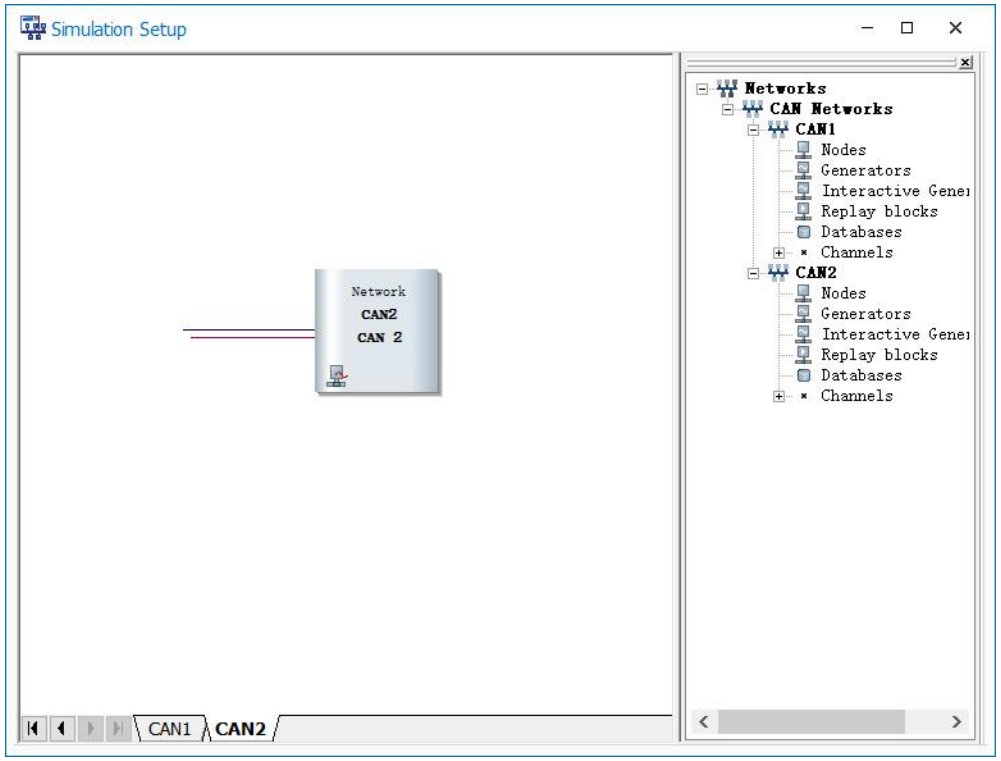


基本配置窗口

> 测量设置与仿真设置：



Measurement Setup

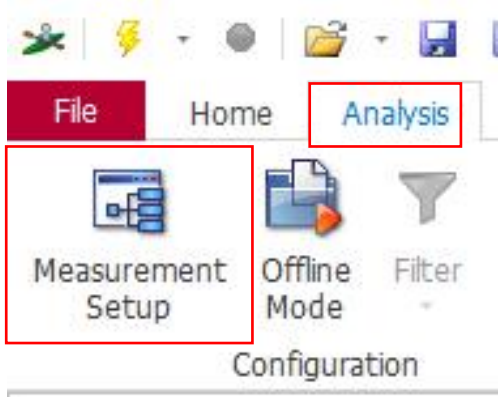


Simulation Setup

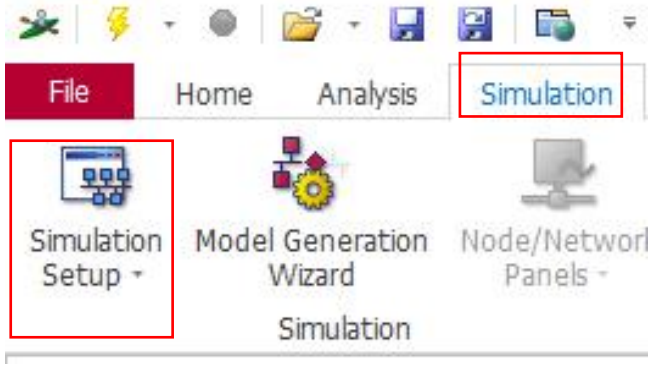
测量和仿真窗口

> 主窗口

- Measurement Setup
  - 分析窗口
  - 功能块
  - 数据记录
  - Online-/Offline数据分析



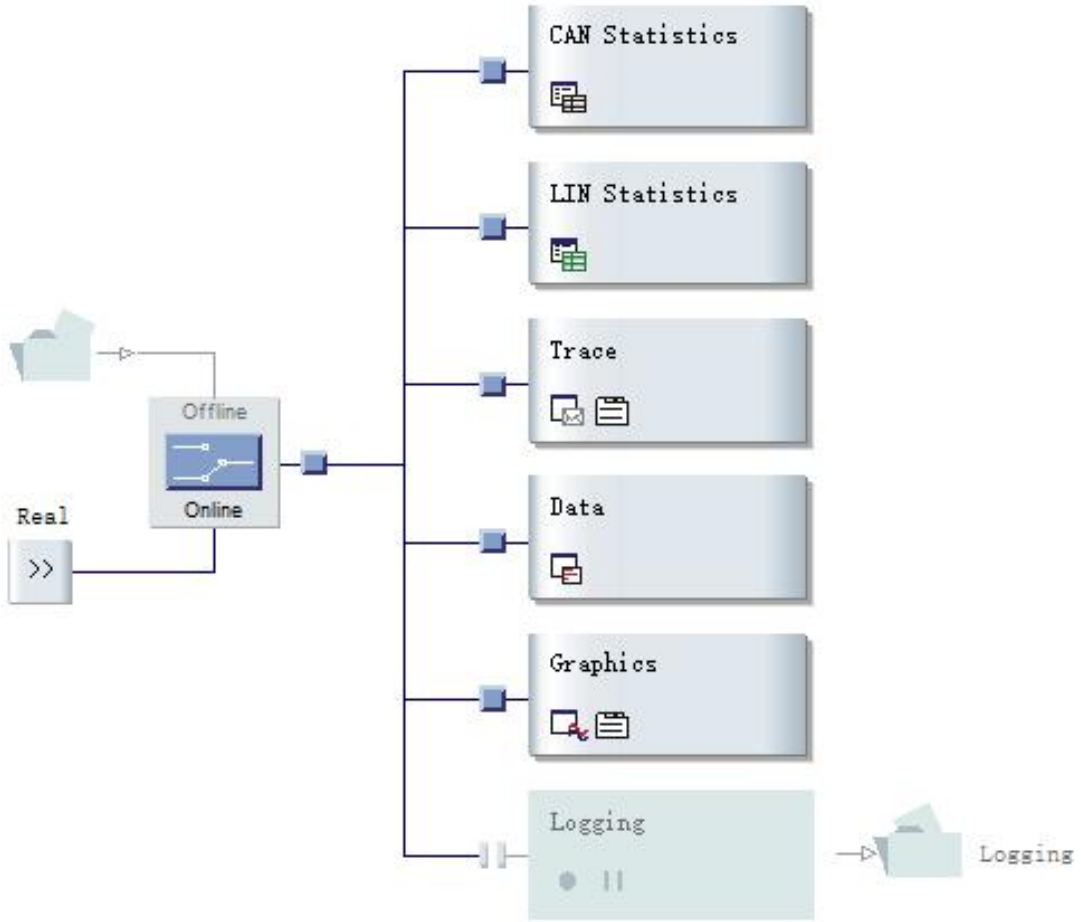
- Simulation Setup
  - 残余总线仿真
  - 报文发送
  - 数据库加载
  - 配置多网络



Measurement Setup

> Measurement Setup

- 管理所有分析窗口和记录
- 配置数据流和处理

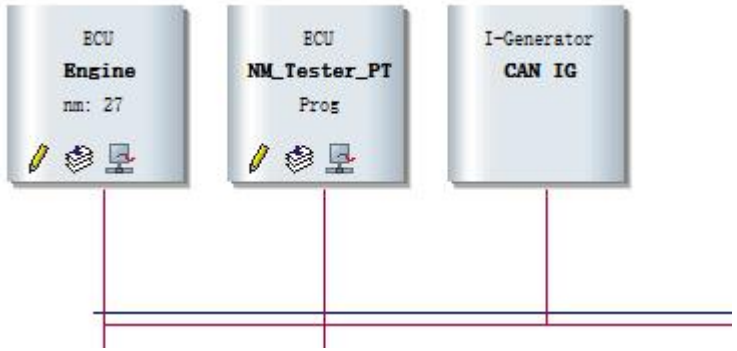


Simulation Setup

> Simulation Setup

根据仿真的程度，功能如下

- 仿真所有节点
- CANoe 脱离真实总线操作
- 完全仿真

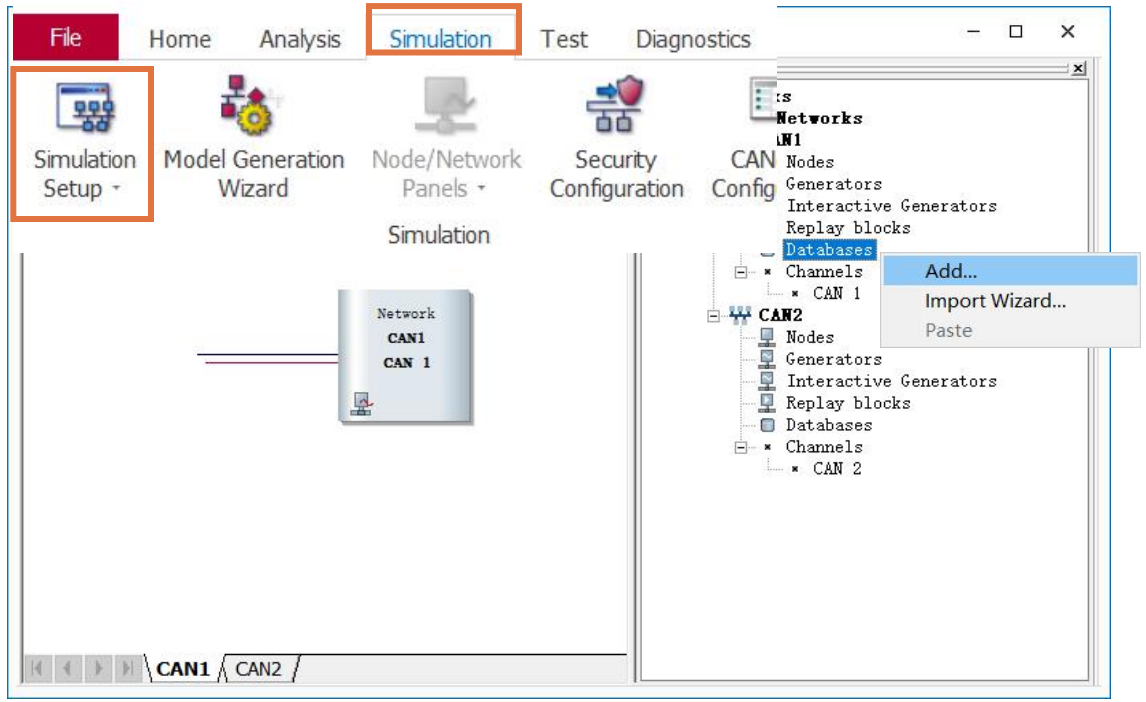


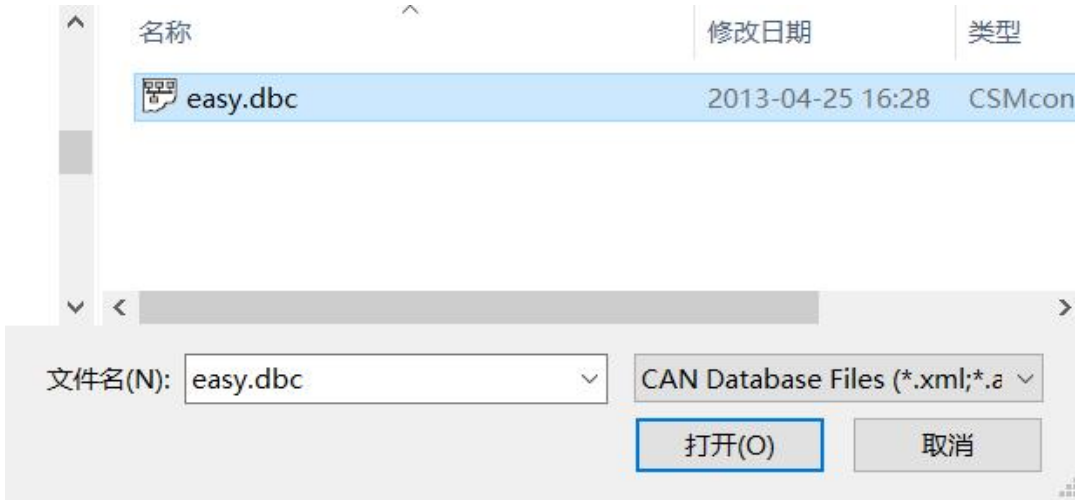
- 残余总线仿真
- 一个或多个节点接入真实总线
- 仿真其余节点

加载DBC数据库

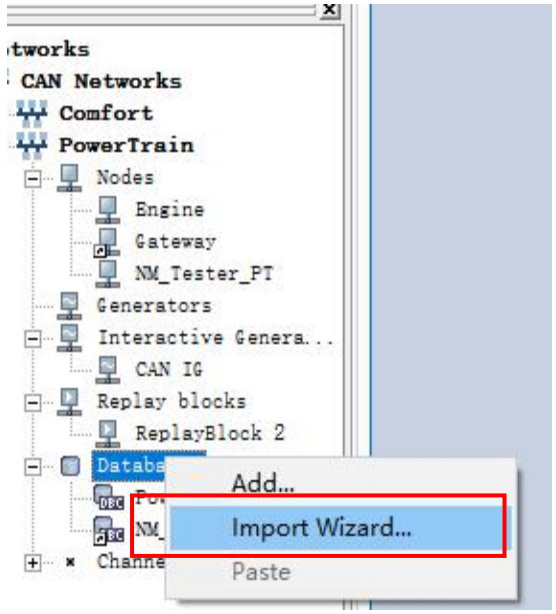
> 加载数据库设置

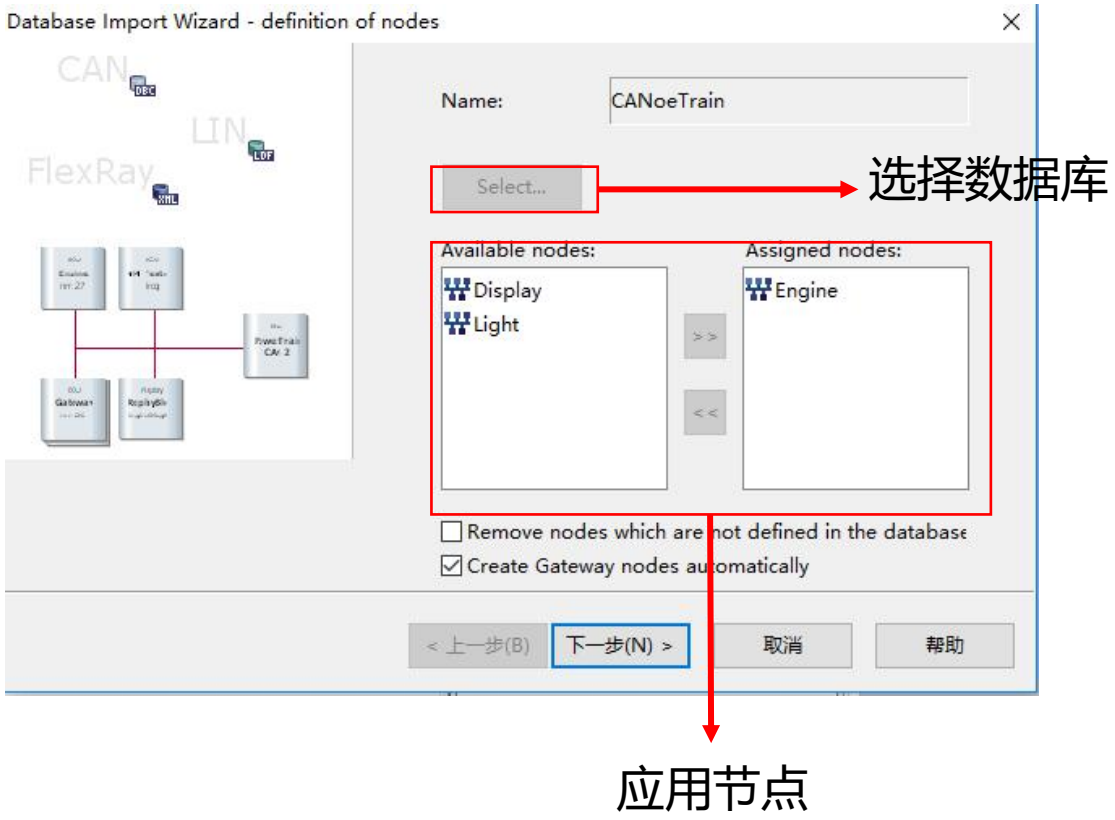
Simulation | Simulation Setup……





加载DBC数据库


tworks
CAN Networks
Comfort
PowerTrain
Nodes
Engine
Gateway
NM_Tester_PT
Generators
Interactive Genera...
CAN IG
Replay blocks
ReplayBlock 2
Add...
Import Wizard...
Paste

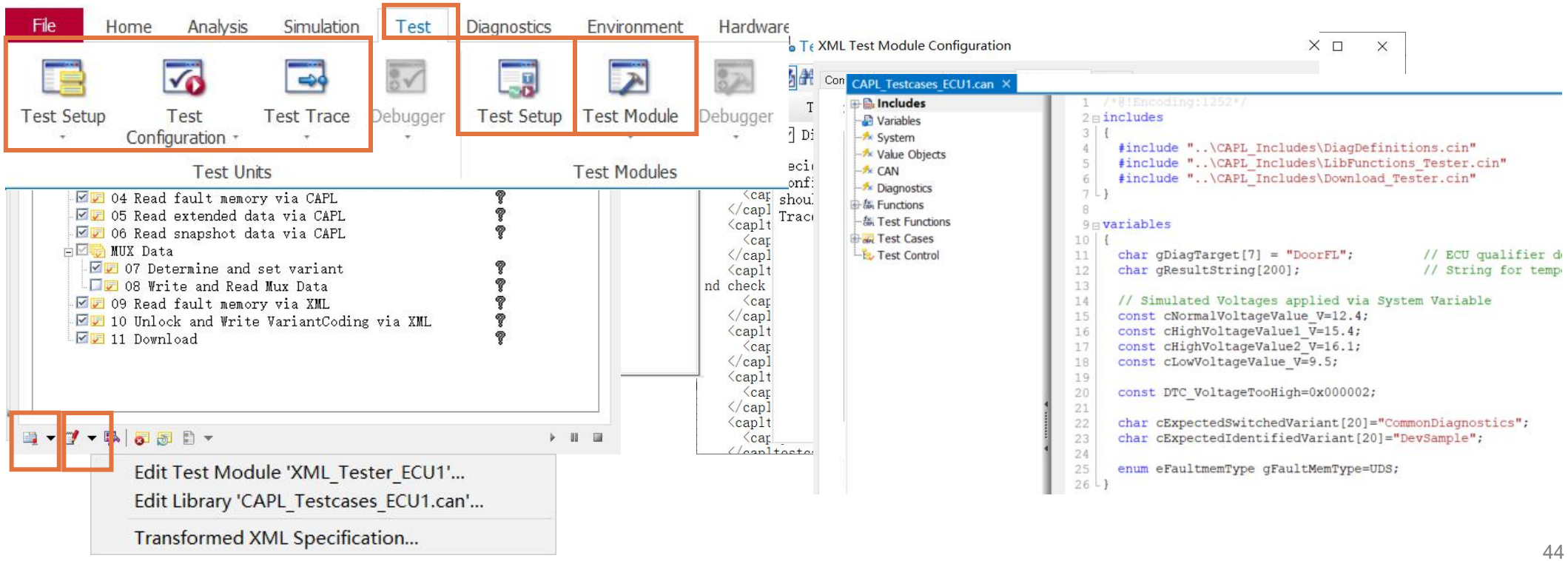

Database Import Wizard - definition of nodes
CAN
LIN
FlexRay
Name:
CANoeTrain
Select...
选择数据库
Available nodes:
Assigned nodes:
Display
Light
Engine
Remove nodes which are not defined in the database
Create Gateway nodes automatically
< 上一步(B)
下一步(N) >
取消
帮助
应用节点

测试窗口

> 测试设置

Test | Test Setup/Test Module……

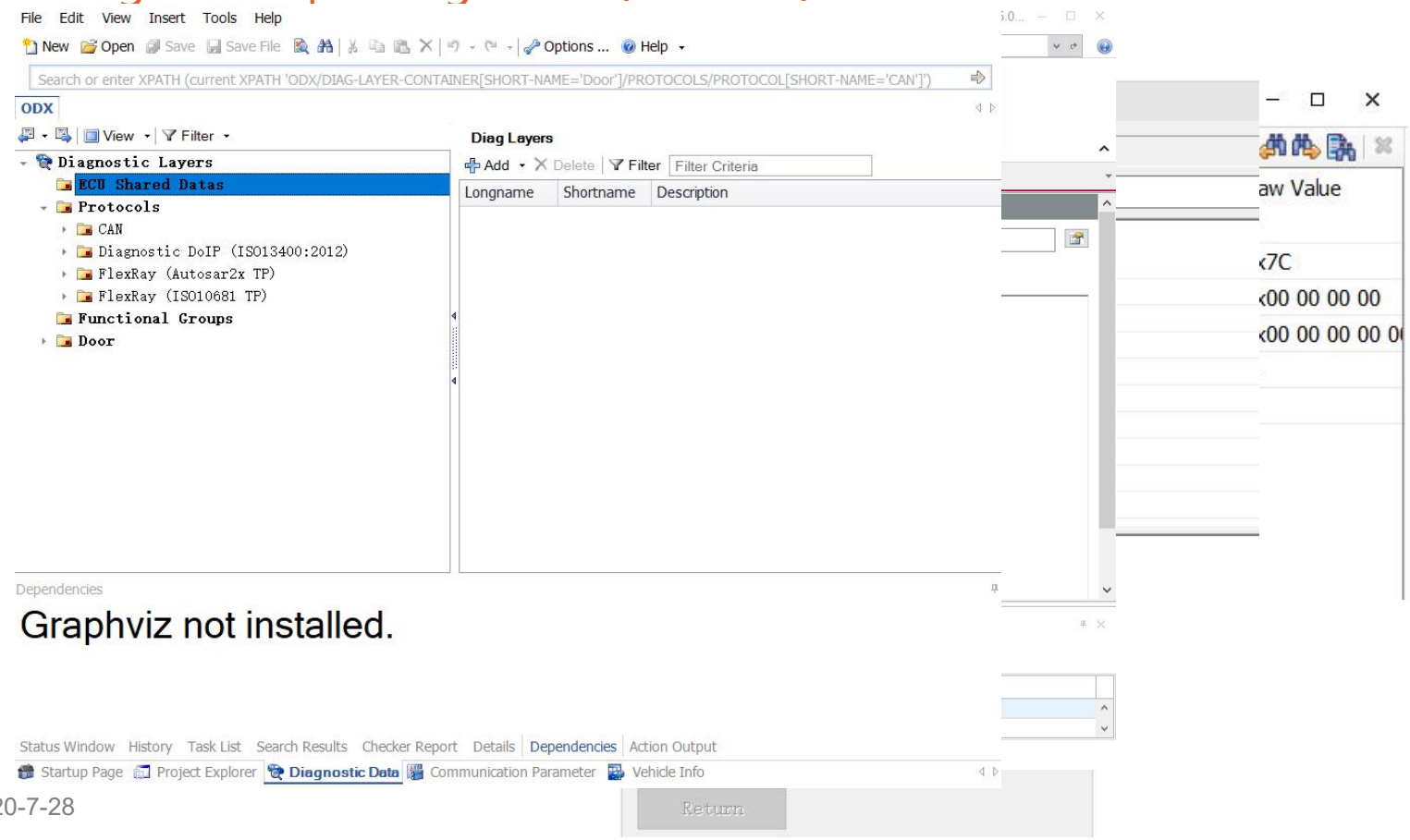
分为Test Units和Test Module两种测试配置：



诊断窗口

> 诊断设置

Diagnostics | Configuration/Control/Tools……



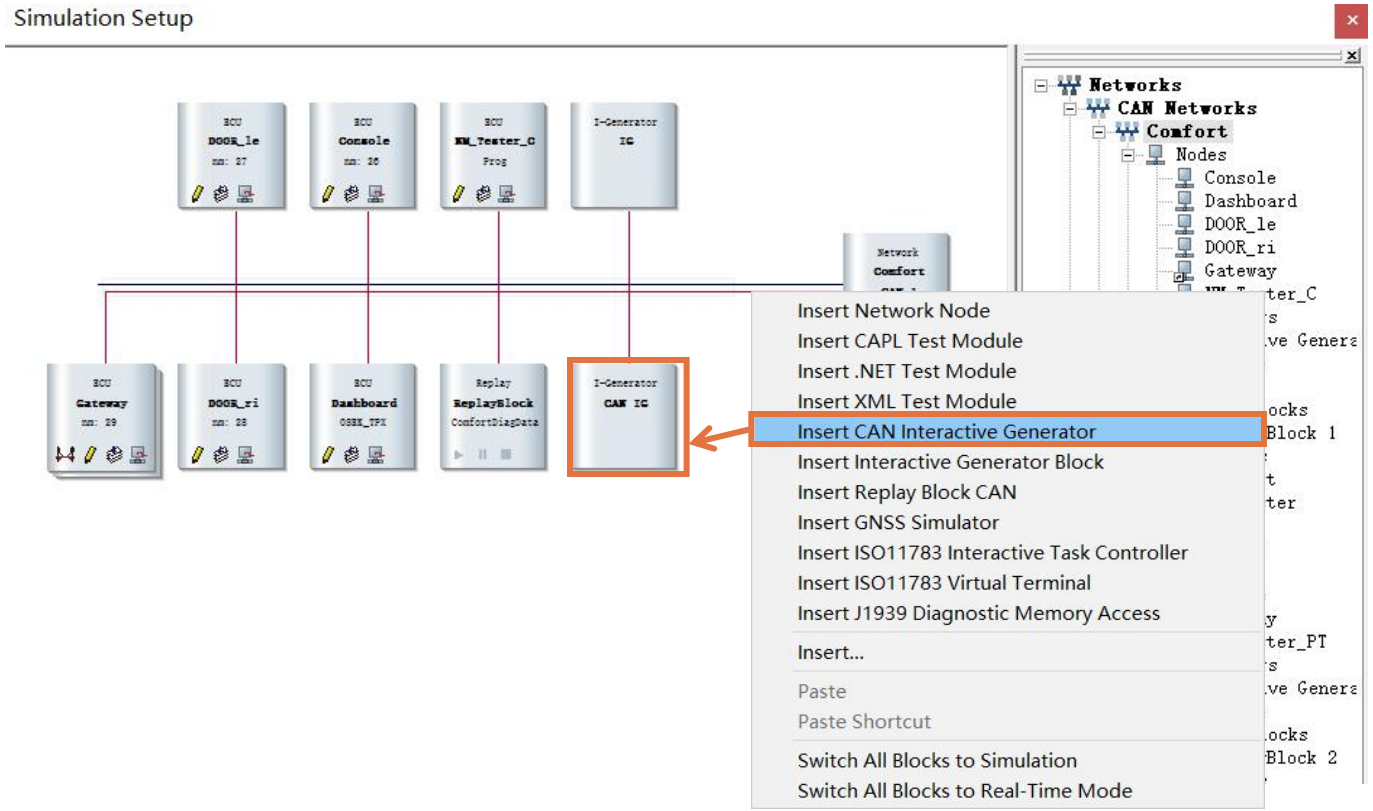
目录

CONTENTS

IG模块

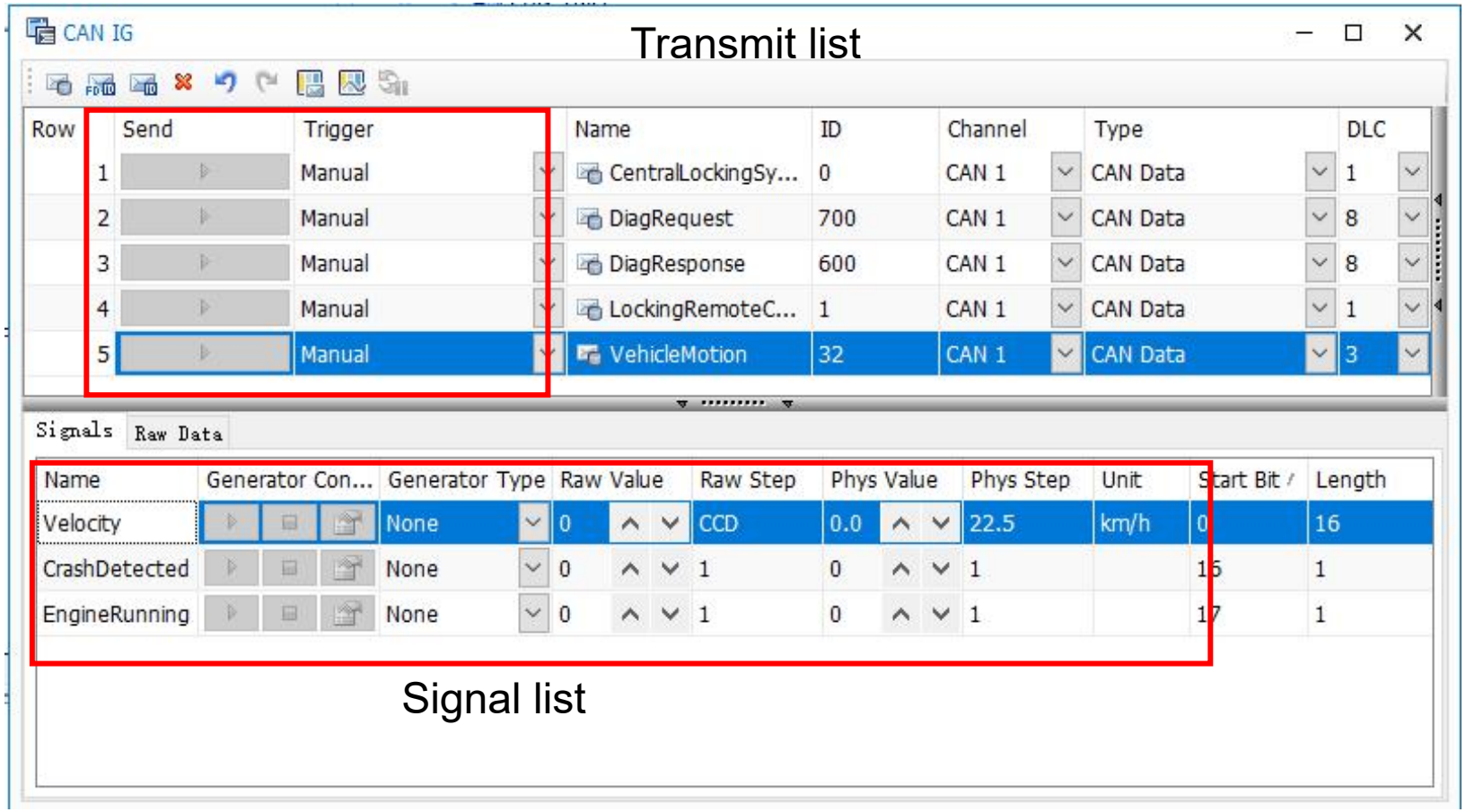
> IG模块

报文发送模块称为IG模块，分为CAN IG模块和IG模块（也包括CAN）：



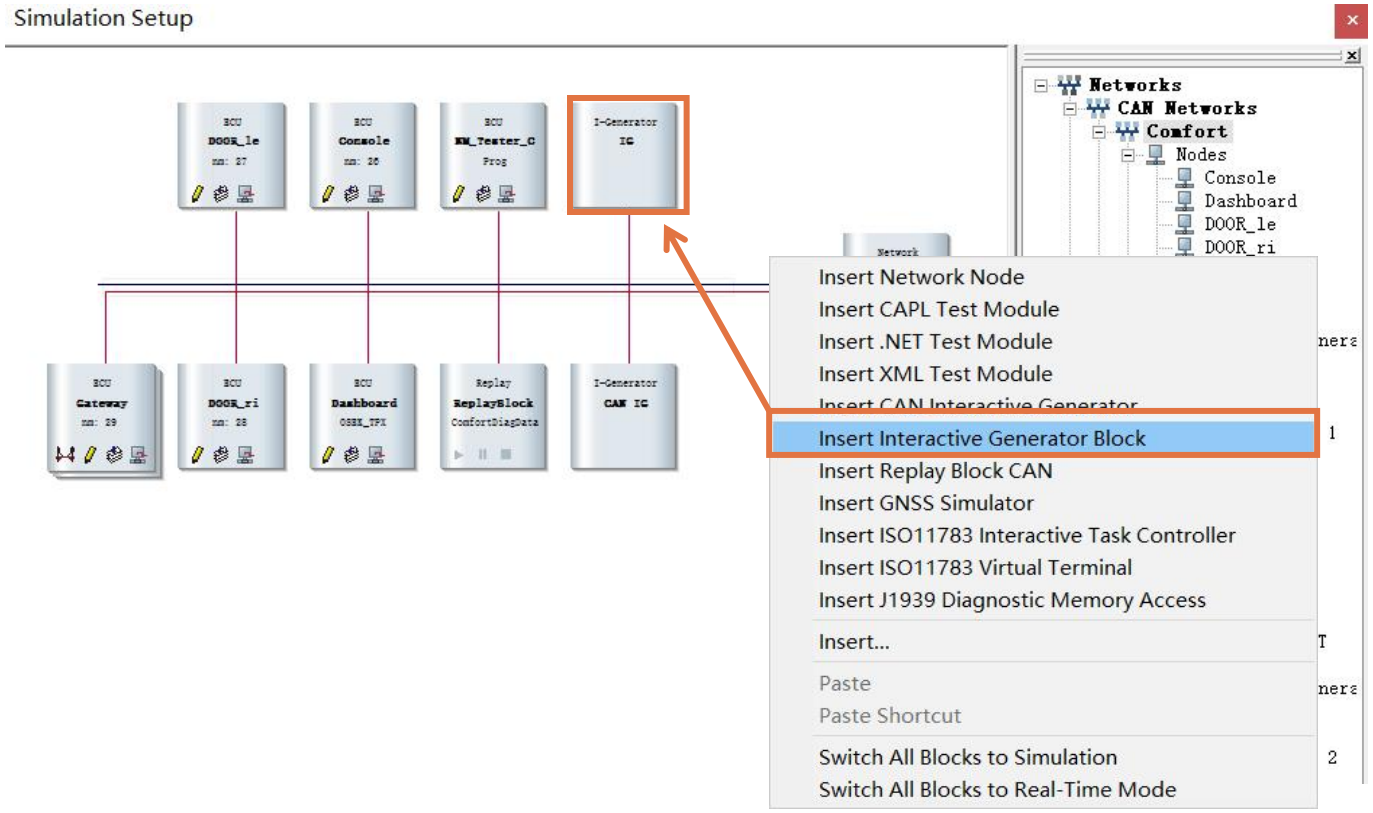
CAN IG模块

> Interactive Generator for CAN



IG模块

> IG模块

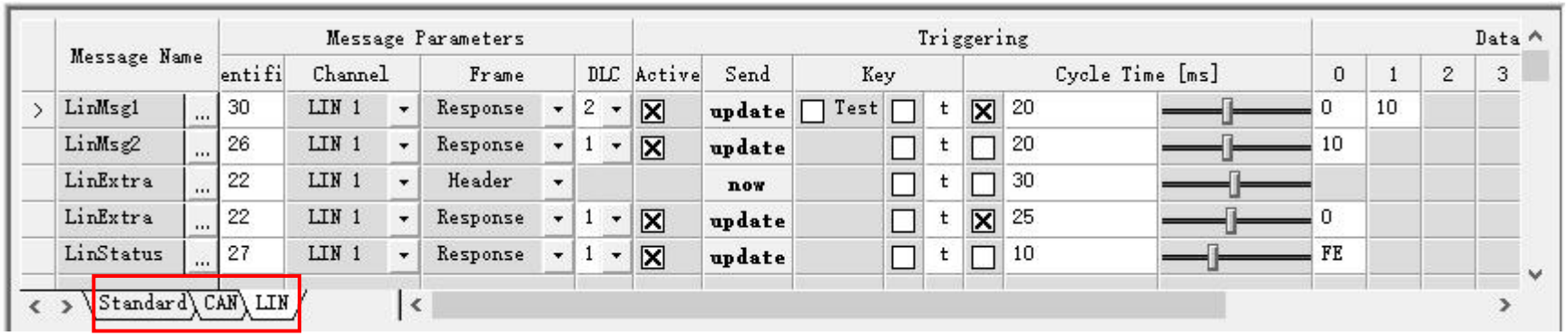

Simulation Setup
Networks
CAN Networks
Comfort
Nodes
Console
Dashboard
DOOR_le
DOOR_ri
Insert Network Node
Insert CAPL Test Module
Insert .NET Test Module
Insert XML Test Module
Insert Interactive Generator Block
Insert Replay Block CAN
Insert GNSS Simulator
Insert ISO11783 Interactive Task Controller
Insert ISO11783 Virtual Terminal
Insert J1939 Diagnostic Memory Access
Insert...
Paste
Paste Shortcut
Switch All Blocks to Simulation
Switch All Blocks to Real-Time Mode

