

# CANoe 基础培训

上海锐勤电子科技有限公司

## CONTENTS



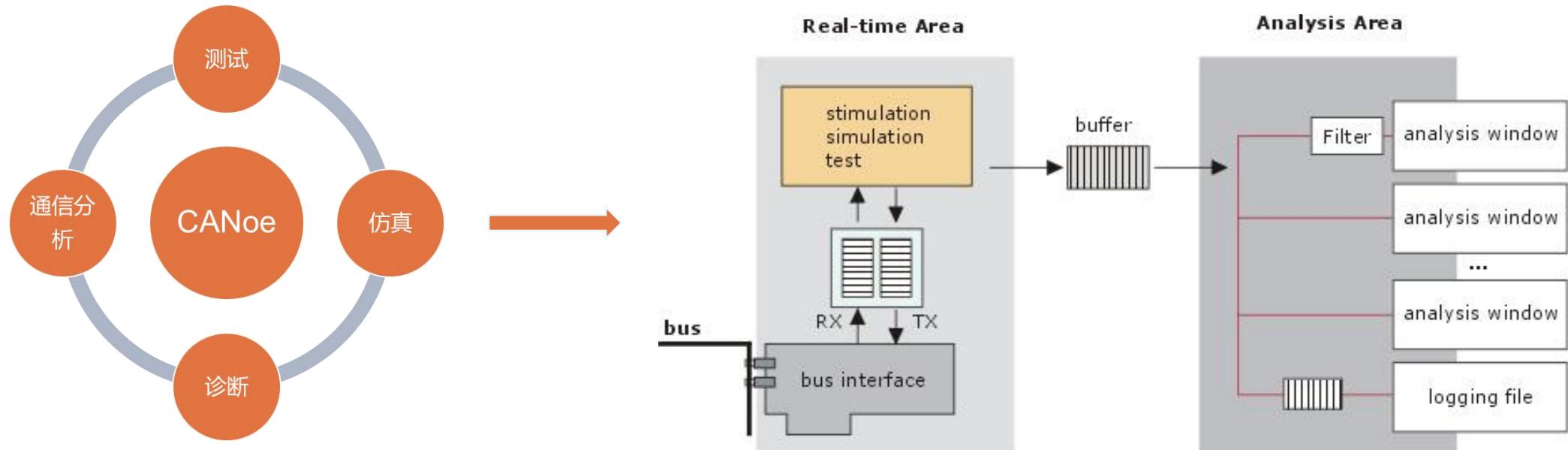
1. 概览
2. 工程创建
3. 报文发送
4. 分析窗口
5. 过滤功能模块
6. 数据记录
7. 离线分析
8. 系统变量环境变量
9. Panel设计
10. CAPL语言

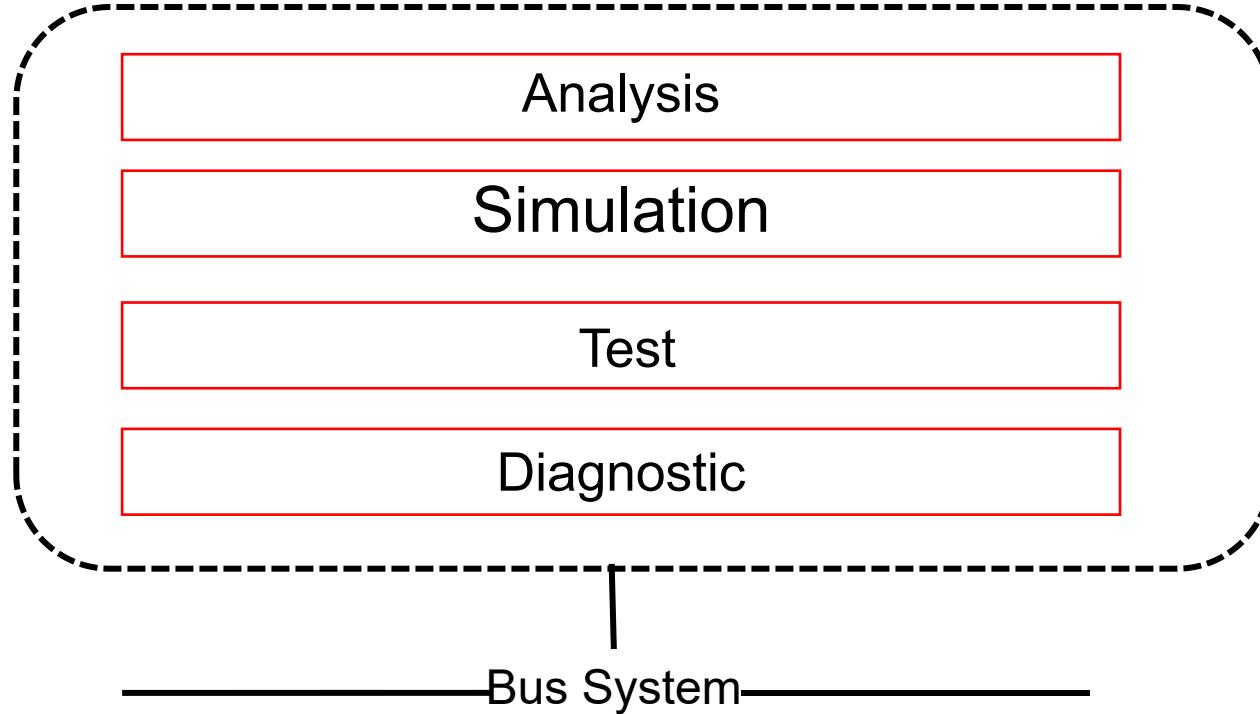
## CONTENTS



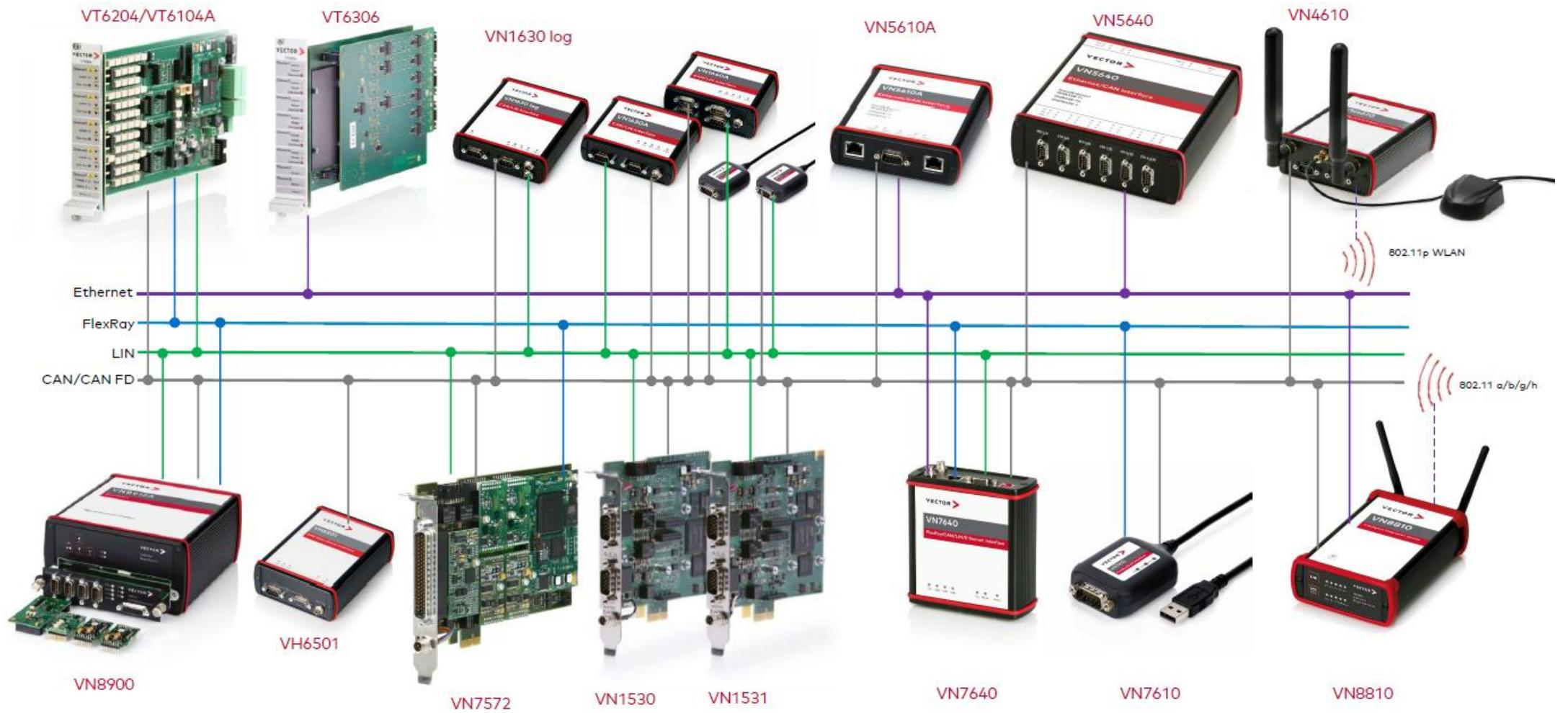
1. 概览
2. 工程创建
3. 报文发送
4. 分析窗口
5. 过滤功能模块
6. 数据记录
7. 离线分析
8. 系统变量环境变量
9. Panel设计
10. CAPL语言

CANoe (CAN Open 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。



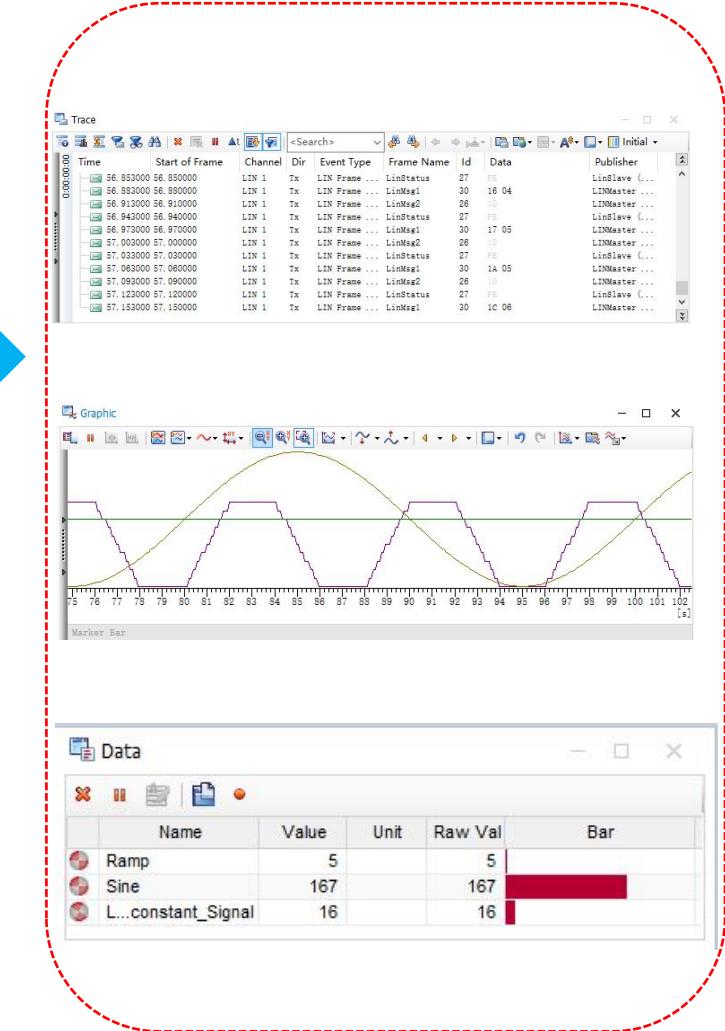
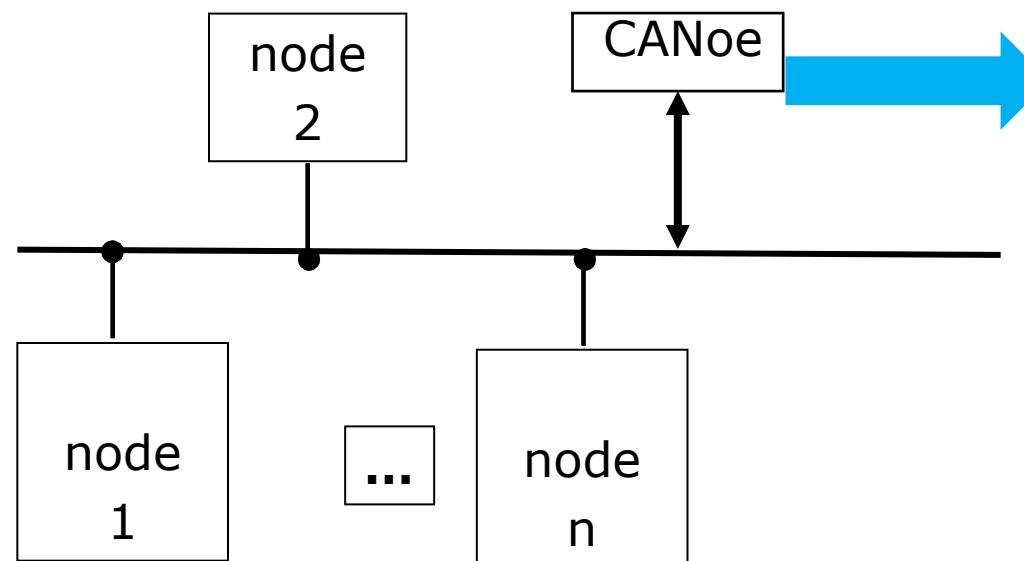


# CANoe基础应用 概述



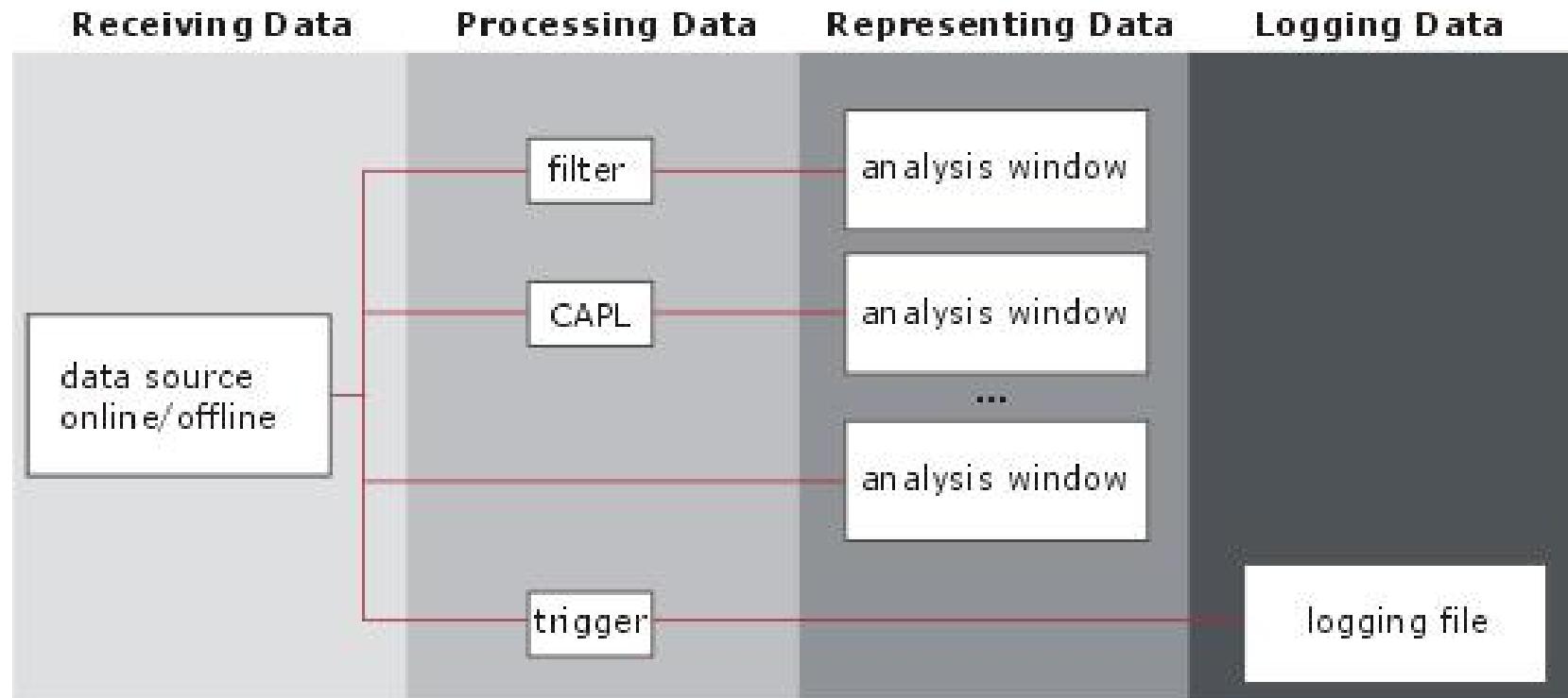


## CANoe 作为分析工具



## > 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

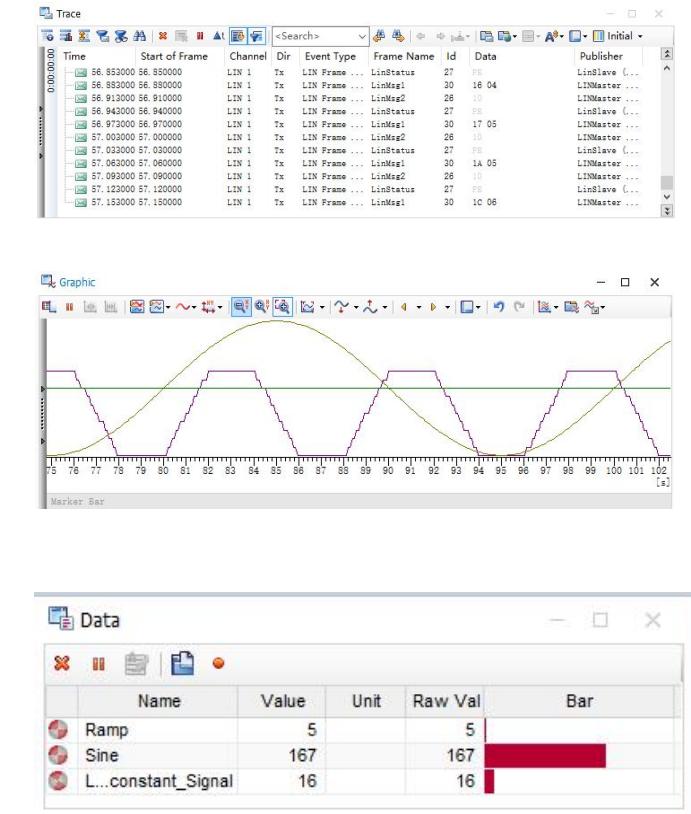
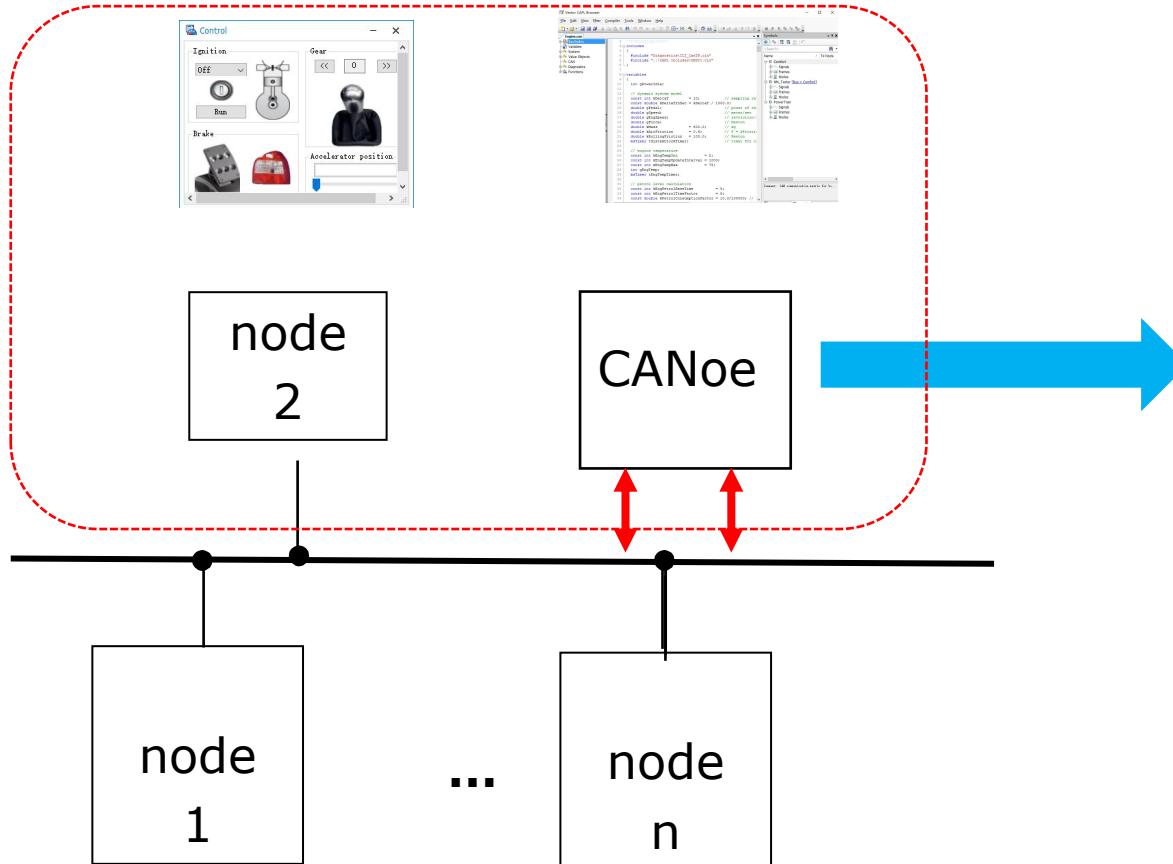