IG模块

> IG模块


触发条件
IG
Message Parameters
Triggering
Data
Message Name
entifi
Channel
Frame
DLC
Active
Send
Key
Cycle Time [ms]
0
1
2
3
LinMsg1
30
LIN 1
Response
2
update
Test
t
20
0
10
LinMsg2
26
LIN 1
Response
1
update
t
20
10
LinExtra
22
LIN 1
Header
now
t
30
LinExtra
22
LIN 1
Response
1
update
t
25
0
LinStatus
27
LIN 1
Response
1
update
t
10
FE
Standard
CAN
LIN
报文数据域
New
Clone
Special Frame
Delete
Cut
Copy
Paste
Layout
SB
Signal Name
Raw Value
Phys Value
Unit
Dec
Phys Step
Inc
Wave form
0
Sine
0
0
13
+
Sine
8
Ramp
10
16
13
+
Ramps and pul
Close
Help
信号列表
Default input mode: LIN

> IG模块—发送列表



	Message Name		Message Parameters				Triggering								Data			
			entifi	Channel	Frame	DLC	Active	Send	Key			Cycle Time [ms]			0	1	2	3
>	LinMsg1	...	30	LIN 1	Response	2	☒	update	☐ Test	☐	t	☒	20		0	10		
	LinMsg2	...	26	LIN 1	Response	1	☒	update		☐	t	☐	20		10			
	LinExtra	...	22	LIN 1	Header			now		☐	t	☐	30					
	LinExtra	...	22	LIN 1	Response	1	☒	update		☐	t	☒	25		0			
	LinStatus	...	27	LIN 1	Response	1	☒	update		☐	t	☐	10		FE			

Standard CAN LIN

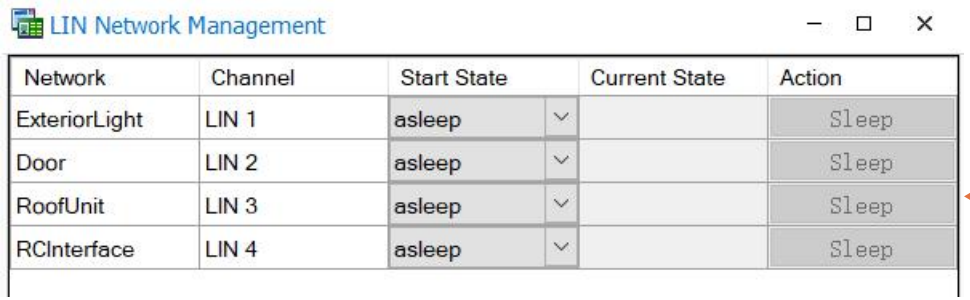
视图方式

- 可配置每条报文的ID、DLC、数据
- 可配置每条报文的发送条件（手动、周期）
- 可通过数据字节/信号值配置每条报文的数据

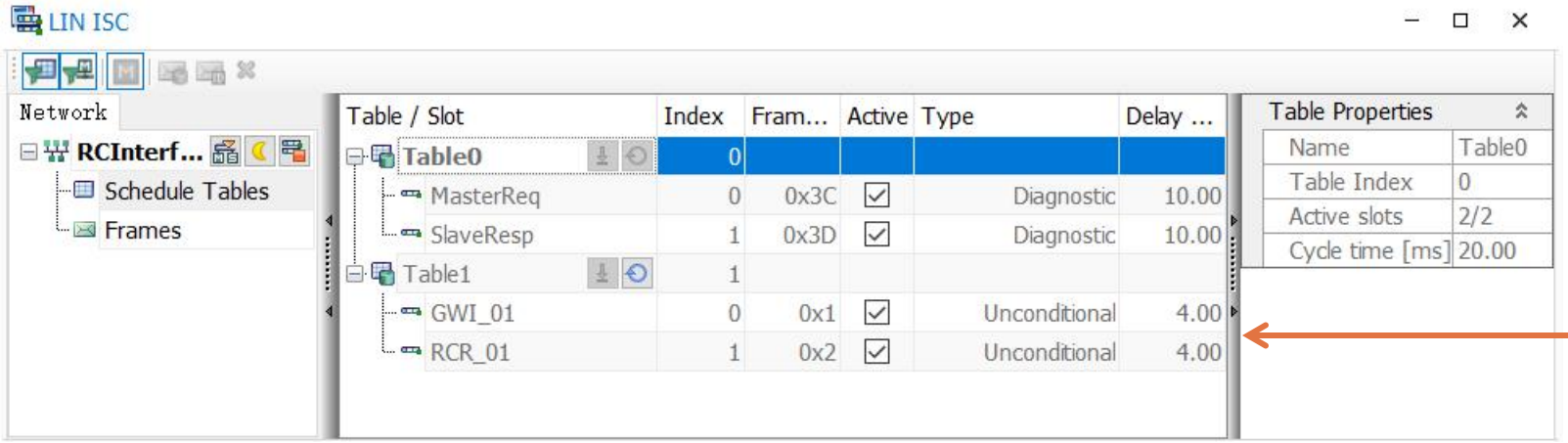
报文发送

> LIN报文管理

对于LIN总线报文，有专门的管理窗口：



报文休眠与唤醒

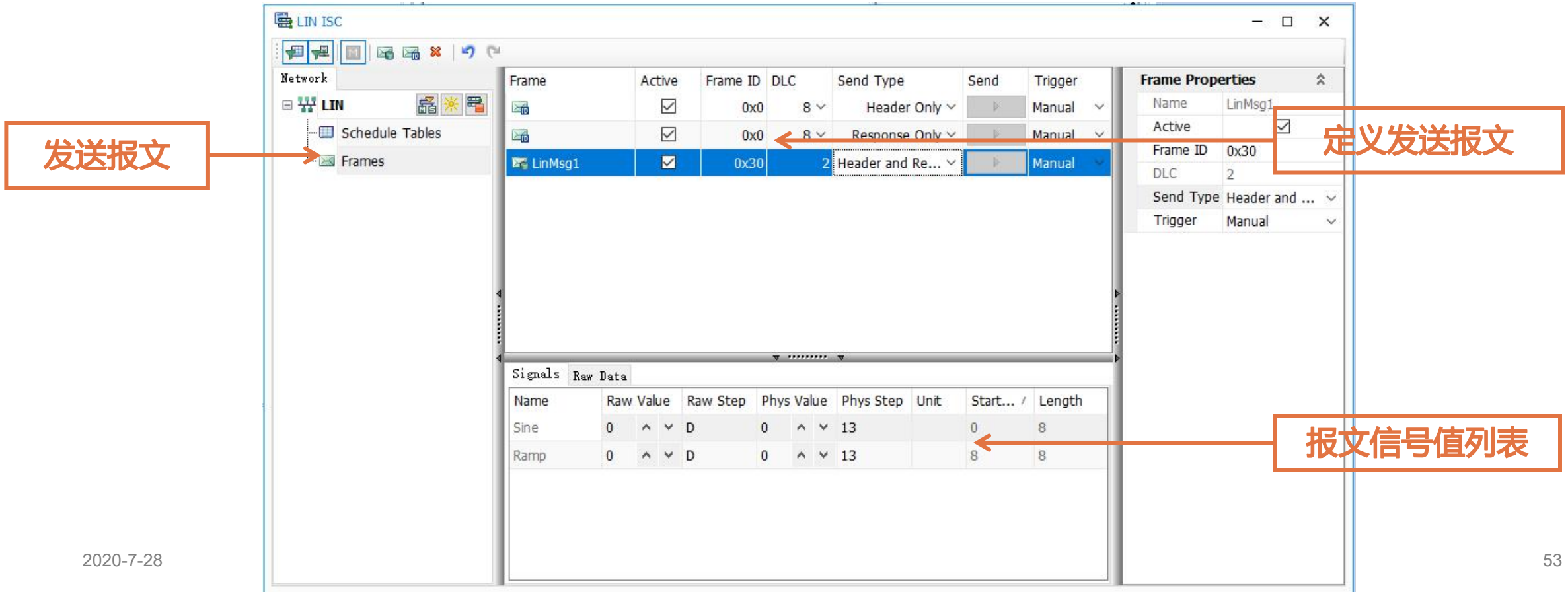


调度表切换

报文发送

> LIN报文管理

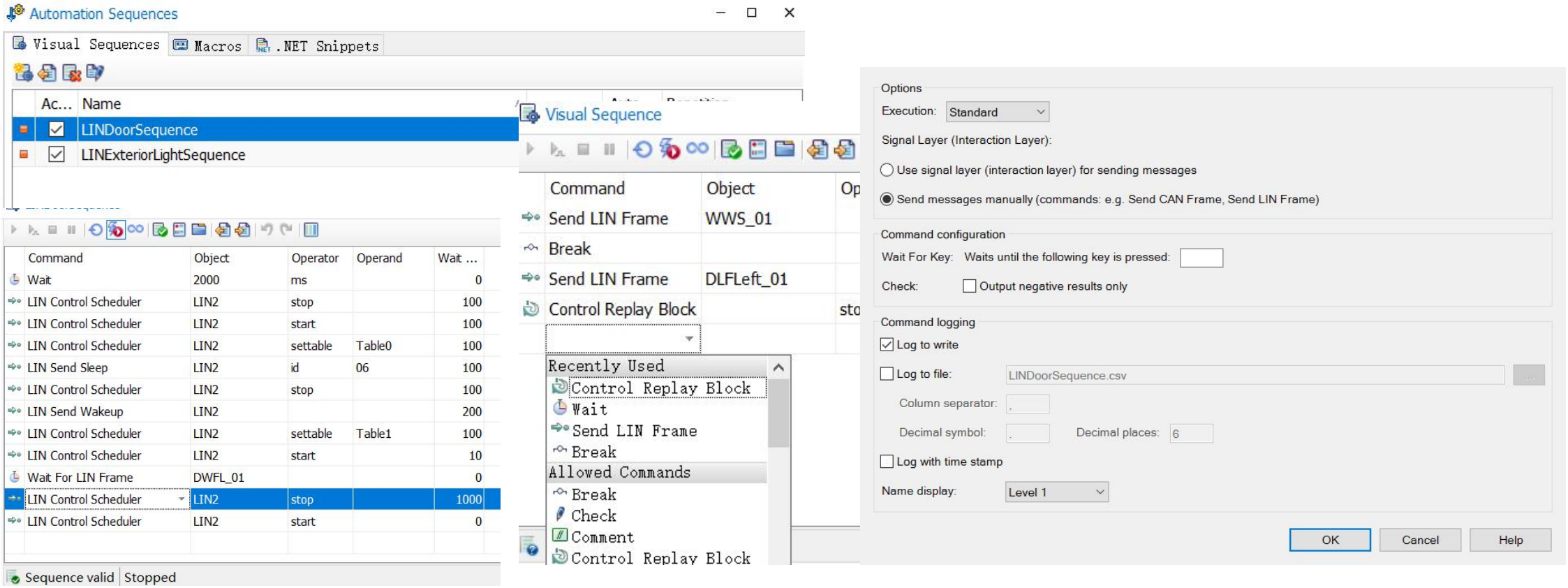
对于LIN总线报文，有专门的管理窗口：



报文发送

> 自动时序

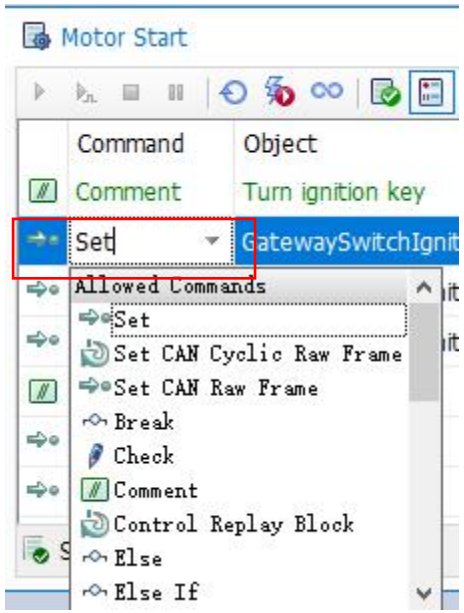
Simulation | Automation |.....

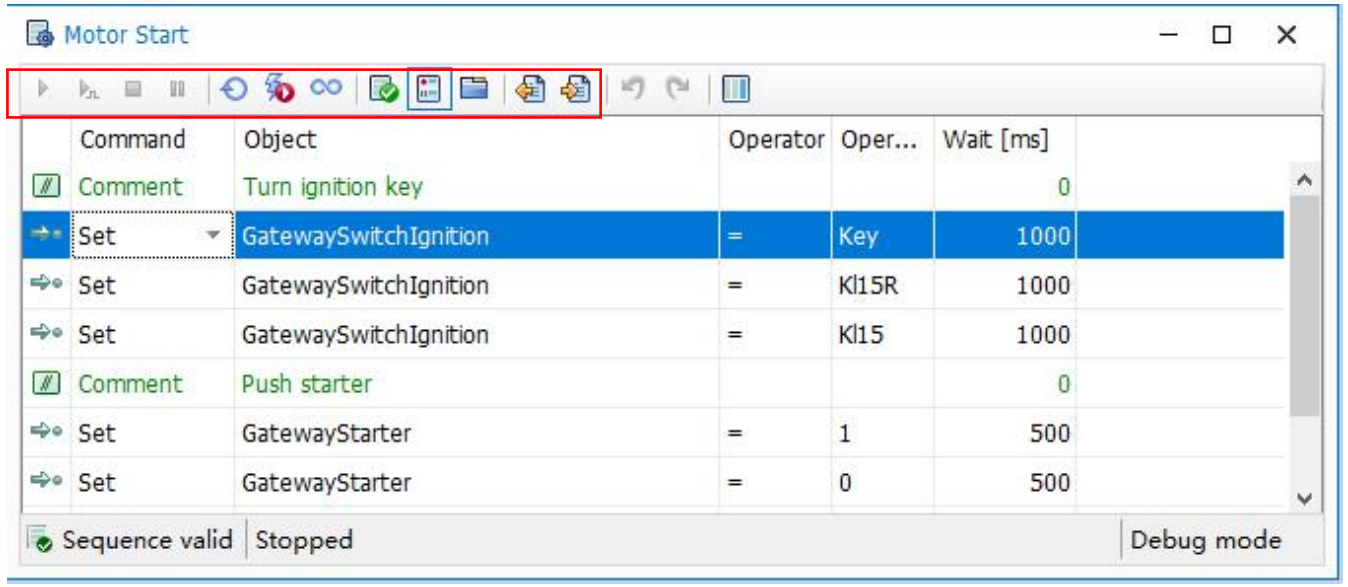


自动发送序列

>自动发送序列

选择和修改序列命令

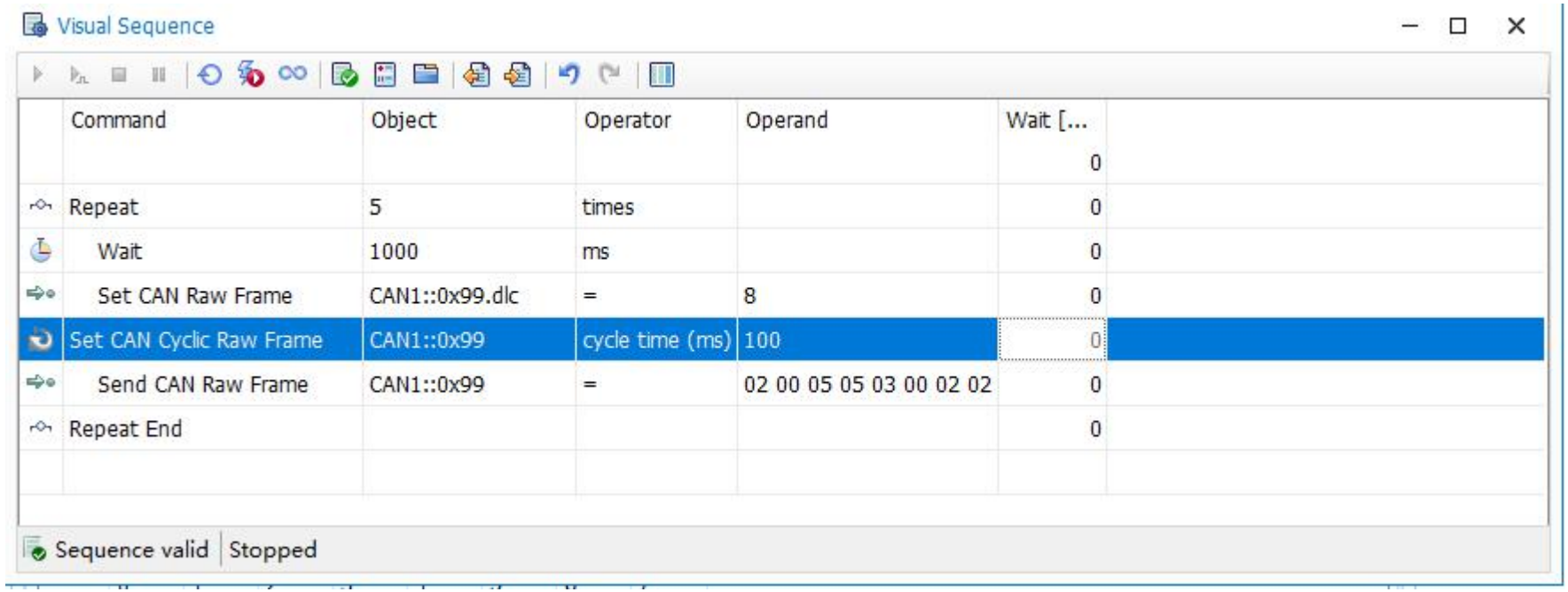




自动发送序列

>自动发送序列

选择和修改序列命令，使用序列发送报文

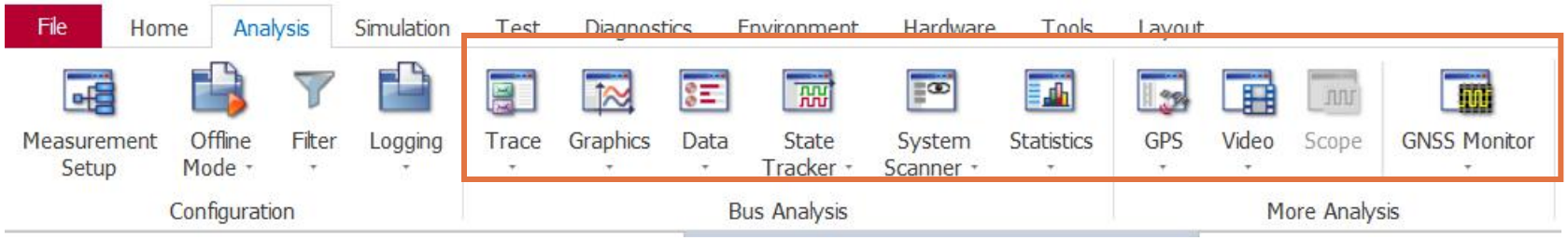


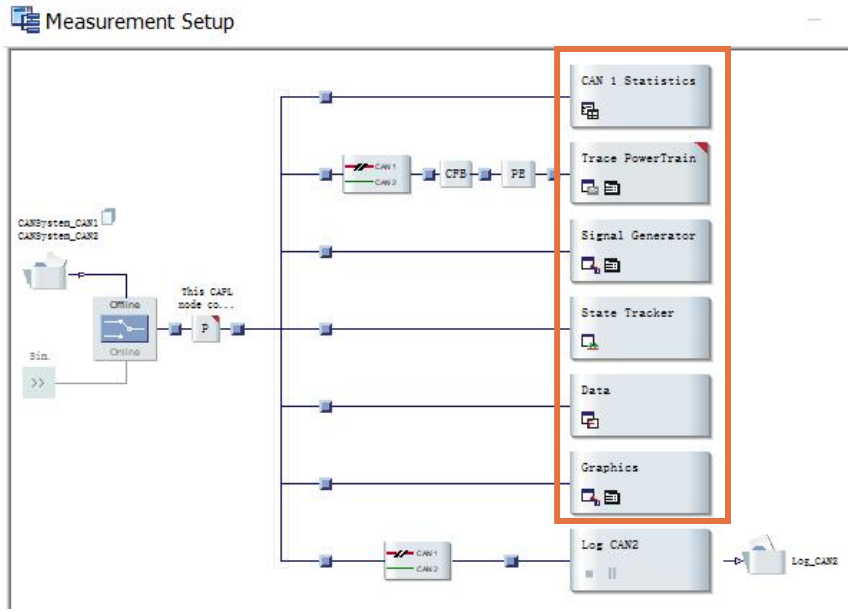
目录

CONTENTS

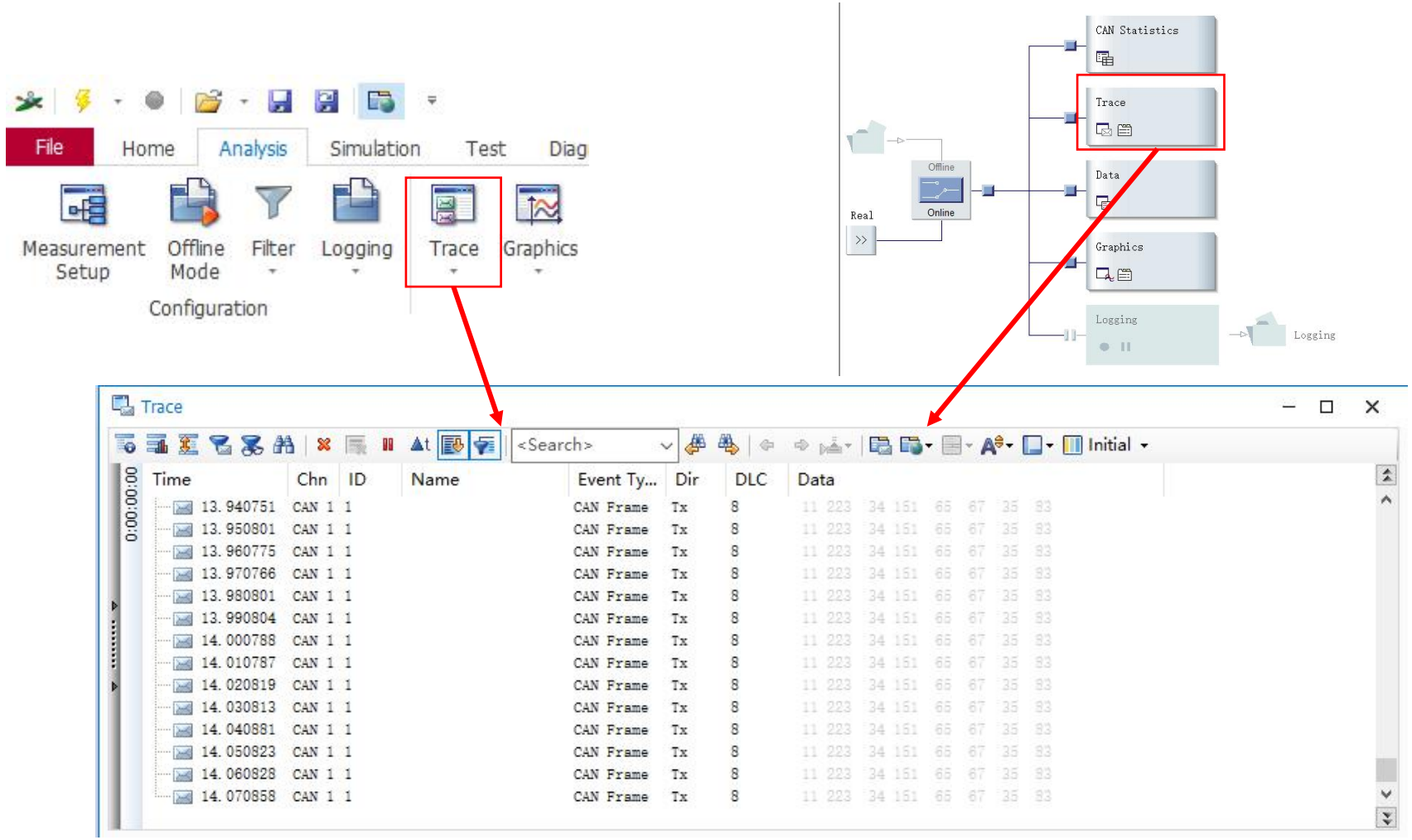
分析窗口

> 窗口类型


File
Home
Analysis
Simulation
Test
Diagnostics
Environment
Hardware
Tools
Layout
Measurement Setup
Offline Mode
Filter
Logging
Trace
Graphics
Data
State Tracker
System Scanner
Statistics
GPS
Video
Scope
GNSS Monitor
Configuration
Bus Analysis
More Analysis


Measurement Setup
CAN 1 Statistics
Trace PowerTrain
Signal Generator
State Tracker
Data
Graphics
Log CAN2
CPB
PE
Offline
Online
Log_CAN2

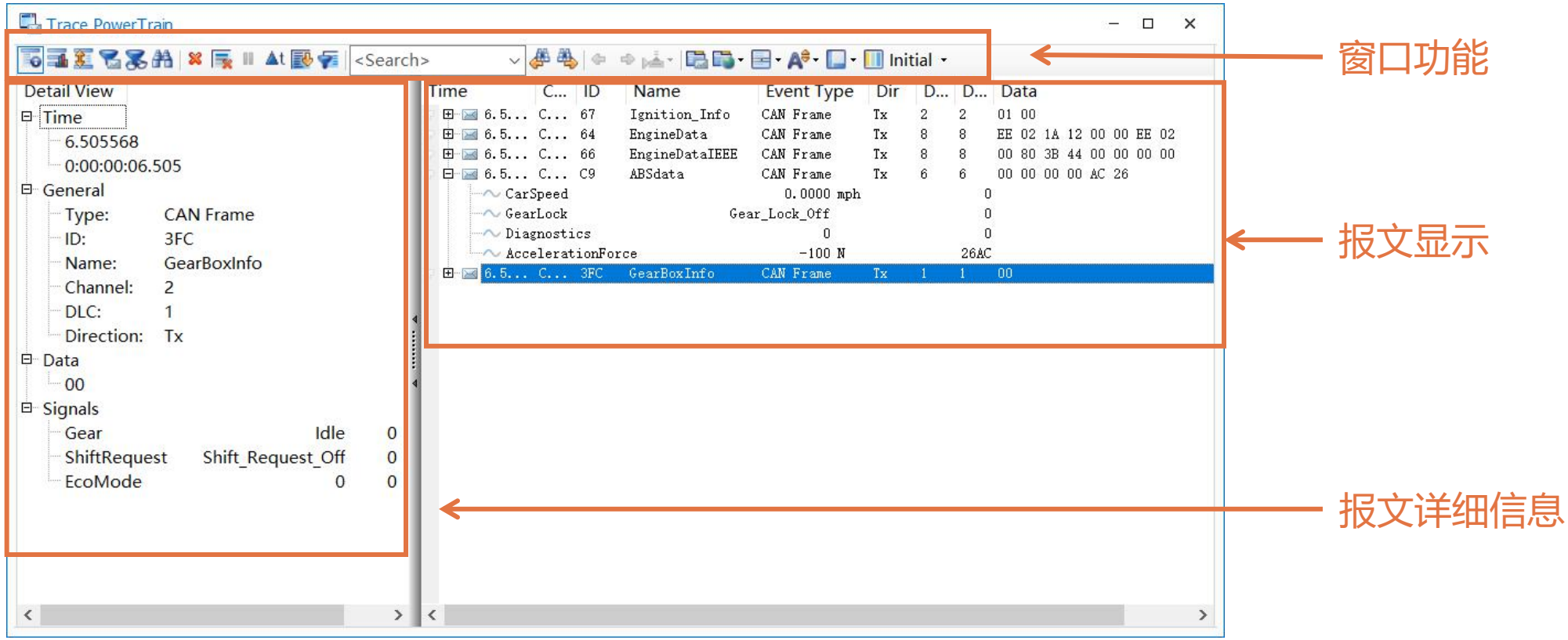
Trace Window



分析窗口

> Trace 窗口

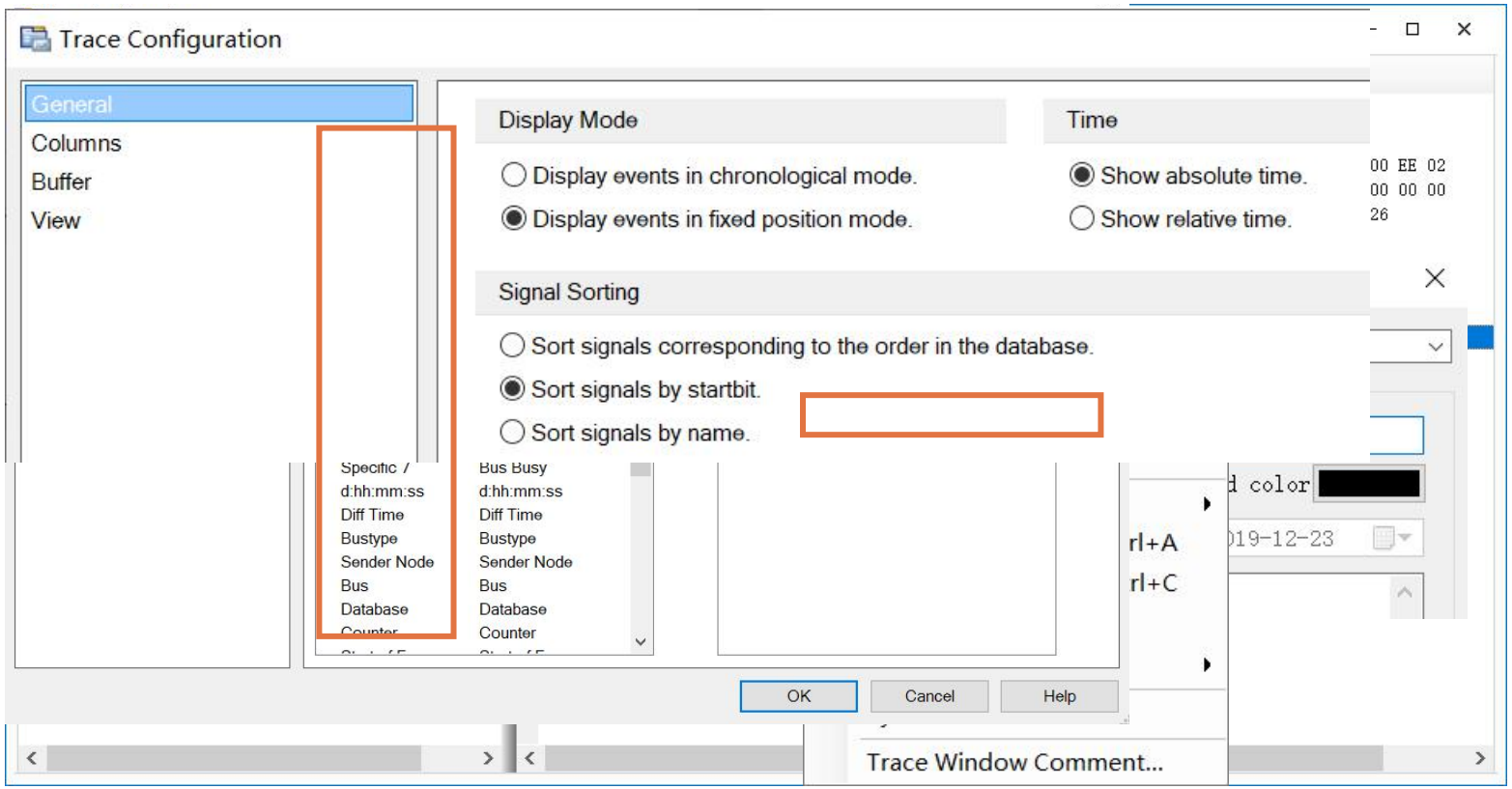
Trace窗口包含报文显示，解析，记录，分析等功能。



分析窗口

> Trace 窗口

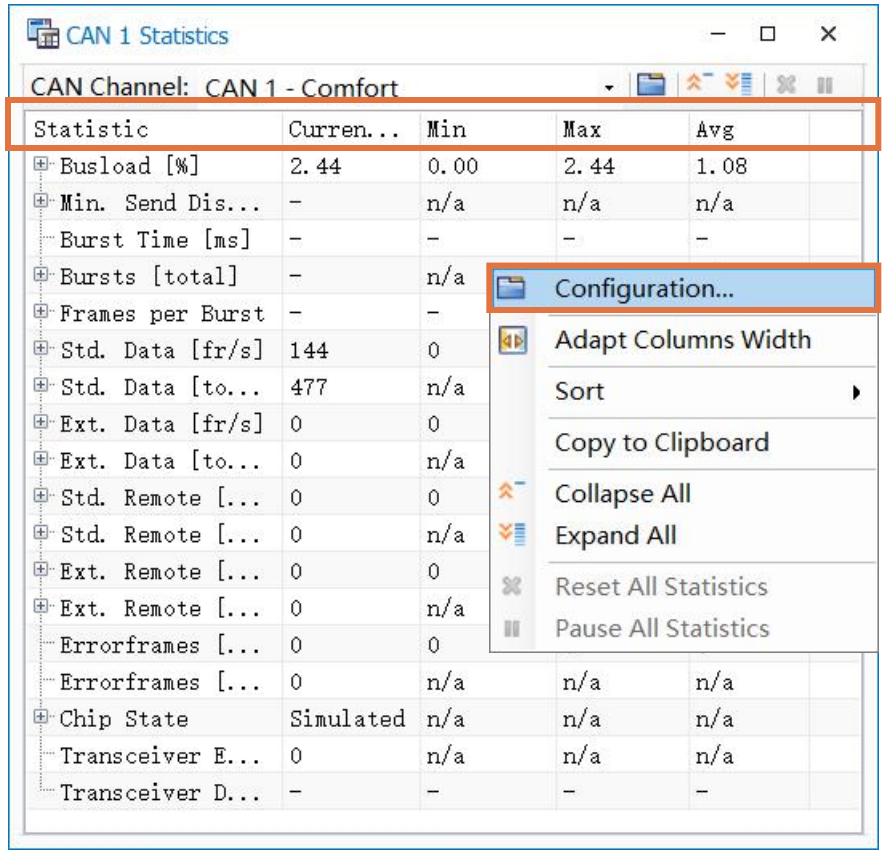
添加标记，增加列项：

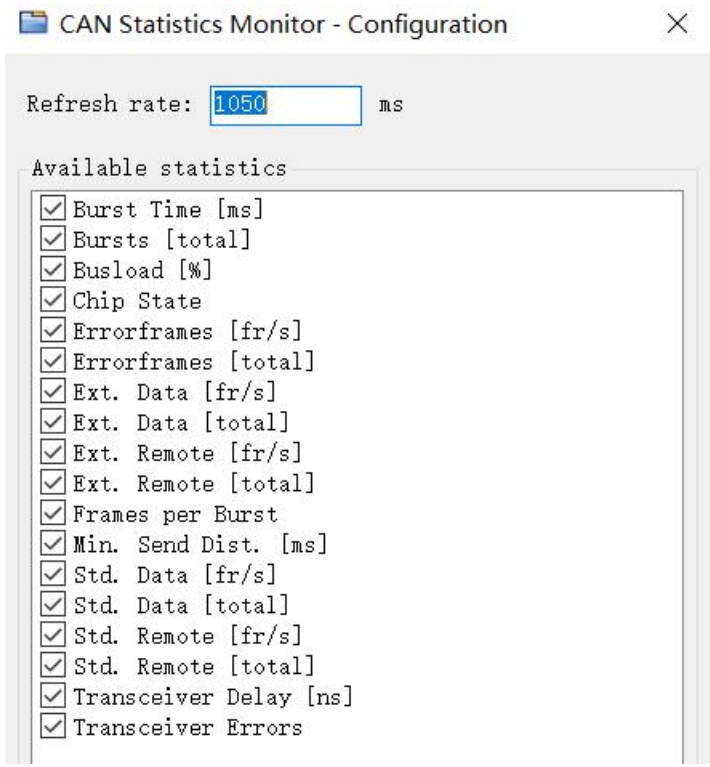


分析窗口

> Statistics窗口

统计窗口，显示总线报文的最大最小值，平均值，当前值信息：

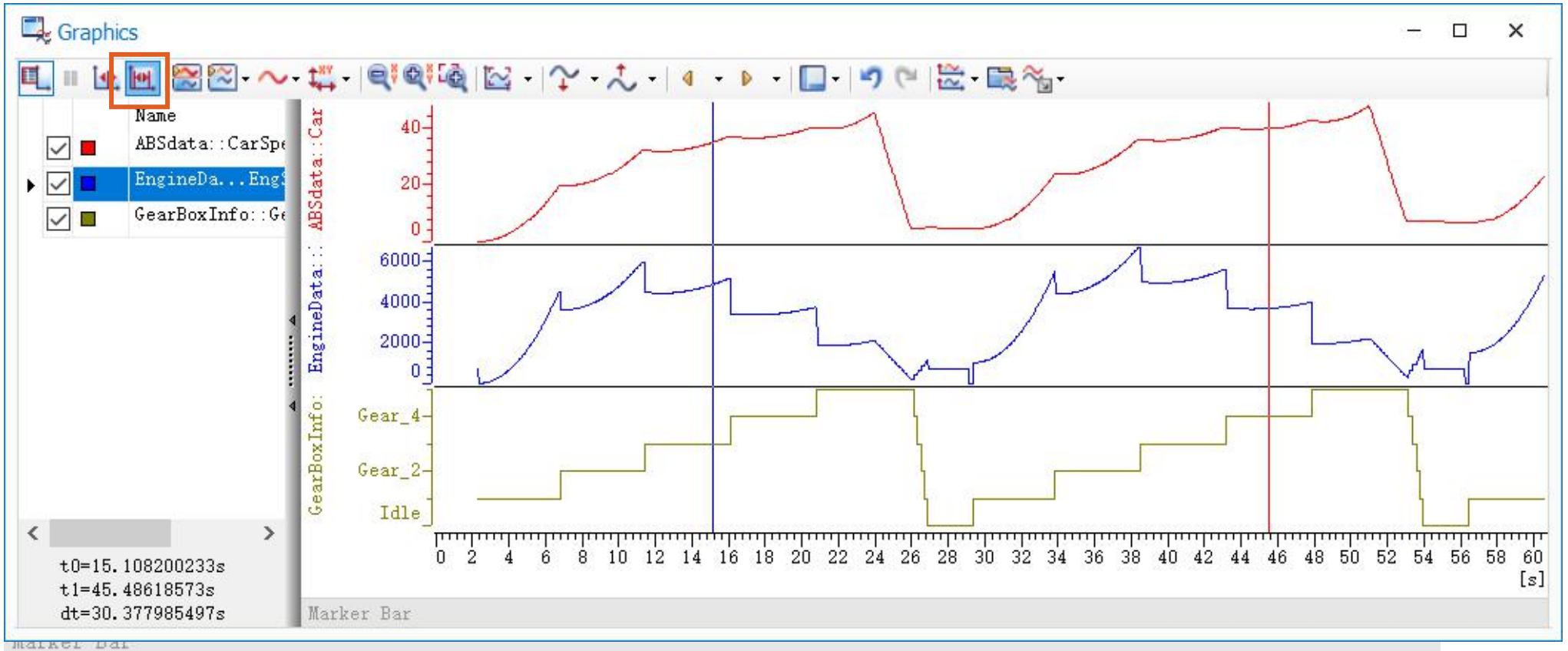




分析窗口-Graphic

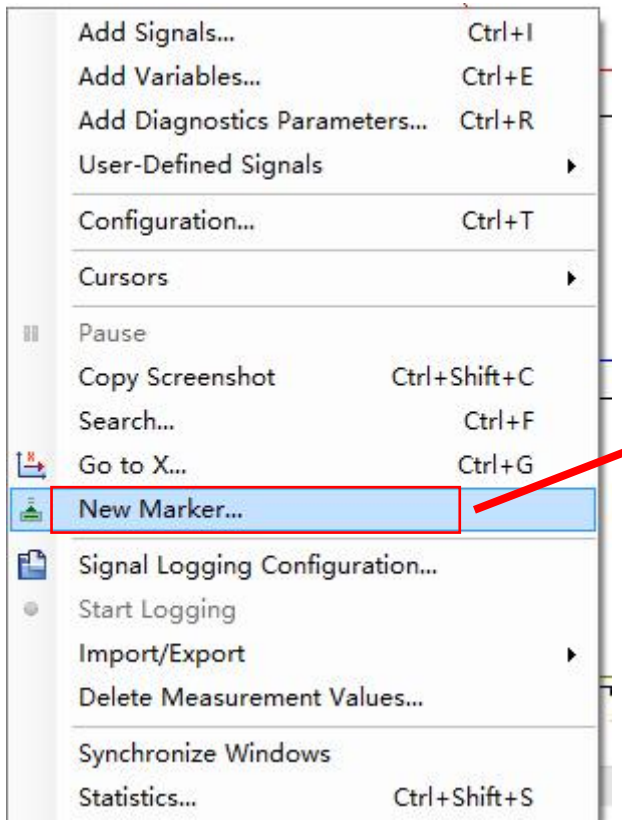
> Graphic 窗口

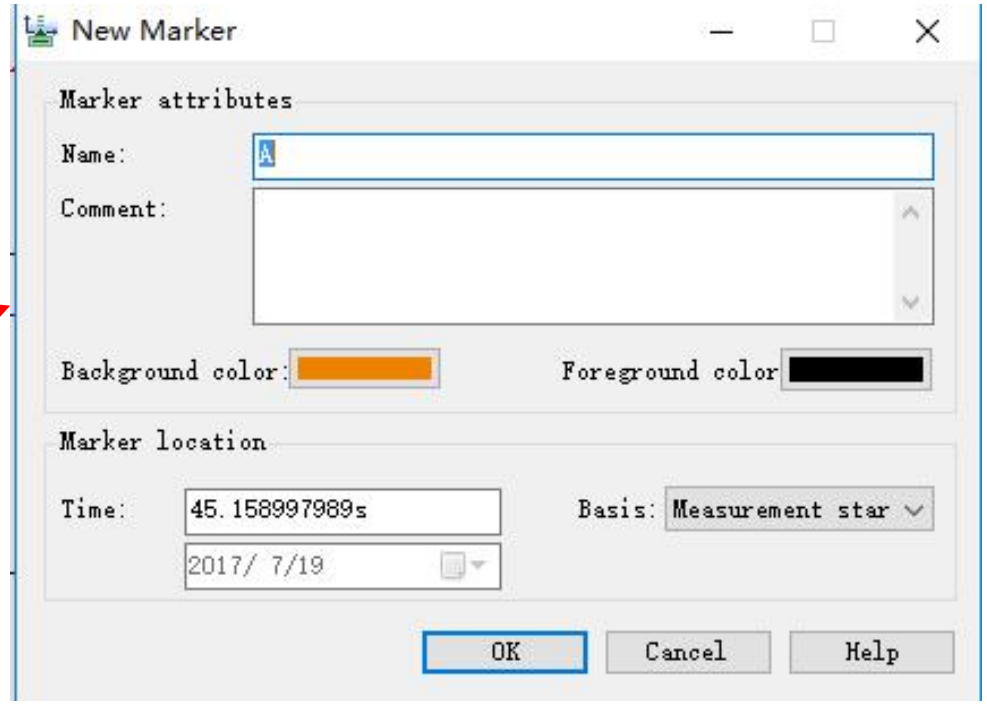
图形窗口提供各种信号的图形显示:



Graphic 窗口

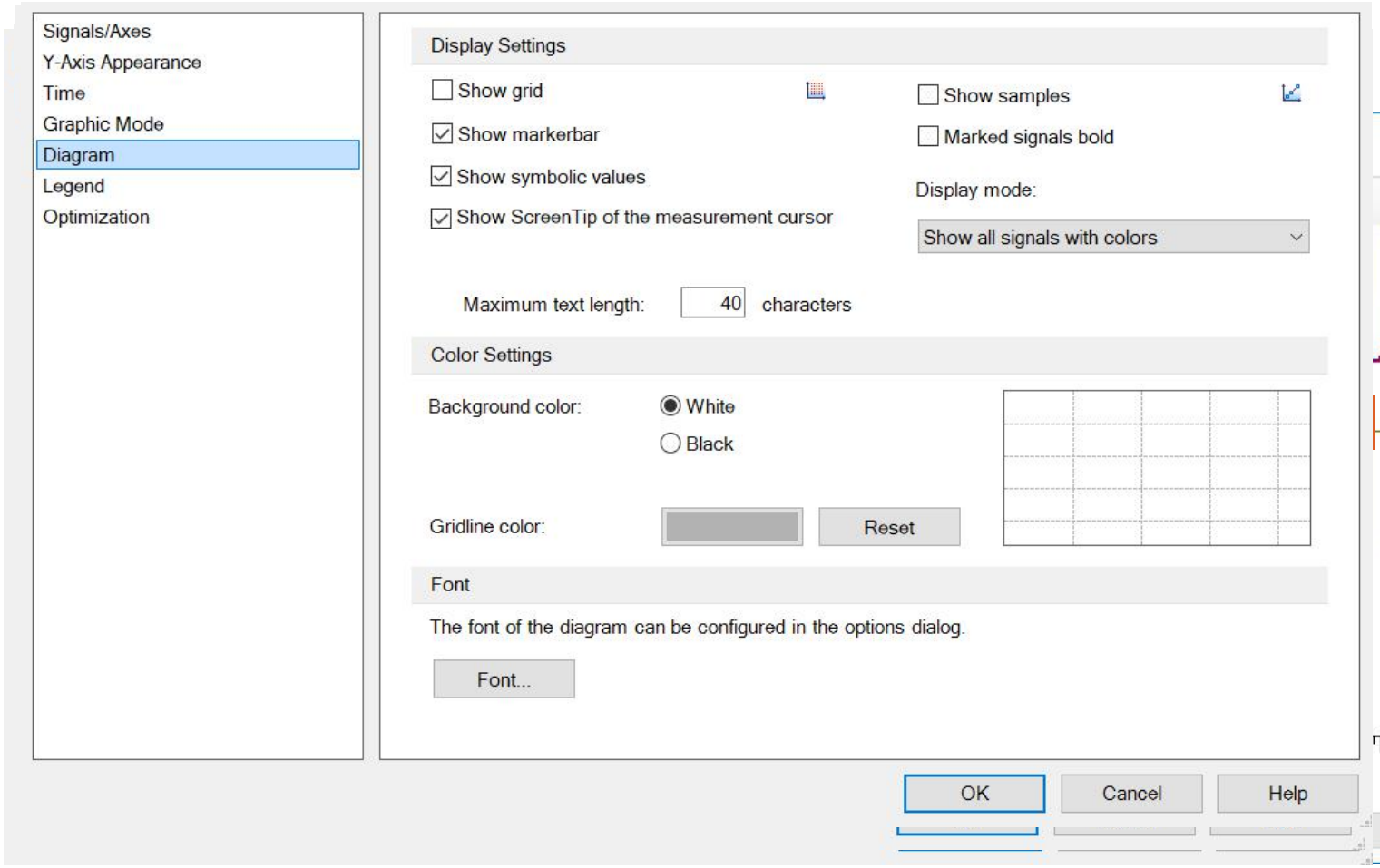
> Marker

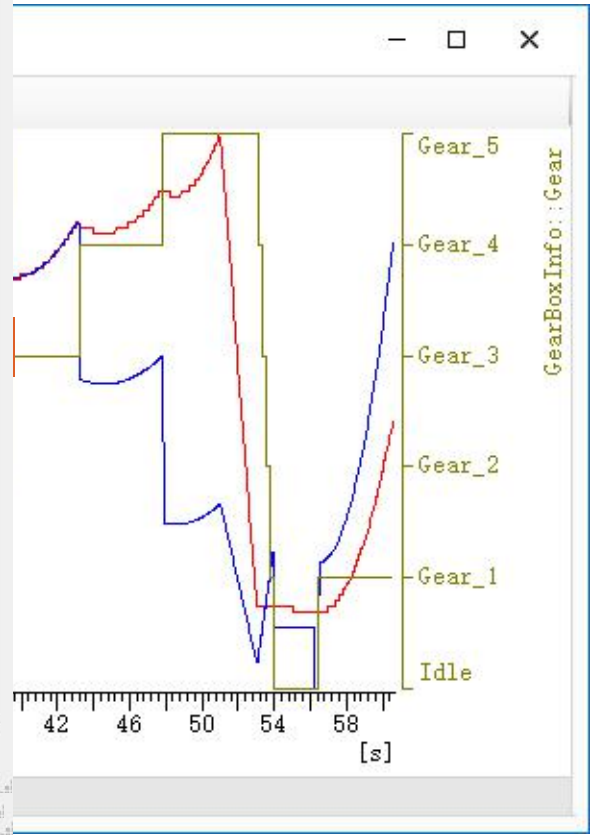




分析窗口

> Graphic 窗口

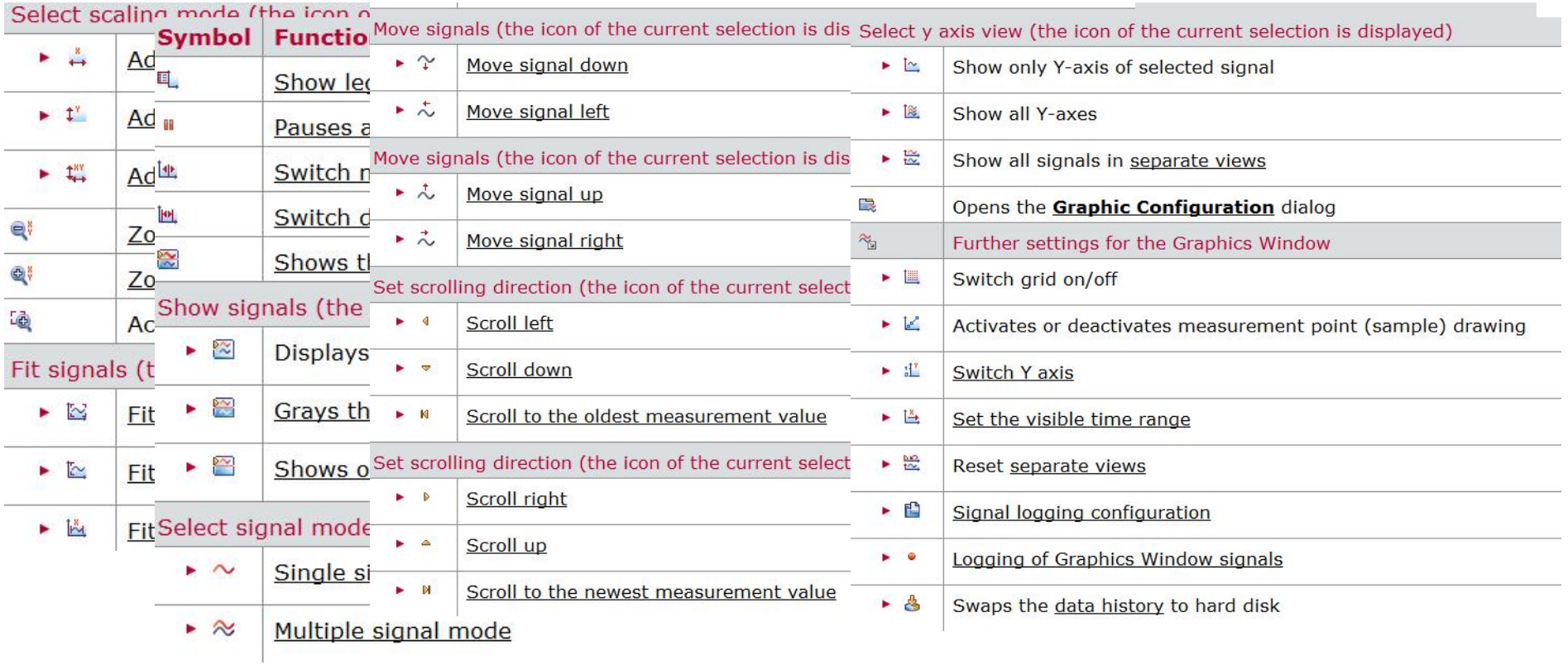




分析窗口

> Graphic 窗口

对于功能栏，所有的图标可以通过help文档查询解释：



Symbol	Function
	Show leg
	Pauses a
	Switch n
	Switch d
	Shows tl
Show signals (the	
	Displays
	Grays th
	Shows o
Select signal mode	
	Single si
	Multiple signal mode

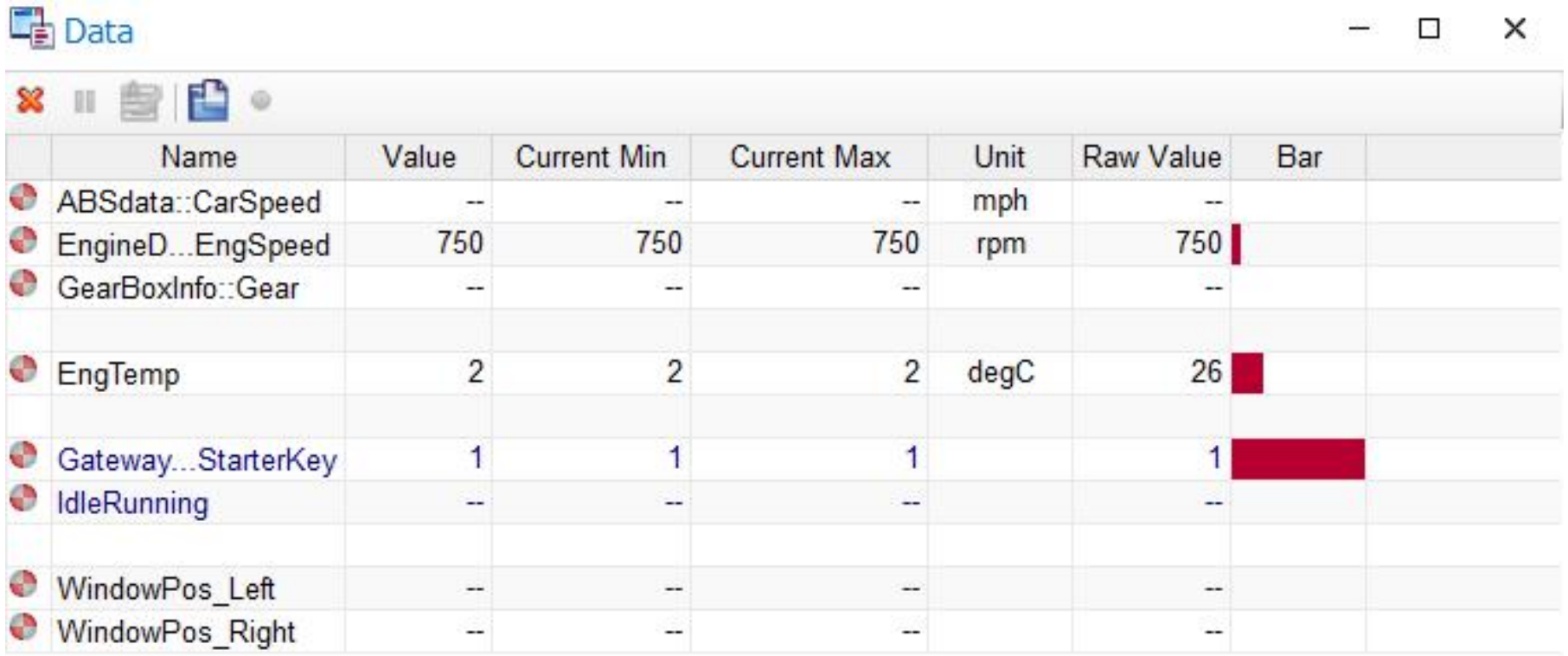
Move signals (the icon of the current selection is dis	
	Move signal down
	Move signal left
Move signals (the icon of the current selection is dis	
	Move signal up
	Move signal right
Set scrolling direction (the icon of the current select	
	Scroll left
	Scroll down
	Scroll to the oldest measurement value
Set scrolling direction (the icon of the current select	
	Scroll right
	Scroll up
	Scroll to the newest measurement value

Select y axis view (the icon of the current selection is displayed)	
	Show only Y-axis of selected signal
	Show all Y-axes
	Show all signals in separate views
	Opens the **Graphic Configuration** dialog
	Further settings for the Graphics Window
	Switch grid on/off
	Activates or deactivates measurement point (sample) drawing
	Switch Y axis
	Set the visible time range
	Reset separate views
	Signal logging configuration
	Logging of Graphics Window signals
	Swaps the data history to hard disk