# CANoe基础应用

## 概述

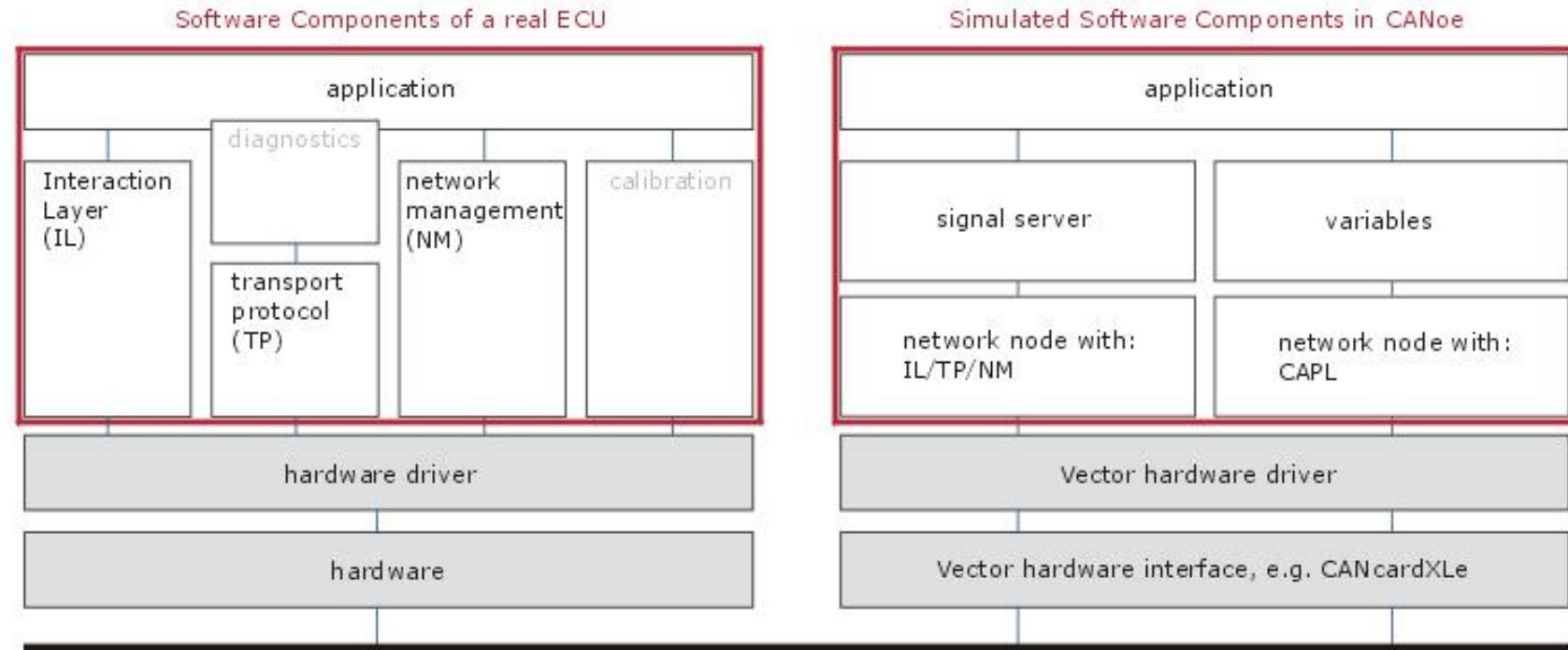


### CANoe 作为仿真工具



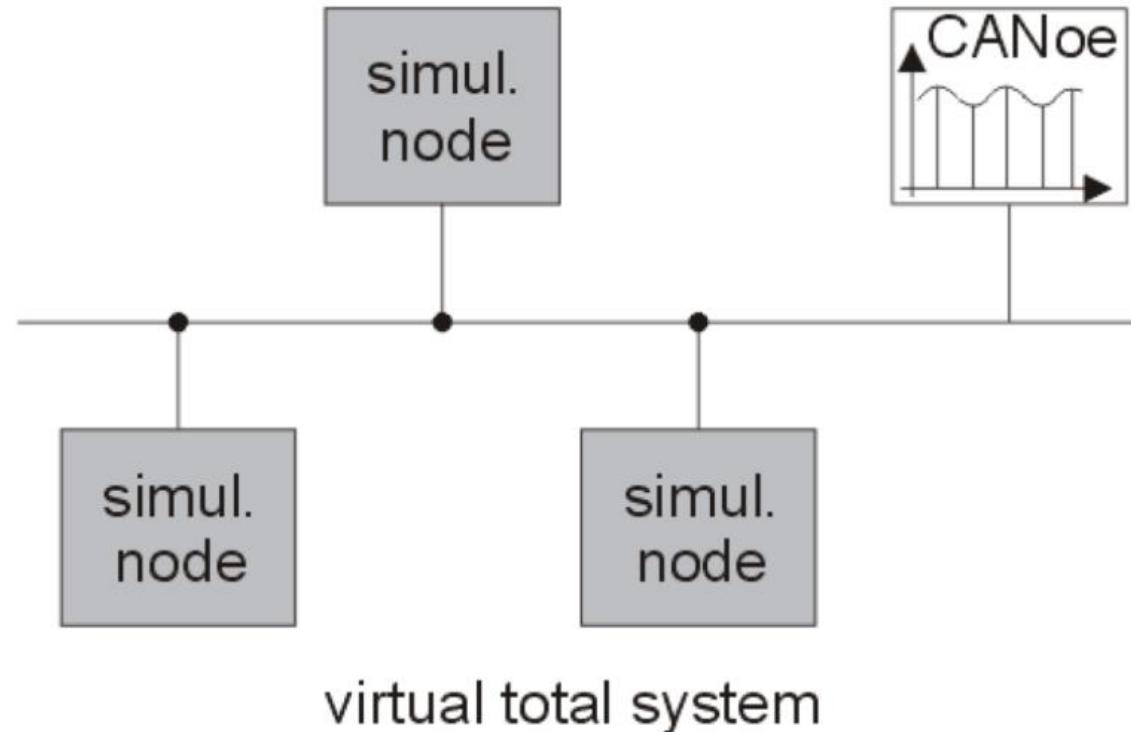
## > 仿真功能

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。



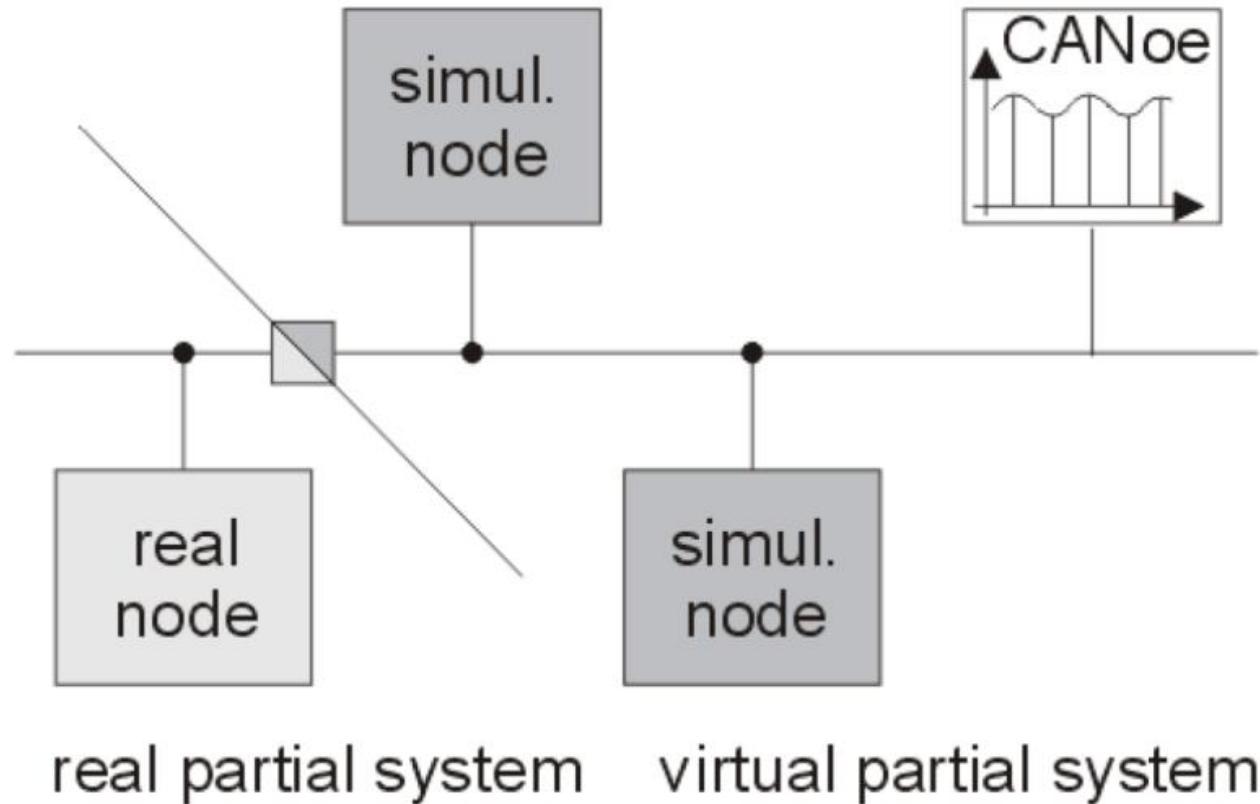
> CANoe在ECU开发中的作用

第一阶段：全仿真网络系统



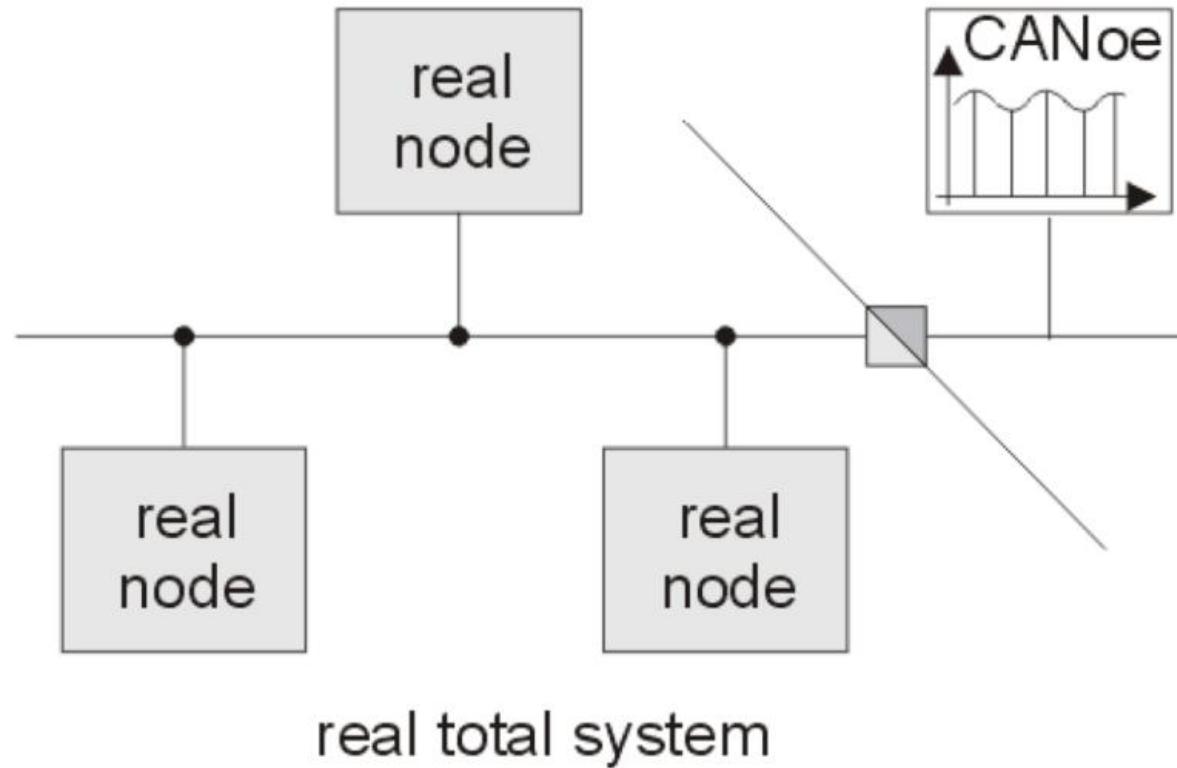
> CANoe在ECU开发中的作用

第二阶段：全真实节点和部分仿真节点共存

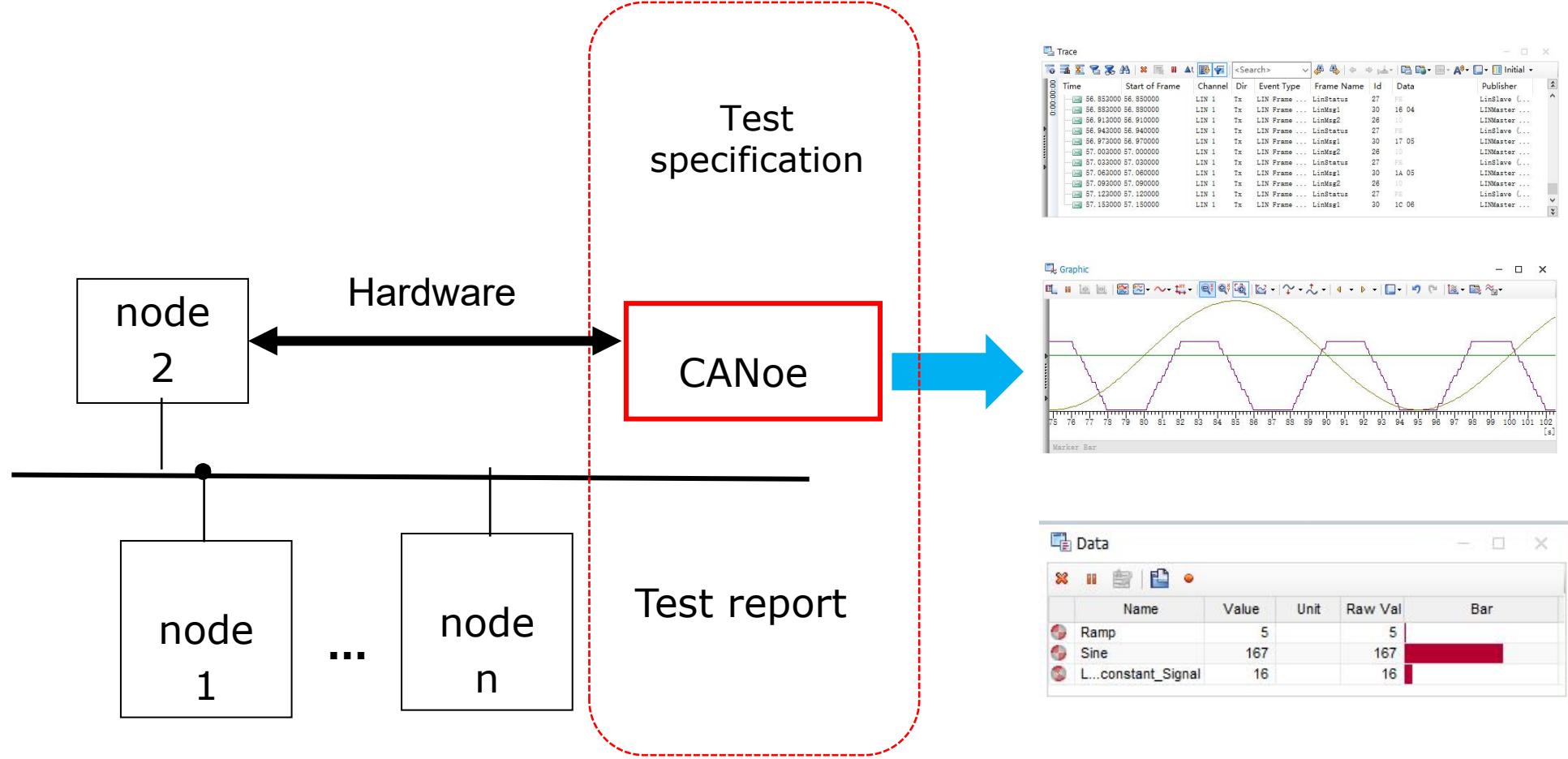


> CANoe在ECU开发中的作用

第三阶段：全真实节点的网络系统

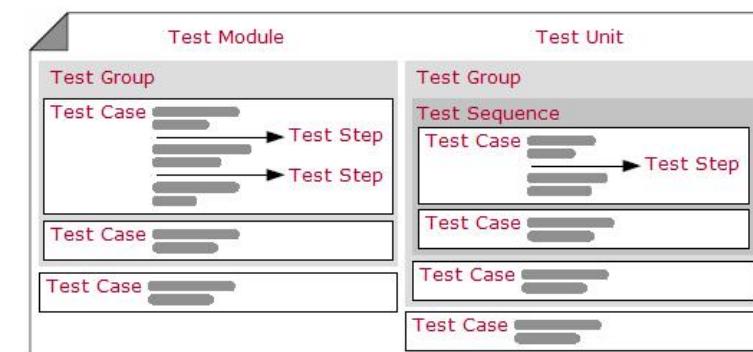
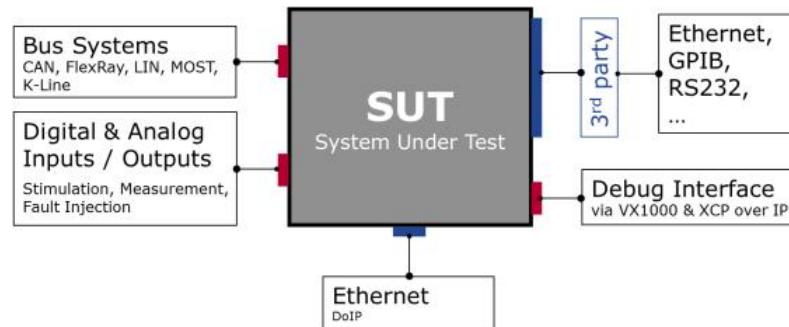
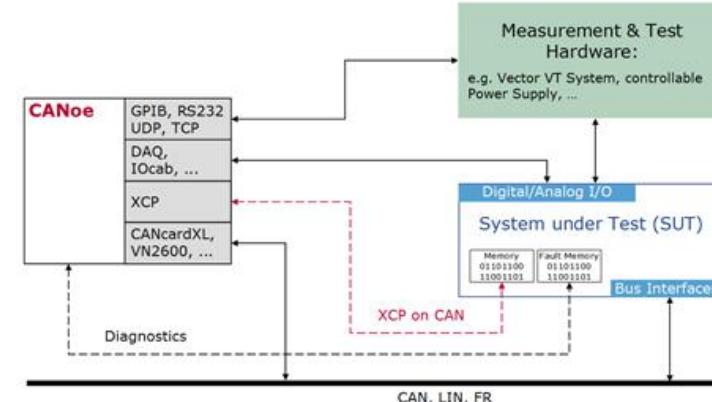
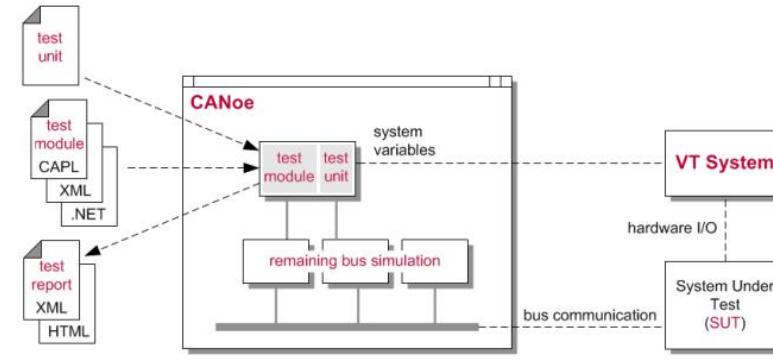