分析窗口

> Data窗口

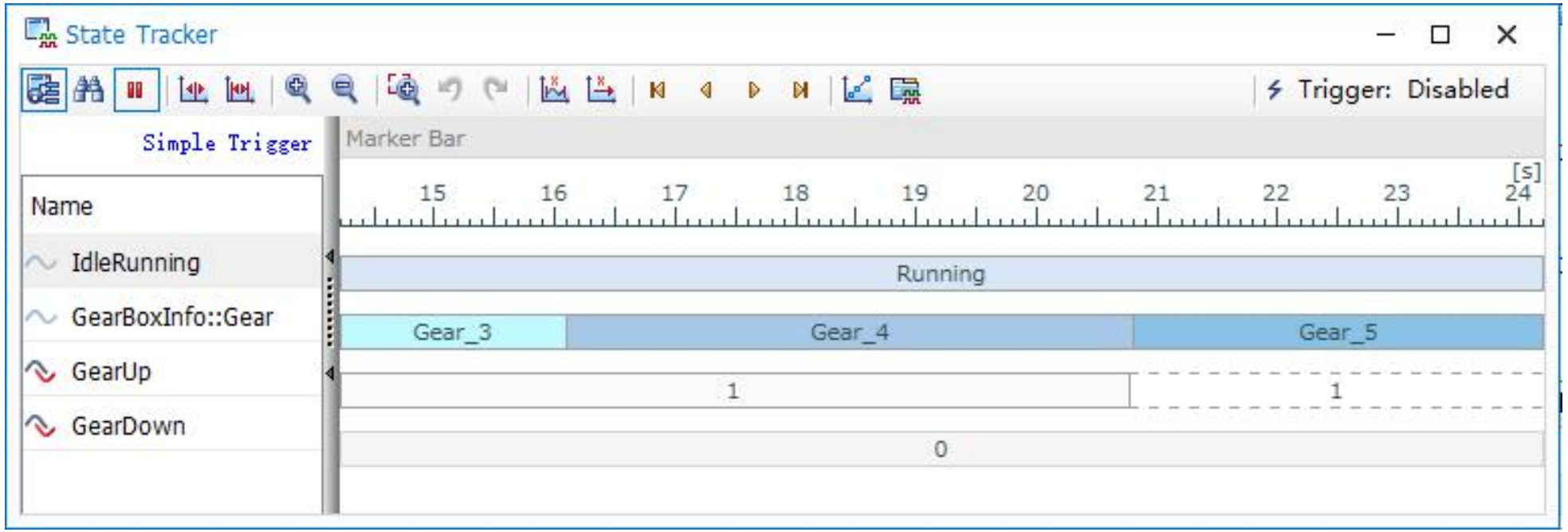
显示信号的名字，单位，初始值等：



分析窗口

> State Tracker窗口

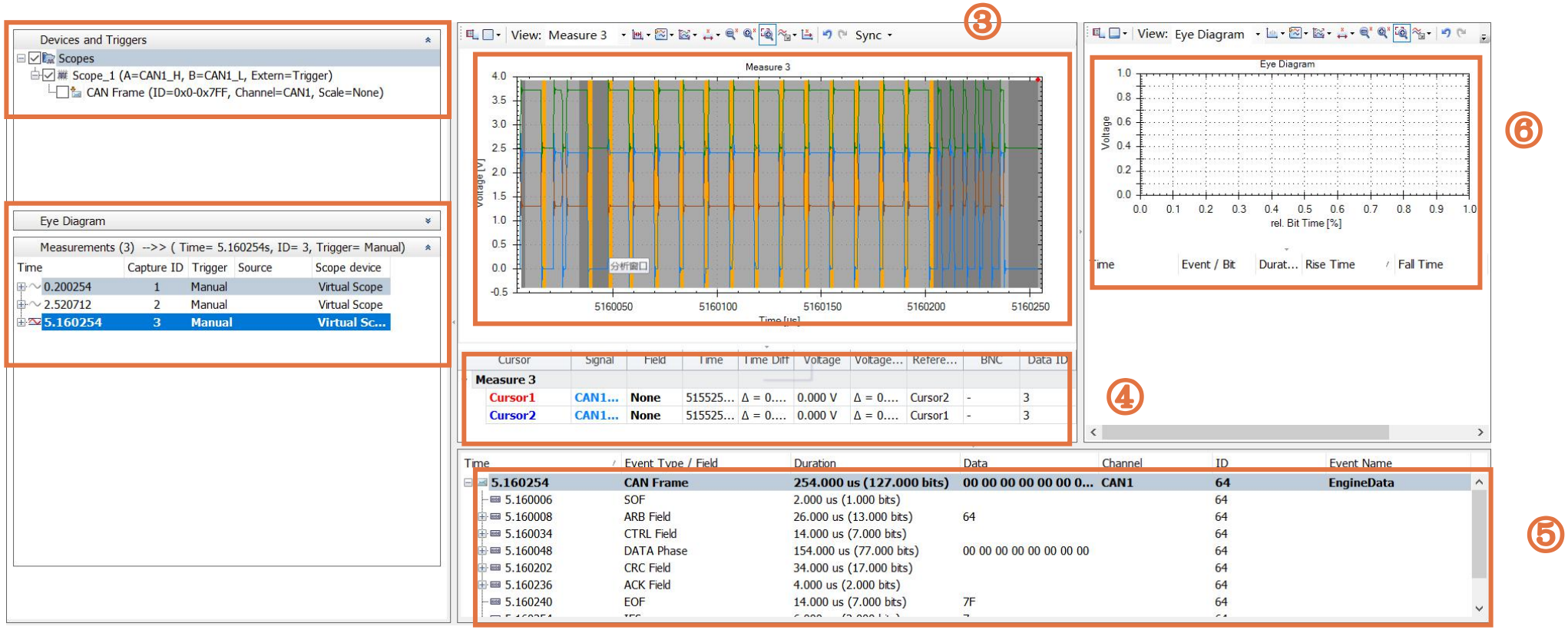
显示比特值和一些状态值：



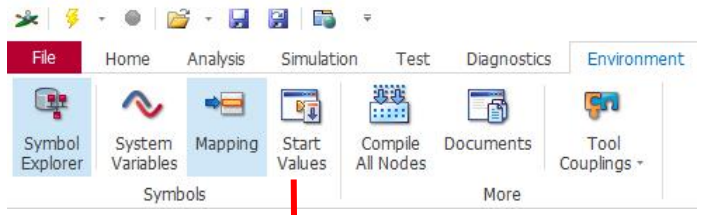
分析窗口

> Scope窗口

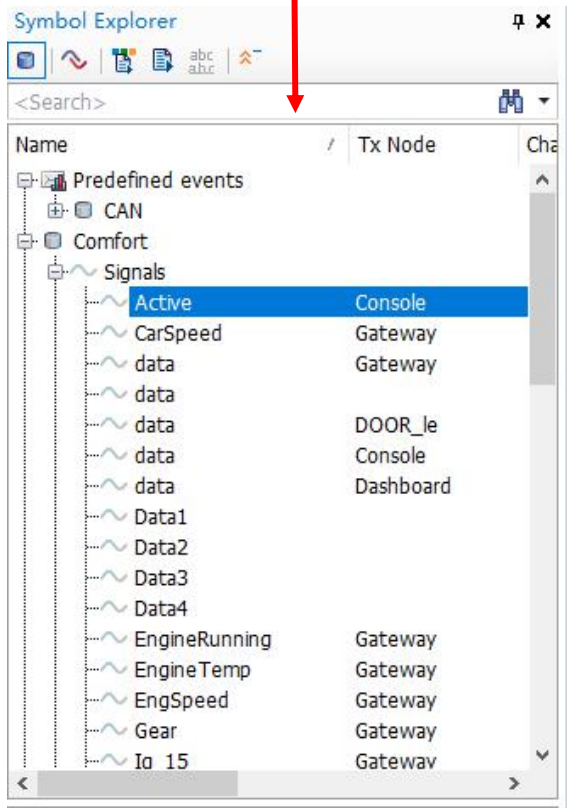
观察总线点评，用于分析总线协议，用Eye 图评估信号品质：



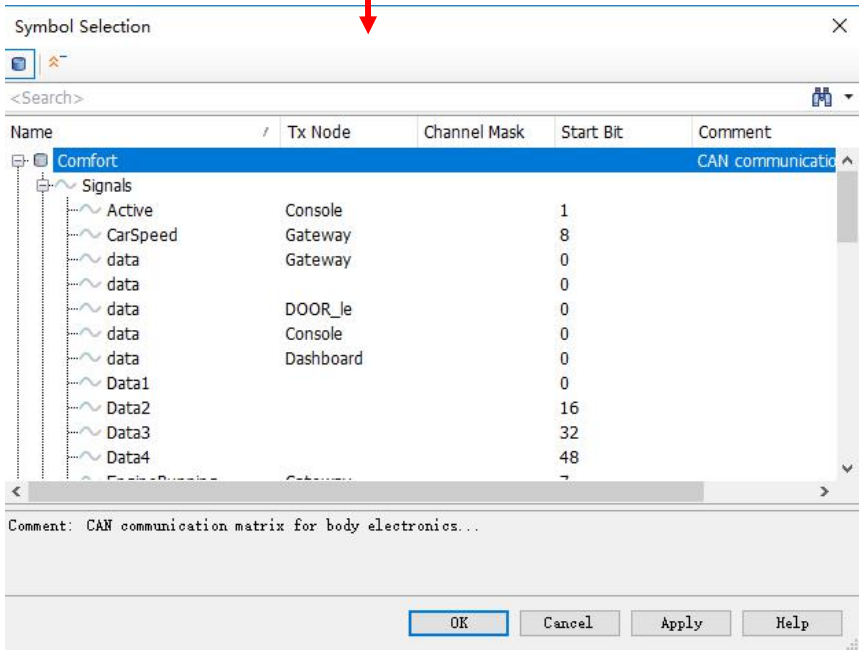
信号添加



Symbol Explorer

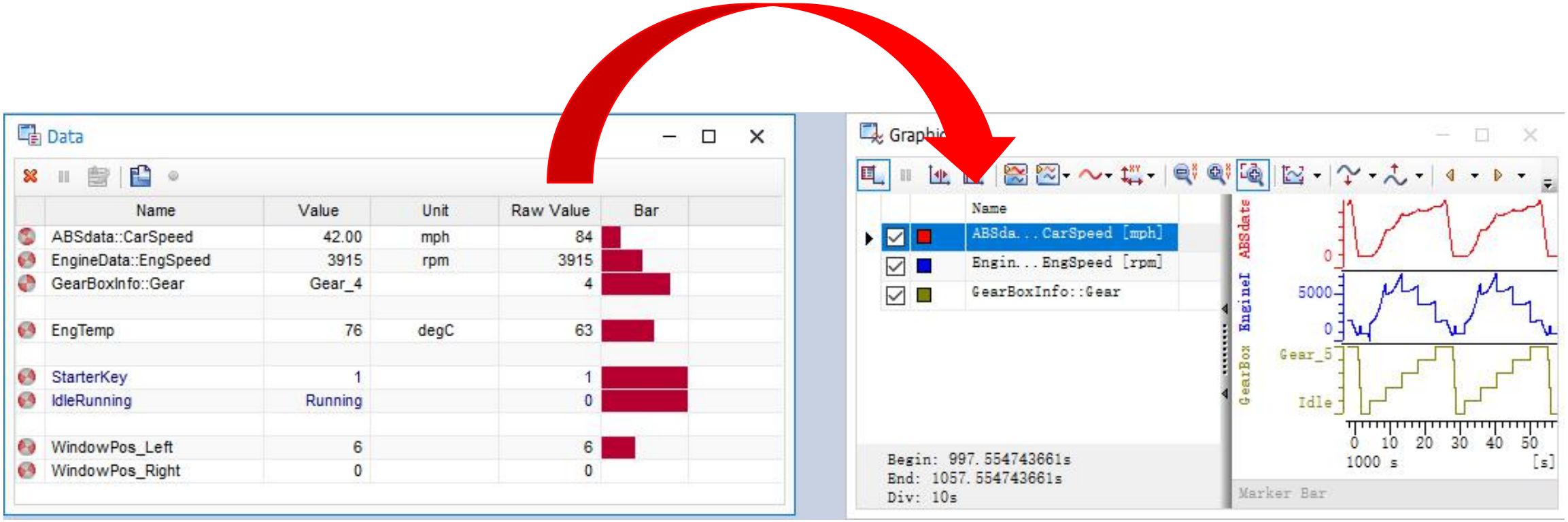


右键快捷菜单



信号添加

通过拖拉方式从另外一个分析窗口



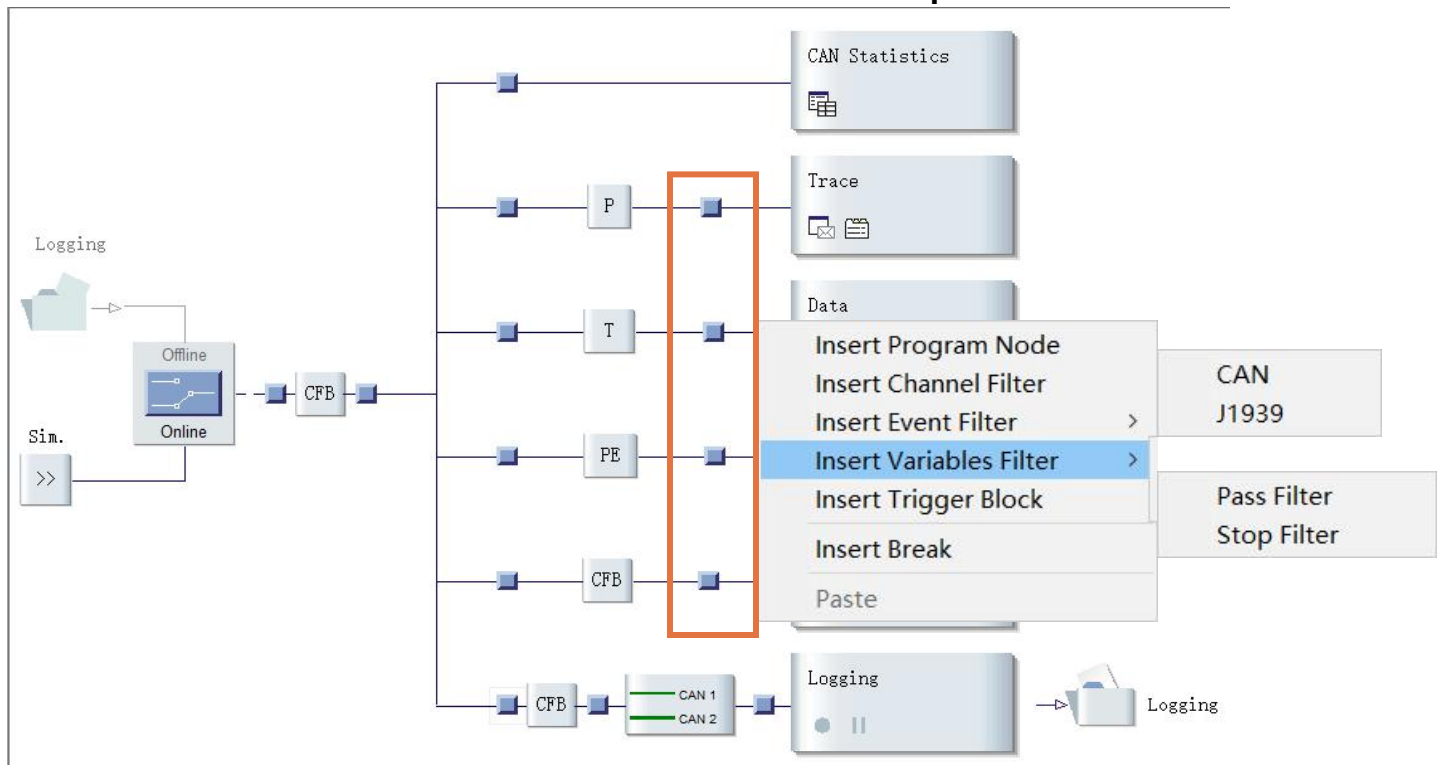
目录

CONTENTS

过滤功能模块

> 过滤功能设置

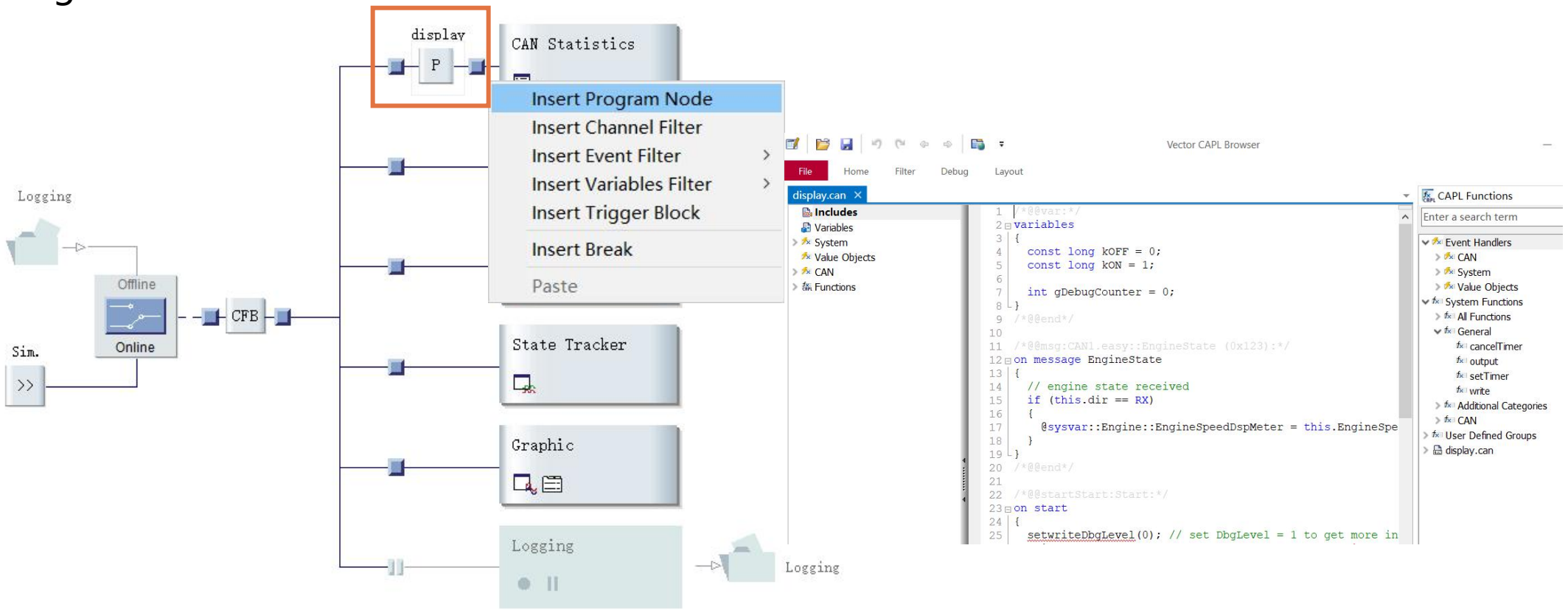
此功能属于分析功能，在Measurment Setup中设置



过滤功能模块

> 过滤功能设置

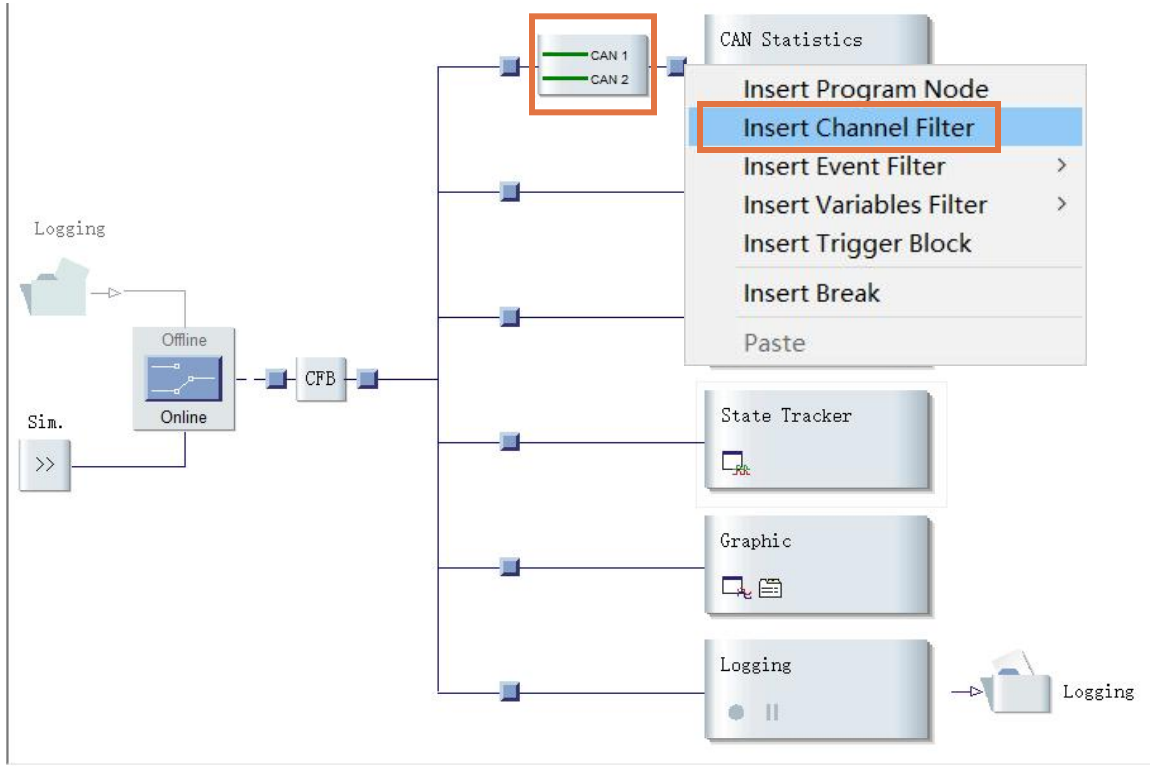
Program Node：

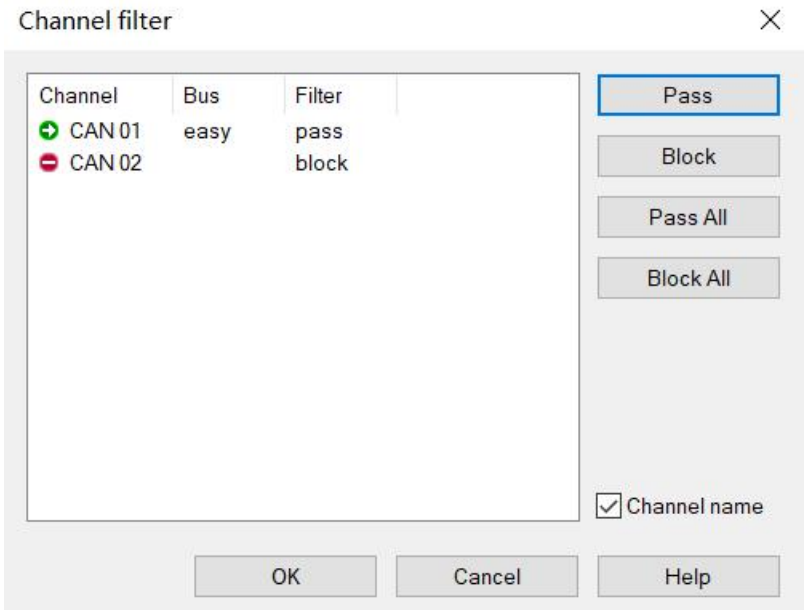


过滤功能模块

> 过滤功能设置

Channel filter:

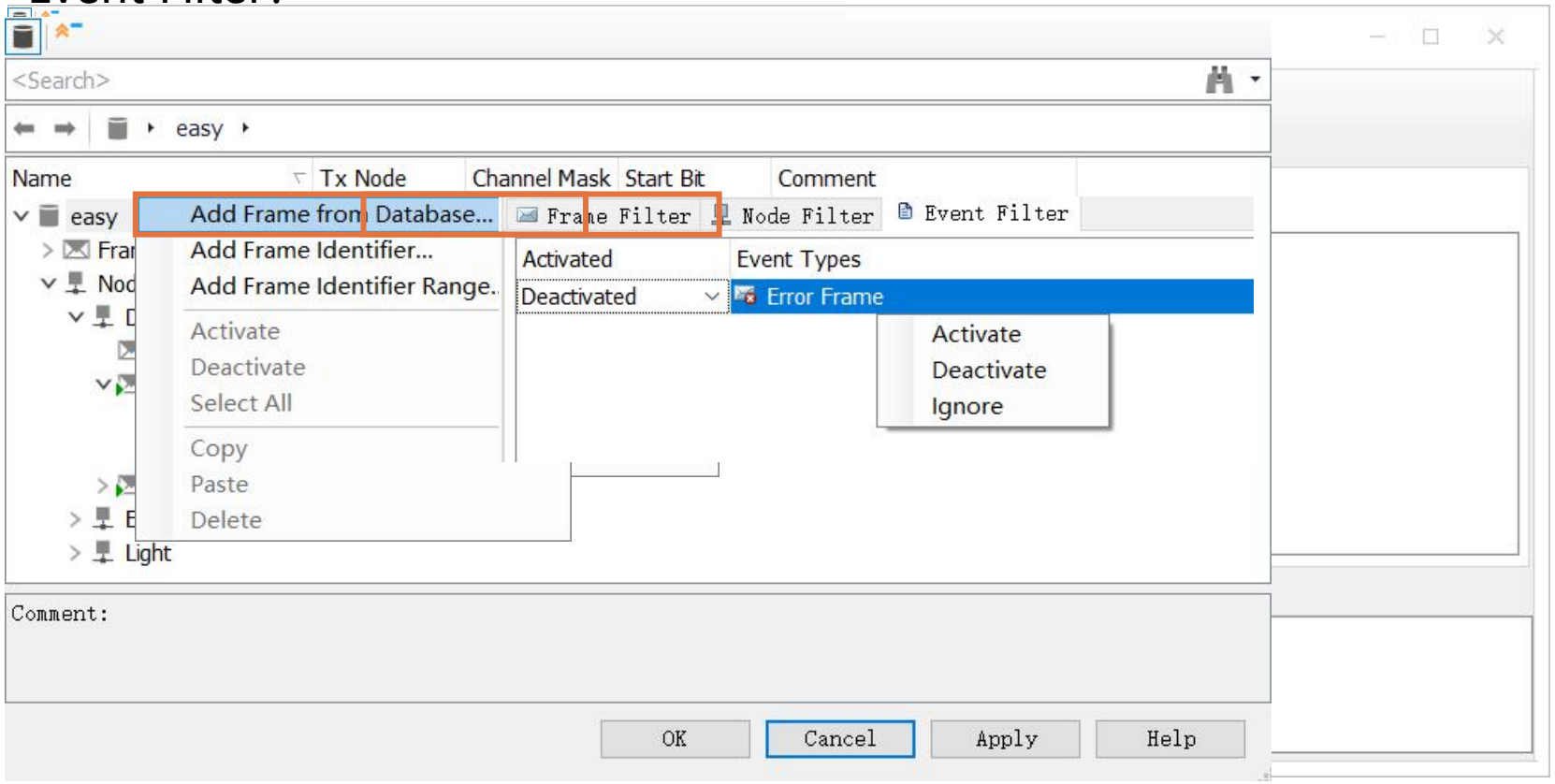




过滤功能模块

> 过滤功能设置

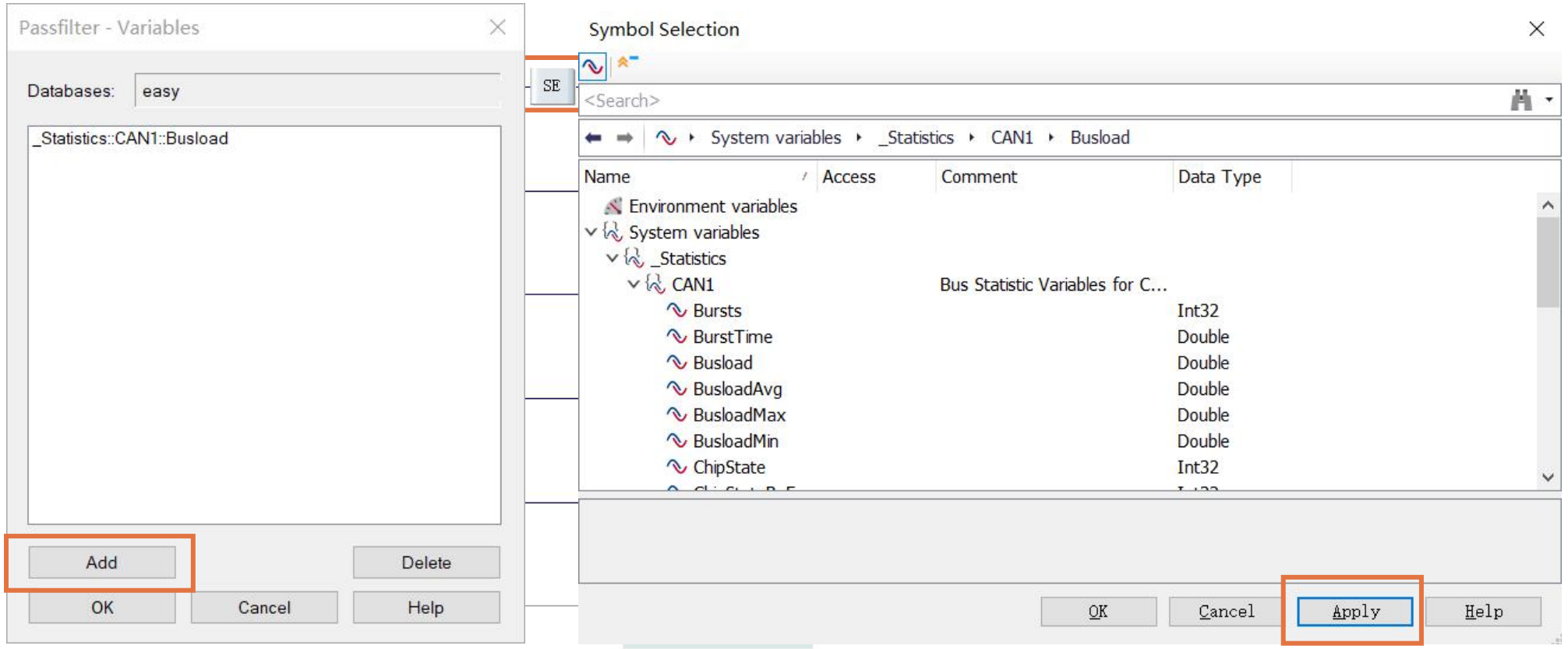
Event Filter：



过滤功能模块

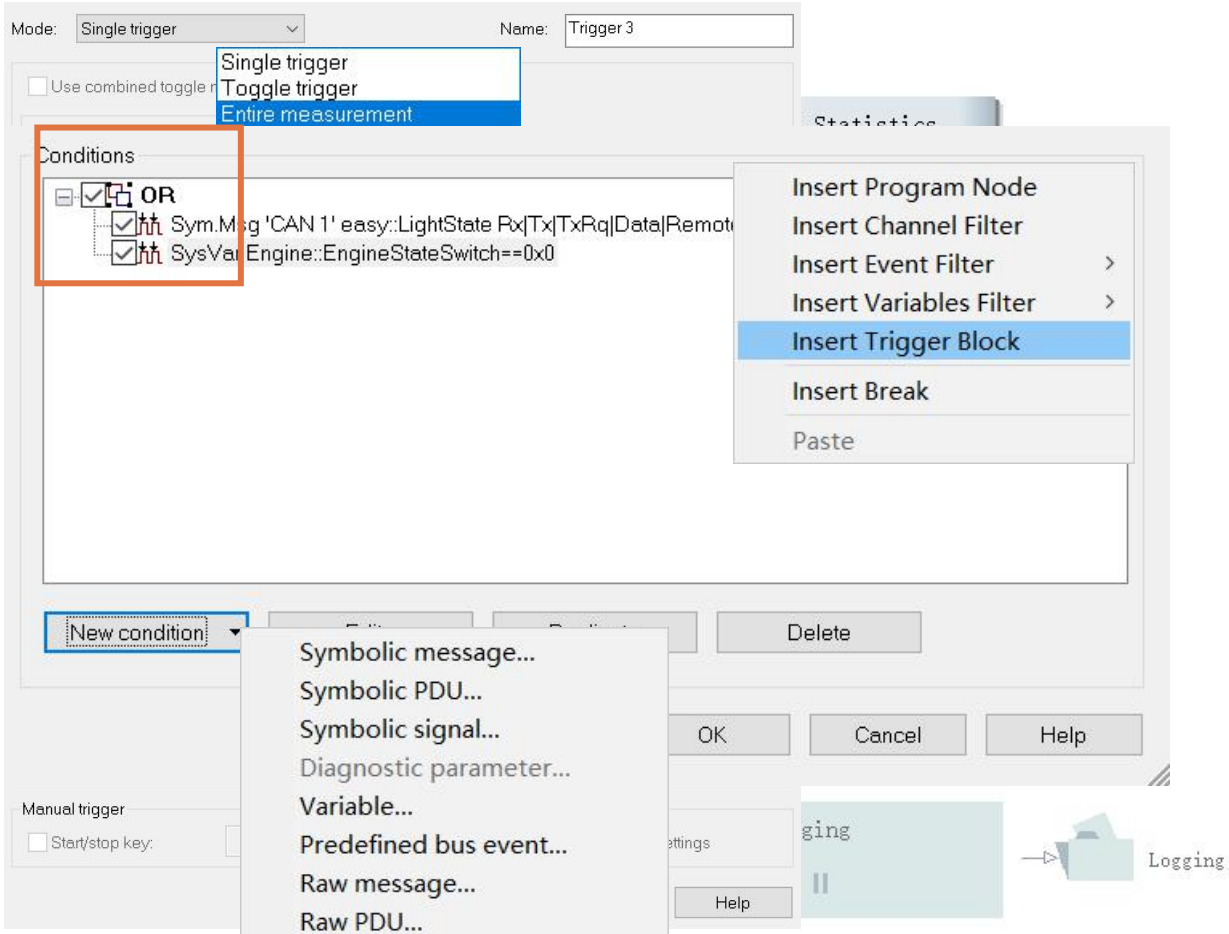
> 过滤功能设置

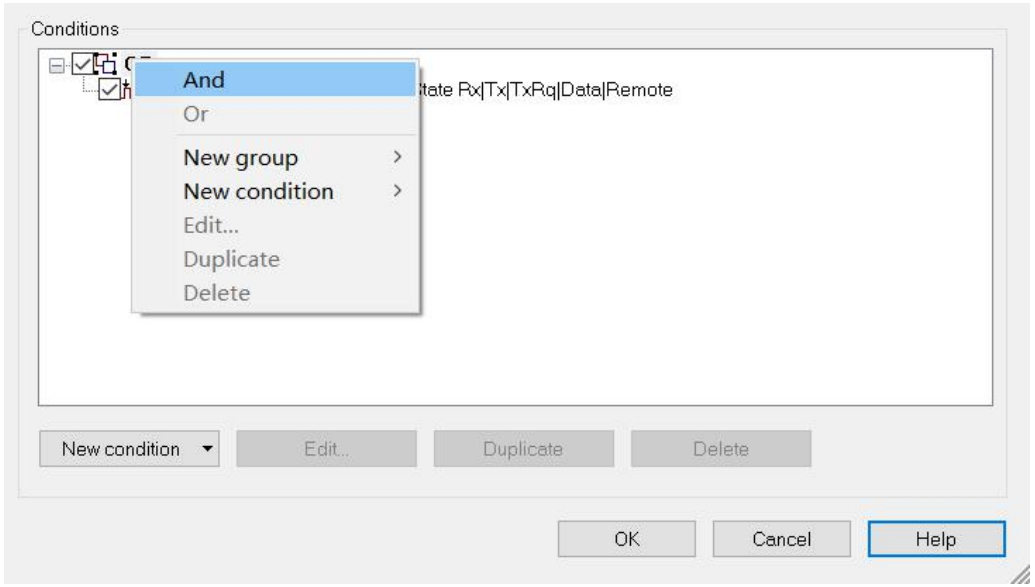
Variables Filter:



过滤功能模块

> 过滤功能设置





目录

CONTENTS

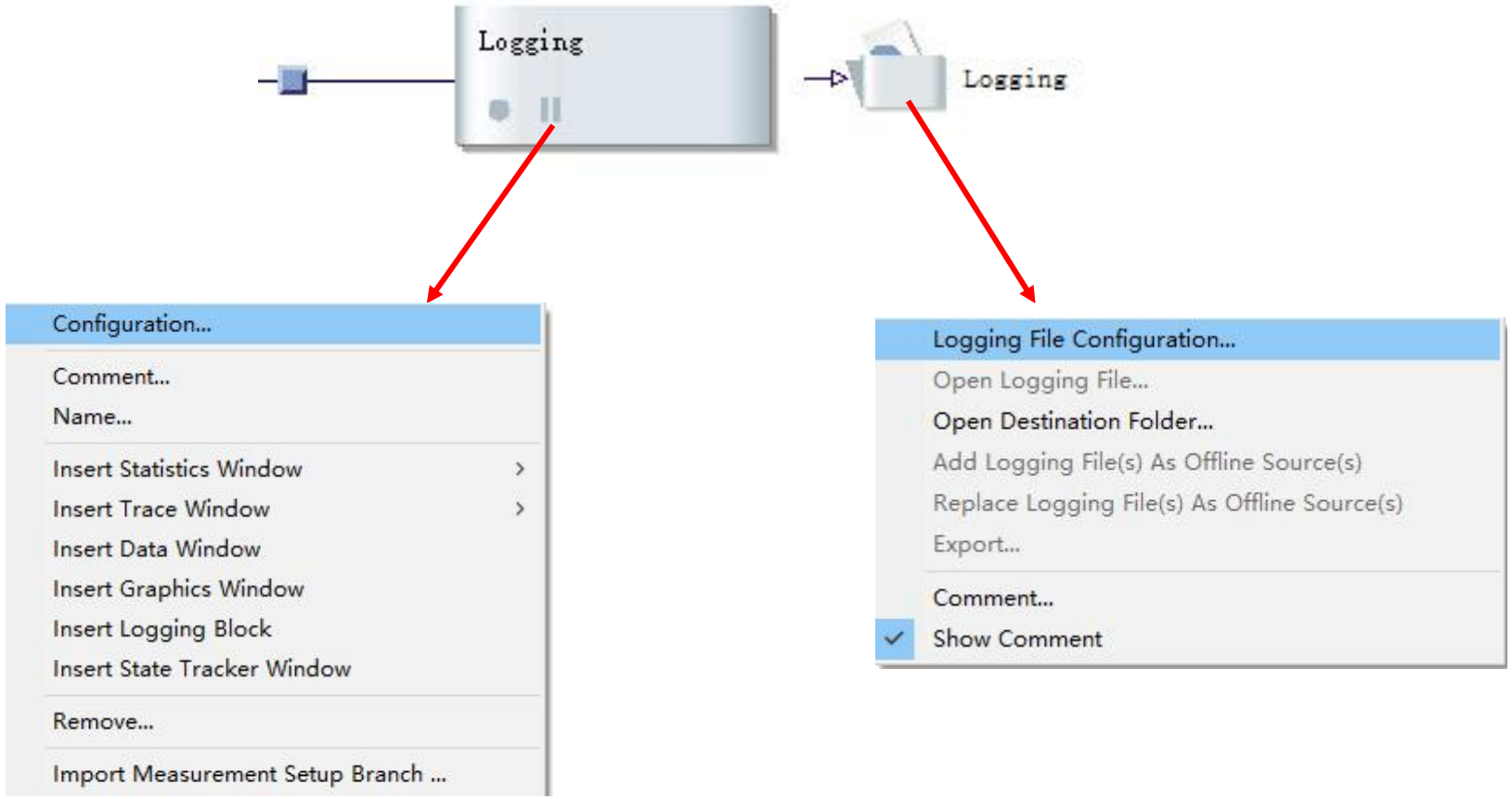
数据格式

报文类记录格式

- ASCII(*.asc)——主要用与和其他的程序进行交互;
- Binary(*.blf)——生成比ASCII格式小得多的报文记录，支持所有的总线系统和协议,