### > CANoe 作为测试工具

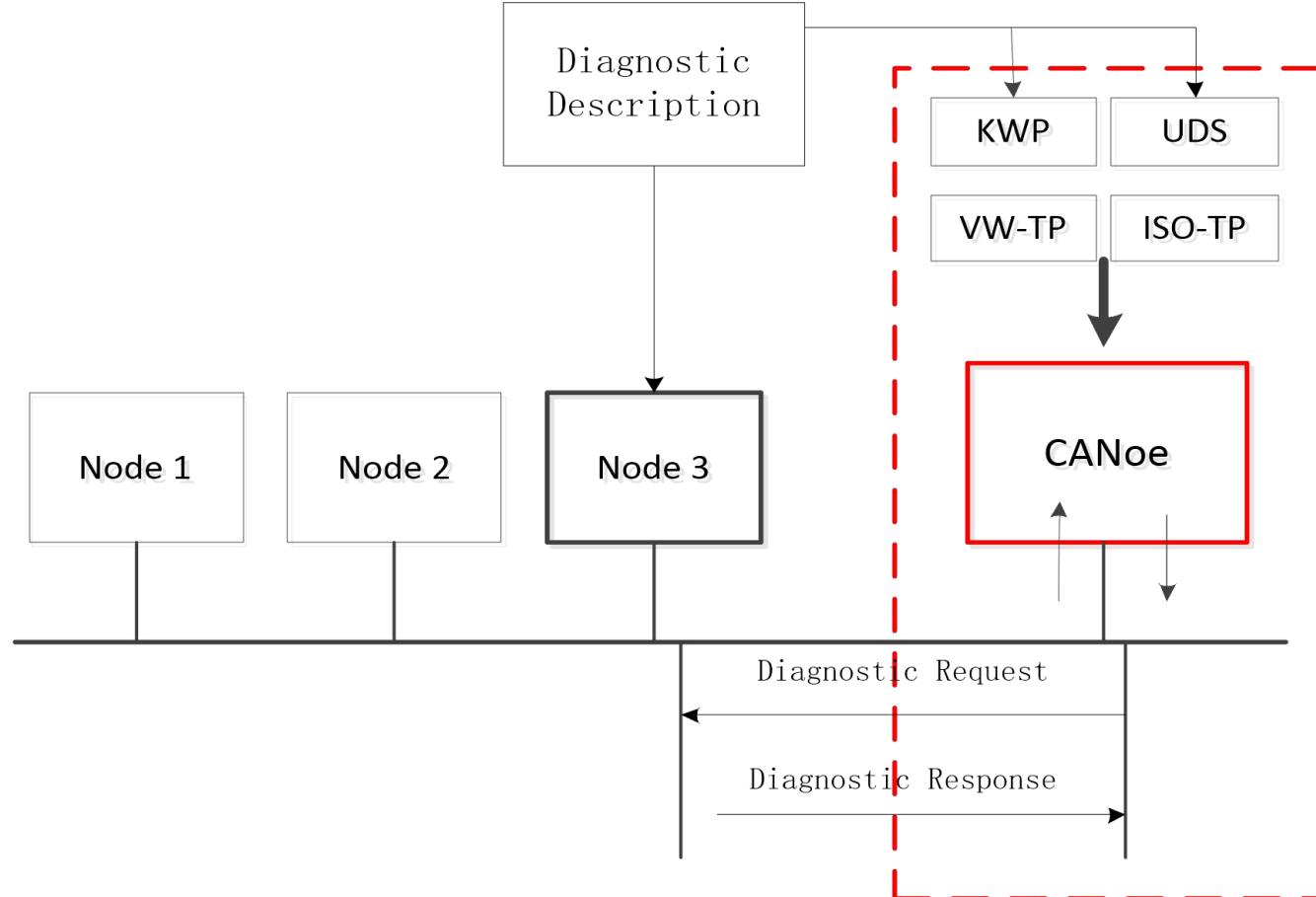


## > 测试功能

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。

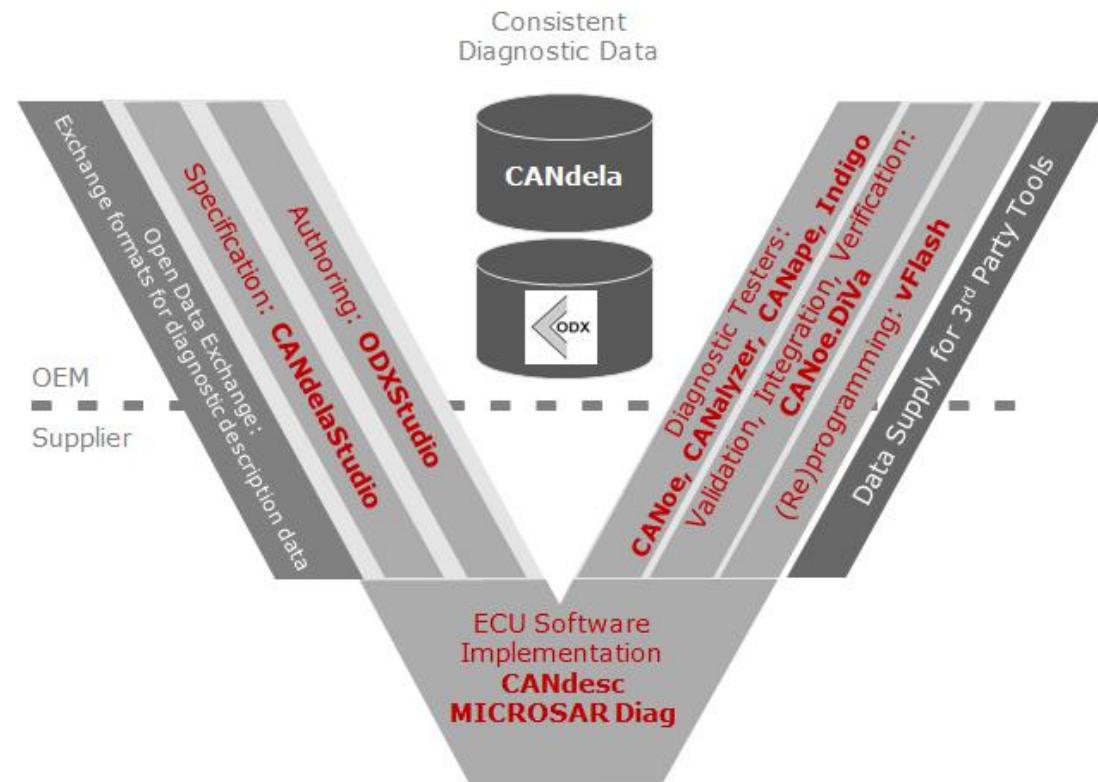


## CANoe 作为诊断工具



## > 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。



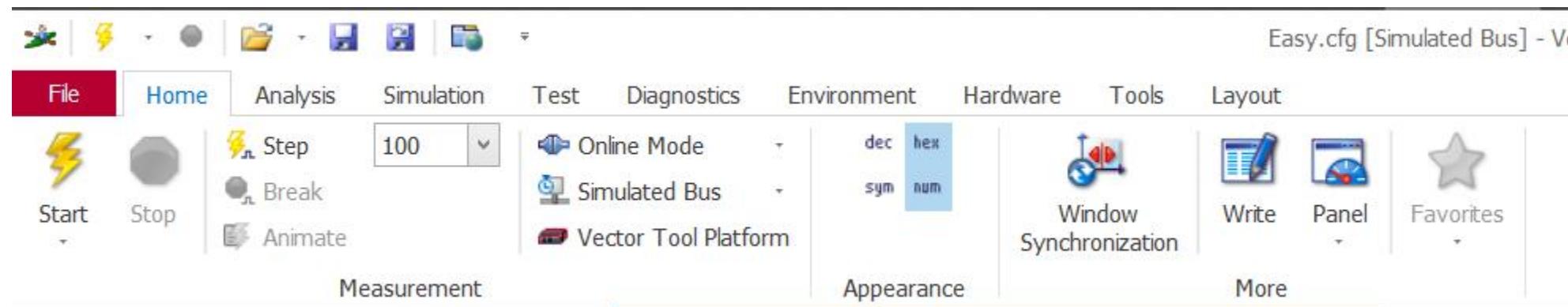
## CONTENTS



1. 概览
2. 工程创建
3. 报文发送
4. 分析窗口
5. 过滤功能模块
6. 数据记录
7. 离线分析
8. 系统变量环境变量
9. Panel设计
10. CAPL语言

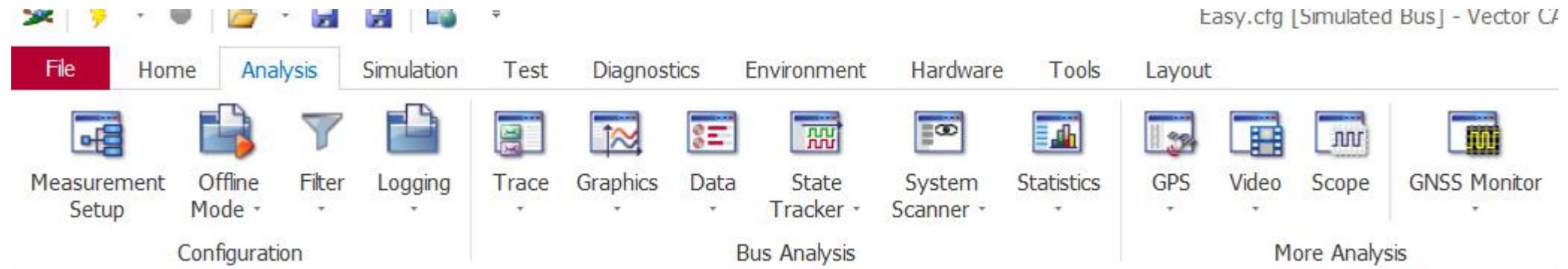
## &gt; CANoe菜单栏

Home | 测量设置，模式切换，数据显示格式等



## &gt; CANoe菜单栏

Analysis | 测试配置, 总线分析, 其他类型信号分析等



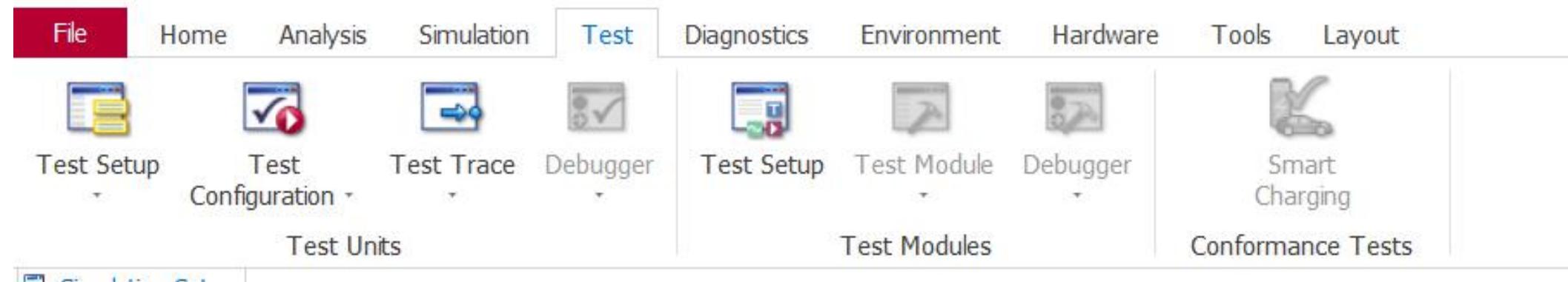
### > CANoe菜单栏

Simulation | 仿真设置，信号发生器，IG模块，激励等



## &gt; CANoe菜单栏

Test | 测试单元设置，测试模块设置等



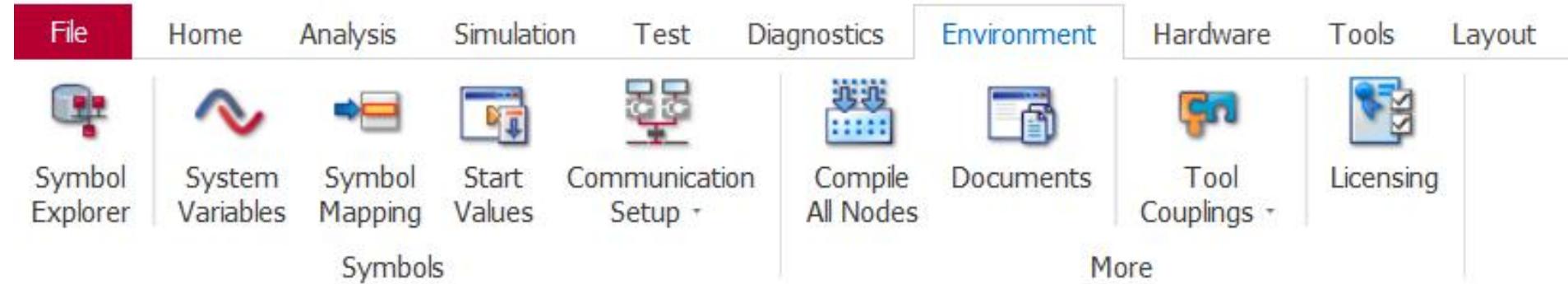
## &gt; CANoe菜单栏

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。



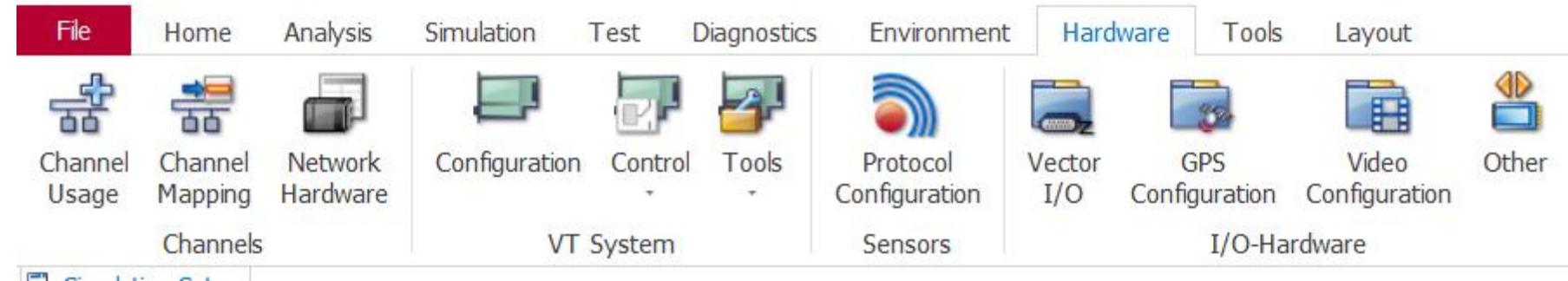
## &gt; CANoe菜单栏

Environment | 对象配置及控制。



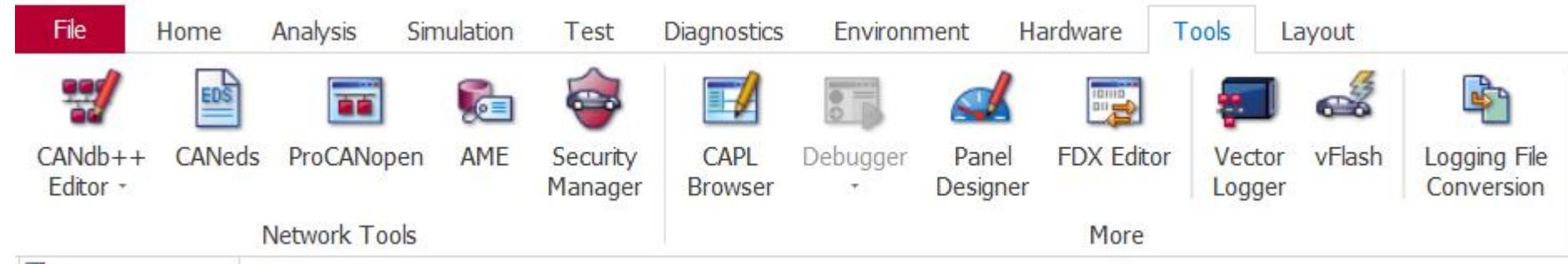
## &gt; CANoe菜单栏

Hardware | 通道设置，VT配置及控制，传感器及I/O硬件控制



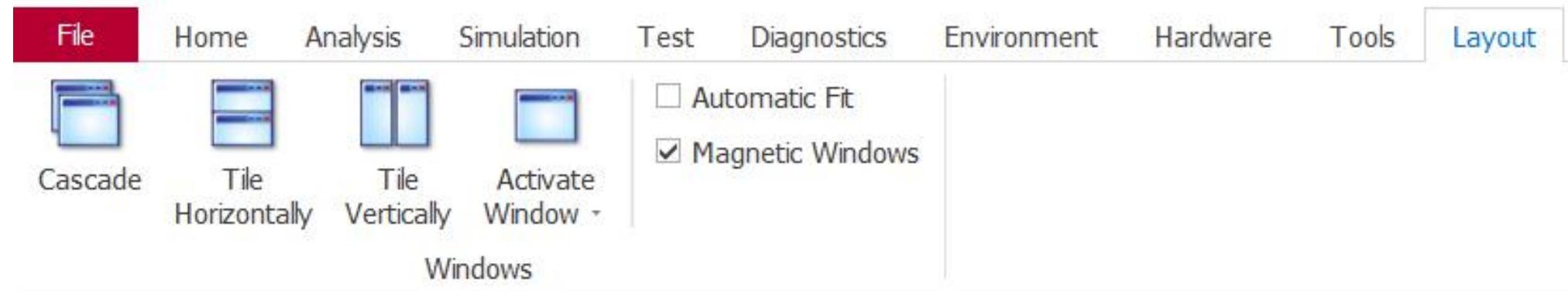
## &gt; CANoe菜单栏

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

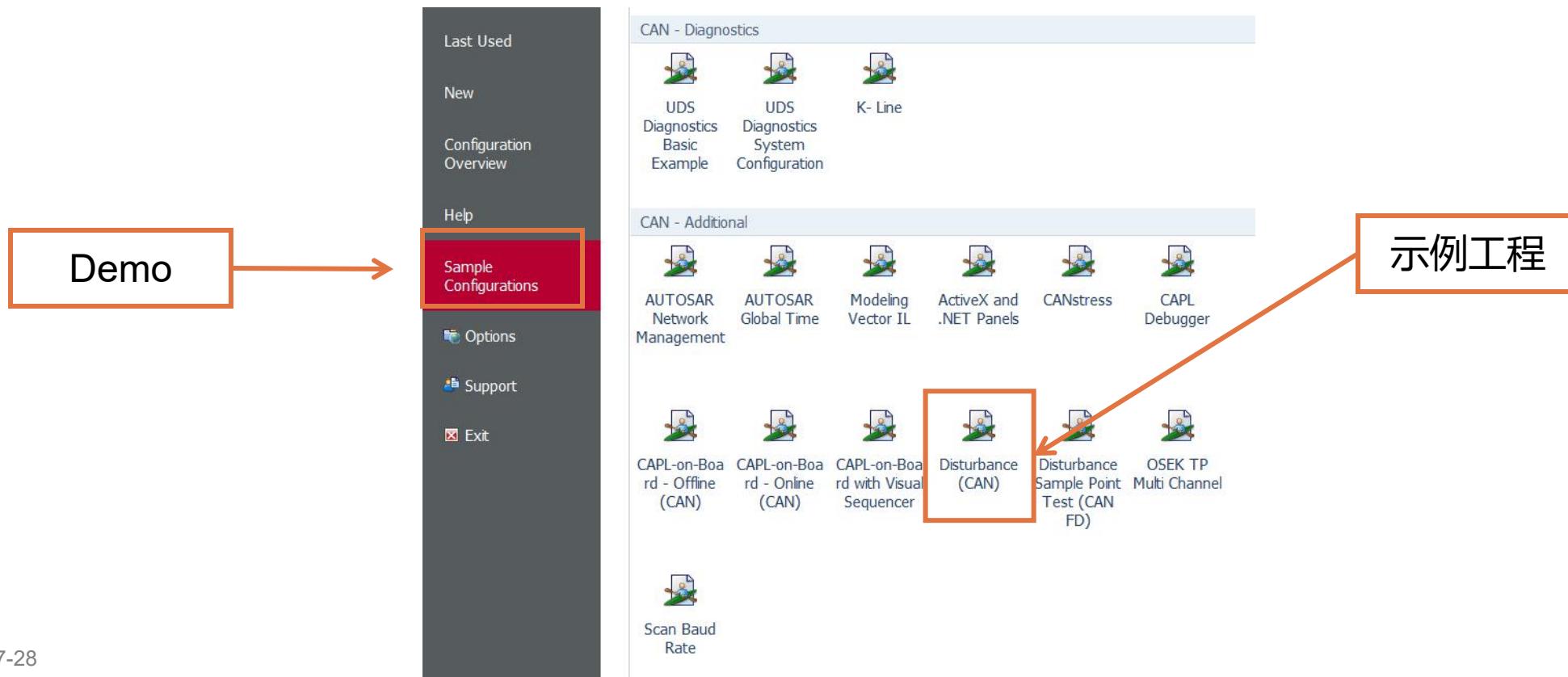


## &gt; CANoe菜单栏

Layout | CANoe界面中窗口的排列方式切换。

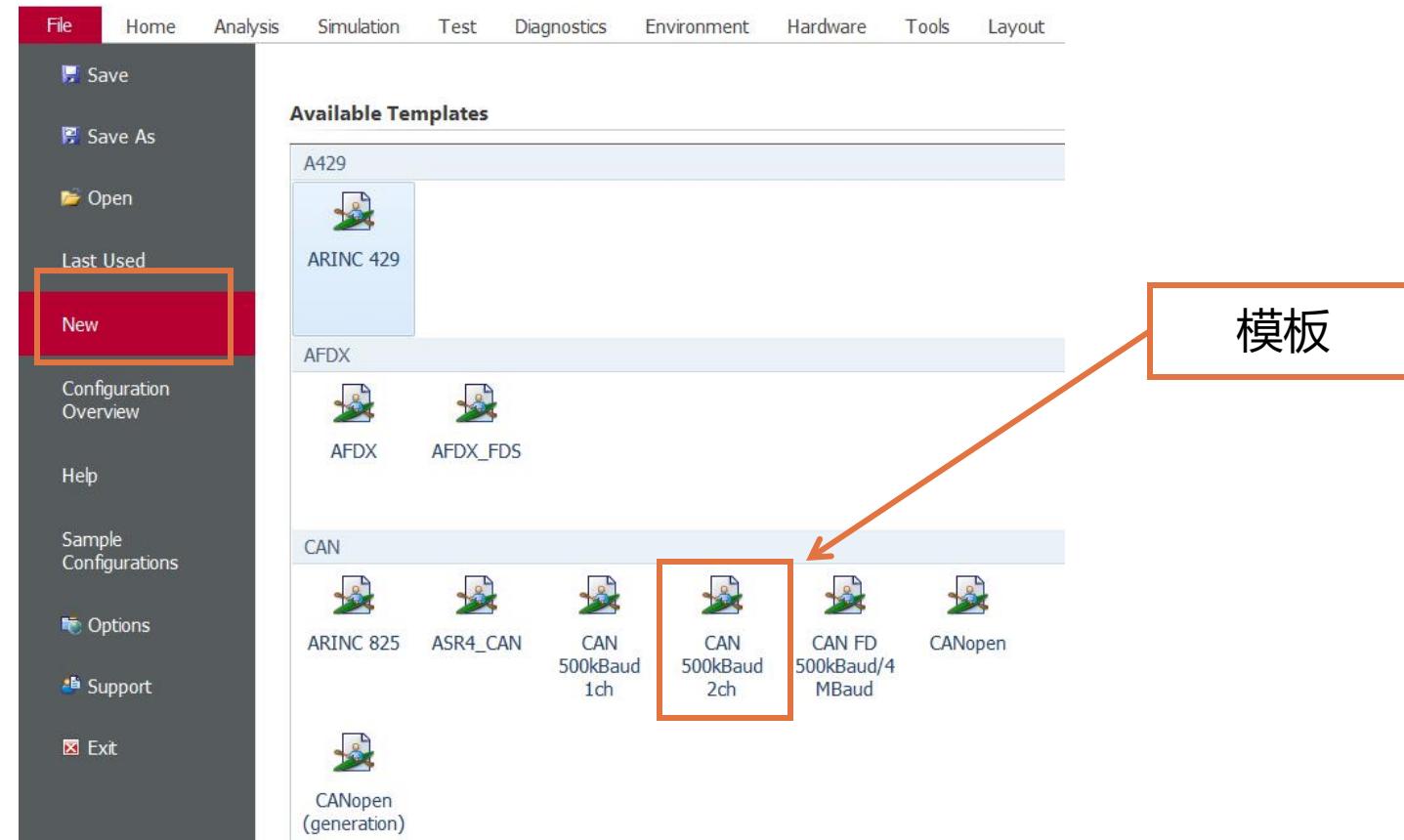


### > Sample Configurations



### > 创建新的配置

新建配置，选择对应的模板 File | New | CAN.....



### > 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

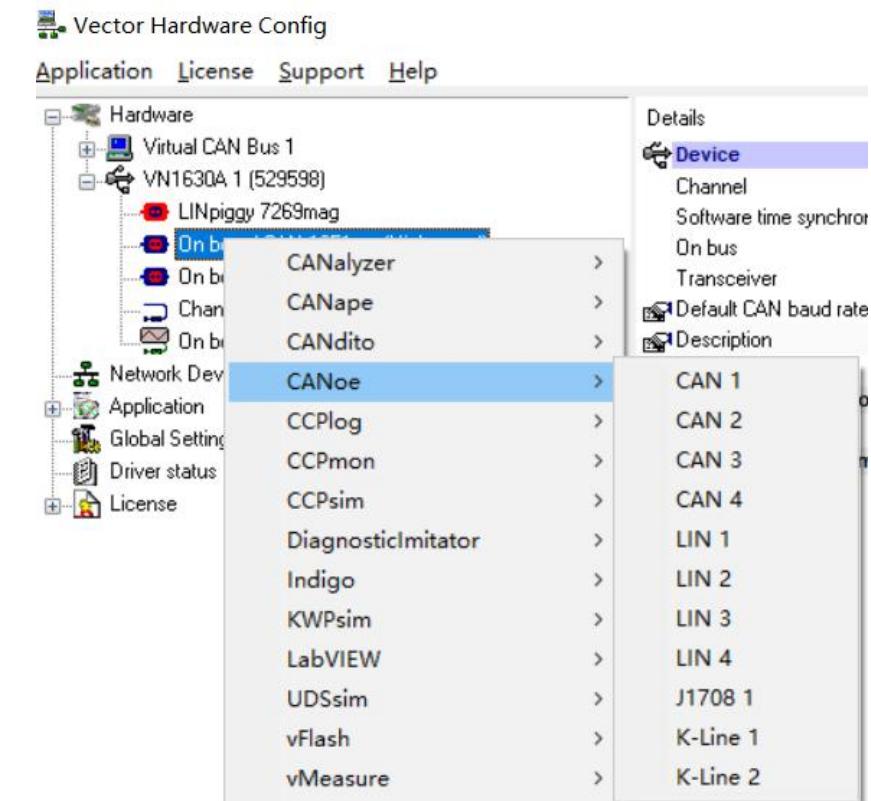
Hardware | Network Hardware..... 硬件通道

Hardware | Channel Mapping..... 通道映射

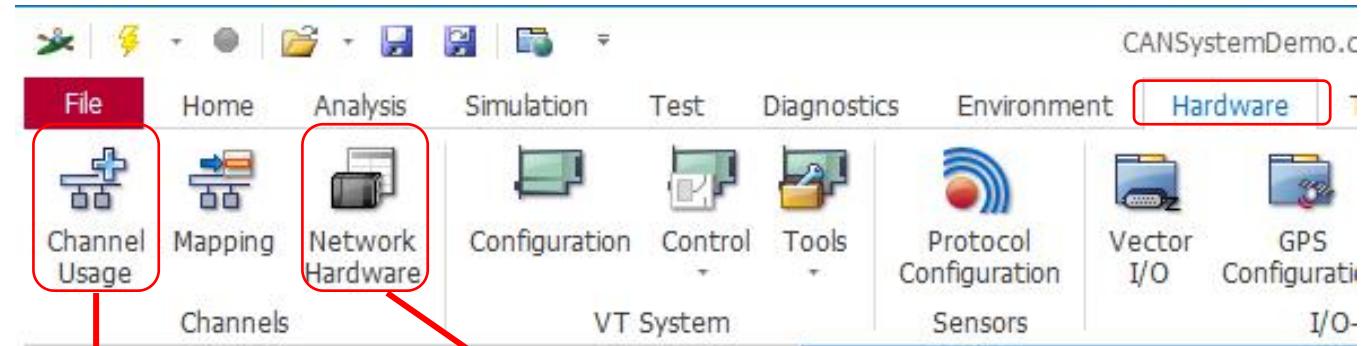
Channel Usage

CAN:	2	LIN:	0	MOST:	0	FlexRay:	0
J1708:	0	Eth:	0	AFDX:	0	A429:	0
K-Line:	0	SENT:	0				

Perform consistency check  
 Display results in Write window



> Configuration-Hardware



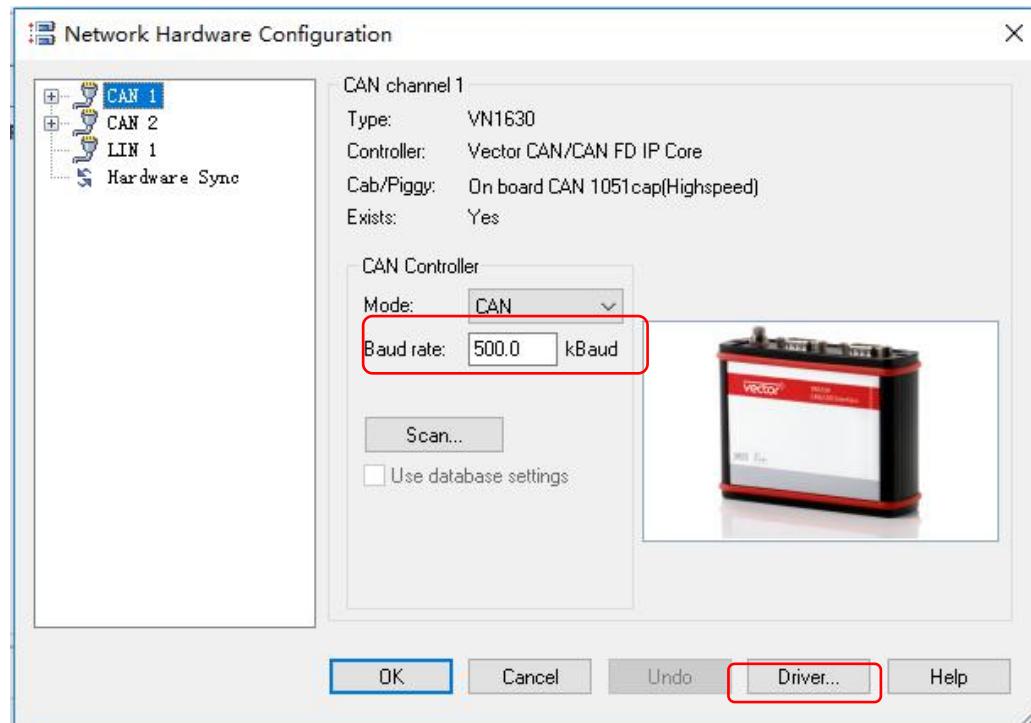
- Channel Usage
  - 设置通道数量
  - 通道全局设置
- Network Hardware
  - 波特率
  - Acknowledge
  - Acceptance Filter
  - 设备管理

## > Channel usage—设置通道数量



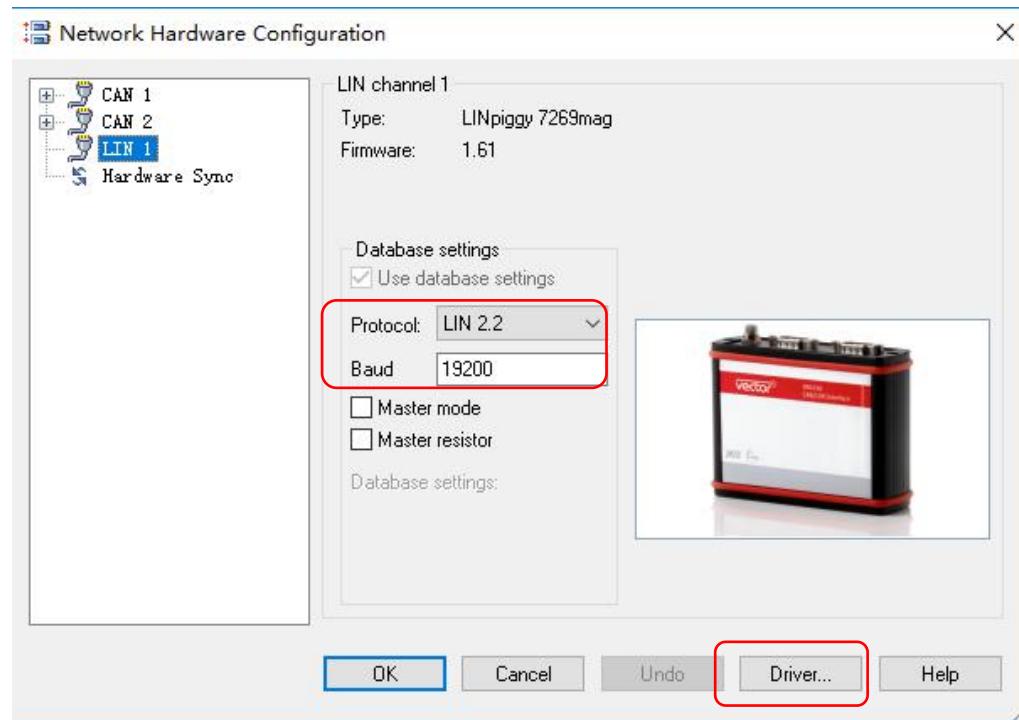
- 修改总线通道数量
- 根据总线类型选择通道数量

## &gt; 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

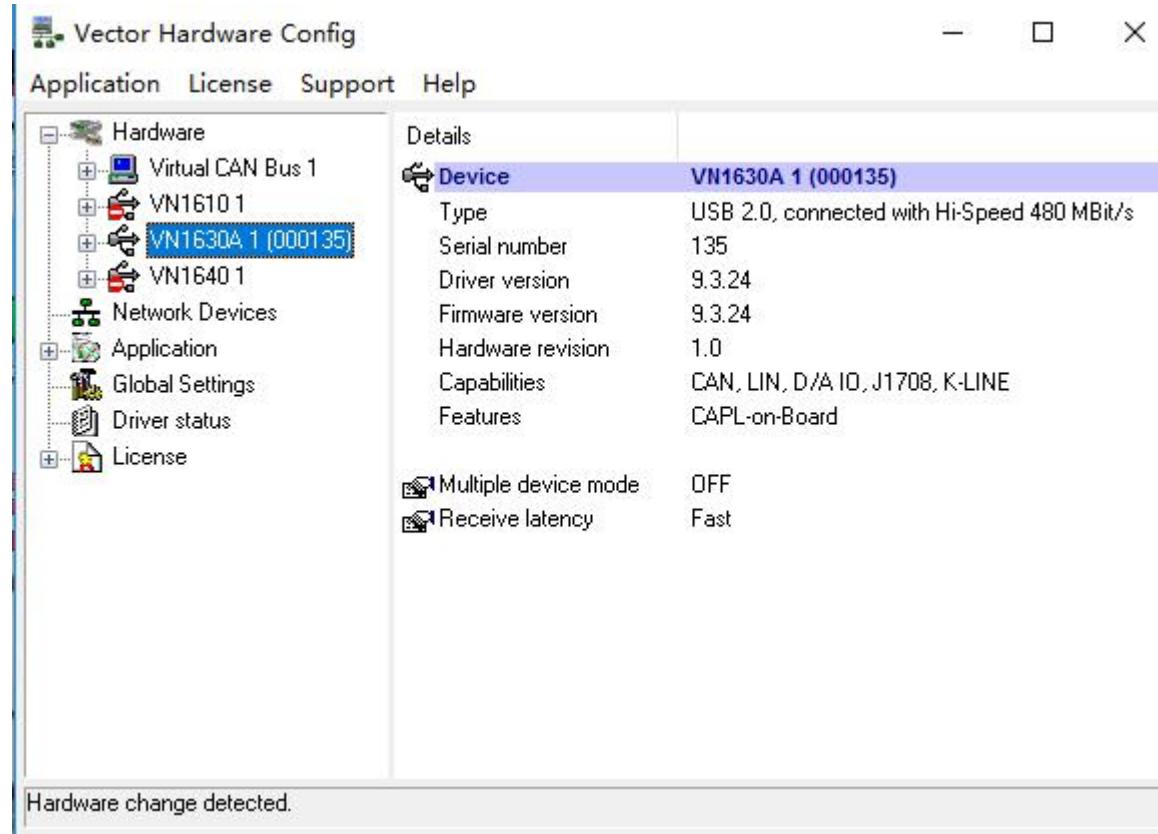
## &gt; Network Hardware—通道设置



- 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

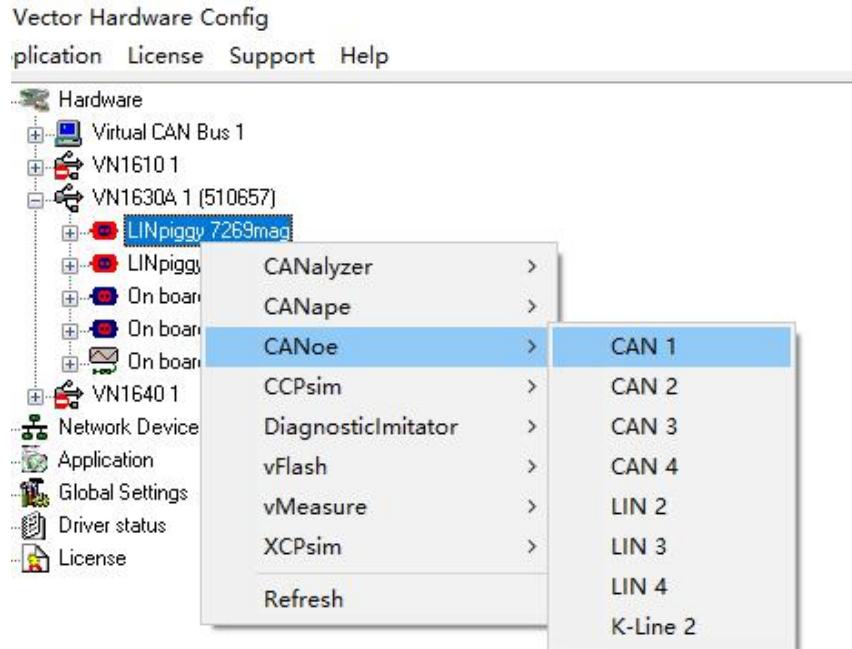
# Vector Hardware Configuration

## > Vector Hardware Configuration



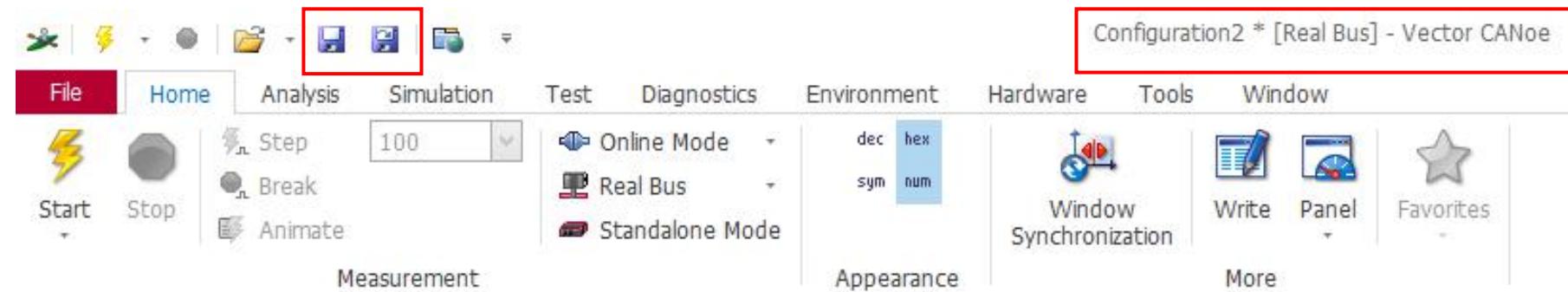
- **Hardware**
  - 硬件接口通道和收发器
  - 应用通道信息
- **Application**
  - 列出已安装工具
  - 硬件通道应用情况
- **General Information**
  - 硬件基本信息
  - 设备状态
- **License**
  - License 信息 (License旧模式)

### > 通道映射

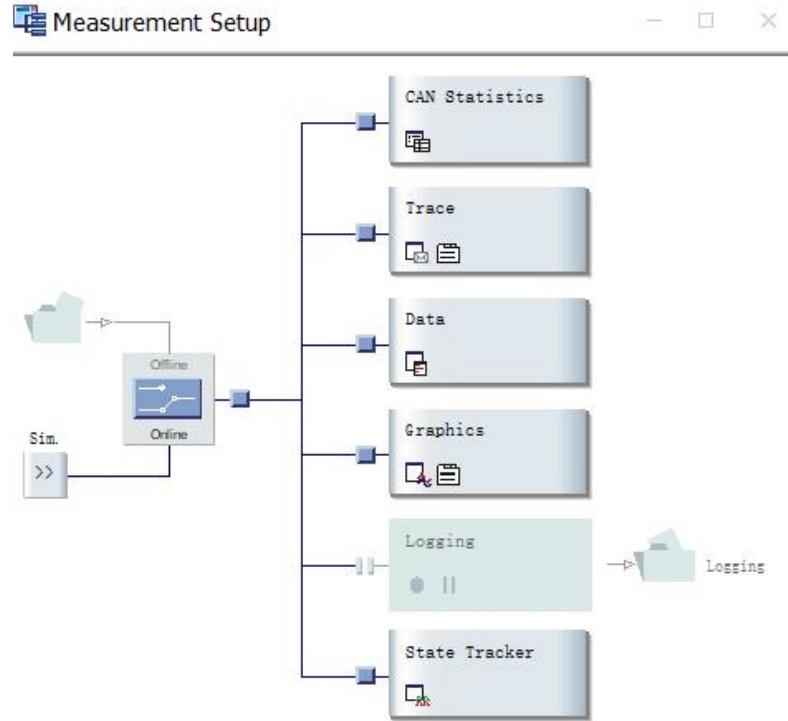


1. 选择 piggy
2. 选择应用软件
3. 选择应用软件的通道

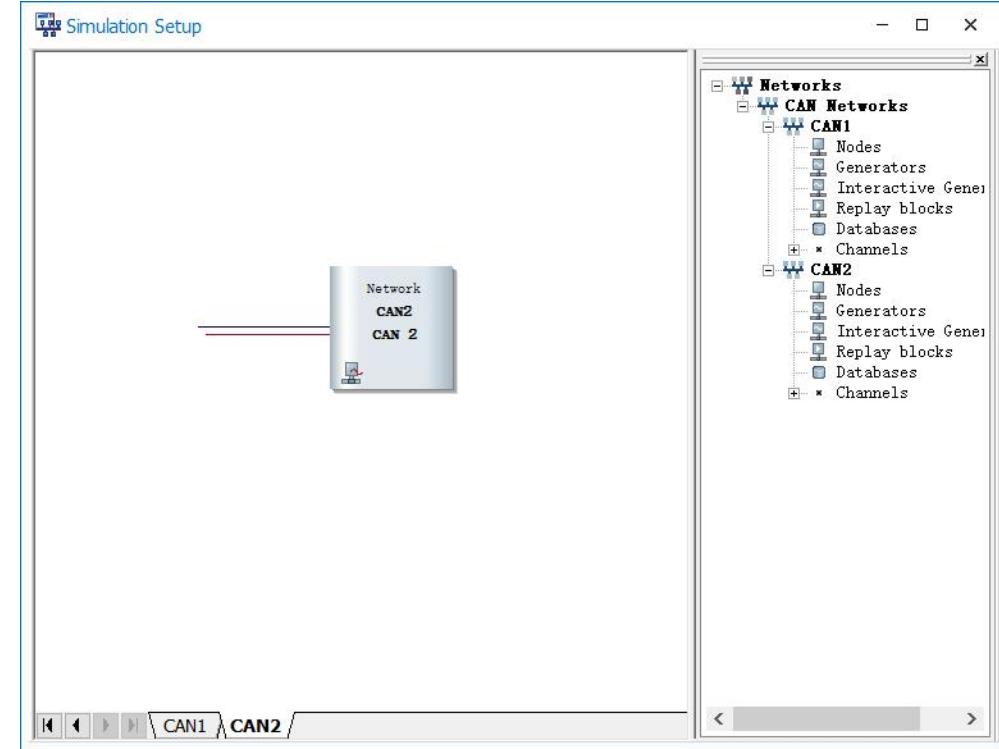
- > 保存工程配置
- > CANoe工程保存文件格式 (\*.cfg)



> 测量设置与仿真设置：



Measurement Setup

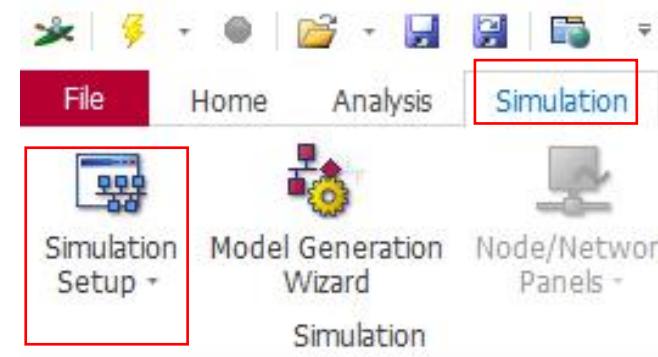
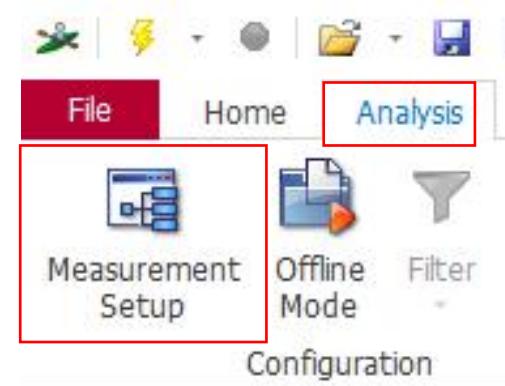


Simulation Setup



## &gt; 主窗口

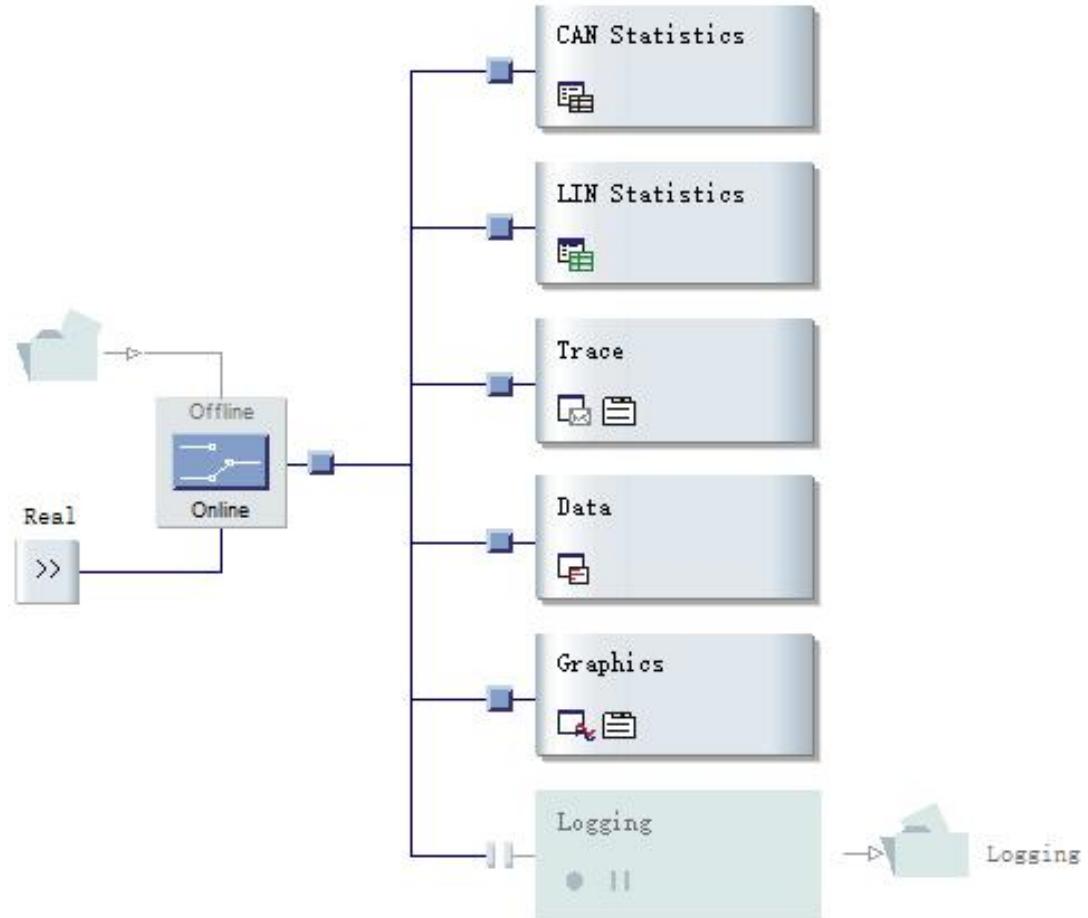
- Measurement Setup
  - 分析窗口
  - 功能块
  - 数据记录
  - Online-/Offline数据分析
- Simulation Setup
  - 残余总线仿真
  - 报文发送
  - 数据库加载
  - 配置多网络