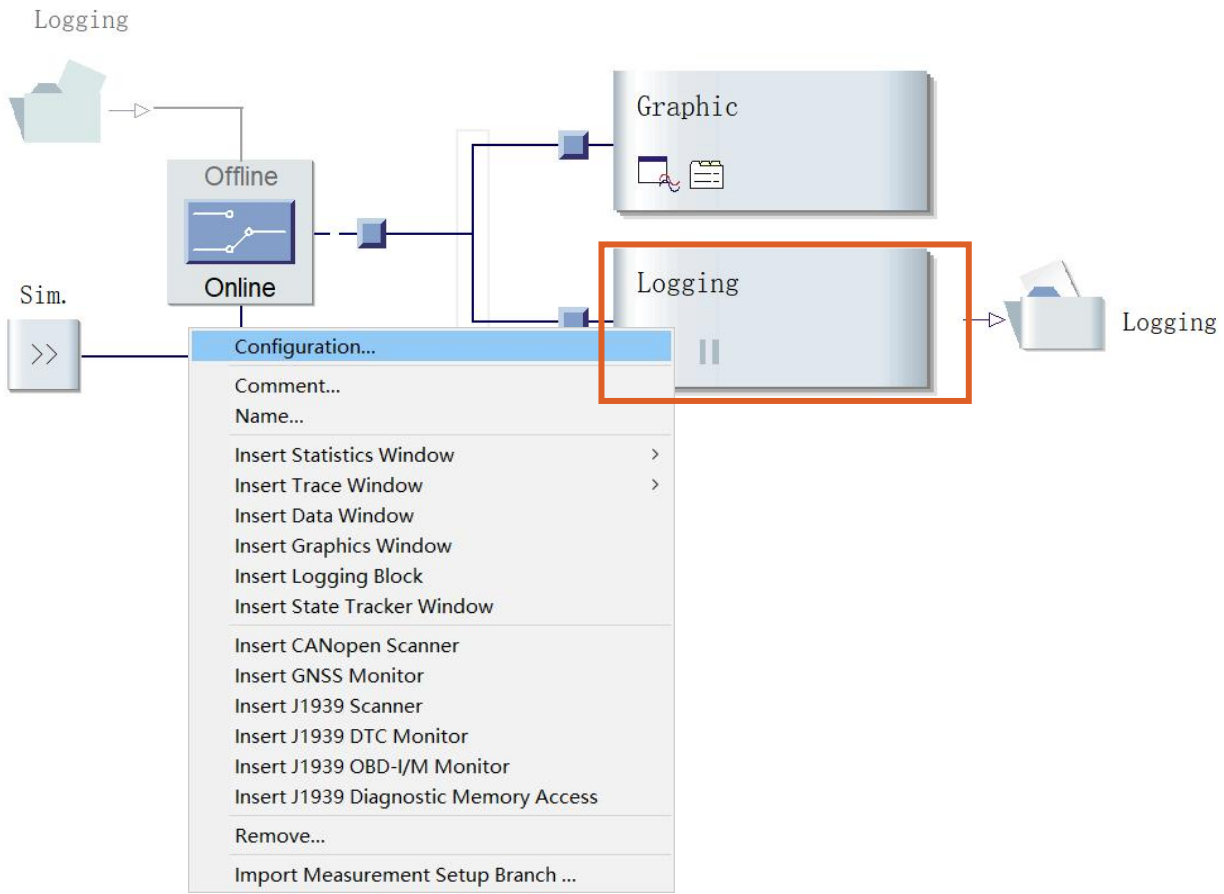
- 信号类记录格式
- Measurement Data Format(*.mdf)
- 支持记录信号，此文件可导入到Graphics 窗口

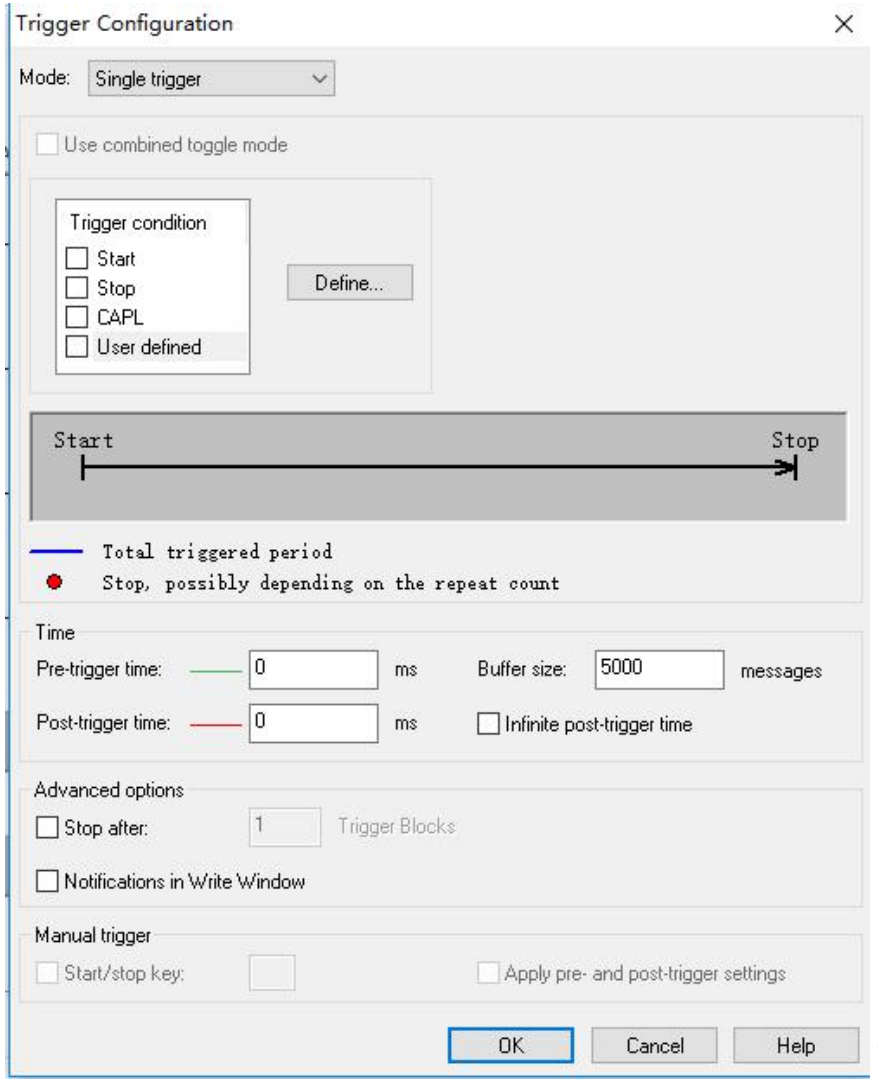
记录模块


Logging
Logging
Configuration...
Comment...
Name...
Insert Statistics Window
Insert Trace Window
Insert Data Window
Insert Graphics Window
Insert Logging Block
Insert State Tracker Window
Remove...
Import Measurement Setup Branch ...
Logging File Configuration...
Open Logging File...
Open Destination Folder...
Add Logging File(s) As Offline Source(s)
Replace Logging File(s) As Offline Source(s)
Export...
Comment...
Show Comment

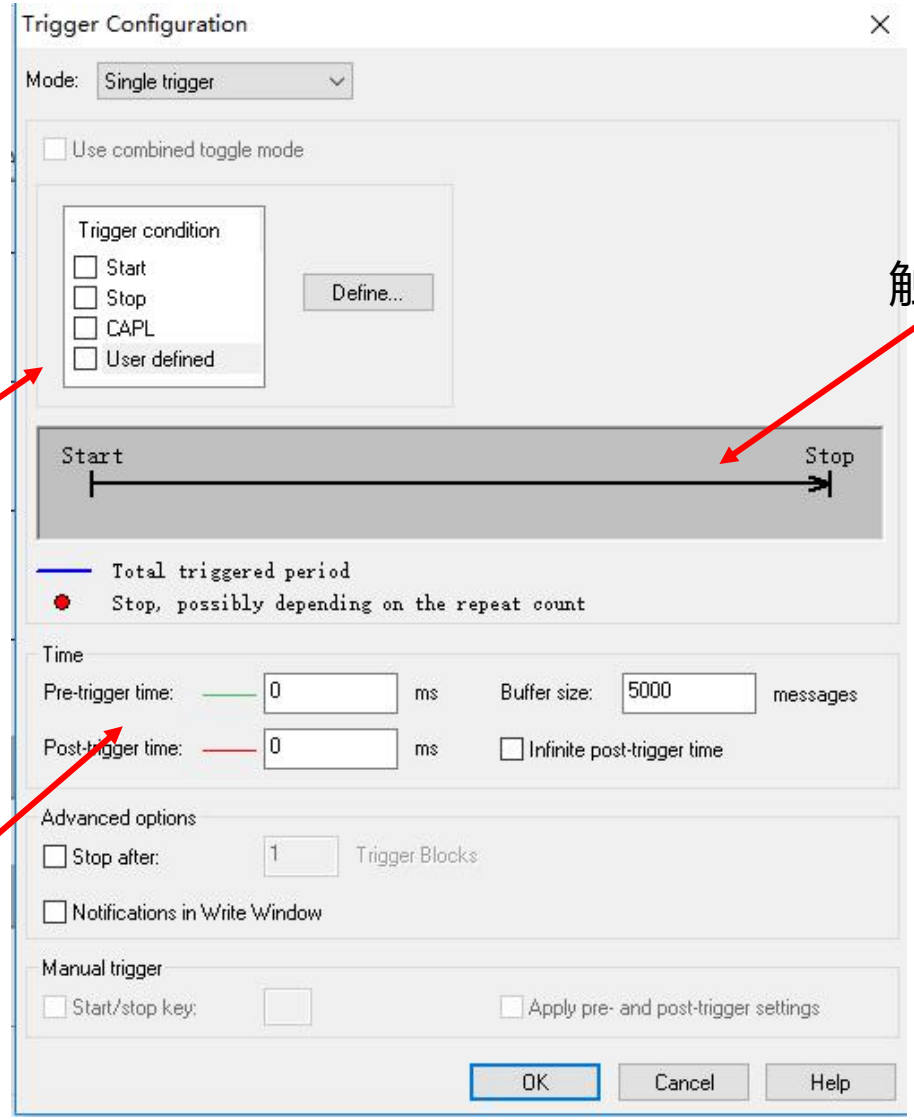
数据记录

> 记录模块


Logging
Offline
Online
Sim.
Graphic
Logging
Logging
Configuration...
Comment...
Name...
Insert Statistics Window
Insert Trace Window
Insert Data Window
Insert Graphics Window
Insert Logging Block
Insert State Tracker Window
Insert CANopen Scanner
Insert GNSS Monitor
Insert J1939 Scanner
Insert J1939 DTC Monitor
Insert J1939 OBD-I/M Monitor
Insert J1939 Diagnostic Memory Access
Remove...
Import Measurement Setup Branch ...


Trigger Configuration
Mode: Single trigger
Use combined toggle mode
Trigger condition
Start
Stop
CAPL
User defined
Define...
Start
Stop
Total triggered period
Stop, possibly depending on the repeat count
Time
Pre-trigger time: 0 ms
Buffer size: 5000 messages
Post-trigger time: 0 ms
Infinite post-trigger time
Advanced options
Stop after: 1 Trigger Blocks
Notifications in Write Window
Manual trigger
Start/stop key:
Apply pre- and post-trigger settings
OK
Cancel
Help

记录模块



触发方式

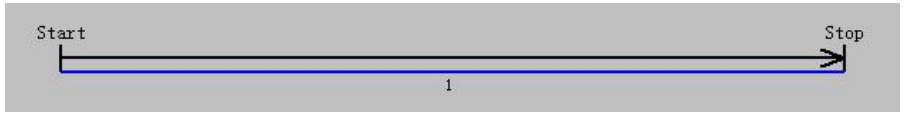
触发记录时间段预览

Pre and Post Trigger time

数据记录

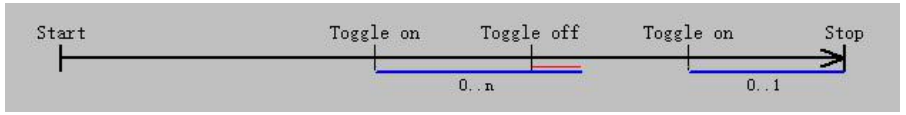
> 记录模块

Entire Measurement
完整测量：从开始测量到结束记录完整数据

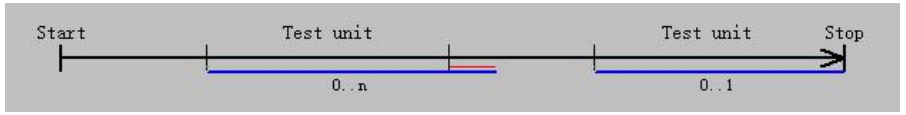


Single Trigger
单触发：一个触发条件，记录触发点前后数据

Toggle Trigger
组合触发：记录两个触发条件中间数据



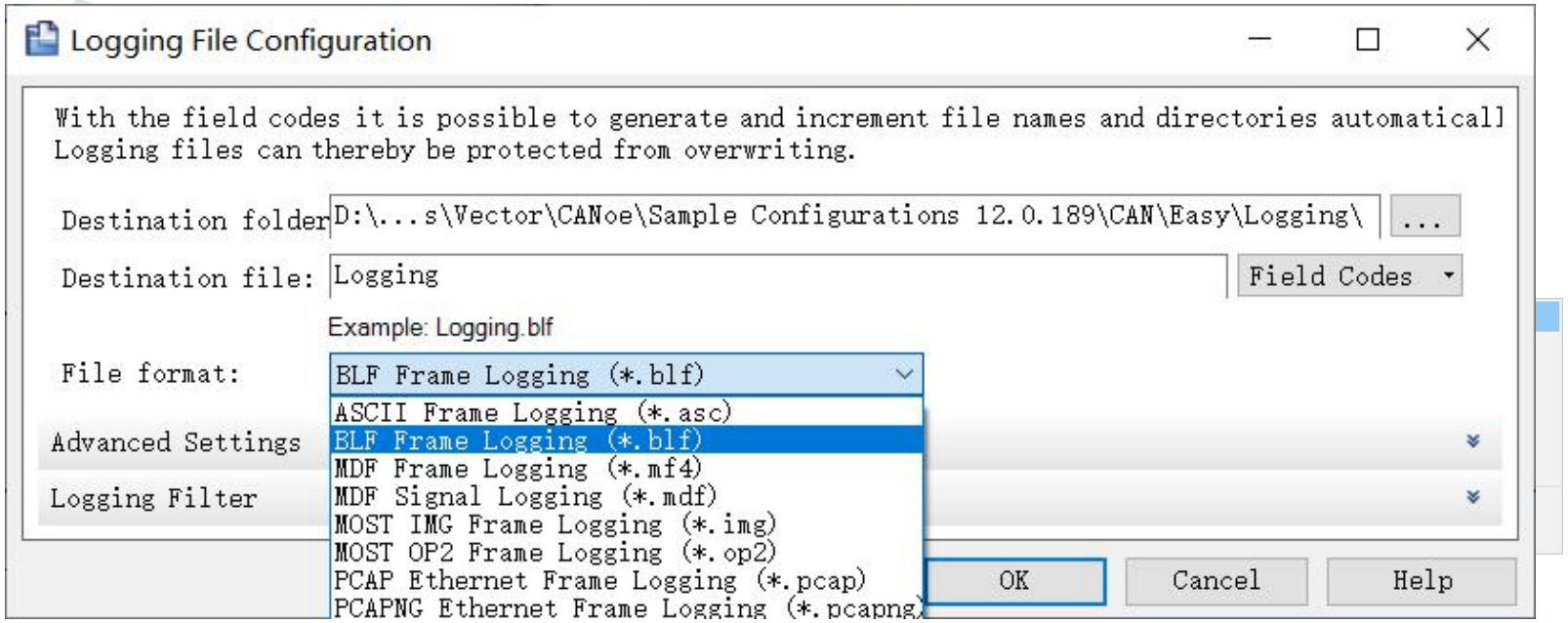
Test Trigger
测试触发：记录测试单元数据



记录

> 记录模块

Logging



数据记录

记录模块

CAPL Logger:

发送一个触发事件到所有CANoe 记录或者触发模块.

```
startLogging();
// starts all Logging Blocks
stopLogging();
// stops all Logging Blocks
startLogging( "Logging 1");
// starts the Logging Block "Logging 1"
stopLogging( "Logging 1");
// stops the Logging Block "Logging 1"
startLogging( "Logging 1", 2000);
// starts the Logging Block "Logging 1" with pre-trigger time 2000
milliseconds.
stopLogging( "Logging 1", 1000);
// stops the Logging Block "Logging 1" with post-trigger time 1000
milliseconds.
```

```
on message 100
{
  write("logging start");
  trigger(); // start logging
  setTimer(logging, 1000); // for 1000 ms
}
on timer logging
{
  trigger(); // Stop logging
}
```

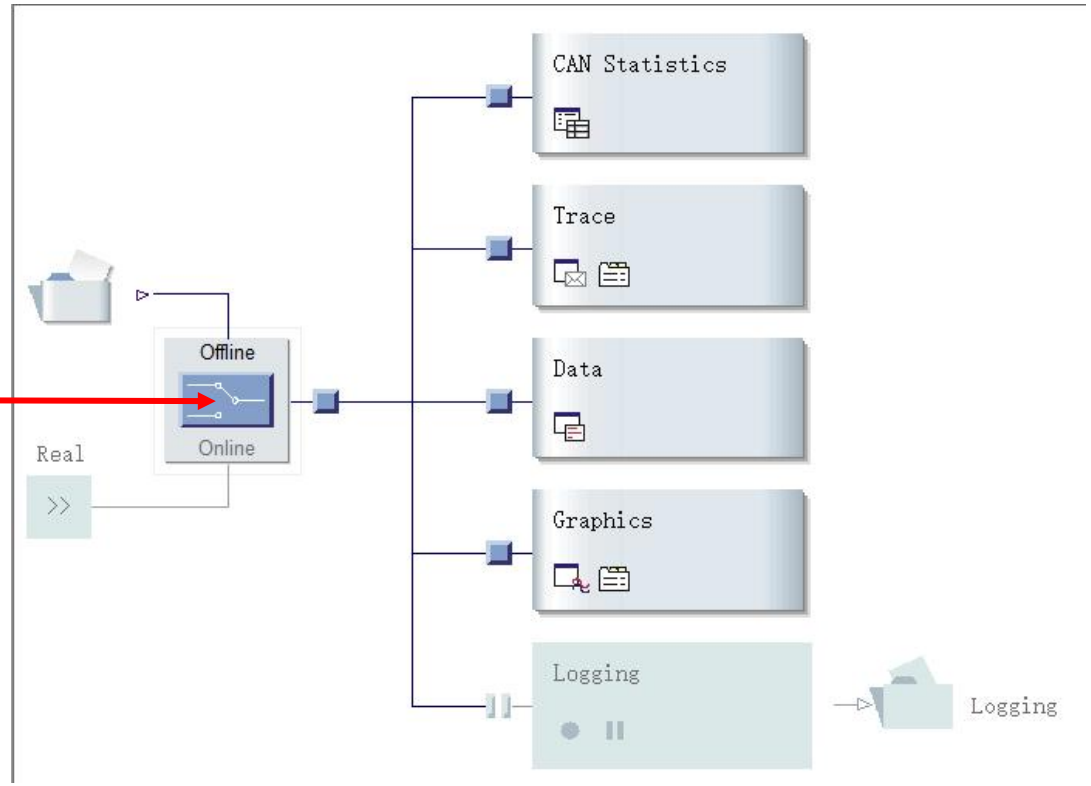
目录

CONTENTS

Offline Mode

- Offline模式下回放记录文件
- 分析记录的数据
- 无需连接真实总线
- 可在分析窗口中分析数据

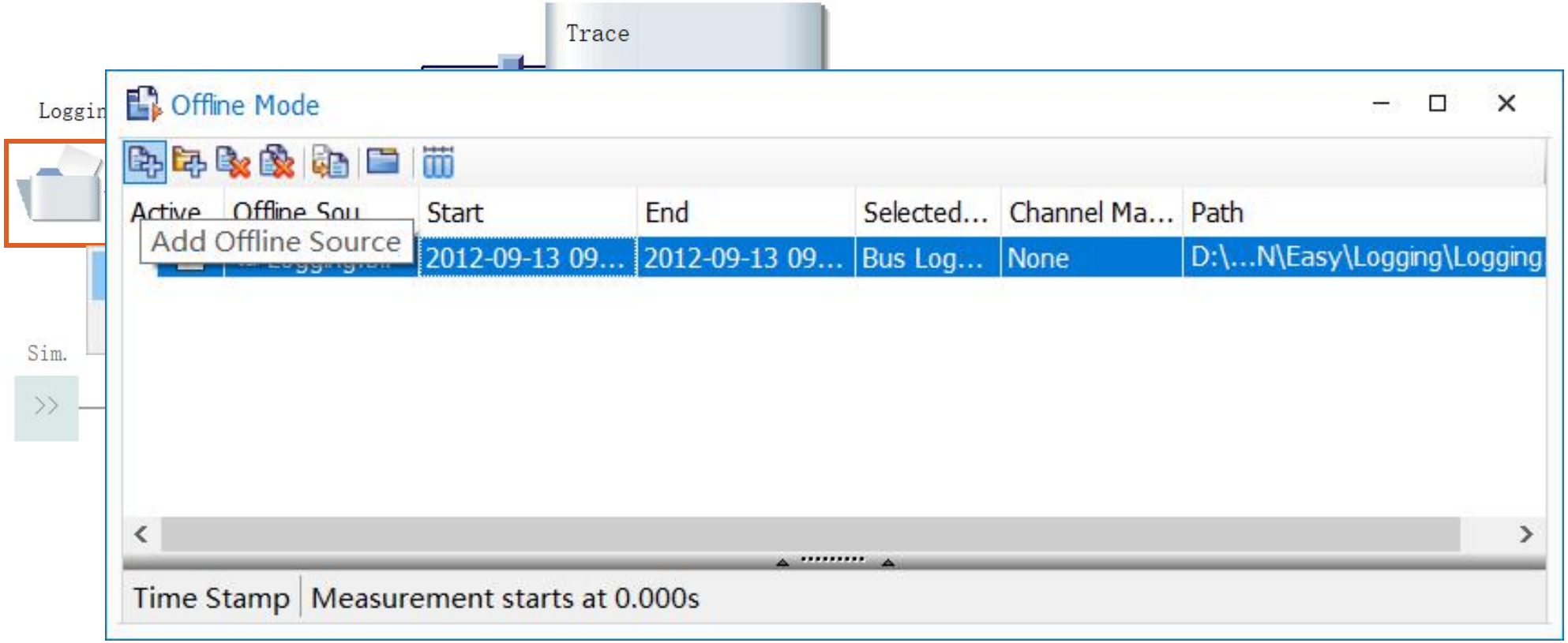
双击切换
Online/Offline



离线数据分析

> 离线模式

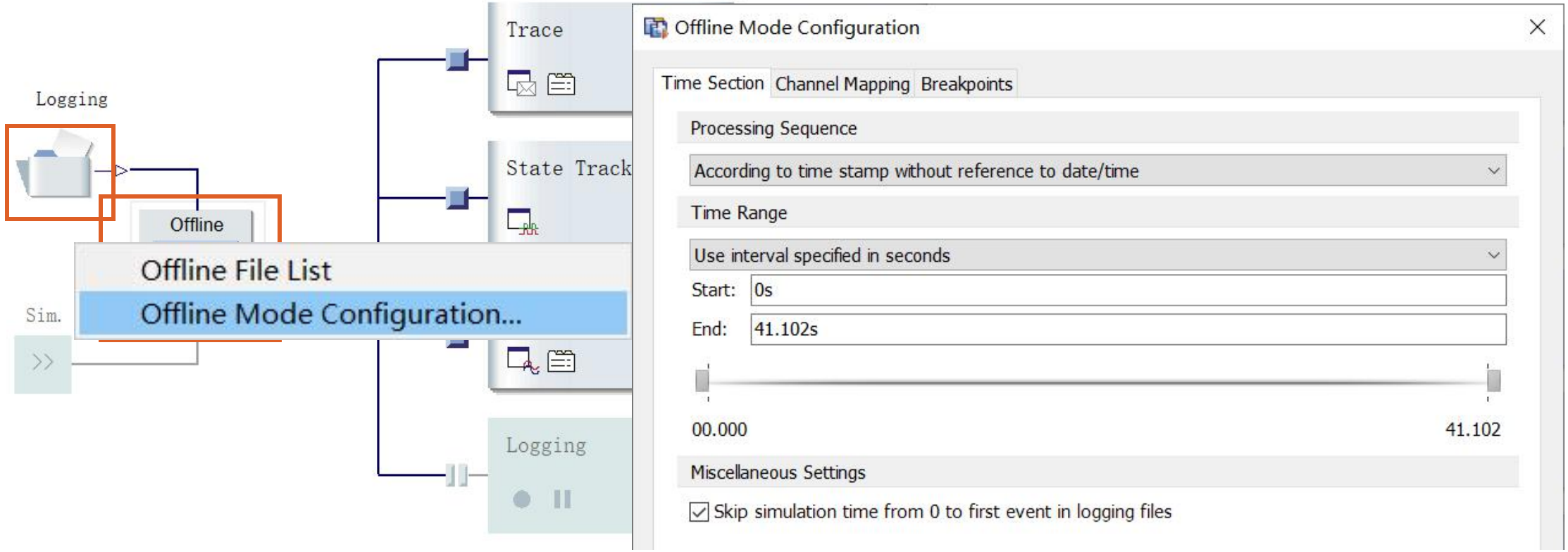
在Measurement Setup窗口进行离线模式的切换：



离线数据分析

> 离线模式

在Measurement Setup窗口进行离线模式的切换：



离线数据分析

> 离线模式

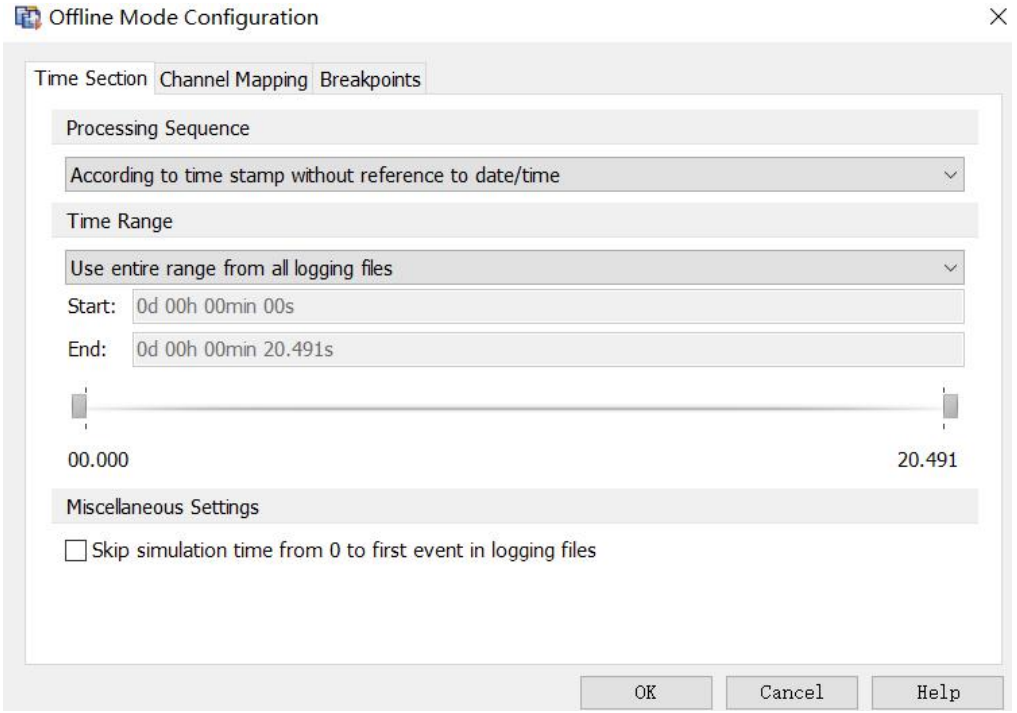
Time section
处理文件的顺序：
1.处理文件的顺序按照时间戳，与日期和时间无关。
2.按照时间和日期的先后顺序，不考虑时间戳的顺序。

CANoe录制的ASC文件格式：
在文件的表头包含起始日期date/time，时间戳是绝对时间或者相对时间

选择处理的文件的时间范围；

跳过从零到第一个事件event的时间。

可以添加多个离线分析文件



离线数据分析

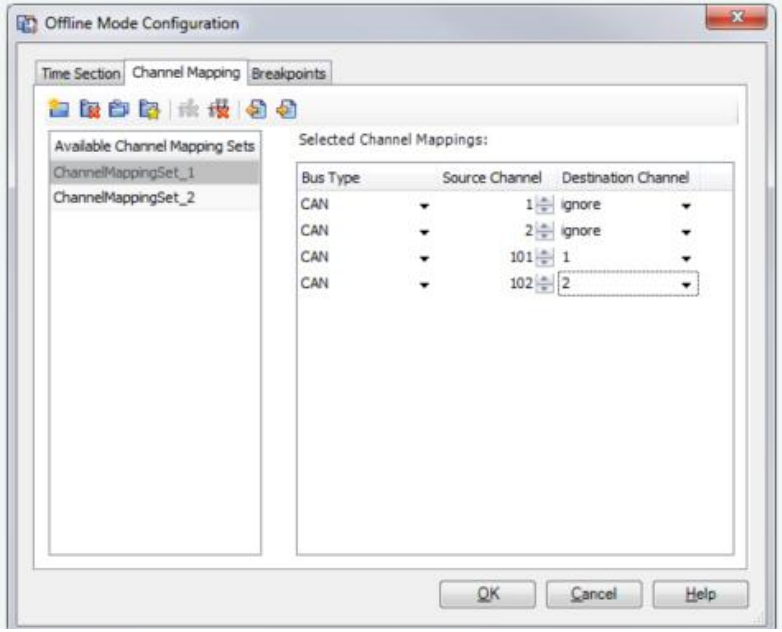
> 离线模式

文件分析时可以映射文件与CANoe的通道

Example

A CAN logger generates a BLF log file with messages on channel **101** and **102**.
In CANoe the two channels **1** and **2** are configured.
The channels in the log file can be mapped to CANoe channels **1** and **2** using channel mapping of the CANoe offline source.

In this case, the dialog looks as follows:

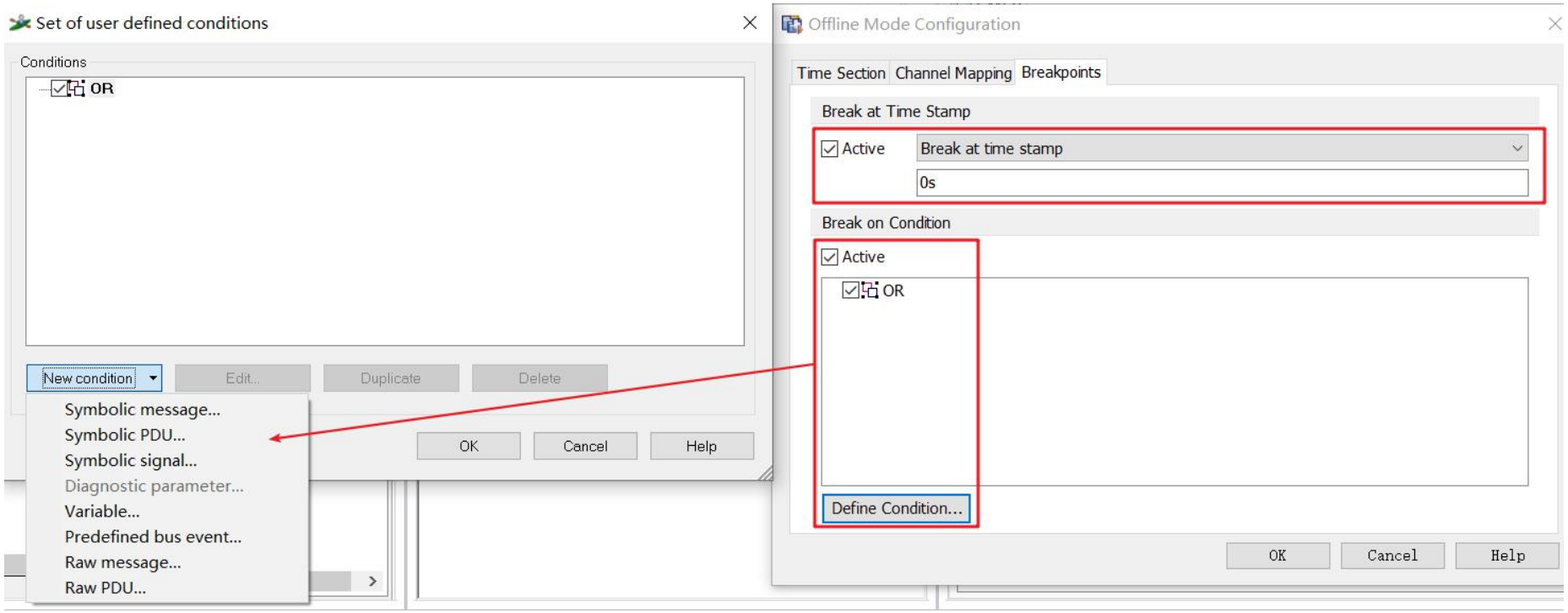


离线数据分析

> 离线模式

断点条件设置：

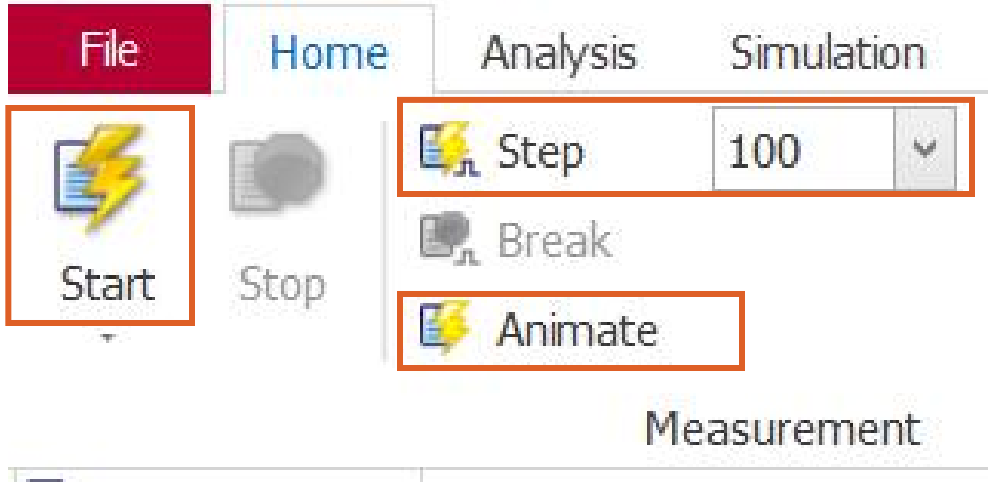
在文件进行回放的时候，可以设置断点，停止回放。断点就是一个条件（报文或者信号值），这个叫做全局断点。断点可以设置为时间条件或者逻辑条件。