## Measurement Setup

### > Measurement Setup

- 管理所有分析窗口和记录
- 配置数据流和处理



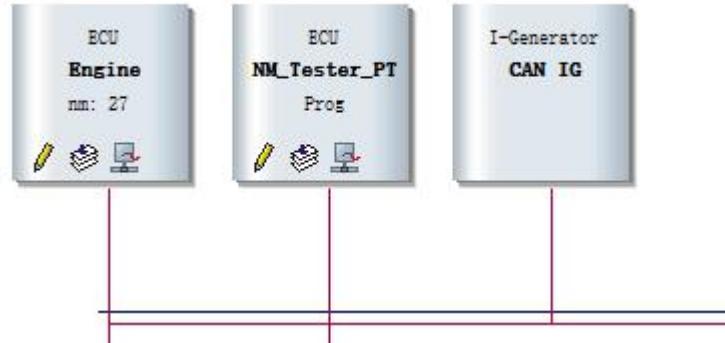
## Simulation Setup

### > Simulation Setup

根据仿真的程度，功能如下

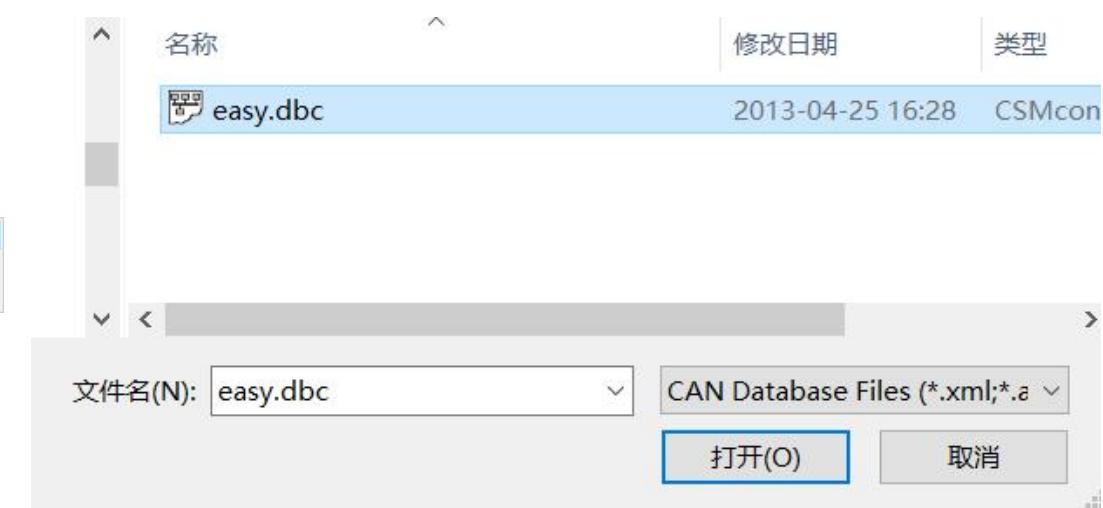
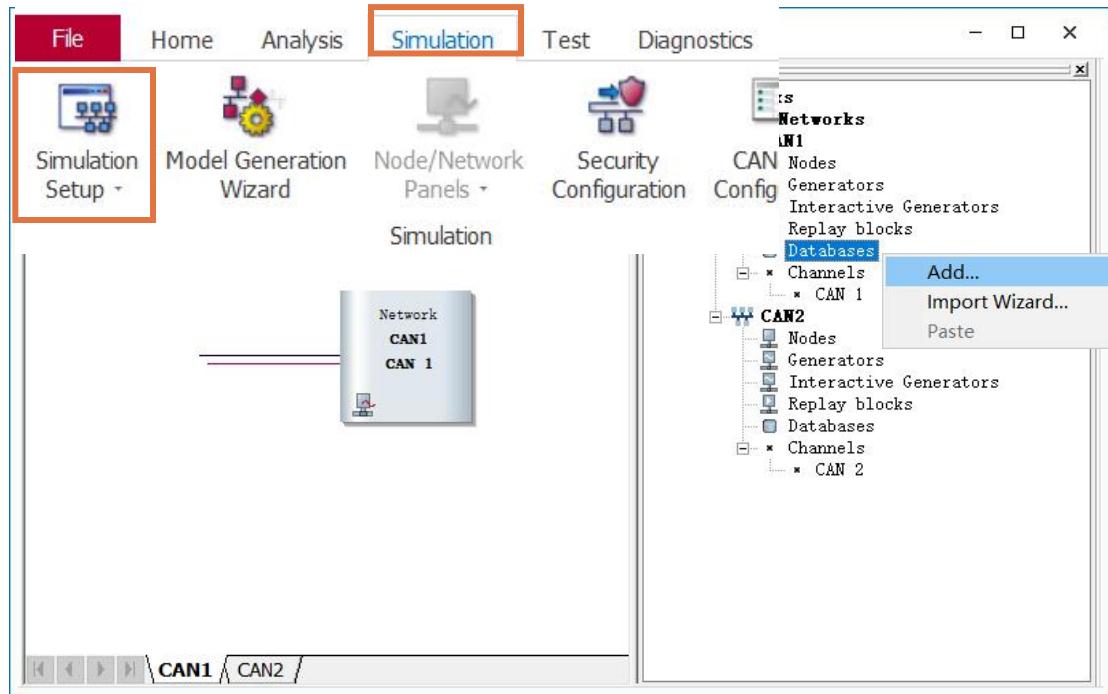
- 仿真所有节点
- CANoe 脱离真实总线操作
- 完全仿真

- 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

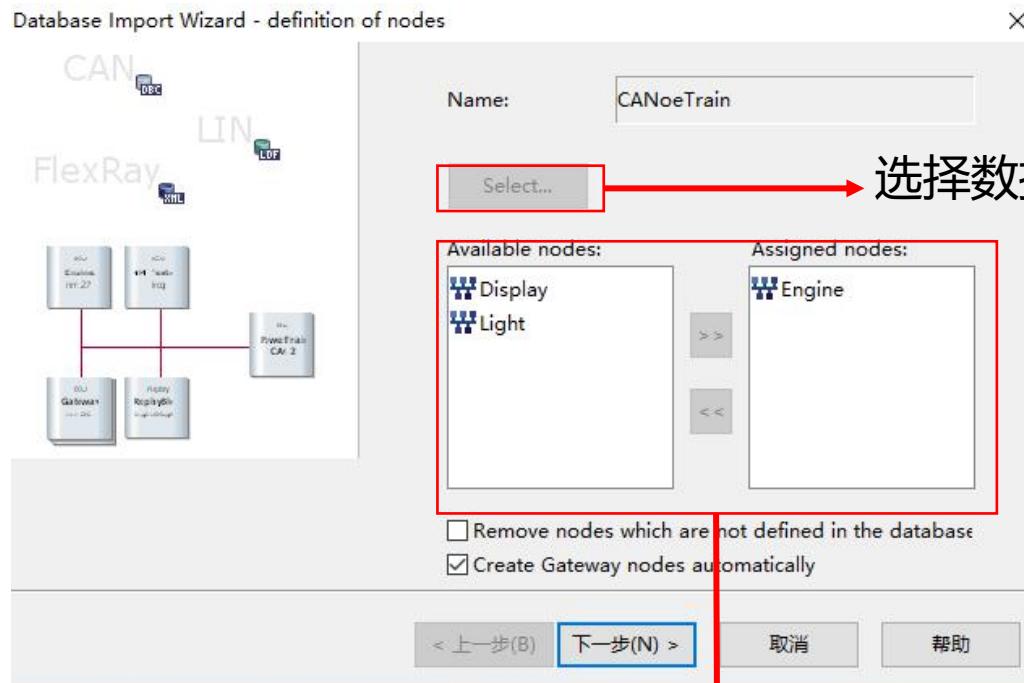
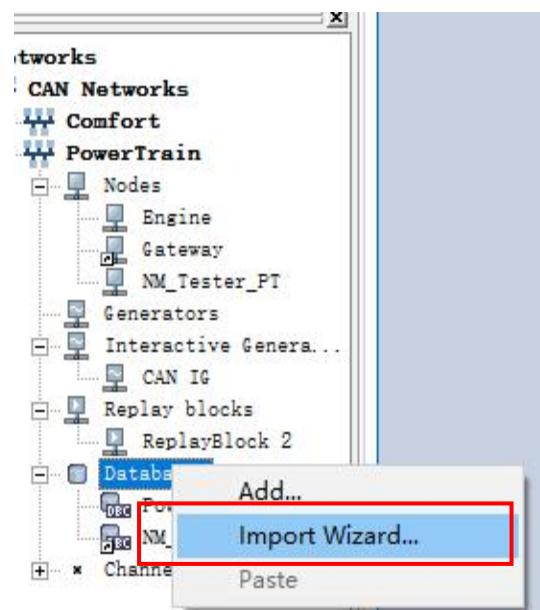


## &gt; 加载数据库设置

Simulation | Simulation Setup.....



# 加载DBC数据库

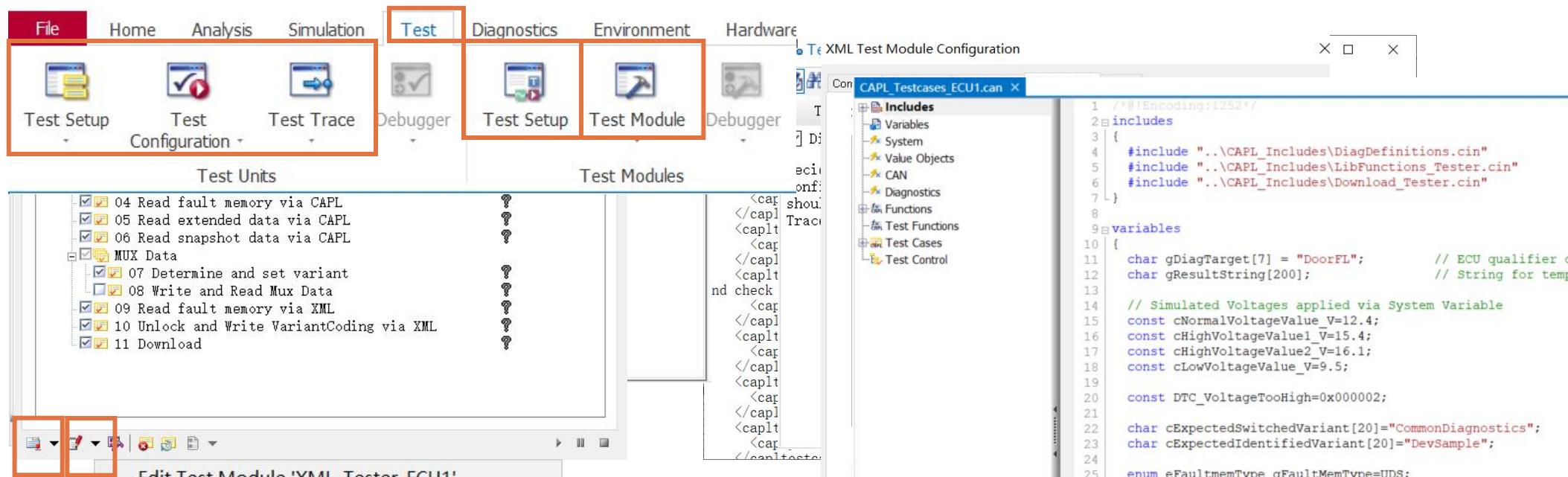


应用节点

## &gt; 测试设置

## Test | Test Setup/Test Module.....

分为Test Units和Test Module两种测试配置：



The screenshot shows the Test Setup/Test Module interface in a software application. The top navigation bar includes File, Home, Analysis, Simulation, Test (highlighted in red), Diagnostics, Environment, and Hardware. The Test tab has two main sections: Test Units and Test Modules.

- Test Units:** Contains icons for Test Setup, Test Configuration, and Test Trace. The "Test Configuration" icon is highlighted in red. A list of test cases is displayed:
  - 04 Read fault memory via CAPL (checked)
  - 05 Read extended data via CAPL (checked)
  - 06 Read snapshot data via CAPL (checked)
  - MUX Data (grouped under a collapsed icon)
    - 07 Determine and set variant (checked)
    - 08 Write and Read Mux Data (unchecked)
    - 09 Read fault memory via XML (checked)
    - 10 Unlock and Write VariantCoding via XML (checked)
    - 11 Download (checked)
- Test Modules:** Contains icons for Test Setup and Test Module. The "Test Module" icon is highlighted in red. Below these are icons for Debugger, Test Cases, and Test Control.

At the bottom, there are buttons for Edit Test Module 'XML\_Tester\_ECU1'..., Edit Library 'CAPL\_Testcases\_ECU1.can'..., and Transformed XML Specification... .

To the right, a code editor window titled "XML Test Module Configuration" displays the following XML code:

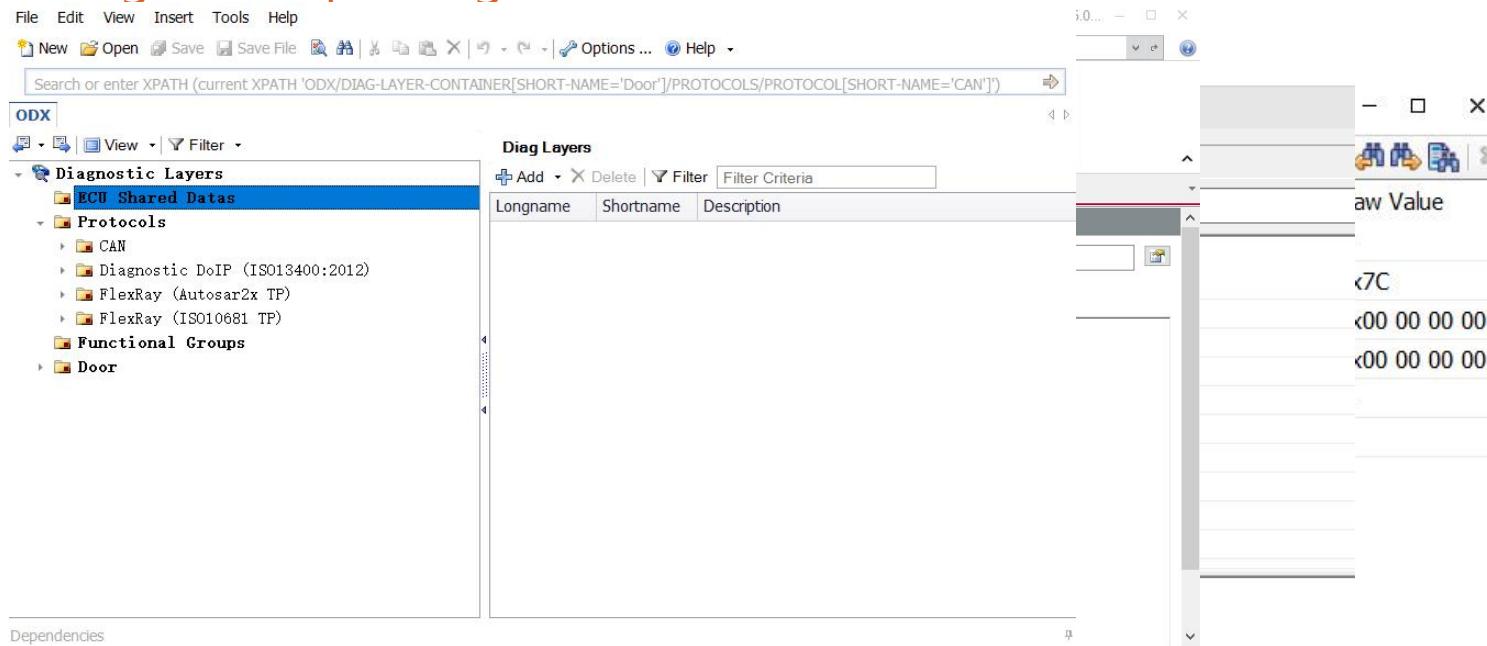
```

<?xml version="1.0" encoding="UTF-8"?>
<t>
  <!-- Encoding:1252 -->
  <includes>
    <include file="..\CAPL_Includes\DiagDefinitions.cin" />
    <include file="..\CAPL_Includes\LibFunctions_Tester.cin" />
    <include file="..\CAPL_Includes\Download_Tester.cin" />
  </includes>
  <variables>
    <variable name="gDiagTarget" value="DoorFL" type="String" />
    <variable name="gResultString" value="" type="String" />
    <variable name="cNormalVoltageValue_V" value="12.4" type="Float" />
    <variable name="cHighVoltageValue1_V" value="15.4" type="Float" />
    <variable name="cHighVoltageValue2_V" value="16.1" type="Float" />
    <variable name="cLowVoltageValue_V" value="9.5" type="Float" />
    <variable name="DTC_VoltageTooHigh" value="0x000002" type="Int" />
    <variable name="cExpectedSwitchedVariant" value="CommonDiagnostics" type="String" />
    <variable name="cExpectedIdentifiedVariant" value="DevSample" type="String" />
  </variables>
  <functions>
    <function name="TestFunction1">
      <!-- Function implementation -->
    </function>
  </functions>
  <testCases>
    <testCase name="Test Case 1">
      <!-- Test case implementation -->
    </testCase>
  </testCases>
  <testControl>
    <!-- Test control implementation -->
  </testControl>
  <!-- Other configuration sections -->
</t>

```

## &gt; 诊断设置

## Diagnostics | Configuration/Control/Tools.....



Graphviz not installed.

Status Window History Task List Search Results Checker Report Details Dependencies Action Output

Startup Page Project Explorer Diagnostic Data Communication Parameter Vehicle Info

## CONTENTS

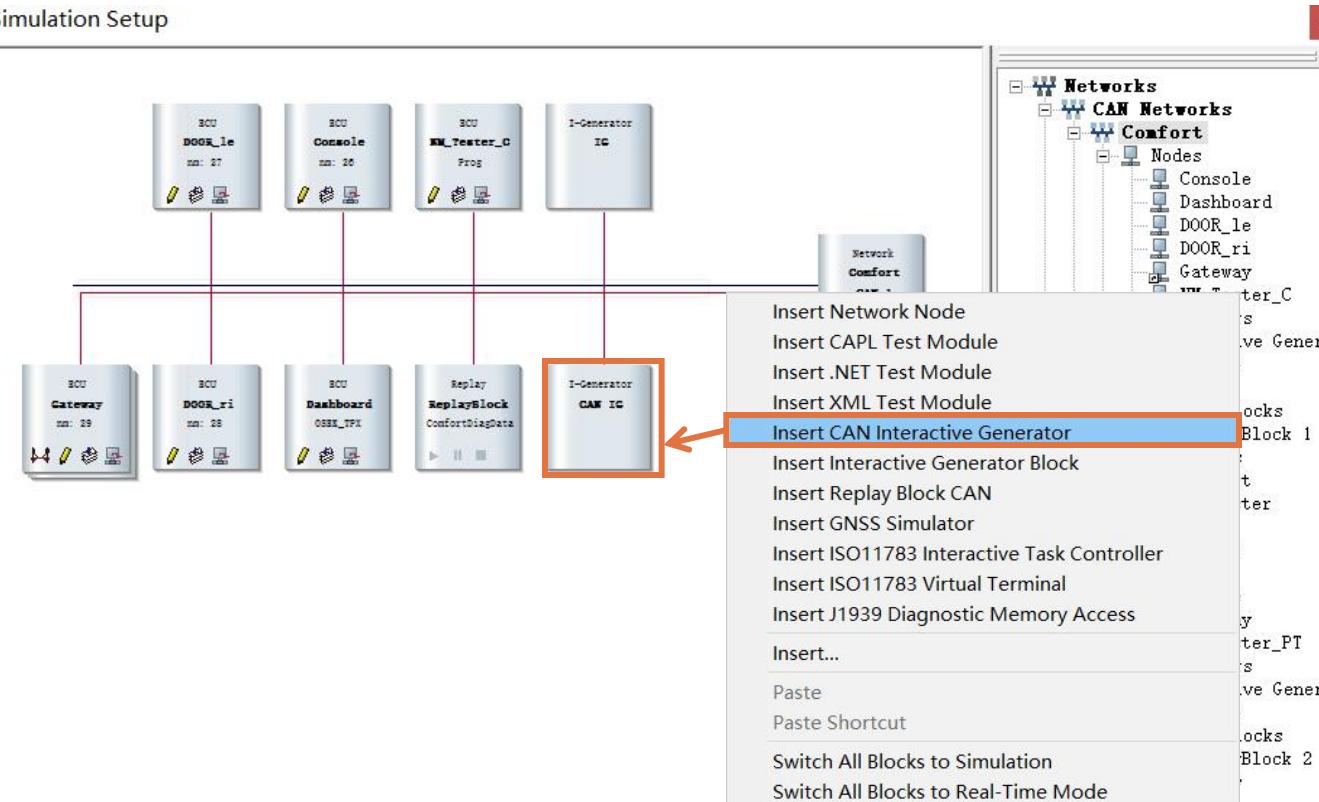


1. 概览
2. 工程创建
3. 报文发送
4. 分析窗口
5. 过滤功能模块
6. 数据记录
7. 离线分析
8. 系统变量环境变量
9. Panel设计
10. CAPL语言

### > 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）：

Simulation Setup



## &gt; Interactive Generator for CAN

CAN IG

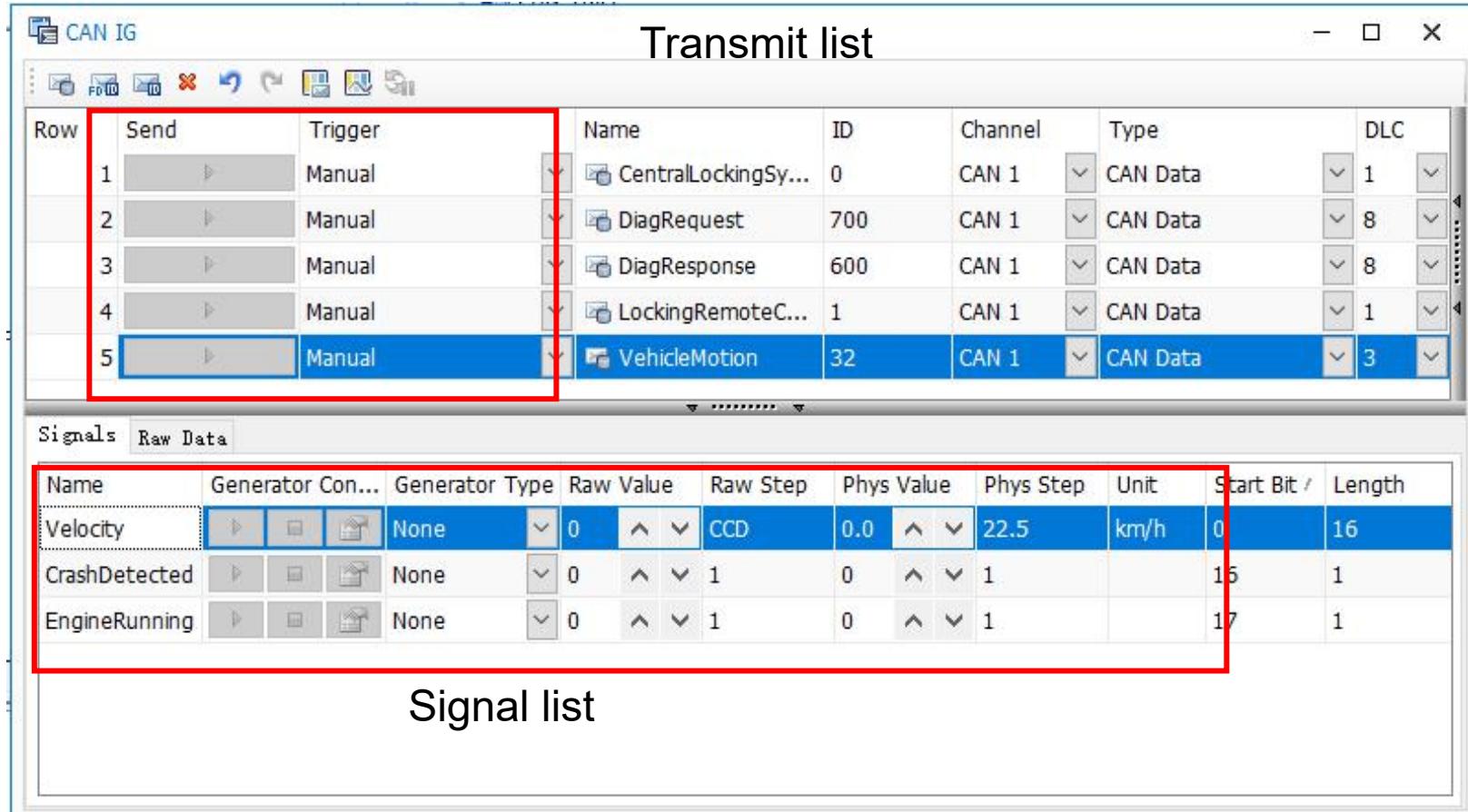
Transmit list

Row	Send	Trigger	Name	ID	Channel	Type	DLC
1	▶	Manual	CentralLockingSy...	0	CAN 1	CAN Data	1
2	▶	Manual	DiagRequest	700	CAN 1	CAN Data	8
3	▶	Manual	DiagResponse	600	CAN 1	CAN Data	8
4	▶	Manual	LockingRemoteC...	1	CAN 1	CAN Data	1
5	▶	Manual	VehicleMotion	32	CAN 1	CAN Data	3

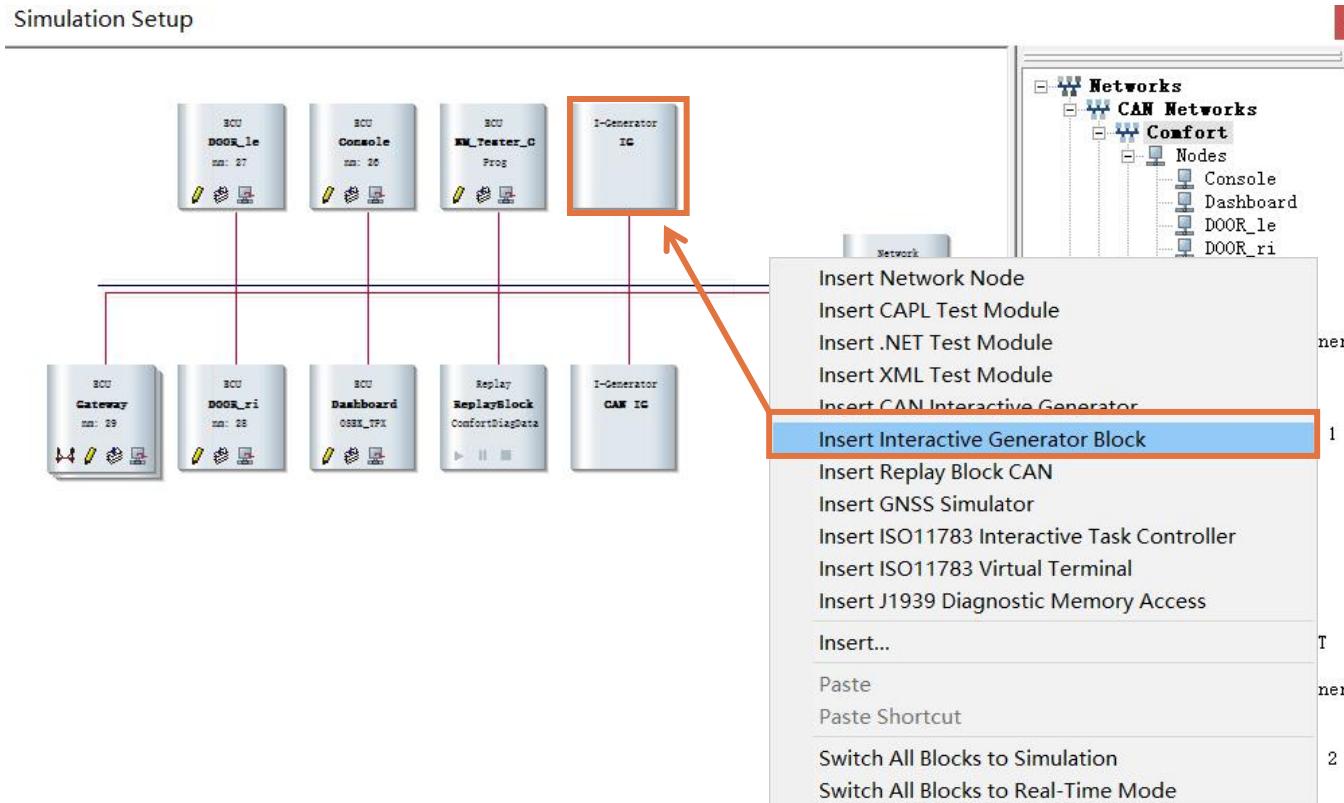
Signals Raw Data

Name	Generator Con...	Generator Type	Raw Value	Raw Step	Phys Value	Phys Step	Unit	Start Bit /	Length		
Velocity	▶	None	0	▲ ▼	CCD	0.0	▲ ▼	22.5	km/h	0	16
CrashDetected	▶	None	0	▲ ▼	1	0	▲ ▼	1		15	1
EngineRunning	▶	None	0	▲ ▼	1	0	▲ ▼	1		17	1

Signal list



## > IG模块



## &gt; IG模块

触发条件

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	<input checked="" type="checkbox"/>	update	<input type="checkbox"/> Test	<input type="checkbox"/> t	<input checked="" type="checkbox"/>	20	0	10		
LinMsg2	26	LIN 1	Response	1	<input checked="" type="checkbox"/>	update	<input type="checkbox"/> t	<input type="checkbox"/> t	<input checked="" type="checkbox"/>	20	10			
LinExtra	22	LIN 1	Header			now	<input type="checkbox"/> t	<input type="checkbox"/> t	<input checked="" type="checkbox"/>	30				
LinExtra	22	LIN 1	Response	1	<input checked="" type="checkbox"/>	update	<input type="checkbox"/> t	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	25	0			
LinStatus	27	LIN 1	Response	1	<input checked="" type="checkbox"/>	update	<input type="checkbox"/> t	<input type="checkbox"/> t	<input checked="" type="checkbox"/>	10	FE			

报文数据域

信号列表

SB	Signal Name	Raw Value	Phys Value	Unit	Dec	Phys Step	Inc	Wave form
0	Sine	0	0		-	13	+	<input checked="" type="checkbox"/> Sine
8	Ramp	10	16		-	13	+	<input checked="" type="checkbox"/> Ramps and pul

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	<input checked="" type="checkbox"/>	update	<input type="checkbox"/>	Test	<input type="checkbox"/>	t	<input checked="" type="checkbox"/>	20	0	10
LinMsg2	26	LIN 1	Response	1	<input checked="" type="checkbox"/>	update	<input type="checkbox"/>	t	<input type="checkbox"/>	20	<input checked="" type="checkbox"/>	10		
LinExtra	22	LIN 1	Header			now	<input type="checkbox"/>	t	<input type="checkbox"/>	30	<input checked="" type="checkbox"/>			
LinExtra	22	LIN 1	Response	1	<input checked="" type="checkbox"/>	update	<input type="checkbox"/>	t	<input checked="" type="checkbox"/>	25	<input checked="" type="checkbox"/>	0		
LinStatus	27	LIN 1	Response	1	<input checked="" type="checkbox"/>	update	<input type="checkbox"/>	t	<input type="checkbox"/>	10	<input checked="" type="checkbox"/>	FE		

视图方式

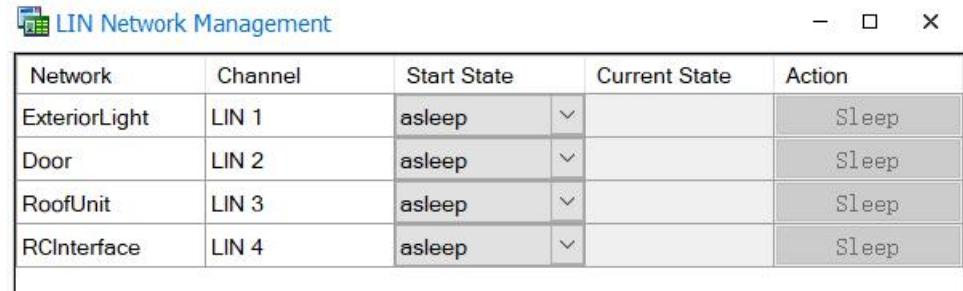
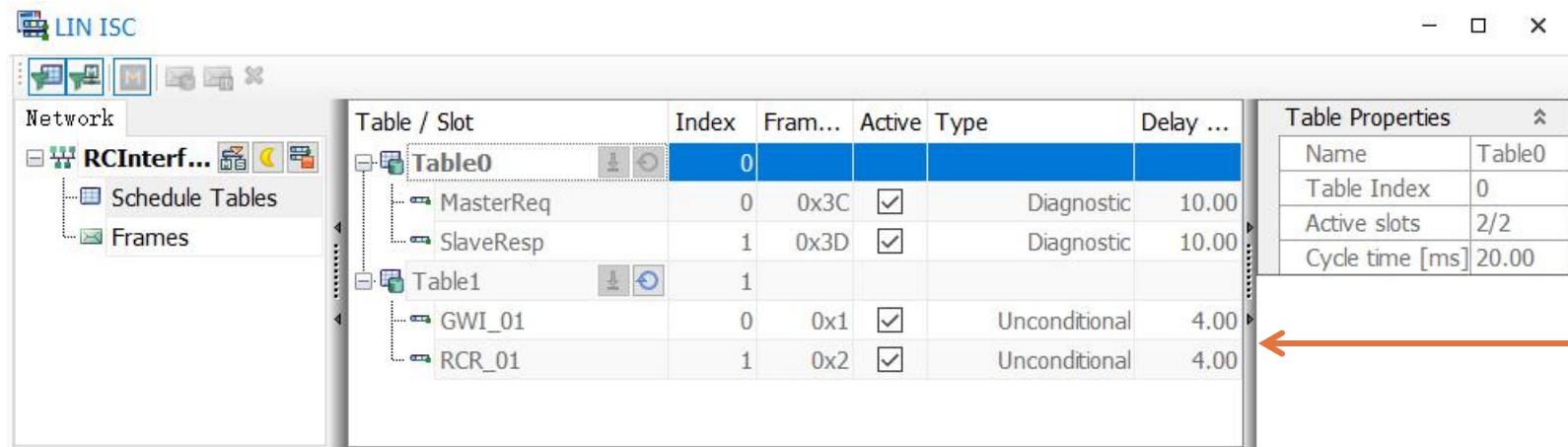
- 可配置每条报文的ID、DLC、数据
- 可配置每条报文的发送条件（手动、周期）
- 可通过数据字节/信号值配置每条报文的数据

## &gt; LIN报文管理

对于LIN总线报文，有专门的管理窗口：

Network	Channel	Start State	Current State	Action
ExteriorLight	LIN 1	asleep	▼	Sleep
Door	LIN 2	asleep	▼	Sleep
RoofUnit	LIN 3	asleep	▼	Sleep
RCInterface	LIN 4	asleep	▼	Sleep

报文休眠与唤醒

The screenshot shows the LIN ISC software interface. On the left, there's a tree view under 'Network' with 'RCInterface' expanded, showing 'Schedule Tables' and 'Frames'. The main area has two tables: 'Table / Slot' and 'Table Properties'. The 'Table / Slot' table lists frames for Table0 and Table1. Table0 contains 'MasterReq' and 'SlaveResp' frames. Table1 contains 'GWI\_01' and 'RCR\_01' frames. The 'Table Properties' table shows details for Table0: Name is Table0, Table Index is 0, Active slots is 2/2, and Cycle time [ms] is 20.00.

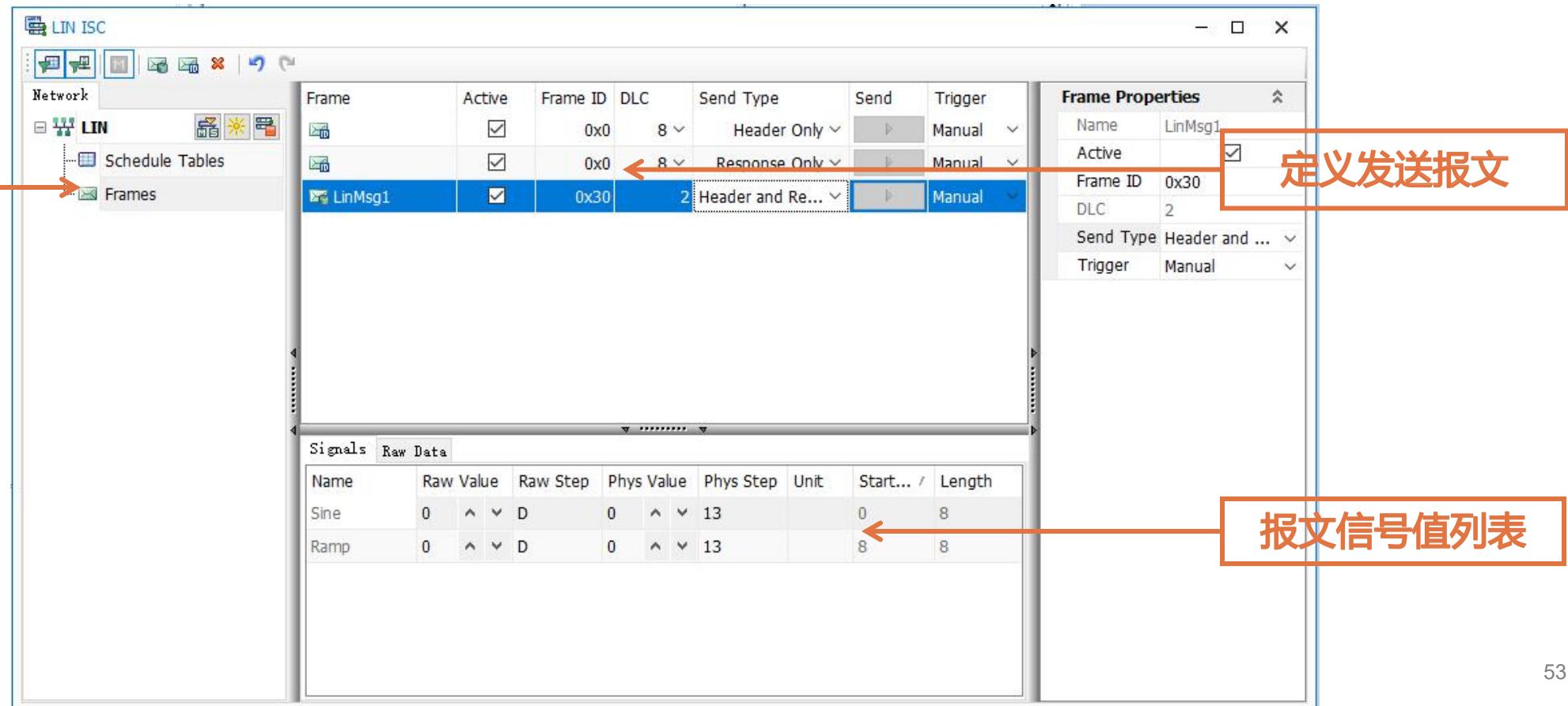
Table / Slot	Index	Fram...	Active	Type	Delay ...
Table0	0	0x3C	<input checked="" type="checkbox"/>	Diagnostic	10.00
Table0	1	0x3D	<input checked="" type="checkbox"/>	Diagnostic	10.00
Table1	1				
Table1	0	0x1	<input checked="" type="checkbox"/>	Unconditional	4.00
Table1	1	0x2	<input checked="" type="checkbox"/>	Unconditional	4.00

Name	Table0
Table Index	0
Active slots	2/2
Cycle time [ms]	20.00

调度表切换

## &gt; LIN报文管理

对于LIN总线报文，有专门的管理窗口：



## &gt; 自动时序

## Simulation | Automation |.....

Automation Sequences

Visual Sequences Macros .NET Snippets

Ac...	Name
<input checked="" type="checkbox"/>	LINDoorSequence
<input checked="" type="checkbox"/>	LINExteriorLightSequence

Visual Sequence

Command	Object	Op
Send LIN Frame	WWS_01	
Break		
Send LIN Frame	DLFLeft_01	
Control Replay Block		stop

Options

Execution: Standard

Signal Layer (Interaction Layer):

Use signal layer (interaction layer) for sending messages

Send messages manually (commands: e.g. Send CAN Frame, Send LIN Frame)

Command configuration

Wait For Key: Waits until the following key is pressed:

Check:  Output negative results only

Command logging

Log to write

Log to file: LINDoorSequence.csv

Column separator: ,

Decimal symbol: . Decimal places: 6

Log with time stamp

Name display: Level 1

OK Cancel Help

Recently Used

- Control Replay Block
- Wait
- Send LIN Frame
- Break

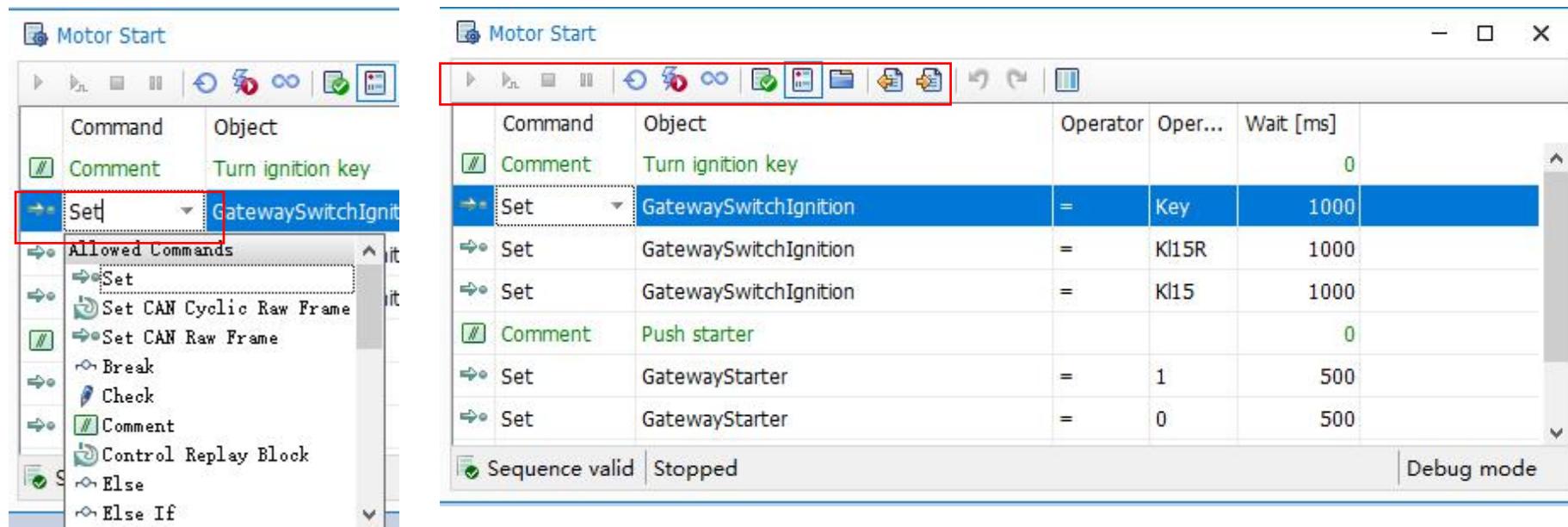
Allowed Commands

- Break
- Check
- Comment
- Control Replay Block

Sequence valid | Stopped

## &gt;自动发送序列

## 选择和修改序列命令



The screenshot shows two windows of a sequence configuration tool for a 'Motor Start' application.

**Left Window (Command Selection):**

- Shows a list of commands under 'Command' and 'Object' columns.
- A specific row is selected: 'Comment' (Turn ignition key) and 'Set' (GatewaySwitchIgnition).
- A red box highlights the 'Set' dropdown menu, which is open to show 'GatewaySwitchIgnition' as the current selection.
- The status bar at the bottom indicates 'Sequence valid' and 'Stopped'.

**Right Window (Sequence View):**

- Shows a detailed view of the sequence steps:
- Step 1: Comment (Turn ignition key), Set (GatewaySwitchIgnition) = Key, Wait [ms] 0.
- Step 2: Set (GatewaySwitchIgnition) = Kl15R, Wait [ms] 1000.
- Step 3: Set (GatewaySwitchIgnition) = Kl15, Wait [ms] 1000.
- Step 4: Comment (Push starter), Set (GatewayStarter) = 1, Wait [ms] 0.
- Step 5: Set (GatewayStarter) = 0, Wait [ms] 500.

- The status bar at the bottom indicates 'Sequence valid' and 'Stopped'.