离线数据分析

> 数据回放

数据回放形式：



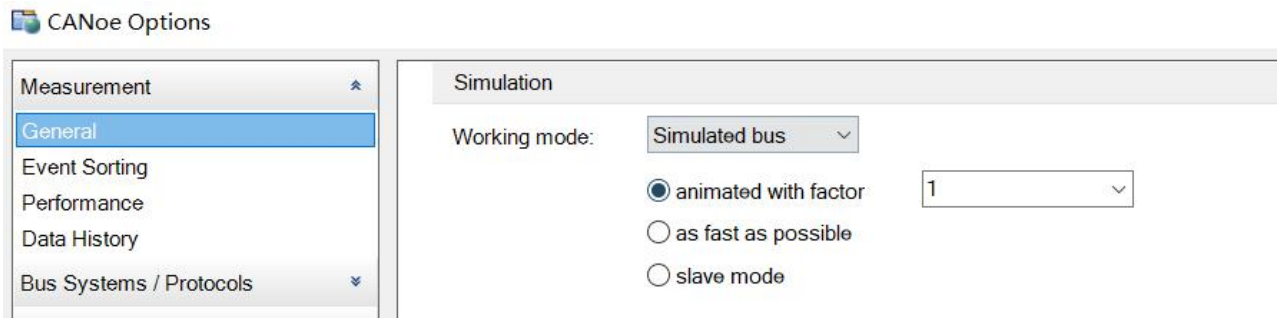
Start 和Stop：整个文件回放

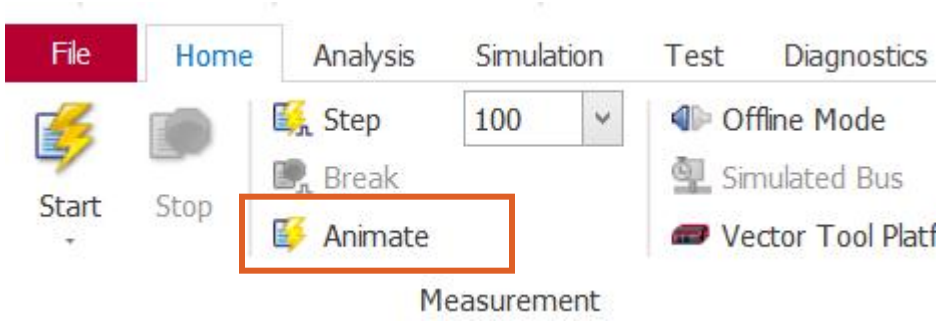
Step：步进回放，可以设置步进的时间

Animate：事件回放，默认

离线数据分析

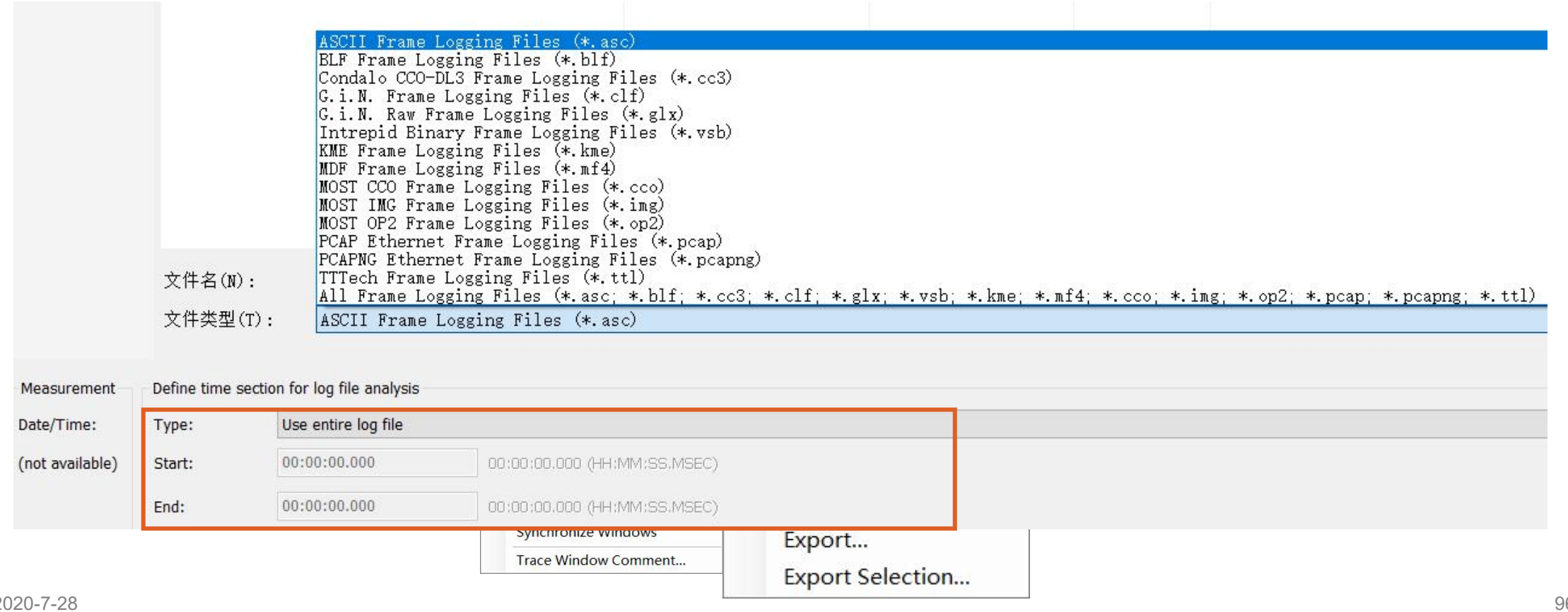
> Animate





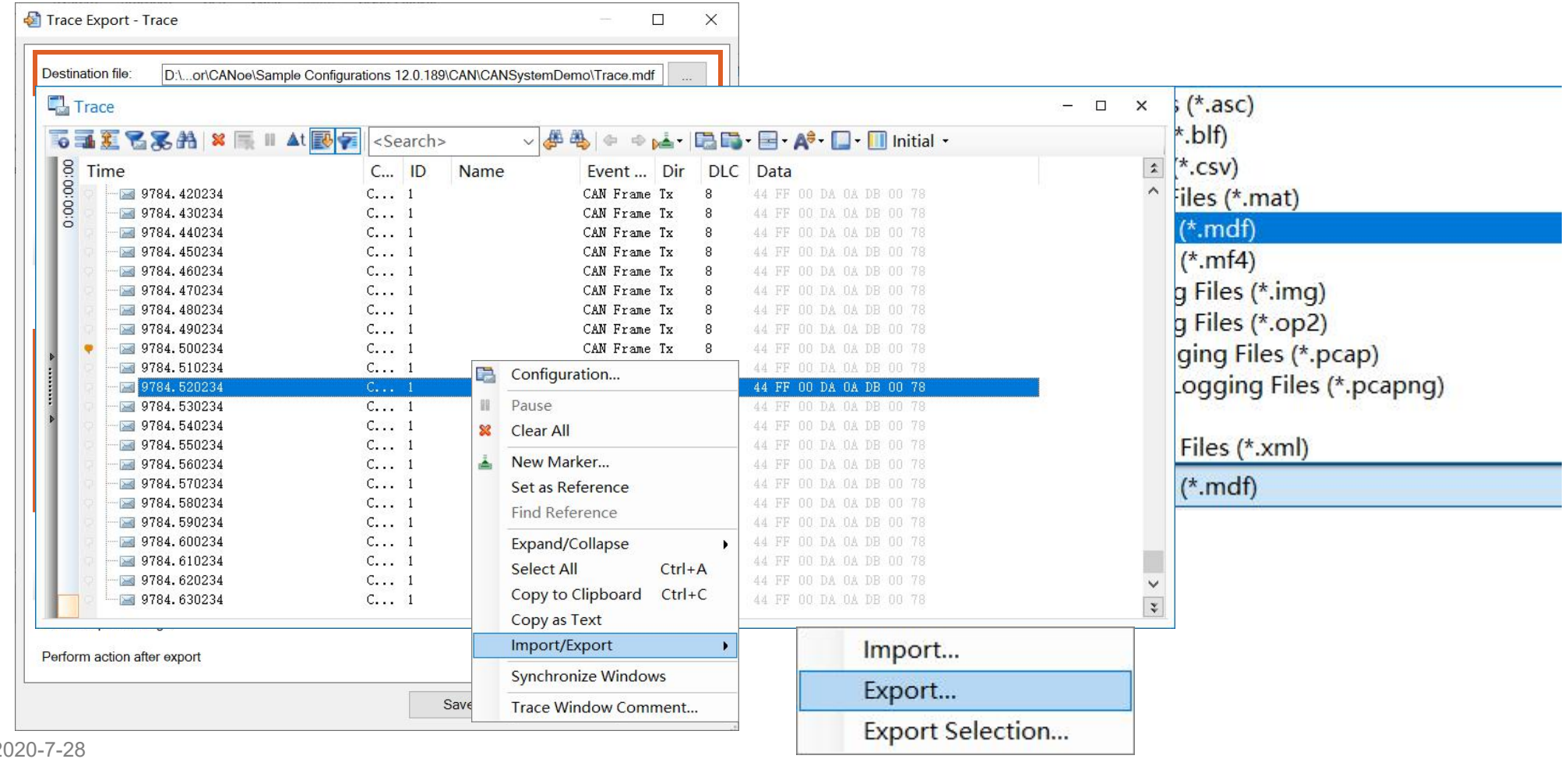
离线数据分析

> Trace窗口导入



离线数据分析

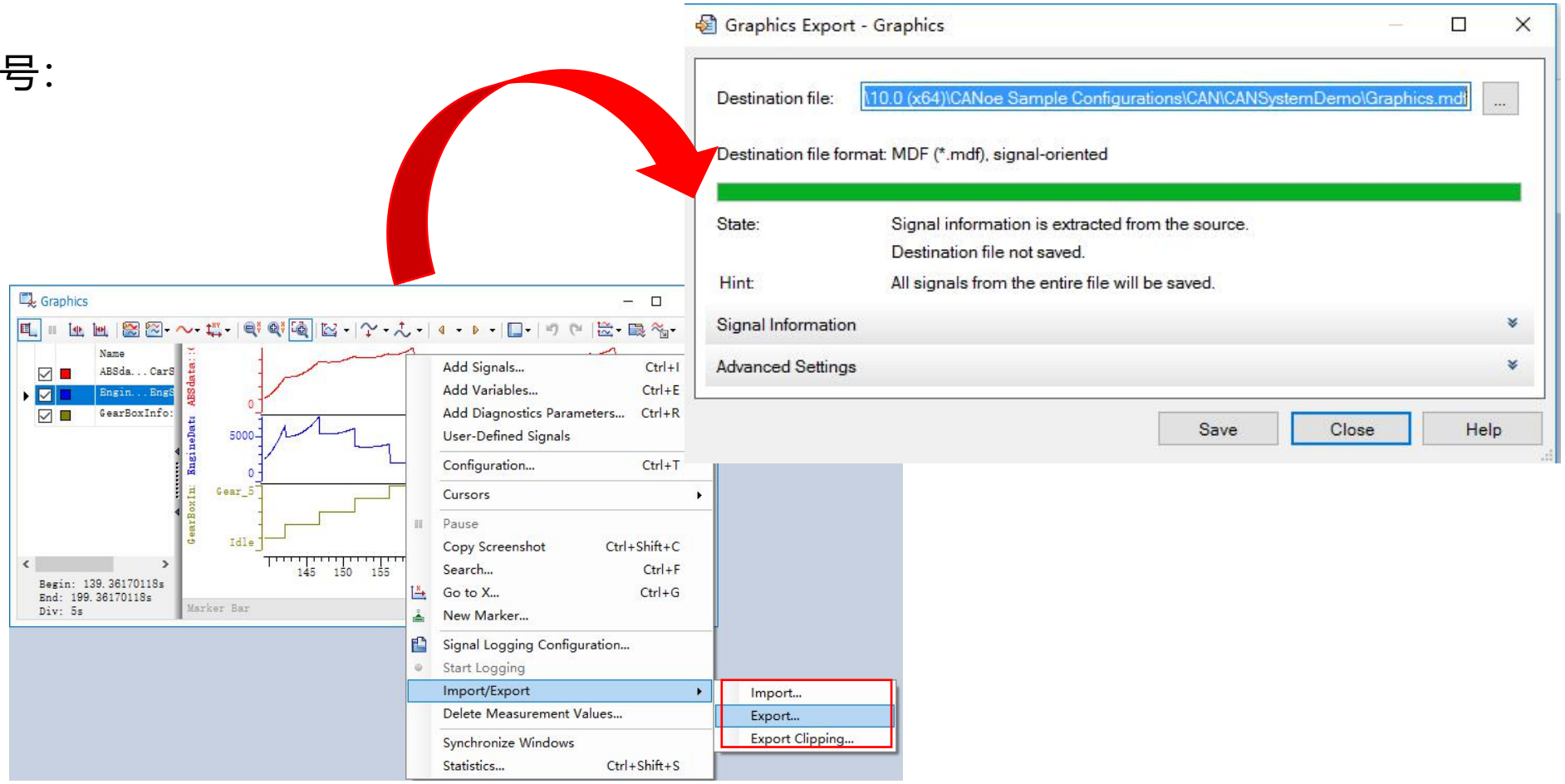
> Trace窗口导出



离线数据分析

> Graphic 窗口导入导出信号

添加信号：



离线数据分析

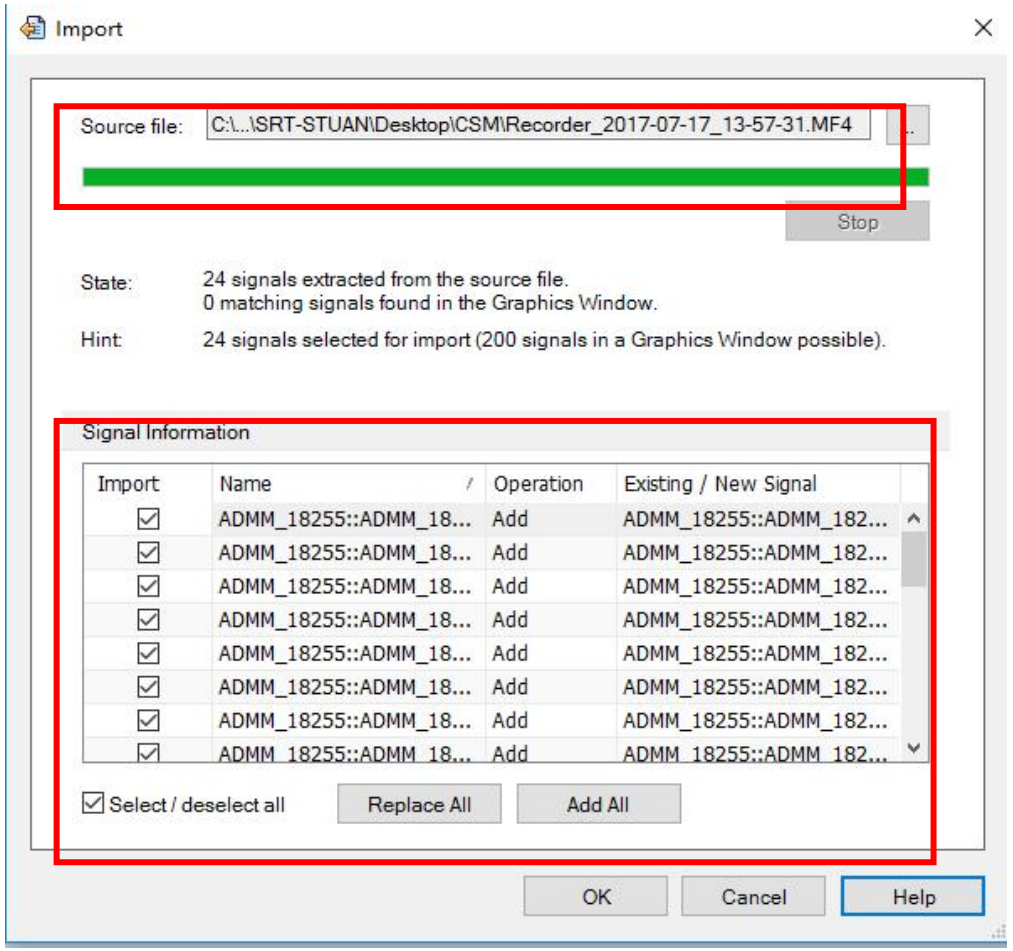
> Graphic 窗口导入信号

添加信号：

源文件

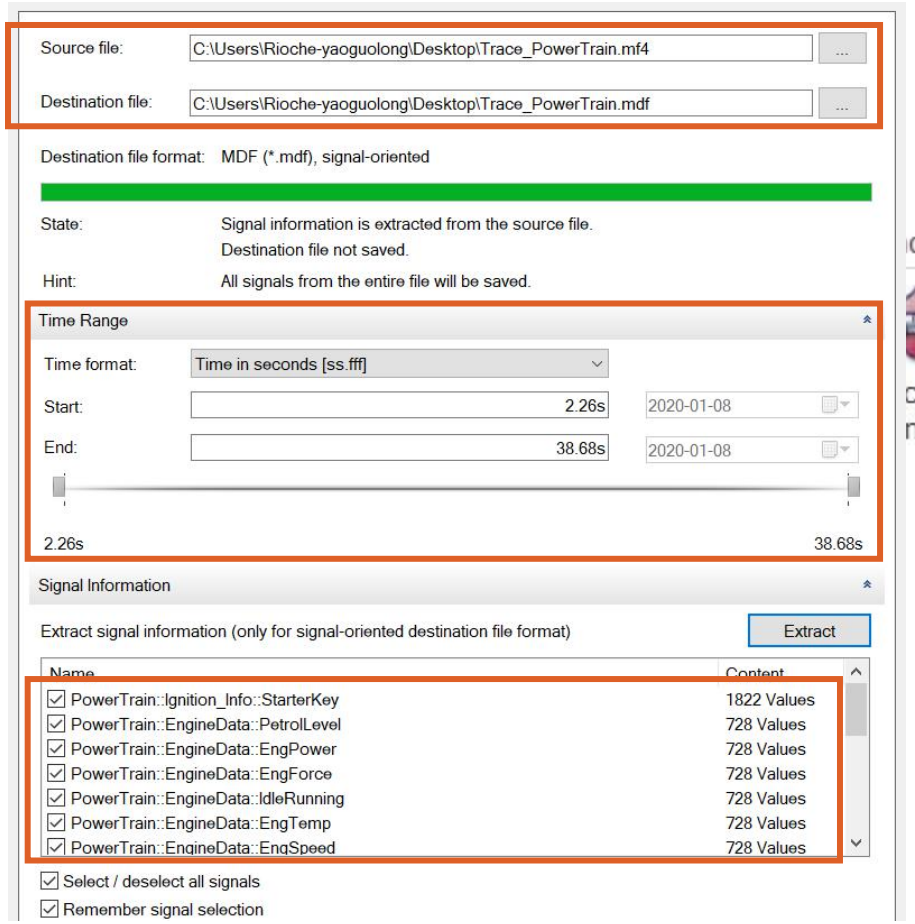
文件中信号状态

信号列表

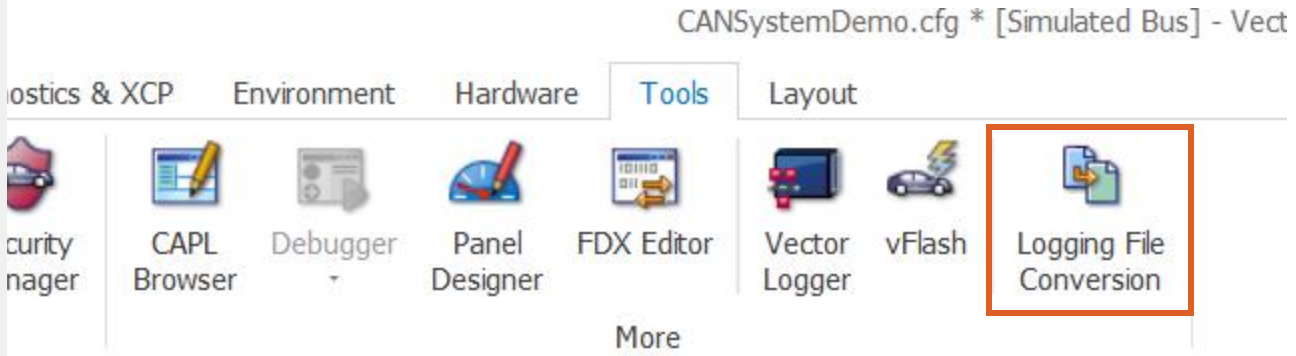


离线数据分析

> 文件转换



源文件选择，目标文件格式及命名



时间范围设置

信号选择

目录

CONTENTS

系统和环境变量

- 系统和环境变量

Environment | System Variables……

- 应用在仿真环境中
- 用于所有分析窗口
- 与Panel控件关联显示
- 作为结果和数据的变量
- 控制节点行为
- 为CANoe扩展实现接口

系统和环境变量

> 对比系统和环境变量

环境变量

> 旧版本模式下CANoe中交互的变量
> 存储位置
DBC 文件

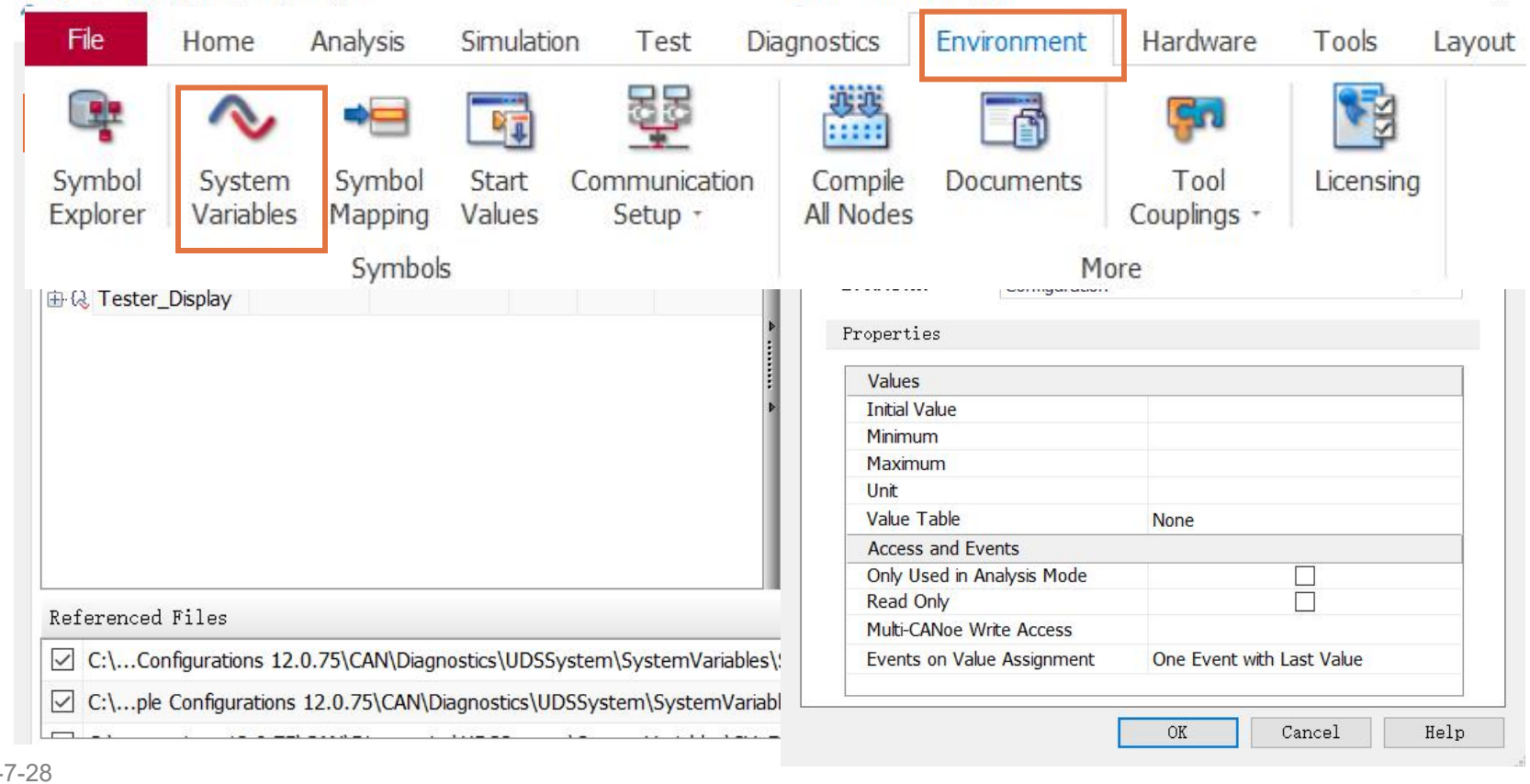
系统变量

> 独立于总线的信息交互变量
> 存储位置
CFG/XML（CANoe的工程文件）
> 可导出 XML 文件

系统变量创建

> 系统和环境变量

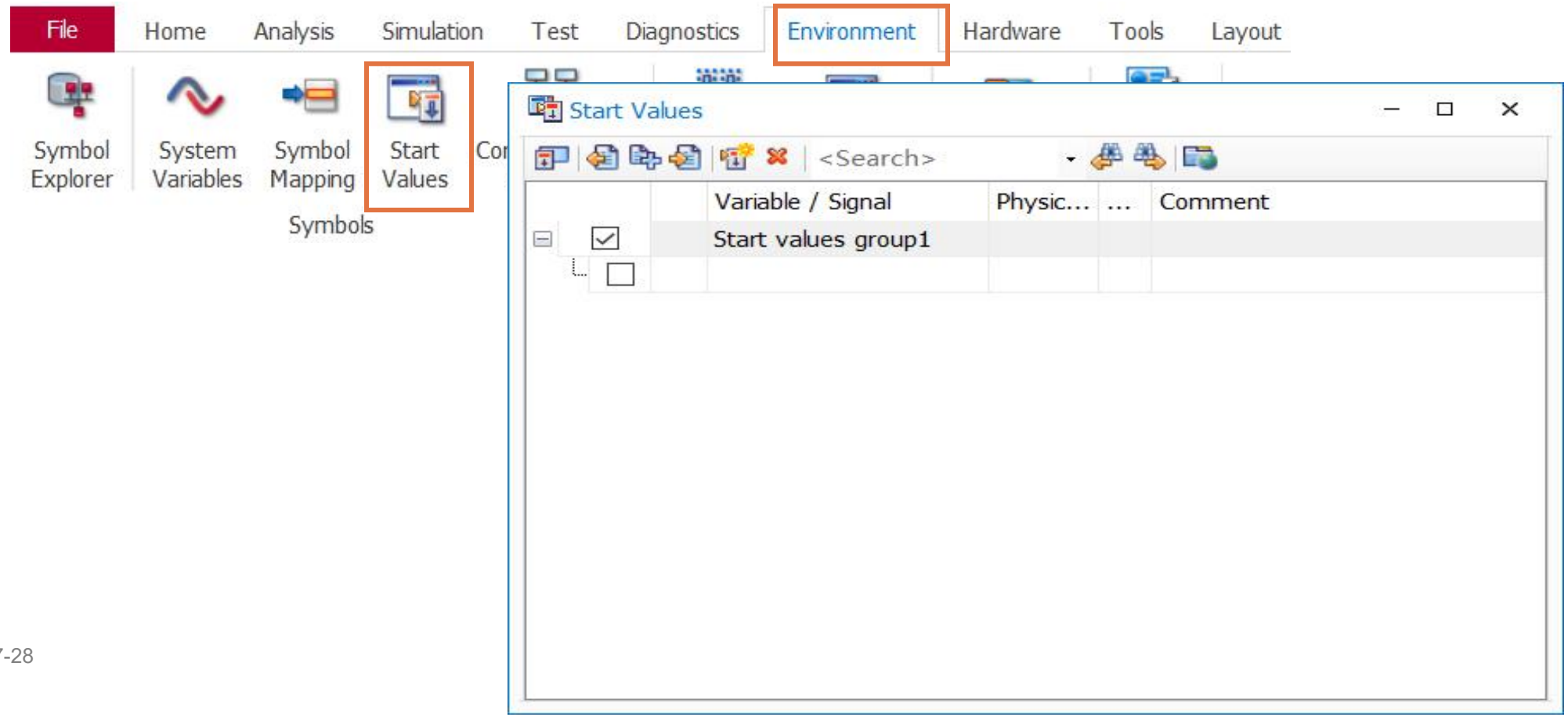
Environment | System Variables……



系统变量初始值

> 系统和环境变量

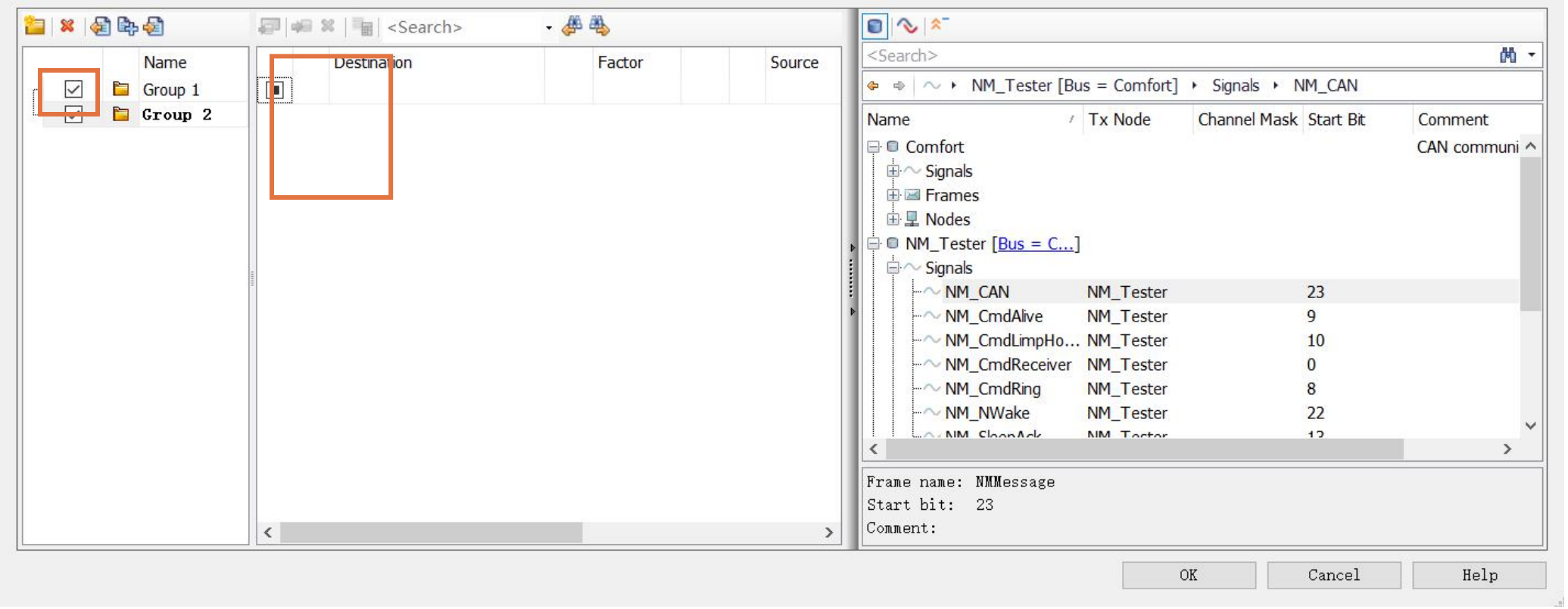
Environment | System Variables……



变量映射

> 系统和环境变量

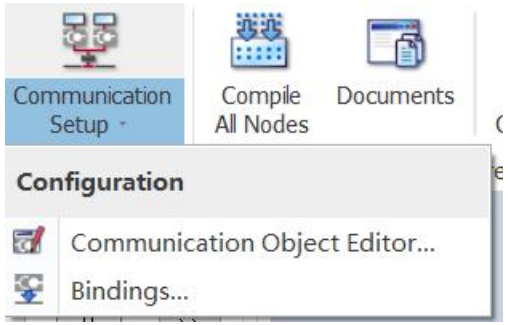
Environment | System Variables……

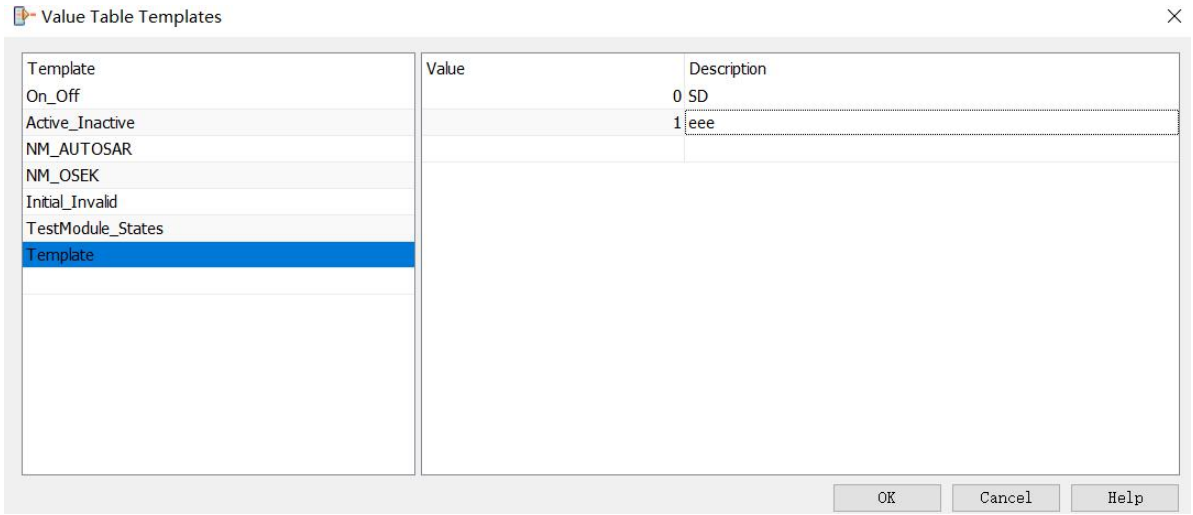


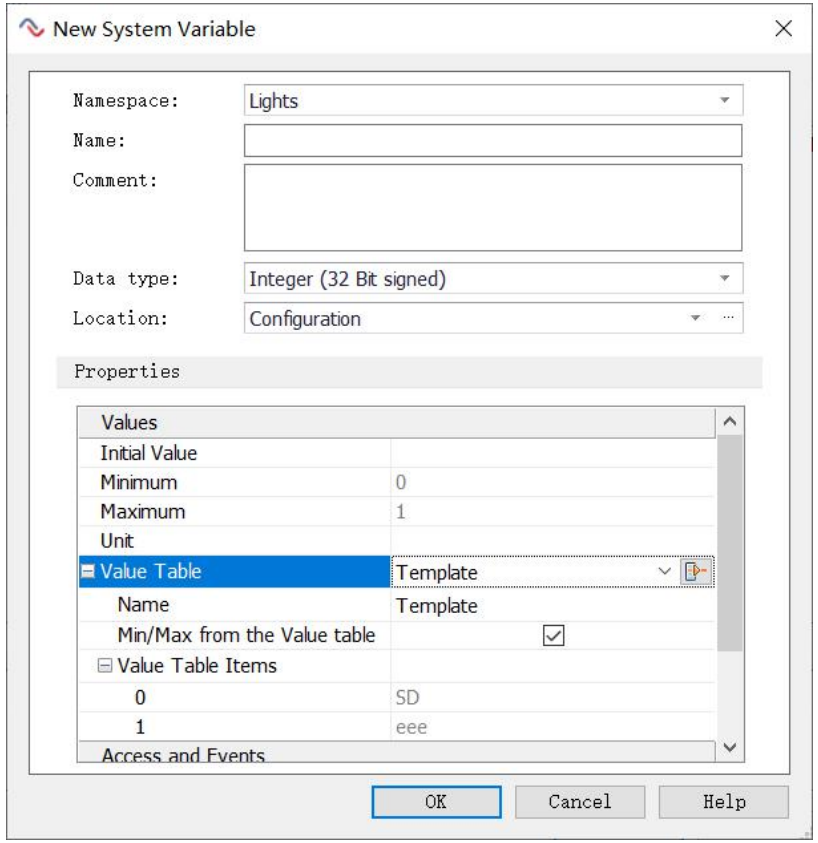
系统和环境变量

> 系统和环境变量

Environment | System Variables……





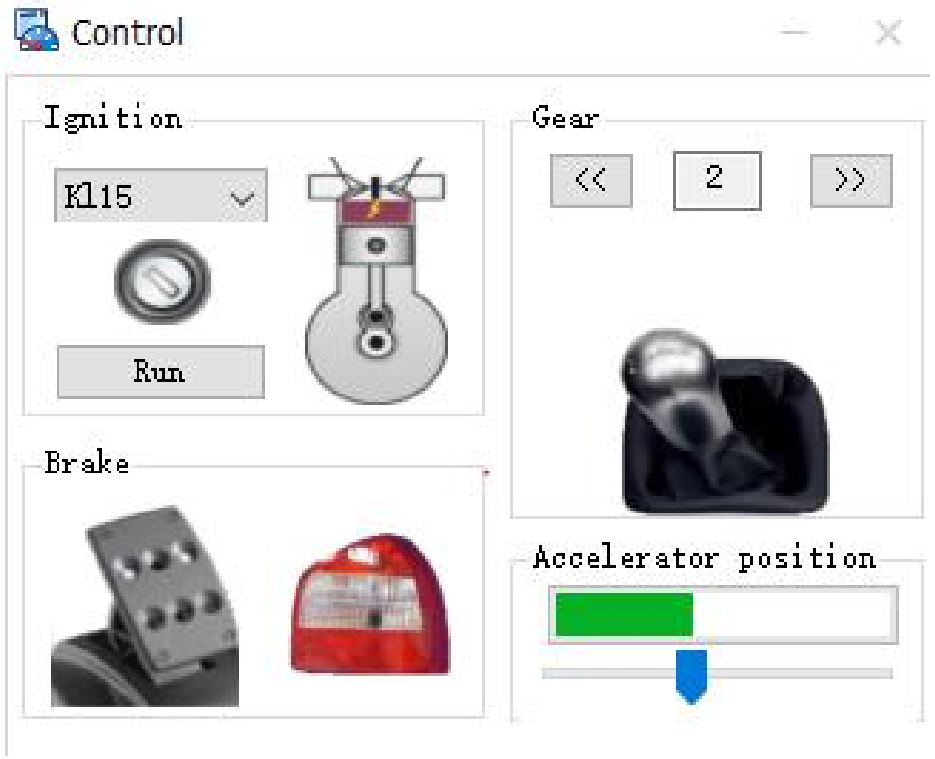


目录

CONTENTS

Panel概述

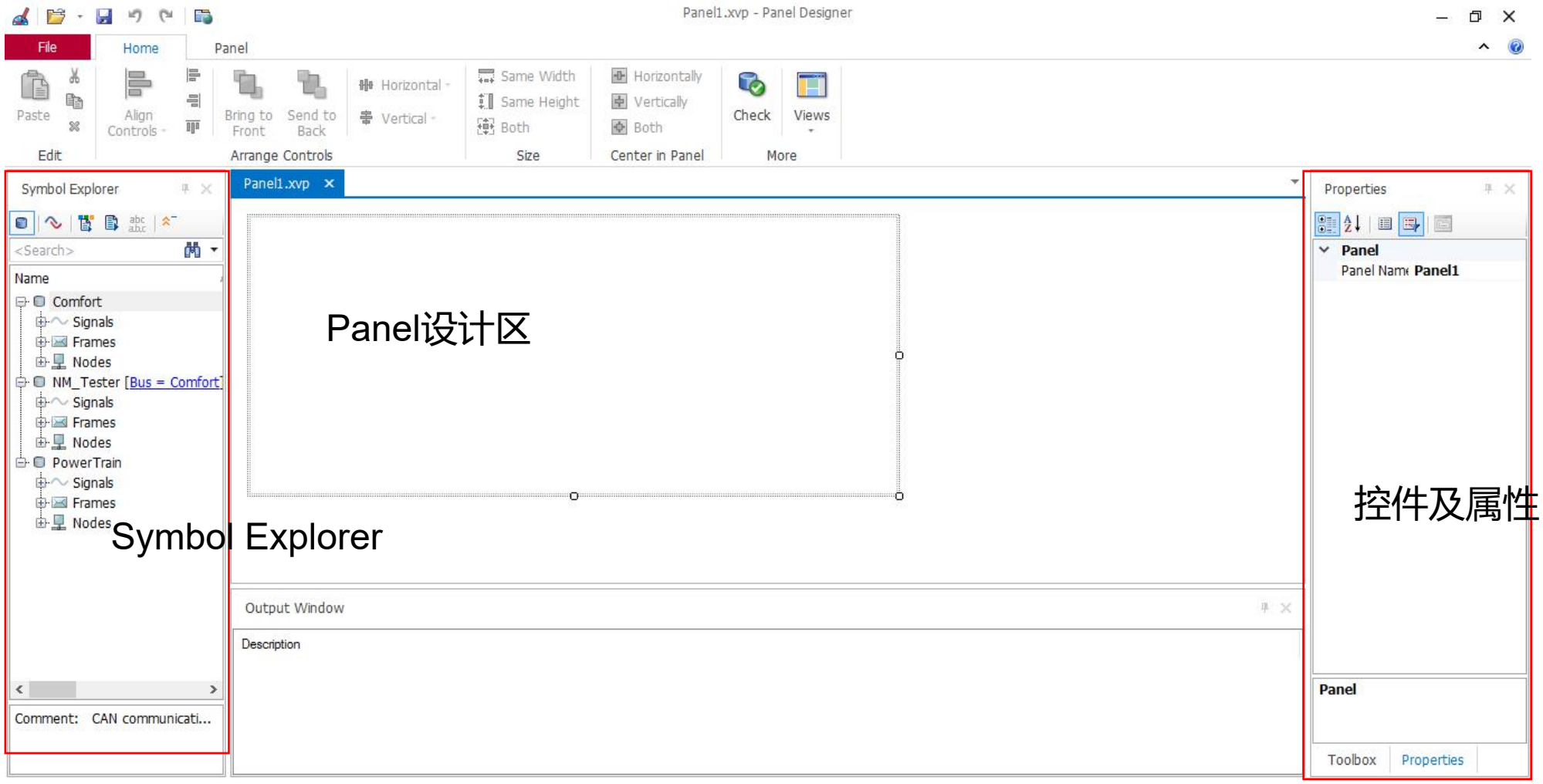
- 创建交互界面
- 显示相关信息
- 输入控制参数



Panel创建

打开Panel Designer

Panel 面板


Panel1.xvp - Panel Designer
File
Home
Panel
Paste
Edit
Align Controls
Bring to Front
Send to Back
Arrange Controls
Horizontal
Vertical
Same Width
Same Height
Both
Size
Horizontally
Vertically
Both
Center in Panel
Check
Views
More
Symbol Explorer
<Search>
Name
Comfort
Signals
Frames
Nodes
NM_Tester [Bus = Comfort]
Signals
Frames
Nodes
PowerTrain
Signals
Frames
Nodes
Comment: CAN communicati...
Panel1.xvp
Panel设计区
Output Window
Description
Properties
Panel
Panel Name Panel1
Panel
Toolbox
Properties
Symbol Explorer
控件及属性

Panel设计

Symbol Explorer
- 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

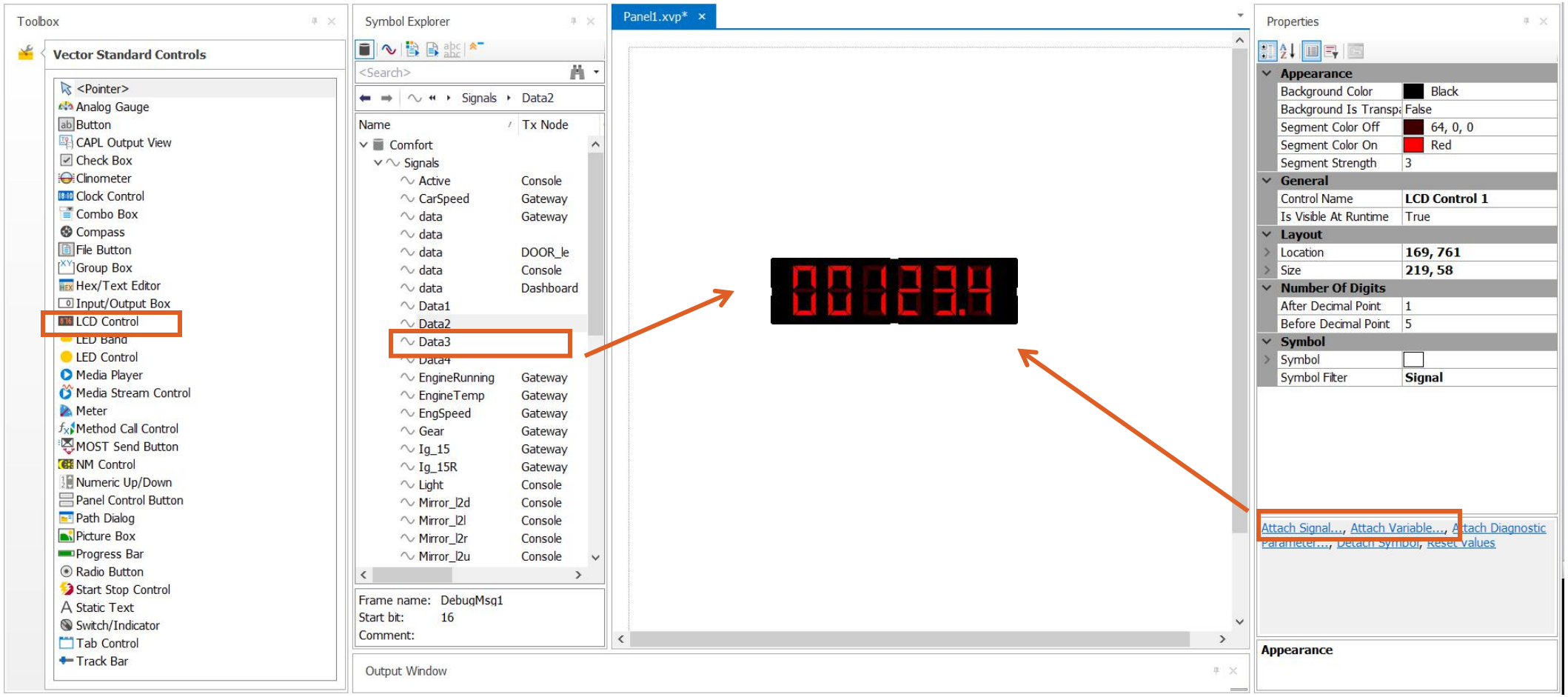
面板区
- 创建面板
- 支持同时编辑多个面板

控件区
- 显示控件
- 双击在工作区产生控件

属性区
- 显示选中控件的相关设置
- 点击某项设置后会在下方出现参数说明

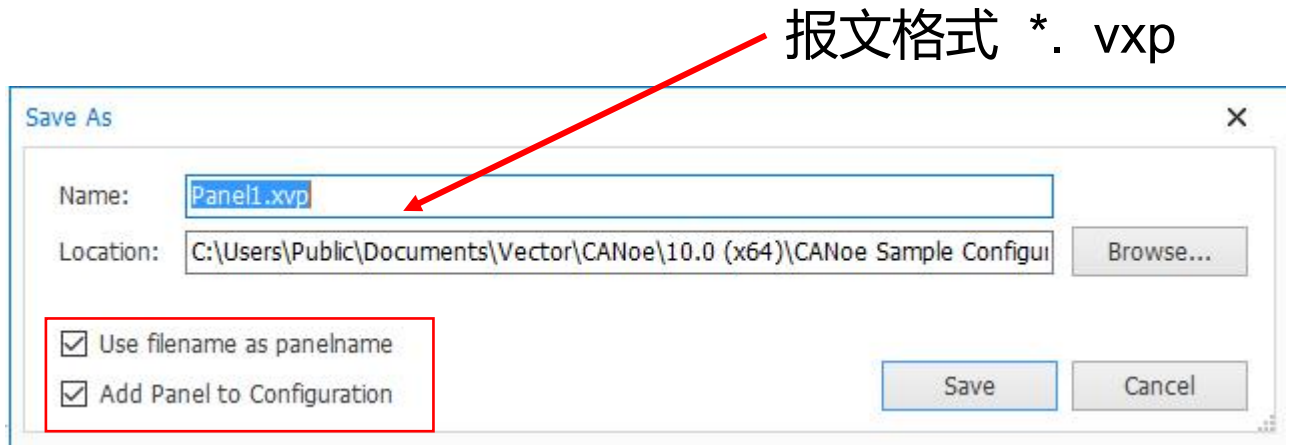
Panel设计

> 控件


Toolbox
Vector Standard Controls
<Pointer>
Analog Gauge
Button
CAPL Output View
Check Box
Clinometer
Clock Control
Combo Box
Compass
File Button
Group Box
Hex/Text Editor
Input/Output Box
LCD Control
LED Band
LED Control
Media Player
Media Stream Control
Meter
Method Call Control
MOST Send Button
NM Control
Numeric Up/Down
Panel Control Button
Path Dialog
Picture Box
Progress Bar
Radio Button
Start Stop Control
Static Text
Switch/Indicator
Tab Control
Track Bar
Symbol Explorer
<Search>
Signals
Data2
Name
Tx Node
Comfort
Signals
Active Console
CarSpeed Gateway
data Gateway
data
data DOOR_le
data Console
data Dashboard
Data1
Data2
Data3
Data4
EngineRunning Gateway
EngineTemp Gateway
EngSpeed Gateway
Gear Gateway
Ig_15 Gateway
Ig_15R Gateway
Light Console
Mirror_l2d Console
Mirror_l2l Console
Mirror_l2r Console
Mirror_l2u Console
Frame name: DebugMsg1
Start bit: 16
Comment:
Panel1.xvp*
Properties
Appearance
Background Color Black
Background Is Transp: False
Segment Color Off 64, 0, 0
Segment Color On Red
Segment Strength 3
General
Control Name LCD Control 1
Is Visible At Runtime True
Layout
Location 169, 761
Size 219, 58
Number Of Digits
After Decimal Point 1
Before Decimal Point 5
Symbol
Symbol
Symbol Filter Signal
Attach Signal..., Attach Variable..., Attach Diagnostic Parameter..., Detach Symbol, Reset Values
Appearance
Output Window

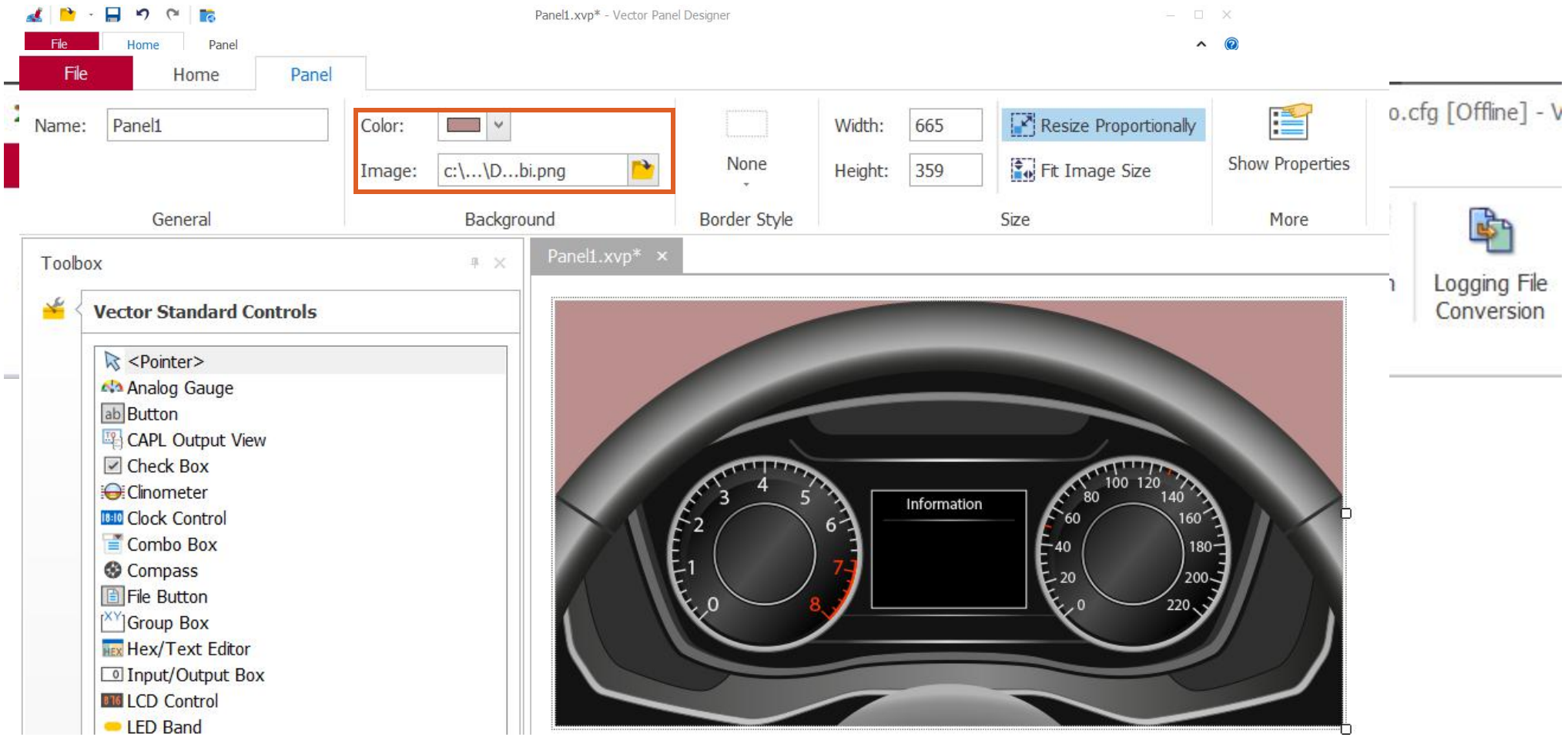
Panels 保存

> 保存格式*. vxp
> 制作好的Panel可以在不同工程中复用



Panel设计

> 工具


Panel1.xvp* - Vector Panel Designer
File
Home
Panel
Name:
Panel1
Color:
Image:
c:\...\D...bi.png
None
Width:
665
Height:
359
Resize Proportionally
Fit Image Size
Show Properties
General
Background
Border Style
Size
More
Logging File Conversion
Toolbox
Vector Standard Controls
<Pointer>
Analog Gauge
Button
CAPL Output View
Check Box
Clinometer
Clock Control
Combo Box
Compass
File Button
Group Box
Hex/Text Editor
Input/Output Box
LCD Control
LED Band
Panel1.xvp*
Information

Panel设计

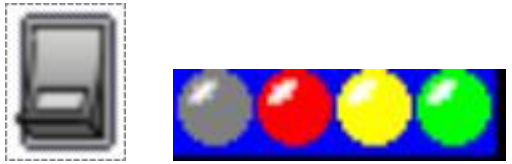
> 控件简介

分类：显示与控制

Signalname:

输入输出

CANoe 开始/停止



状态显示

Button

动作触发

控件

Panel 控件

Button

◆用于触发已配置的动作
◆发动机启动，一键落锁等
◆参数设置
  ➢只有0和1两个状态

Check box

◆用来控制和显示元素
◆仿真过程中注入错误
◆信号设定为特定的数值

Combo box

◆控制和显示元素的符号
◆设置发动机启动序列
◆设置手刹状态

Input/Output box

◆控制和元素的输入与输出信息
◆显示轮速
◆设置车速到某个特定值
◆超出界限值，显示红色

控件

Panel 控件

Progress bar

◆ 在一定范围内显示进度

Track bar

◆ 在一定范围内控制和显示元素

Clock Control

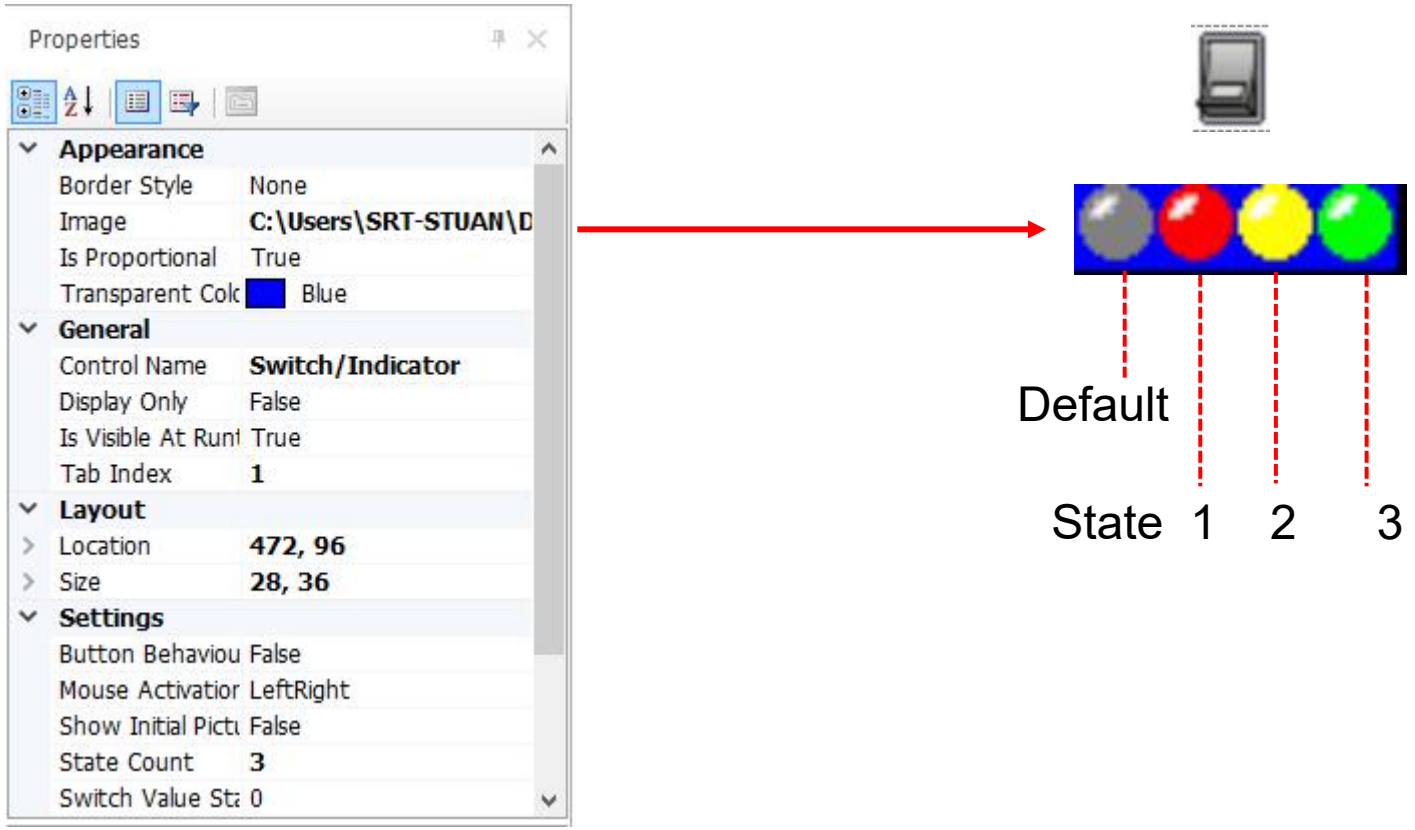
◆ 显示时钟，默认为PC机时间

LCD Control

◆显示浮点型数字

状态控件

位图设计控件显示界面



控件函数

Panel CAPL 接口函数

- SetControlBackColor
- SetControlForeColor
- SetControlColors
- SetDefaultControlColors
- setControlVisibility
- openPanel
- closePanel
- enableControl

输出参数到控件中

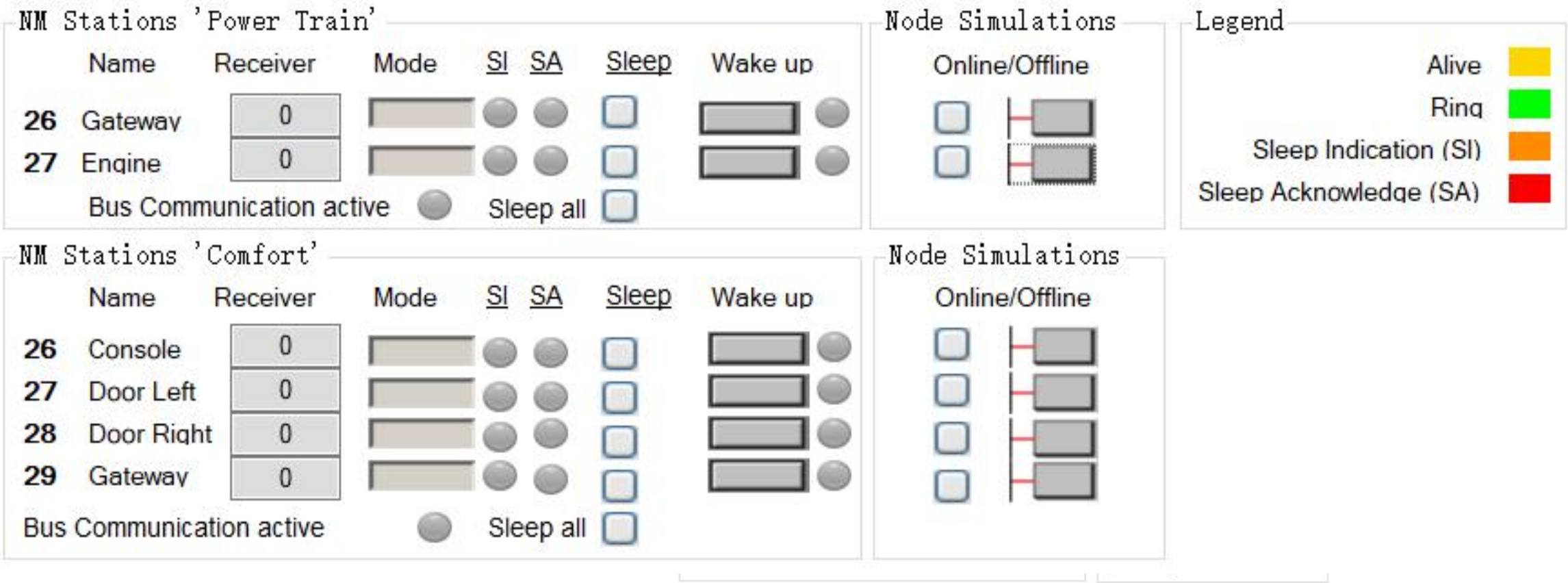
putValueToControl (“面板名称”， “**CAPL output view**”, 输出对象， 行参数， 列参数)

deleteControlContent

仅限于**CAPL output view**控件。

Panel设计

> 概览


NM Stations 'Power Train'
Name
Receiver
Mode
SI
SA
Sleep
Wake up
26 Gateway
0
27 Engine
0
Bus Communication active
Sleep all
Node Simulations
Online/Offline
Legend
Alive
Ring
Sleep Indication (SI)
Sleep Acknowledge (SA)
NM Stations 'Comfort'
Name
Receiver
Mode
SI
SA
Sleep
Wake up
26 Console
0
27 Door Left
0
28 Door Right
0
29 Gateway
0
Bus Communication active
Sleep all
Node Simulations
Online/Offline