## &gt;自动发送序列

选择和修改序列命令，使用序列发送报文

Visual Sequence

Command	Object	Operator	Operand	Wait [...]
Repeat	5	times		0
Wait	1000	ms		0
Set CAN Raw Frame	CAN1::0x99.dlc	=	8	0
Set CAN Cyclic Raw Frame	CAN1::0x99	cycle time (ms)	100	0
Send CAN Raw Frame	CAN1::0x99	=	02 00 05 05 03 00 02 02	0
Repeat End				0

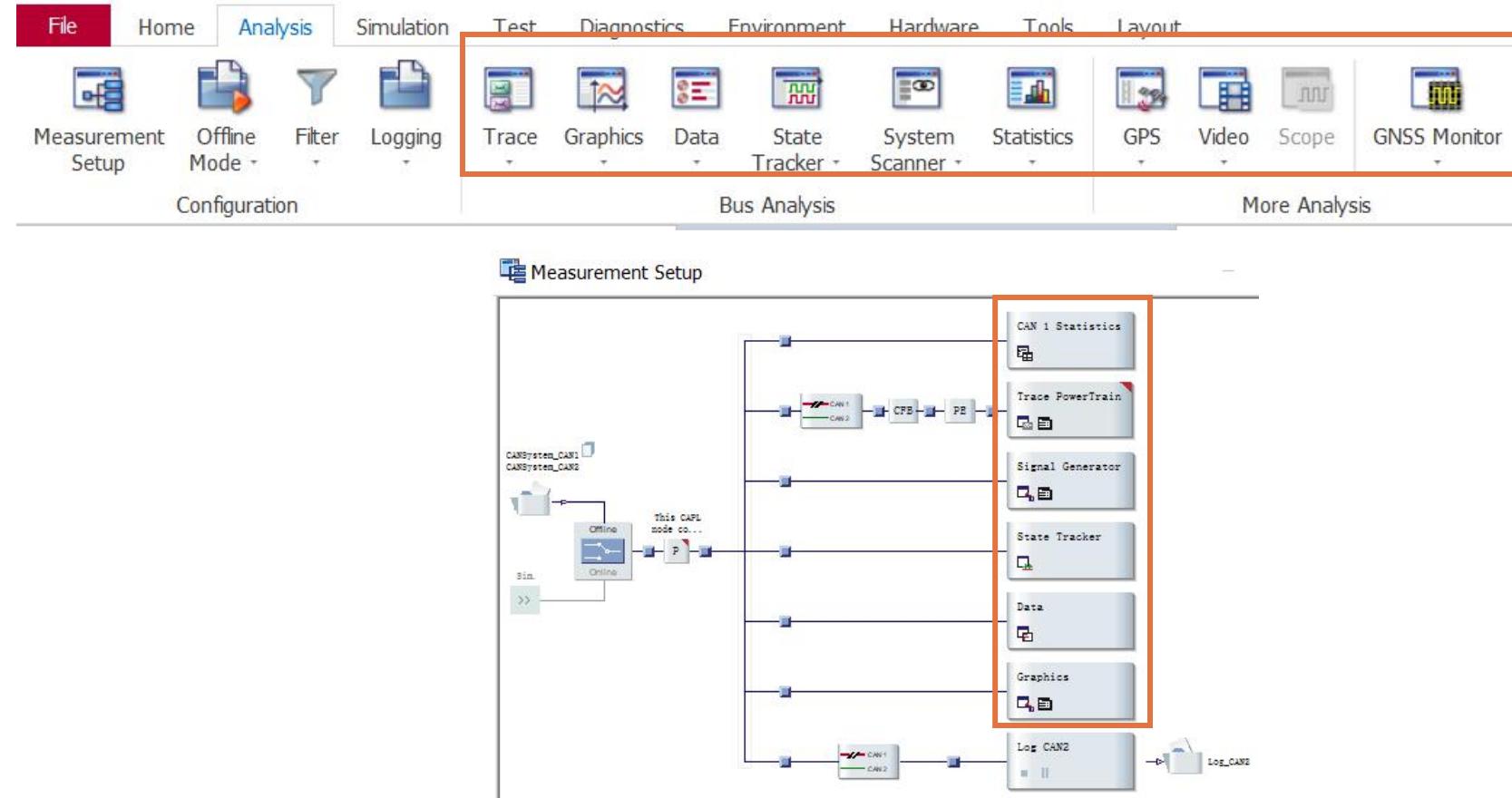
Sequence valid | Stopped

## CONTENTS

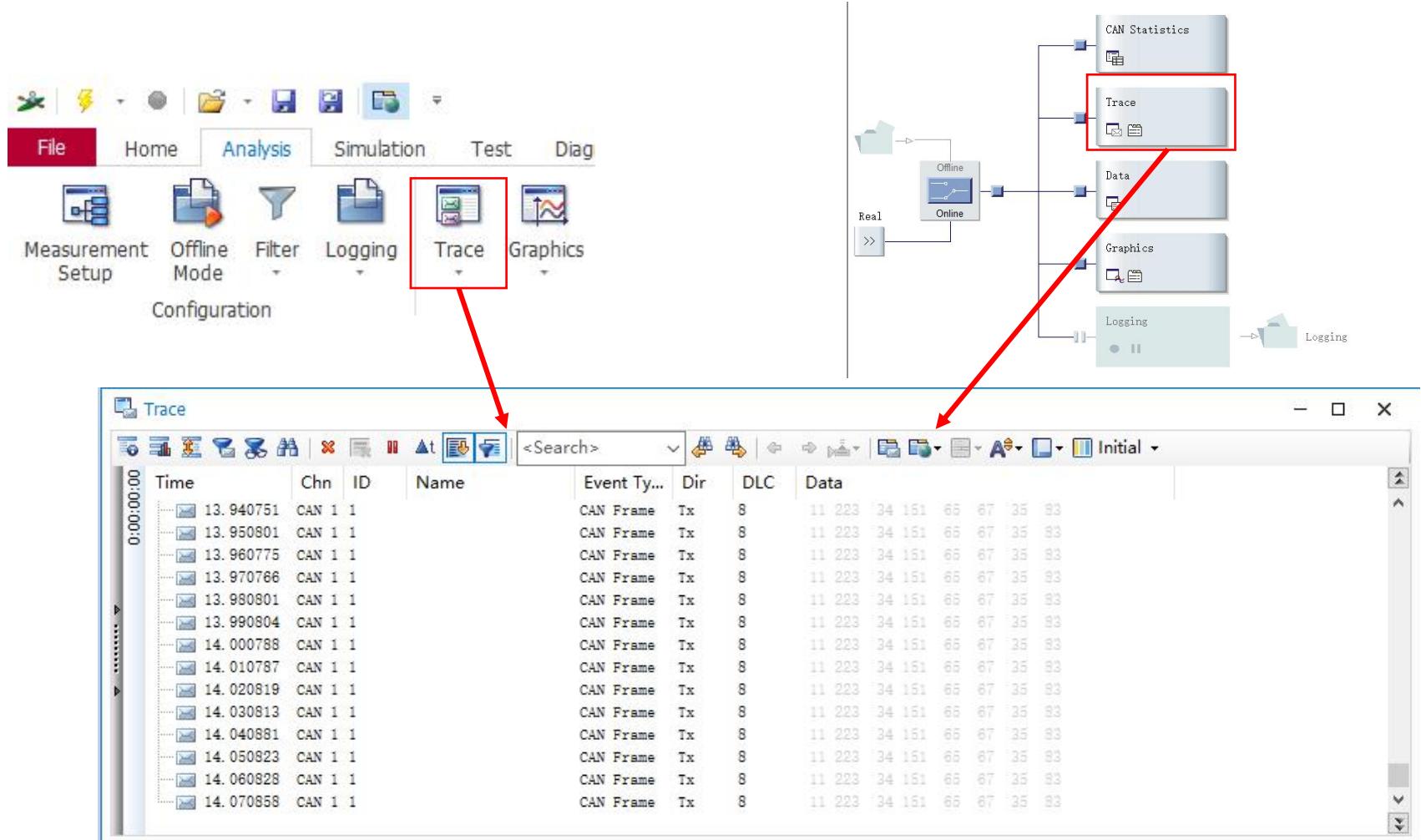


1. 概览
2. 工程创建
3. 报文发送
4. 分析窗口
5. 过滤功能模块
6. 数据记录
7. 离线分析
8. 系统变量环境变量
9. Panel设计
10. CAPL语言

## &gt; 窗口类型



## Trace Window



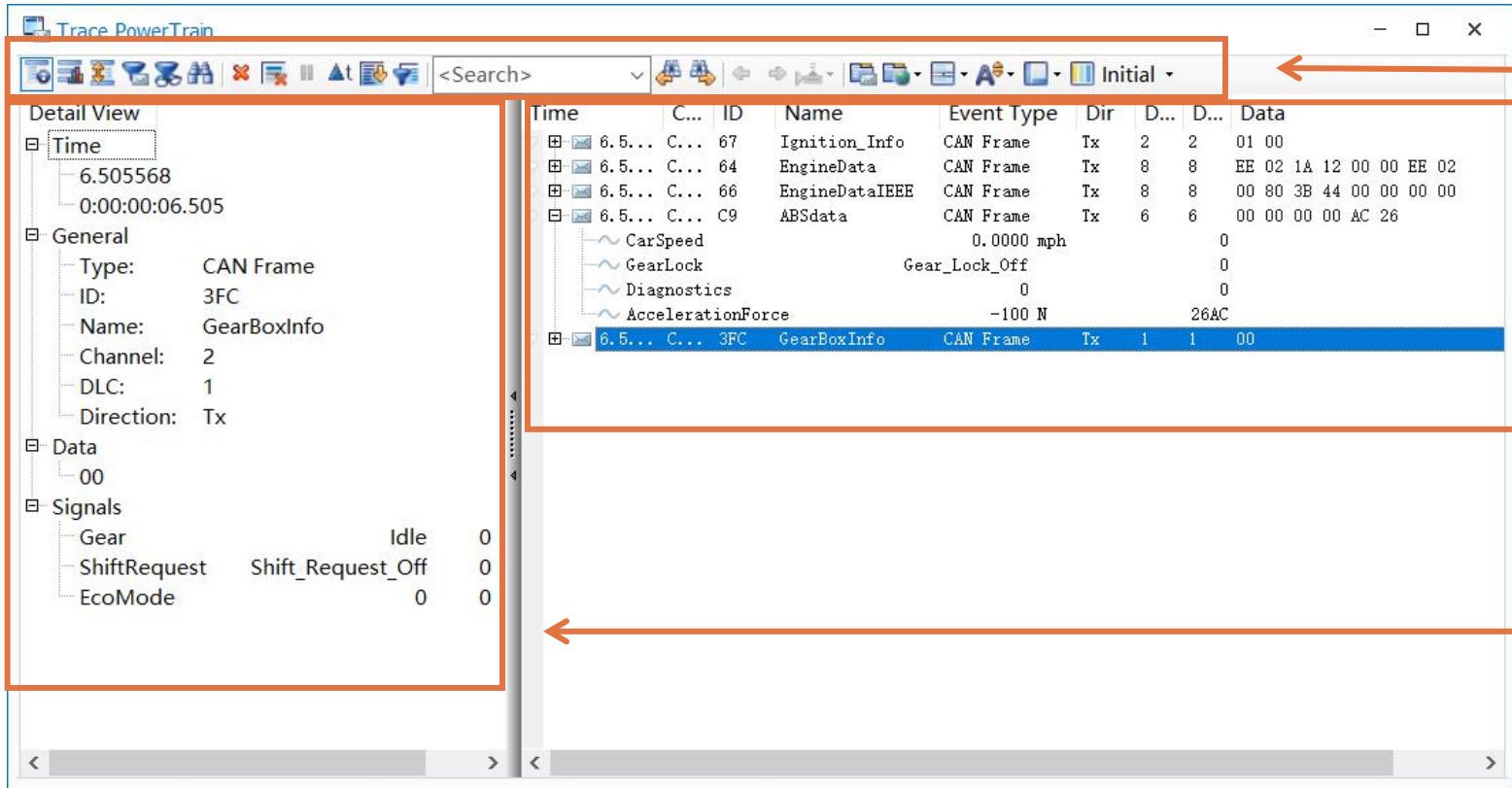
The image shows the Trace Window interface with a red box highlighting the 'Trace' button in the toolbar. A red arrow points from this button to a block diagram where the 'Trace' block is also highlighted with a red box. The block diagram illustrates the flow of data from various sources (Real, Offline, Online) through a 'Trace' block, which then connects to 'Data', 'Graphics', and 'Logging' blocks.

**Trace Window Interface:**

Time	Chn	ID	Name	Event Ty...	Dir	DLC	Data
13.940751	CAN 1	1		CAN Frame	Tx	8	11 223 34 151 65 67 35 93
13.950801	CAN 1	1		CAN Frame	Tx	8	11 223 34 151 65 67 35 93
13.960775	CAN 1	1		CAN Frame	Tx	8	11 223 34 151 65 67 35 93
13.970766	CAN 1	1		CAN Frame	Tx	8	11 223 34 151 65 67 35 93
13.980801	CAN 1	1		CAN Frame	Tx	8	11 223 34 151 65 67 35 93
13.990804	CAN 1	1		CAN Frame	Tx	8	11 223 34 151 65 67 35 93
14.000788	CAN 1	1		CAN Frame	Tx	8	11 223 34 151 65 67 35 93
14.010787	CAN 1	1		CAN Frame	Tx	8	11 223 34 151 65 67 35 93
14.020819	CAN 1	1		CAN Frame	Tx	8	11 223 34 151 65 67 35 93
14.030813	CAN 1	1		CAN Frame	Tx	8	11 223 34 151 65 67 35 93
14.040881	CAN 1	1		CAN Frame	Tx	8	11 223 34 151 65 67 35 93
14.050823	CAN 1	1		CAN Frame	Tx	8	11 223 34 151 65 67 35 93
14.060828	CAN 1	1		CAN Frame	Tx	8	11 223 34 151 65 67 35 93
14.070858	CAN 1	1		CAN Frame	Tx	8	11 223 34 151 65 67 35 93

## &gt; Trace 窗口

Trace窗口包含报文显示，解析，记录，分析等功能。



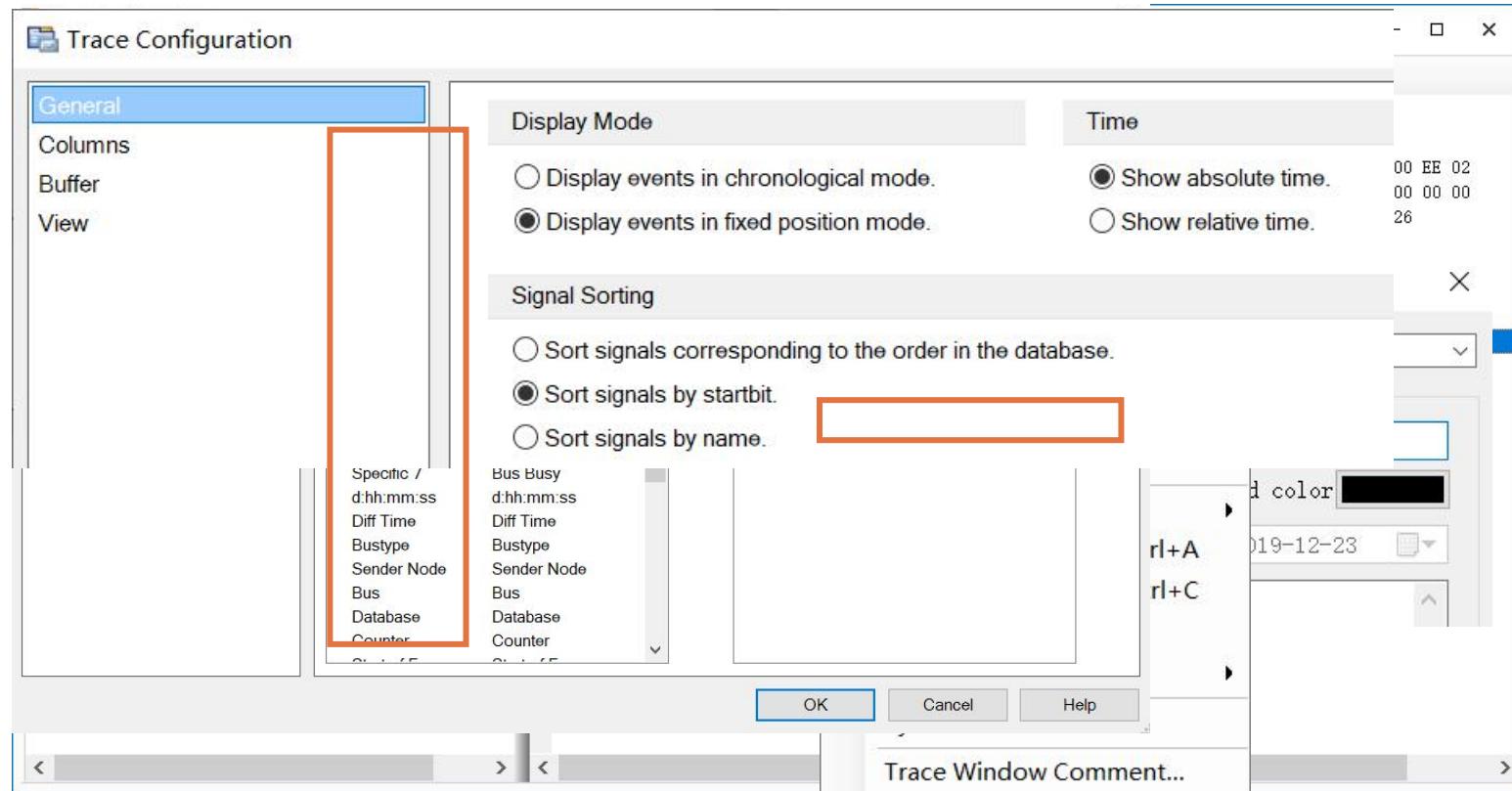
The screenshot shows the Trace PowerTrain window with three main sections highlighted by orange arrows:

- 窗口功能 (Window Functions):** Points to the top toolbar containing various icons for search, filter, and export.
- 报文显示 (Message Display):** Points to the central table view showing CAN frames. The table includes columns for Time, C..., ID, Name, Event Type, Dir, D..., D..., and Data. One row is selected, showing a CAN frame from address C9 with data fields like CarSpeed, GearLock, Diagnostics, and AccelerationForce.
- 报文详细信息 (Message Detailed Information):** Points to the left pane which displays detailed information for the selected message, including Time, General settings (Type: CAN Frame, ID: 3FC, etc.), and Data signals (Gear, ShiftRequest, EcoMode).

Time	C...	ID	Name	Event Type	Dir	D...	D...	Data
6.505568	C...	67	Ignition_Info	CAN Frame	Tx	2	2	01 00
6.505568	C...	64	EngineData	CAN Frame	Tx	8	8	EE 02 1A 12 00 00 EE 02
6.505568	C...	66	EngineDataIEEE	CAN Frame	Tx	8	8	00 80 3B 44 00 00 00 00
6.505568	C...	C9	ABSdata	CAN Frame	Tx	6	6	00 00 00 00 AC 26
			CarSpeed			0.0000 mph		0
			GearLock			Gear_Lock_Off		0
			Diagnostics			0		0
			AccelerationForce			-100 N		26AC
6.505568	C...	3FC	GearBoxInfo	CAN Frame	Tx	1	1	00

## &gt; Trace 窗口

添加标记，增加列项：



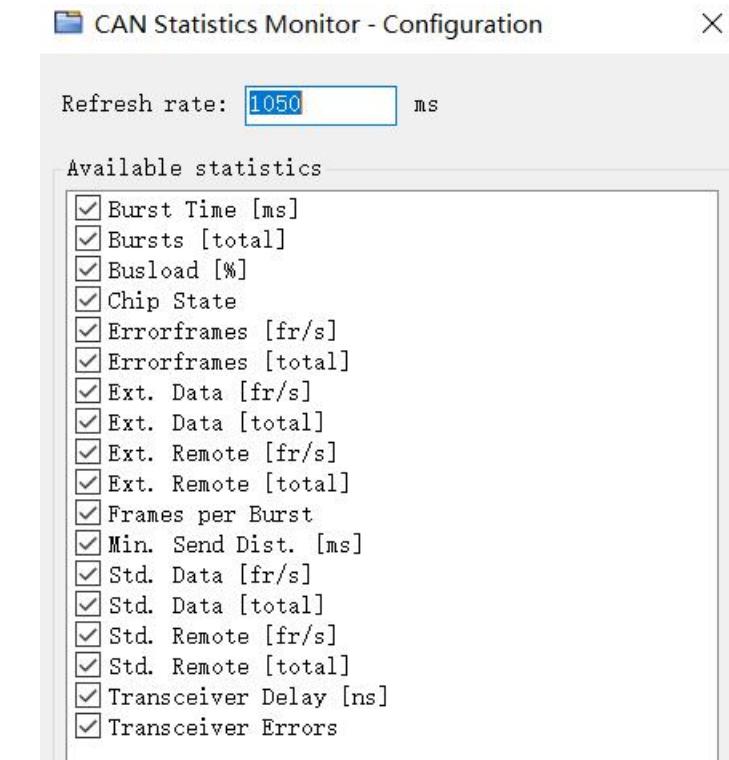
## &gt; Statistics窗口

统计窗口，显示总线报文的最大最小值，平均值，当前值信息：

CAN 1 Statistics

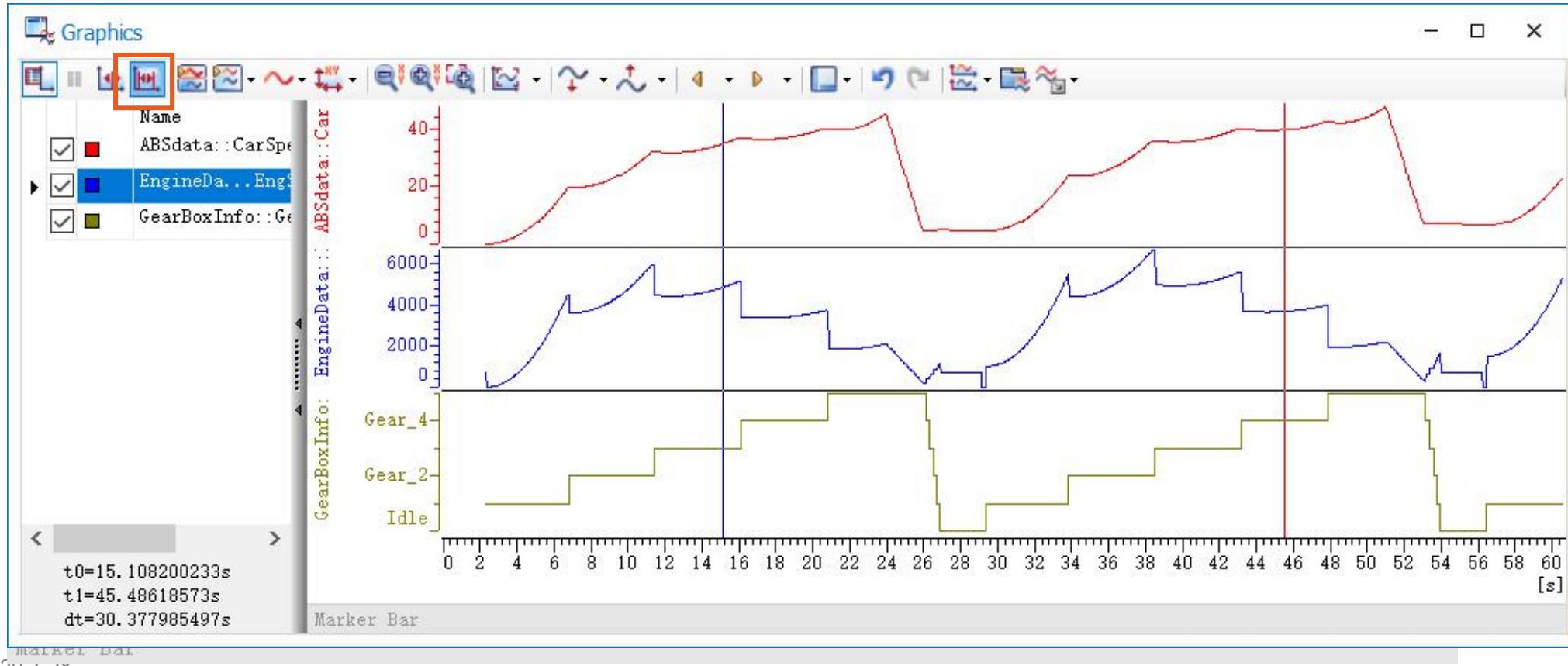
CAN Channel: CAN 1 - Comfort

Statistic	Curren...	Min	Max	Avg
Busload [%]	2.44	0.00	2.44	1.08
Min. Send Dist.	-	n/a	n/a	n/a
Burst Time [ms]	-	-	-	-
Bursts [total]	-	n/a	-	-
Frames per Burst	-	-	-	-
Std. Data [fr/s]	144	0	-	-
Std. Data [to...]	477	n/a	-	-
Ext. Data [fr/s]	0	0	-	-
Ext. Data [to...]	0	n/a	-	-
Std. Remote [...]	0	0	-	-
Std. Remote [...]	0	n/a	-	-
Ext. Remote [...]	0	0	-	-
Ext. Remote [...]	0	n/a	-	-
Errorframes [...]	0	0	n/a	n/a
Errorframes [...]	0	n/a	n/a	n/a
Chip State	Simulated	n/a	n/a	n/a
Transceiver E...	0	n/a	n/a	n/a
Transceiver D...	-	-	-	-