目录

CONTENTS

CAPL语言

> 概述

特征

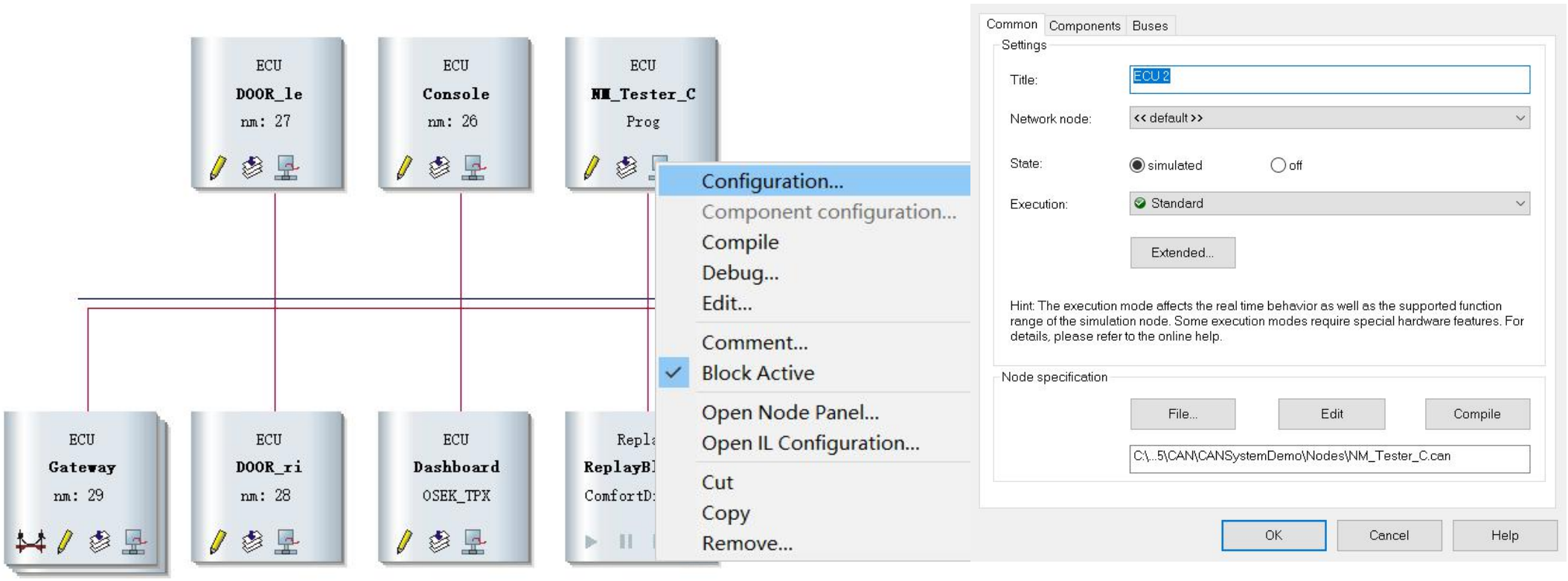
1，CAPL全称Communication Access Programming Language，专用于CANalyzer和CANoe软件开发的程序语言
2，类C语言，在语法及概念上与C语言类似
3，事件驱动，时间事件，按键事件，报文事件等

应用

1，节点仿真
2，网络仿真
3，节点测试

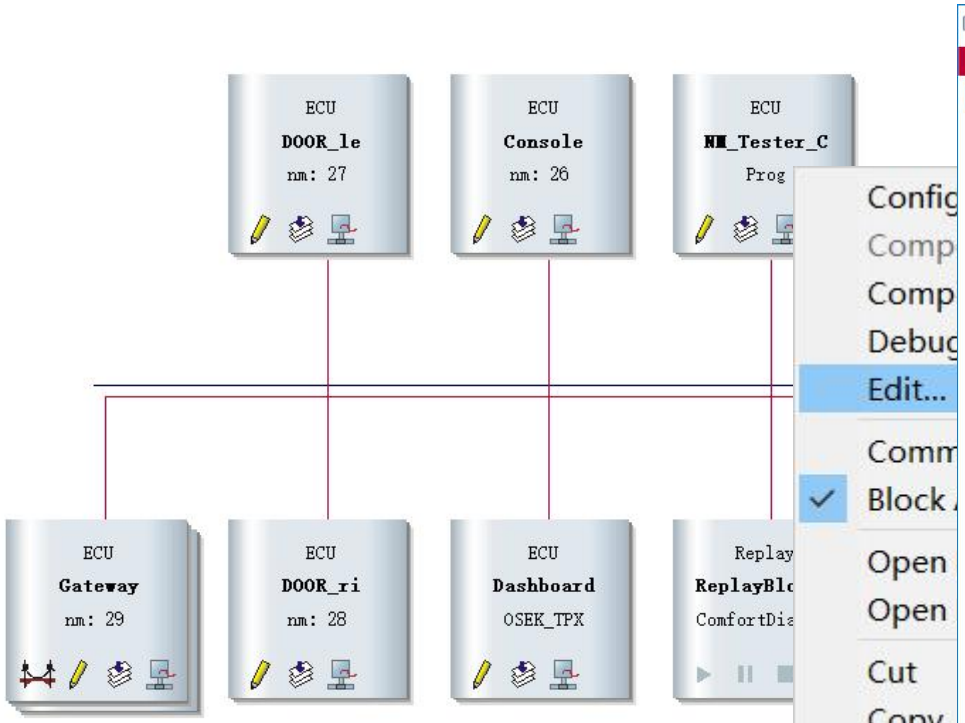
CAPL语言

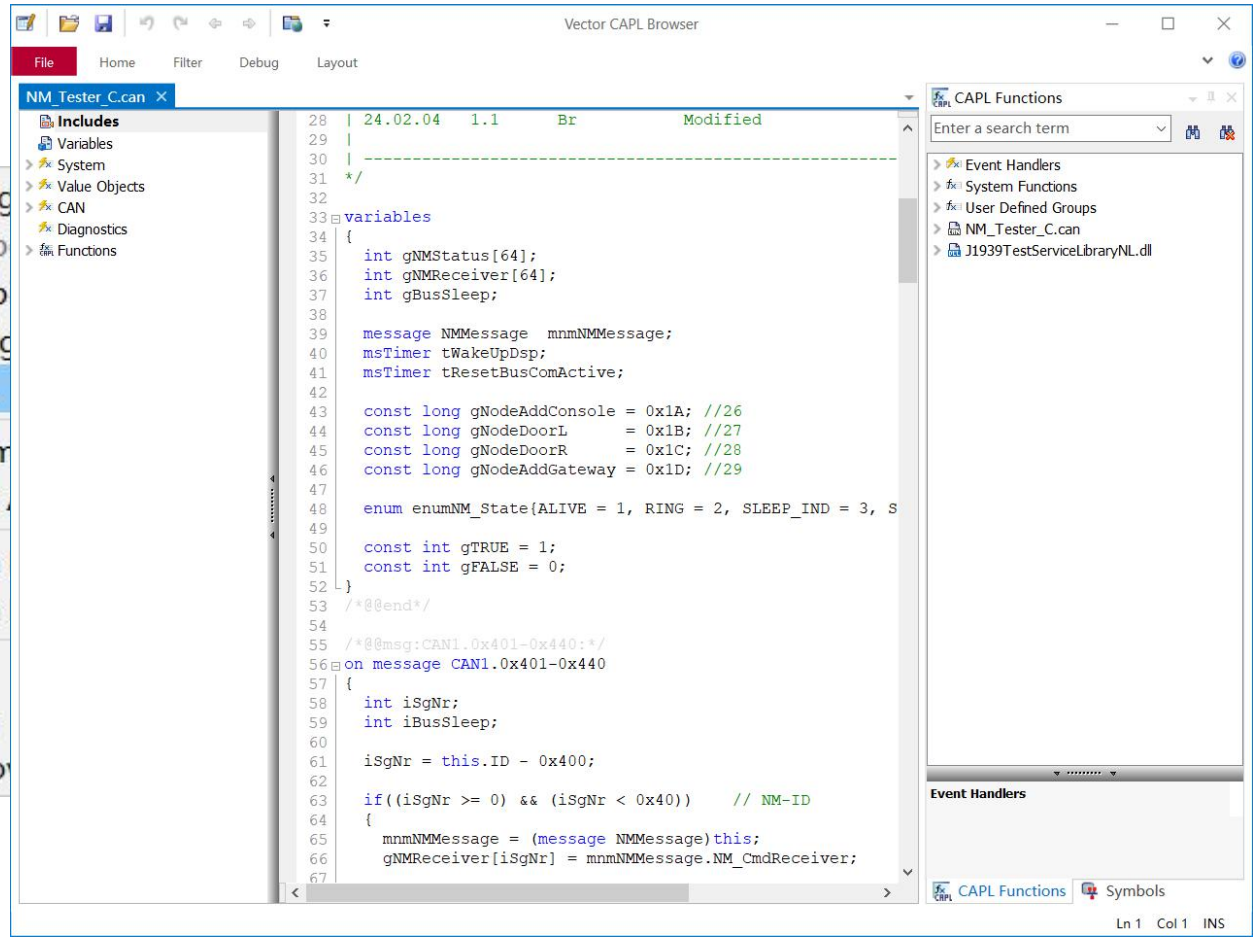
> 配置


ECU
DOOR_le
nm: 27
ECU
Console
nm: 26
ECU
NM_Tester_C
Prog
ECU
Gateway
nm: 29
ECU
DOOR_ri
nm: 28
ECU
Dashboard
OSEK_TPX
Configuration...
Component configuration...
Compile
Debug...
Edit...
Comment...
Block Active
Open Node Panel...
Open IL Configuration...
Cut
Copy
Remove...
Common
Components
Buses
Settings
Title:
ECU 2
Network node:
<< default >>
State:
simulated
off
Execution:
Standard
Extended...
Hint: The execution mode affects the real time behavior as well as the supported function range of the simulation node. Some execution modes require special hardware features. For details, please refer to the online help.
Node specification
File...
Edit
Compile
C:\...5\CAN\CANSystemDemo\Nodes\NM_Tester_C.can
OK
Cancel
Help

CAPL语言

> 配置

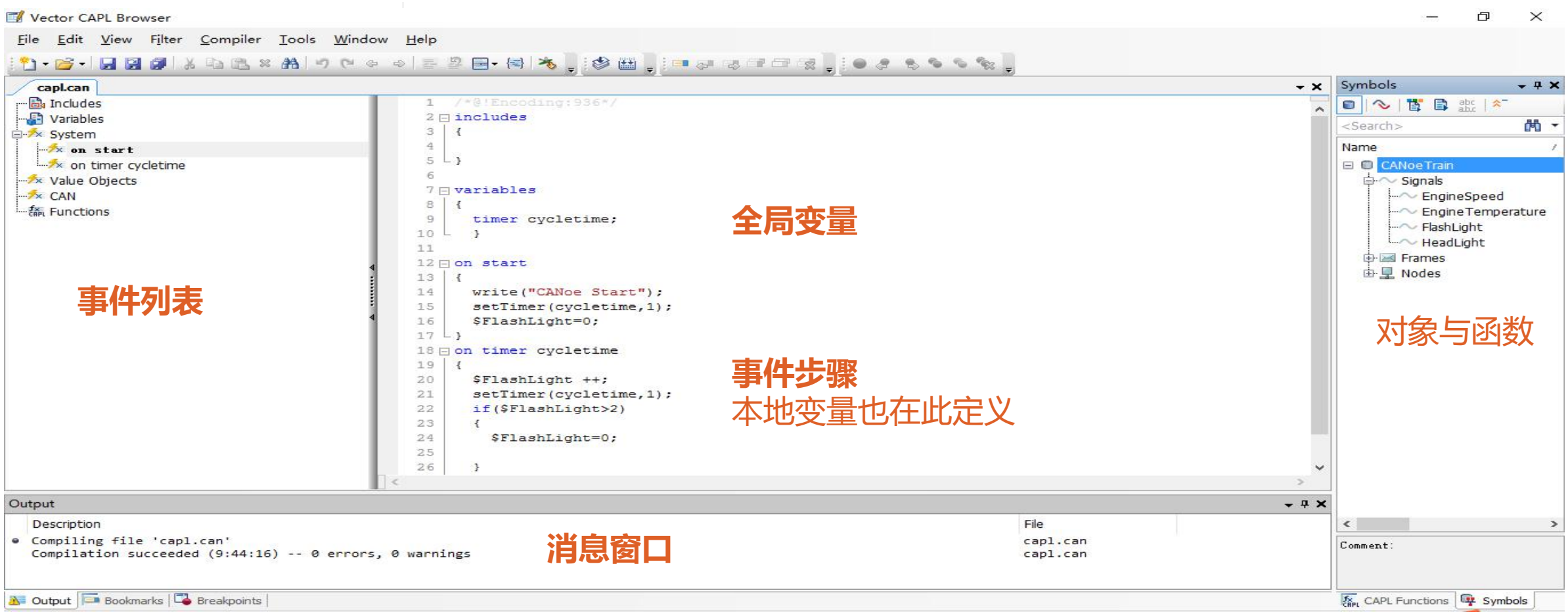





```
28  | 24.02.04   1.1      Br          Modified
29  |
30  | ----------------------------------------------------------
31  */
32
33  variables
34  {
35    int gNMStatus[64];
36    int gNMReceiver[64];
37    int gBusSleep;
38
39    message NMMessage  mnmNMMessage;
40    msTimer tWakeUpDsp;
41    msTimer tResetBusComActive;
42
43    const long gNodeAddConsole = 0x1A; //26
44    const long gNodeDoorL      = 0x1B; //27
45    const long gNodeDoorR      = 0x1C; //28
46    const long gNodeAddGateway = 0x1D; //29
47
48    enum enumNM_State{ALIVE = 1, RING = 2, SLEEP_IND = 3, S
49
50    const int gTRUE = 1;
51    const int gFALSE = 0;
52  }
53  /*@@end*/
54
55  /*@@msg:CAN1.0x401-0x440:*/
56  on message CAN1.0x401-0x440
57  {
58    int iSgNr;
59    int iBusSleep;
60
61    iSgNr = this.ID - 0x400;
62
63    if((iSgNr >= 0) && (iSgNr < 0x40))    // NM-ID
64    {
65      mnmNMMessage = (message NMMessage)this;
66      gNMReceiver[iSgNr] = mnmNMMessage.NM_CmdReceiver;
67
```

CAPL

> CAPL编辑器



CAPL

> CAPL编辑器

变量

```
Variables
{
  … //申明全局变量
}
```

事件处理

```
on start {
 … //过程指令块
}
on message xxx
{
 … //过程指令块
}
on key ‘1’
{
… //过程指令块
 }
```

自定义函数

```
My_function_1(Para_1, Para_2, …)
{
  … //函数体
}
…
My_function_n(Para_1, Para_2, …)
{
  … //函数体
}
```

CAPL 数据类型

> CAPL 数据类型

Type		Name	Bit	Note
Whole numbers	signed	int	16	
		long	32	
		int64	64	
	unsigned	byte	8	
		word	16	
		dword	32	
		qword	64	
Floating decimal numbers		float	64	according to IEEE
		double	64	according to IEEE
Individual character		char	8	
Message variables	for CAN	message		for CAN messages
Timer variables	No unit	timer		time unit s (and/or ns)
	Unit milliseconds	mstimer		time unit ms

CAPL 条件和循环

> CAPL条件和循环

Conditions

if

```
if      (a == 100) write ("Value is 100");
else if (a == 500) write ("Value is 500");
else               write ("Value is not 100 or 500");
```

switch

```
switch (a)
{
   case 100: write ("Value is 100"); break;
   case 500: write ("Value is 500"); break;
   default:  write ("Value is not 100 or 500"); break;
}
```

Loops

for

```
for (a = 0; a < 100; a++)
{
   write ("Value of %d", a);
}
```

while

```
while (a < 100)
{
   write ("Value of %d", a);
   a++;
}
```

do while

```
do
{
   write ("Value of %d", a);
   a++;
}  while (a < 100)
```

CAPL

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	PreStart	CANoe初始化时执行	on preStart
	Start	测量开始时执行	on start
	StopMeasuremet	测量结束时执行	on stopMeasurement
CAN控制器事件	BusOff	硬件检测到BusOff时执行	on busOff
	ErrorActive	硬件检测到ErrorActive时执行	on errorActive
	ErrorPassive	硬件检测到ErrorPassiv时执行	on errorPassive
	WarningLimit	硬件检测到WarningLimi时执行	on warningLimit
CAN消息事件	自定义	接收到指定的消息时执行	on message Message
时间事件	自定义	定时时间时执行	on timer Timer
键盘事件	自定义	指定的键被下时执行	on key Key
错误帧事件	Error Frame	硬件每次检测到错误帧时执行	on errorFrame
环境变量事件	自定义	指定的环境变量值改变时执行	on envVar EnvVar

CAPL

> CAPL事件

On key ‘a’	对按键 ‘a’ 反应
On key ‘ ’	对按键 ‘空格’ 反应
On key F1	对按键 ‘F1’反应
On key ctrlF12	对按键 ‘ctrl+F12’ 反应
On key Page Up	对按键 ‘Page Up’ 反应
On key Home	对按键 ‘Home’ 反应
On key *	对所有按键反应

CAPL

CAPL计时器事件

- 定义一个timer 变量

```
Timer <Name TimerVariable>;          //unit: s
msTimer <Name TimerVariable>;        //unit: ms
```

- 设置或者取消Timer

```
setTimer   (<Name TimerVariable>, <Value>);
setTimerCyclic   (<Name TimerVariable>, <Value>);
cancelTimer   (<Name TimerVariable>);
```

- 定义事件步骤

```
on timer  <Name TimerVariable>
{
    //code to be executed
```

CAPL

CAPL计时器示例

```
msTimer  cycletime;       //unit: ms
int i ;
On key    'a'
{
      setTimerCyclic    (cycletime , 1000);
      write (  "Timer was started"  );
}
on timer  cycletime
 {
       i++;
       write (  "Timer cycle pass: %d"  ,i);
}
```

定时器声明示例

- 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

CAPL

> Write 输出窗口

```
int h=100;
char ch =‘a’;
write(“Hundred as a number:%d, %x”, h, h);
write(“Hundred as a string:%s”, ” s100”);
```

CAPL

> CAPL message 语法

语法	解释
on message 123	React to message 123 (dec), regardless of receiving chip
on message 0x123x	React to message 123 (hex), regardless of receiving chip
on message EngineData	React to message EngineData
on message CAN1.123	React to message 123 if it is received by CAN1 chip
on message *	React to all messages
on message CAN2.*	React to all messages received by CAN2 chip
on message 0,1,10-20	React to messages 0, 1 and 10 through 20

CAPL

> CAPL message结构

使用CAPL发送报文，需要先定义报文

✓ 定义一个报文

Syntax: message<Message ID > <Name of variable>

✓ 多种报文对象信息

Syntax: <Name of variable>.selector =value

✓ 输出报文

Syntax: output (<Name of variable>);

CAPL

CAPL message示例

- Message

1. 定义 CAN-Message

```
message can2.125 msg = { // define CAN-Message
    dlc = 1,
    byte(0) = 1
  };
```

2. 在CAN channel 3输出错误帧

```
output (msg); // output Message
output (CAN3.errorFrame);
//output error frame on CAN channel 3
```

3. On Message事件

```
on message CAN2.* {
    message CAN1.* msg;
    if(this.dir != rx) return; //important!
    msg = this;
    output(msg);
    }
```

CAPL

> CAPL message示例

1. 定义 CAN-Message

```
message 0x125 msg ;// define CAN-Message
msg .dlc = 8;
msg .byte(0) = 1;
msg.byte(7)=6;
output(msg);
```

2.读取CAN-Message

```
On message *
{
If (this.id==0x125)
write(  “Msg ID =0x%x,Msg.DLC=0x%.2x,”
this .ID, this .dlc);
}
```

CAPL

> CAPL CAN Selectors

Type	Description	Value range
BYTE(n) **WORD(n)** **DWORD(n)** **QWORD(n)**	Are used to evaluate and write the data bytes (int, long and char are also possible)	n = 0...7 n = 0...6 n = 0...4 n = 0
ID	Message identifier	0...0x7FF (0...2047), 0...0x1FFFFFFF (0...536870911)
DLC	Message length	0...8
CAN	Channel number	1...32
DIR	Direction of transmission	RX(0), TX(1), TXREQUEST(2)
TIME	Time stamp	Resolution 10 microseconds (CANcardXL)
RTR	Selector for distinguishing between Data and Remote frames	RTR = 0 → Data frame RTR = 1 → Remote frame
MSGFLAGS	Special information from the CAN controller	0x04 → System in single-wire operation 0x08 → Message in "High Voltage" mode 0x10 → Remote frame 0x40 → Transmit Acknowledge (DIR = Tx) 0x80 → Request Acknowledge
TYPE	For more efficient evaluation → a combination of RTR and DIR	See Notes page

CAPL

> 按键事件

```
on key ‘a’
{
message 0x123  Msg;
Msg.temperature.phys=50;
Msg.speed.phys=4000;
output(Msg);
}
on key ‘b’
{
message 100 Msg10;
Msg10.dlc=1;
Msg10.byte(0)=0x0B;
output(Msg10);
}
```

```
on message 0x10
{
if(this.byte(2)==0xFF)
write(“Third byte of the message is invalid”);
}
on message Msg
{
if(this.temperature.phys>=150)
write(“Warning: critical temperature”);
}
```

CAPL

PG

✓ 1.Defined PG

```
variables {
    pg 0xFEE1 pg_1 = { // Definition of a parameter group
};
```

✓ 2.output PG

```
on key F1 {
    output (pg_1); // Output parameter group
}
```

CAPL

PG

- pg TC1 pg_tc1 = {
  SA = 0x34,
  DA = 0x24
  };
  output(pg_tc1);

CAPL

PG

3. PG Event

```
on pg 0xFEE9
      //React to receipt of the parameter group 0xFEE9 ("FuelConsumption");


on pg RetarderFluids
      //React to parameter group "RetarderFluids"


on pg 0xFF05-0xFF09
      //React to parameter group number 0xFF05 to 0xFF09
```

CAPL

CAPL 信号示例

读信号和设置信号

✓ 读信号值

```
value = getSignal(LightState::OnOff); // form 1
value = getSignal("LightState::OnOff"); // form 2
value = getSignal("CAN1::LightState::OnOff");//form 2, with channel
```

✓ 设置信号值

```
setSignal(LightState::OnOff, 1.0); // form 1
setSignal("LightState::OnOff", 1.0); // form 2
setSignal("CAN1::LightState::OnOff", 1.0); // form 2, with channel
```

CAPL

> CAPL 信号示例

信号事件（signal event）

```
on signal LightSwitch::OnOff
    {
      v1 = this.raw;
      v2 = $LightSwitch::OnOff.raw;
    }
```

系统变量事件（system variables event）

```
on sysvar <sysvarname>
on sysvar sysvar::NMTester::NMOnOff
{
//is called with each value change
}
```

CAPL

> CAPL 信号示例

@可以代替getvalue和putvalue使用，获取或者定义系统变量的值

```
intValue = @EnvLightState;
@EnvTurnSignal = $LightState::TurnSignal;
```

同样，可以使用@this对系统变量的值进行处理

此用法适用于int和float类型的变量

String和data类型系统变量不能这样使用。

CAPL

Sending LIN messages

Output(linFrame msg)

Example

// Reconfigure response of the frame defined in database linFrame MotorControl

```
linFrame MotorControl frameMotorControl

frameMotorControl.RTR = 0;

frameMotorControl.byte(0)=0xDF;

frameMotorControl.byte(1)=0x20;

output(frameMotorControl);
```

CAPL

RTR

This flag is only required when a frame is being sent with output().

Meaning:

0: Response data will be reconfigured.

1: Frame header is applied to the bus - only works with the LIN hardware in Master mode.

CAPL

Sending LIN messages

Output(linFrame msg)
example

```
// Send the frame header
frameMotorControl.RTR = 1;
output(frameMotorControl);
```

CAPL

Transmits a frame header for a specific LIN frame.
linTransmitHeader(linFrame frame)

// Transmit a LIN header with the id 0x01

linFrame 0x01 frm1;

linTransmitHeader(frm1);

CAPL

linUpdateResponse(linFrame frame);
Updates the response data of a specific LIN frame

```
// Update the response of the LIN frame with the id 0x04
on linFrame 0x04
{
  long i;
  byte frm4data[8];

  for(i = 0; i < linGetDlc(this.id); i++)
  {
    frm4data[i] = this.byte(i) + 1;
  }
linUpdateResponse(this.id, linGetDlc(this.id), frm4data);
}
```

CAPL

CAPL 关键字this

在用于接收CAN对象或环境变量的事件过程中，对象的数据结构由关键字this指定。

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

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

- on message 101
  {
  float a = 0;
  a = this.Signal1.phys;
  }

CAPL

> CAPL on envVar

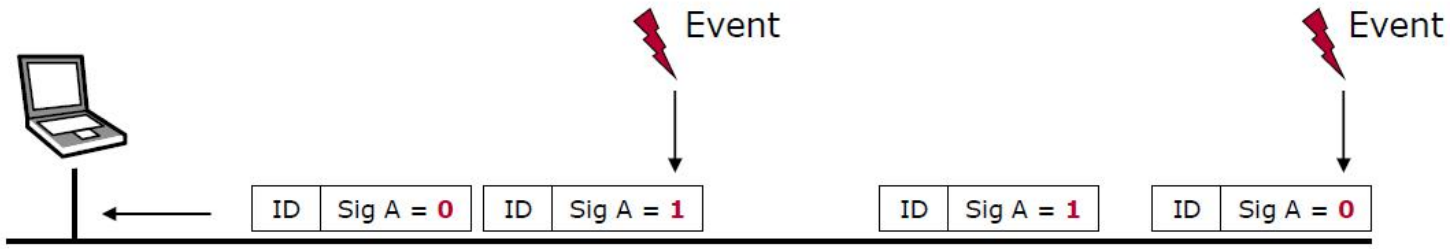
下面例子，对环境变量Switch的变化做出反应，并且把值赋给报文控制器的信号Stop。

```
on envvar Switch {

message Controller msg; // 声明传输的CAN报文

msg.Stop = getvalue(this); // 读switch的值，  // 将值赋给信号Stop

output(msg); // 输出报文到总线上

}
```

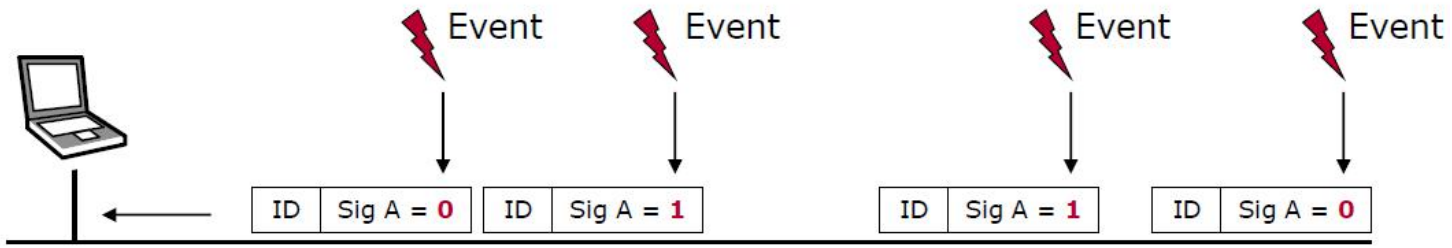
CAPL

> CAPL on signal on signal update

- On signal change: `on signal <DB Signal>`



- On signal update: `on signal_update <DB Signal>`



CAPL

> CAPL 综合示例

```
variables
{ timer cycletime;
  message 0x11 msg2;
  message 0x22 msg3;}
on start{
  write("start");
  setTimer (cycletime,1);
  msg2.DLC=2;
  msg3.dlc=1;
  msg3.byte(0)=0;
  msg2.byte(0)=2;
  msg2.byte(1)=3;
 @sysvar::sys::sys1 =0;
}
```

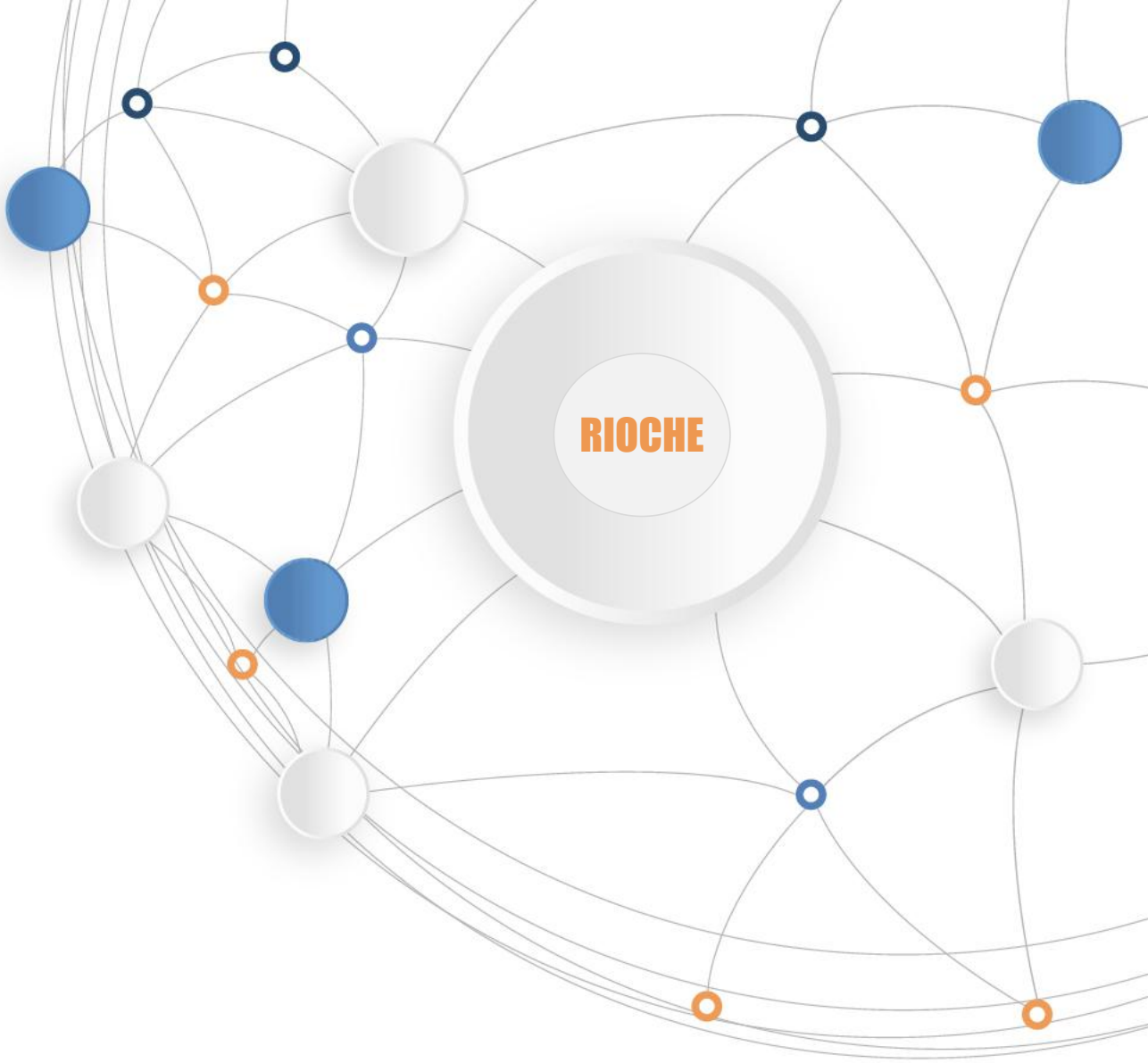
```
on key's‘
{
  message EngineState msg1;
  msg1.EngineSpeed.phys=100;
  output (msg1);
}
on timer cycletime
{
  setTimer (cycletime,1);
  output(msg2);
  @sysvar::sys::sys1++;
  if (@sysvar::sys::sys1 >2)
  {
     @sysvar::sys::sys1 =0;
  }
  //write("sys1=%d“,@sysvar::sys::sys1);
}
```

```
on message 0x11
{
  msg3.byte(0)=this.byte(0)+this.byte(1);
  output(msg3);
}
```

结束！

Thanks.

锐意进取，勤勉致精.



上海锐勤电子科技有限公司