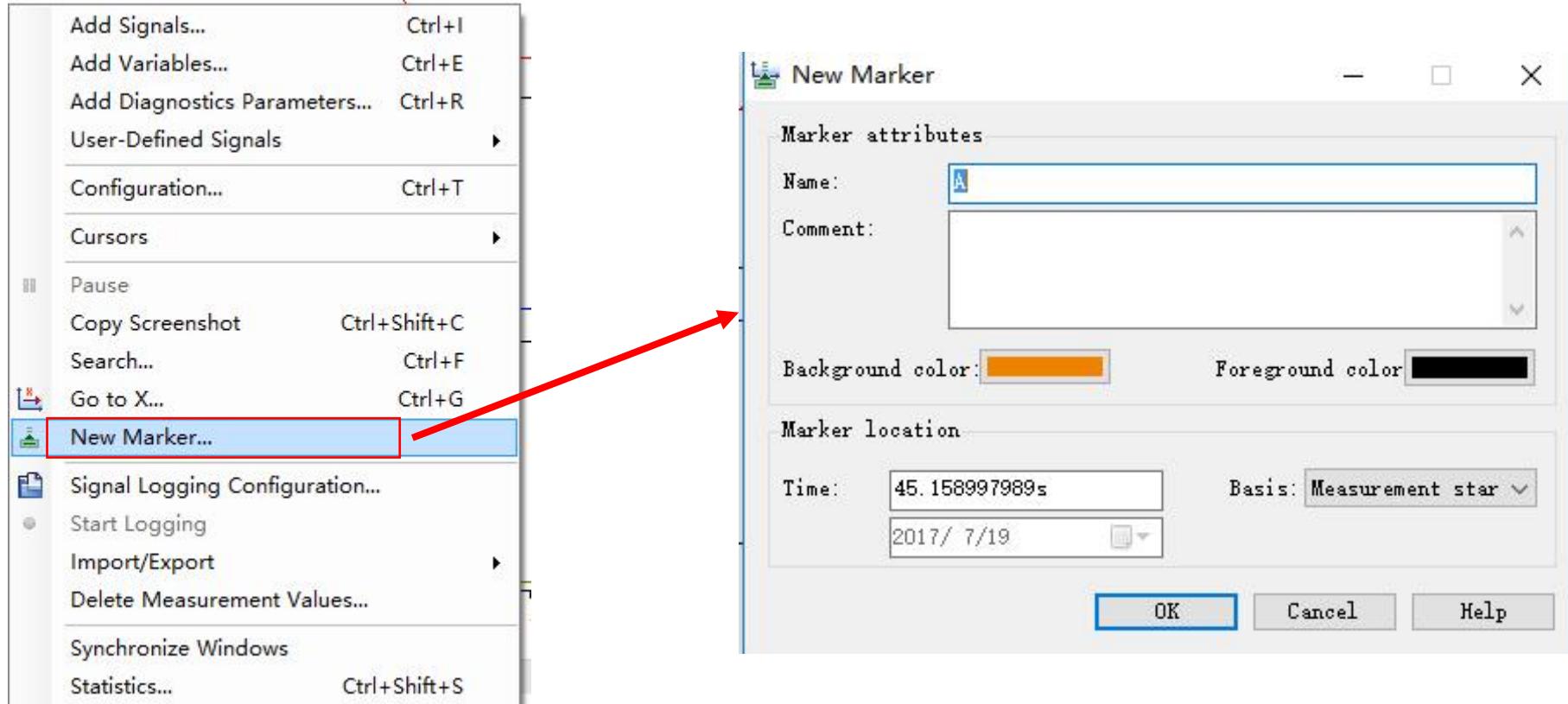


## &gt; Graphic 窗口

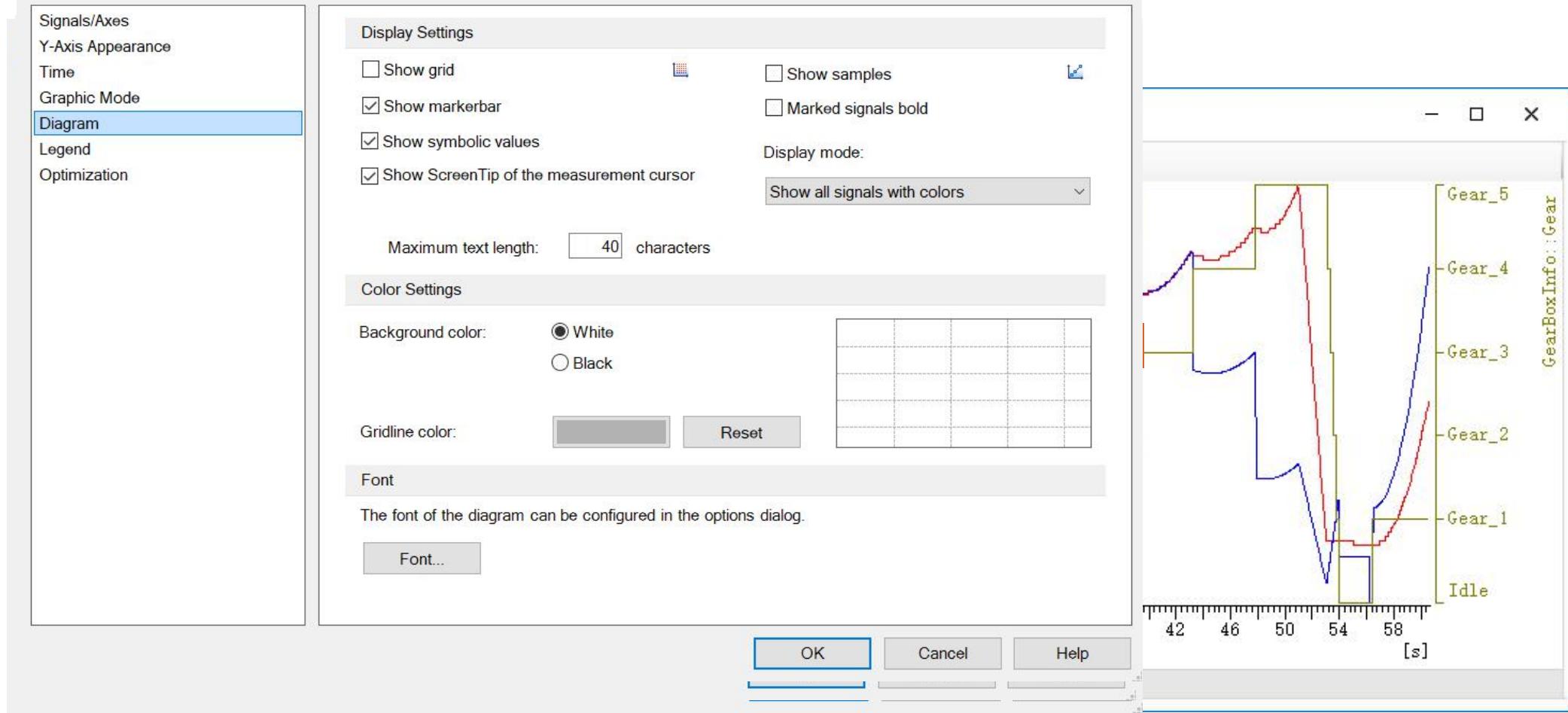
图形窗口提供各种信号的图形显示：



## &gt; Marker



### > Graphic 窗口



The screenshot shows the 'Graphic' window settings dialog and a plot window.

**Left Panel (Settings):**

- Signals/Axes
- Y-Axis Appearance
- Time
- Graphic Mode
- Diagram** (selected)
- Legend
- Optimization

**Display Settings:**

- Show grid
- Show samples
- Show markerbar
- Show symbolic values
- Show ScreenTip of the measurement cursor

Display mode: Show all signals with colors

Maximum text length: 40 characters

**Color Settings:**

Background color:  White  Black

Gridline color:  Reset

**Font:**

The font of the diagram can be configured in the options dialog.

Font...

**Buttons at the bottom:**

OK Cancel Help

**Plot Window:**

The plot shows multiple signals over time (x-axis, 42 to 58 seconds). The signals represent gear positions: Gear\_5 (red), Gear\_4 (blue), Gear\_3 (green), Gear\_2 (yellow), Gear\_1 (orange), and Idle (grey). A vertical legend on the right identifies the gears. A yellow box highlights a specific event around 50 seconds where Gear\_5 transitions from red to blue.

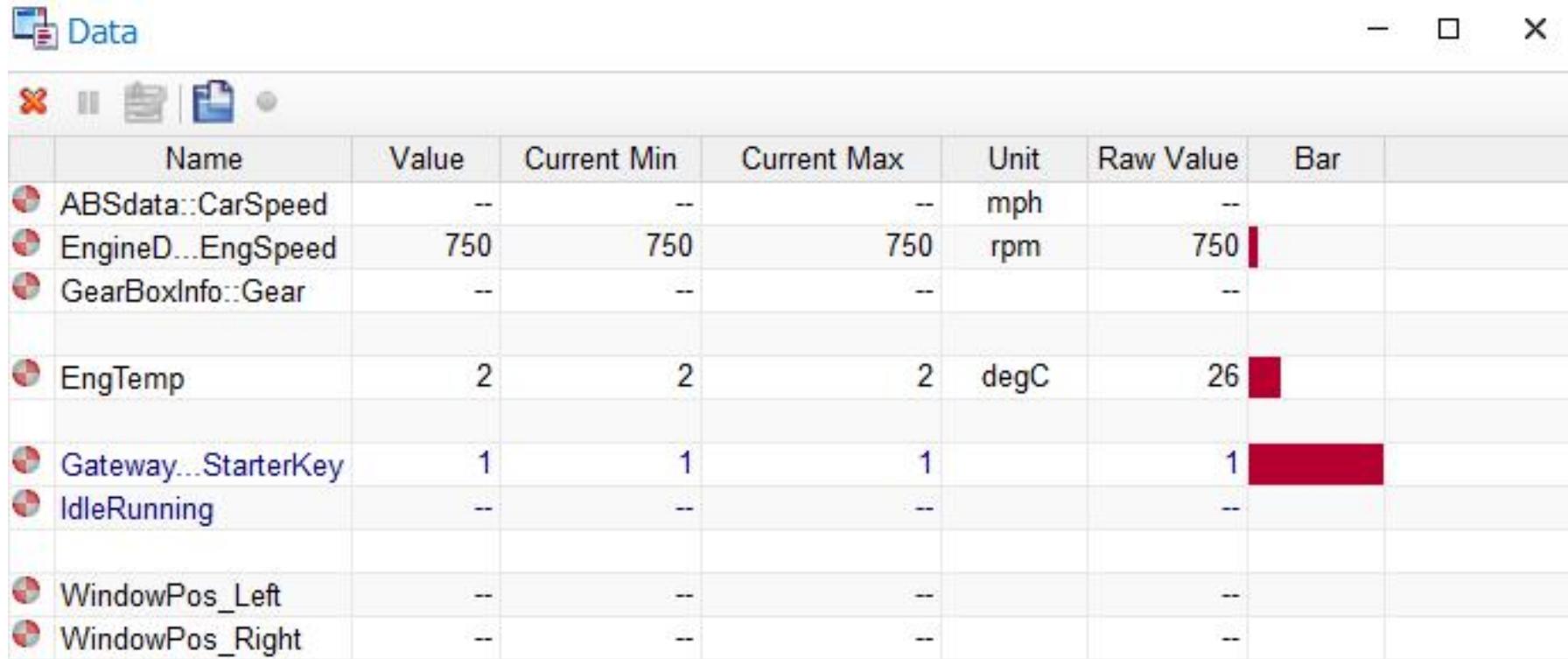
### > Graphic 窗口

对于功能栏，所有的图标可以通过help文档查询解释：

Select scaling mode (the icon of the current selection is displayed)		
►	<b>Symbol</b>	<b>Function</b>
►	Adaptive	Show legend
►	Paused	Pauses a signal
►	Switch	Move signals (the icon of the current selection is displayed)
	Zoom	Move signal down
	Zoom	Move signal left
	Zoom	Move signal right
	Shows signals	Move signal up
	Fit signals	Set scrolling direction (the icon of the current selection is displayed)
►	Fit	Displays
►	Grays	Scroll left
►	Fit	Shows oldest
►	Fit	Shows newest
►	Select signal mode	Set scrolling direction (the icon of the current selection is displayed)
►	Single signal mode	Scroll right
►	Multiple signal mode	Scroll up
		Scroll to the oldest measurement value
		Scroll to the newest measurement value
		Set the visible time range
		Reset separate views
		Signal logging configuration
		Logging of Graphics Window signals
		Swaps the data history to hard disk

### > Data窗口

显示信号的名字，单位，初始值等：

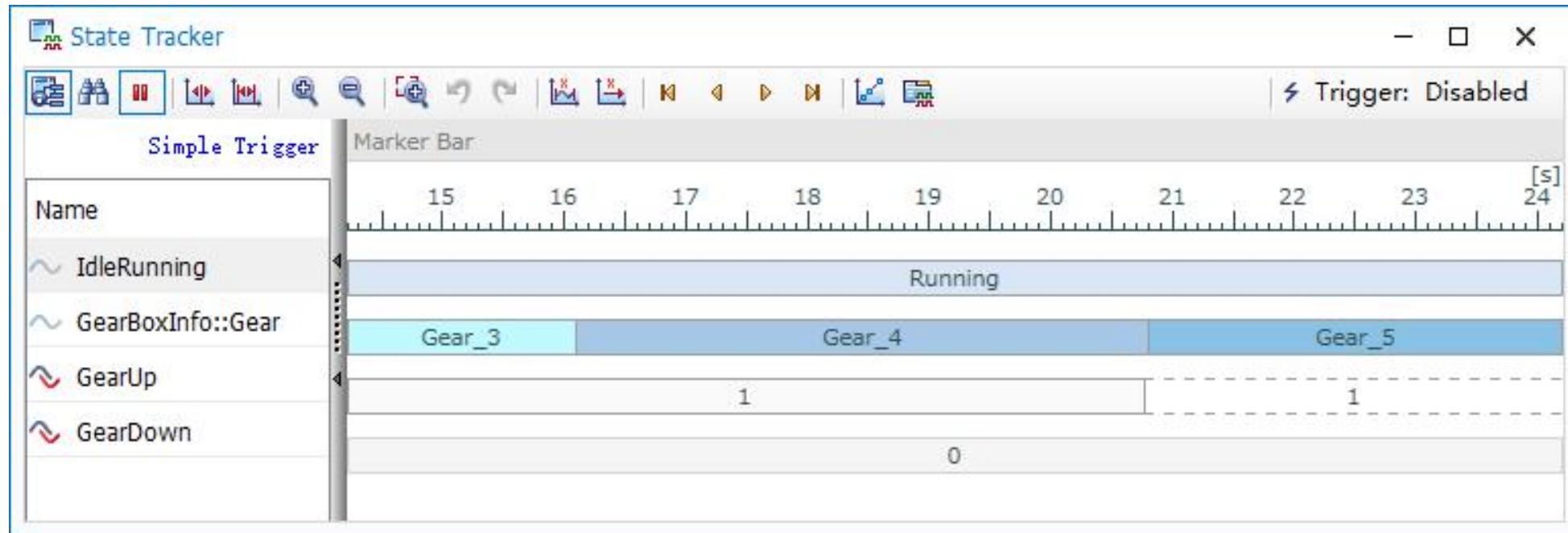


The screenshot shows the CANoe Data window interface. The title bar says "Data". Below it is a toolbar with icons for close, minimize, maximize, and other functions. The main area is a table with the following columns: Name, Value, Current Min, Current Max, Unit, Raw Value, Bar. The table lists several signals:

Name	Value	Current Min	Current Max	Unit	Raw Value	Bar
ABSdata::CarSpeed	--	--	--	mph	--	
EngineD...EngSpeed	750	750	750	rpm	750	<div style="width: 100%; background-color: red;"></div>
GearBoxInfo::Gear	--	--	--		--	
EngTemp	2	2	2	degC	26	<div style="width: 100%; background-color: red;"></div>
Gateway...StarterKey	1	1	1		1	<div style="width: 100%; background-color: red;"></div>
IdleRunning	--	--	--		--	
WindowPos_Left	--	--	--		--	
WindowPos_Right	--	--	--		--	

> State Tracker窗口

显示比特值和一些状态值：



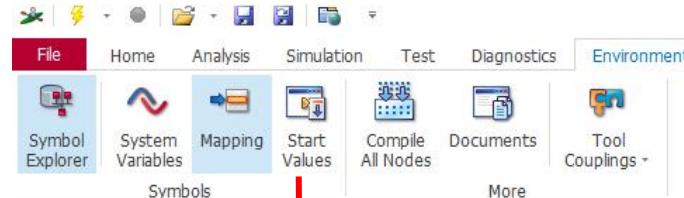
### > Scope窗口

观察总线点评，用于分析总线协议，用Eye图评估信号品质：



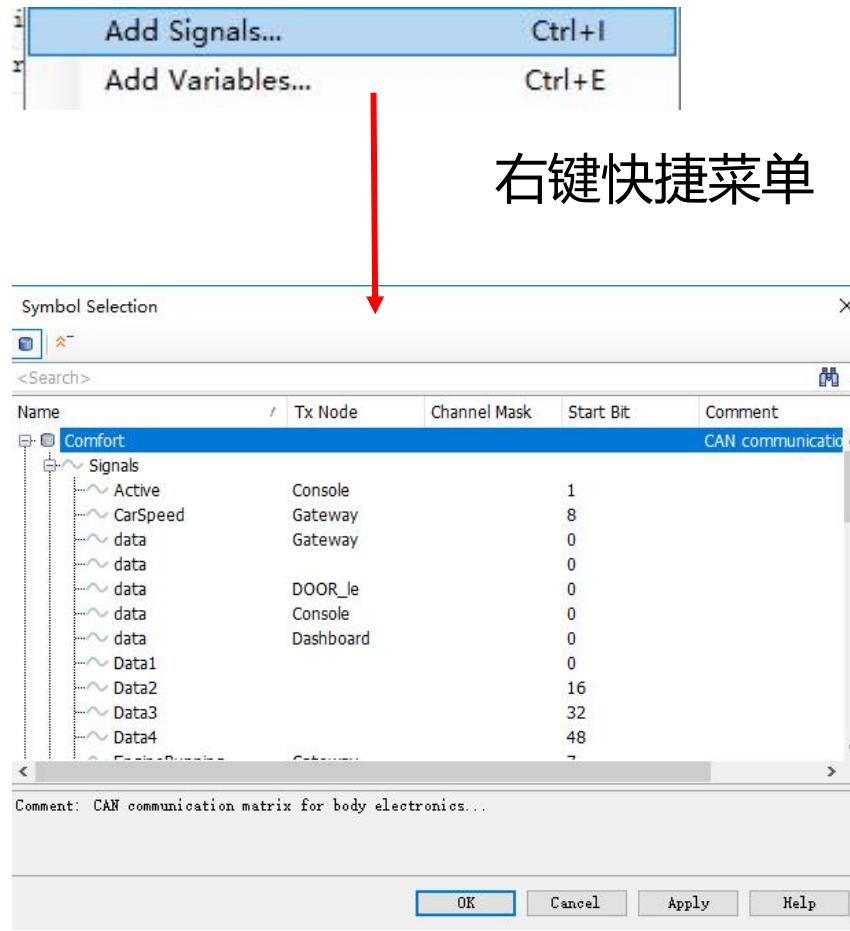


## Symbol Explorer

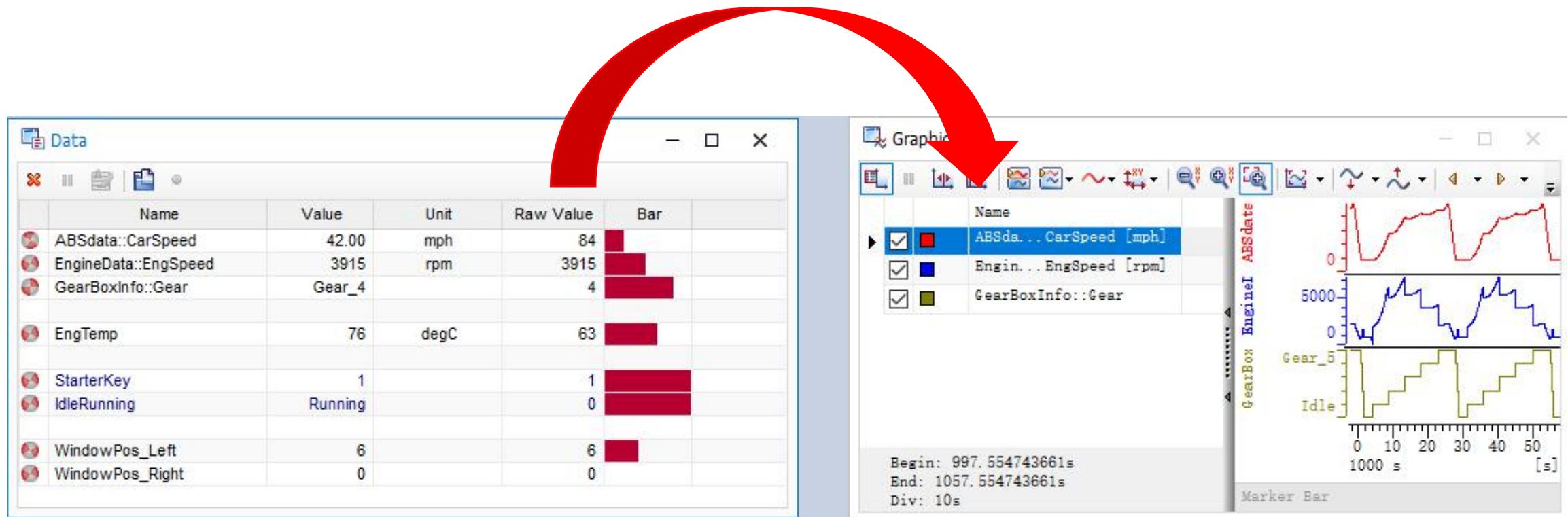


A screenshot of the 'Symbol Explorer' window. It shows a tree view of symbols under 'Predefined events' and 'Comfort'. Under 'Comfort', there is a 'Signals' node which is expanded, showing various signals like 'Active', 'CarSpeed', 'data', etc., each associated with a 'Tx Node' (Console, Gateway, etc.). A red arrow points from the text '右键快捷菜单' down to the 'Start Values' icon in the ribbon bar.

## 右键快捷菜单



通过拖拉方式从另外一个分析窗口



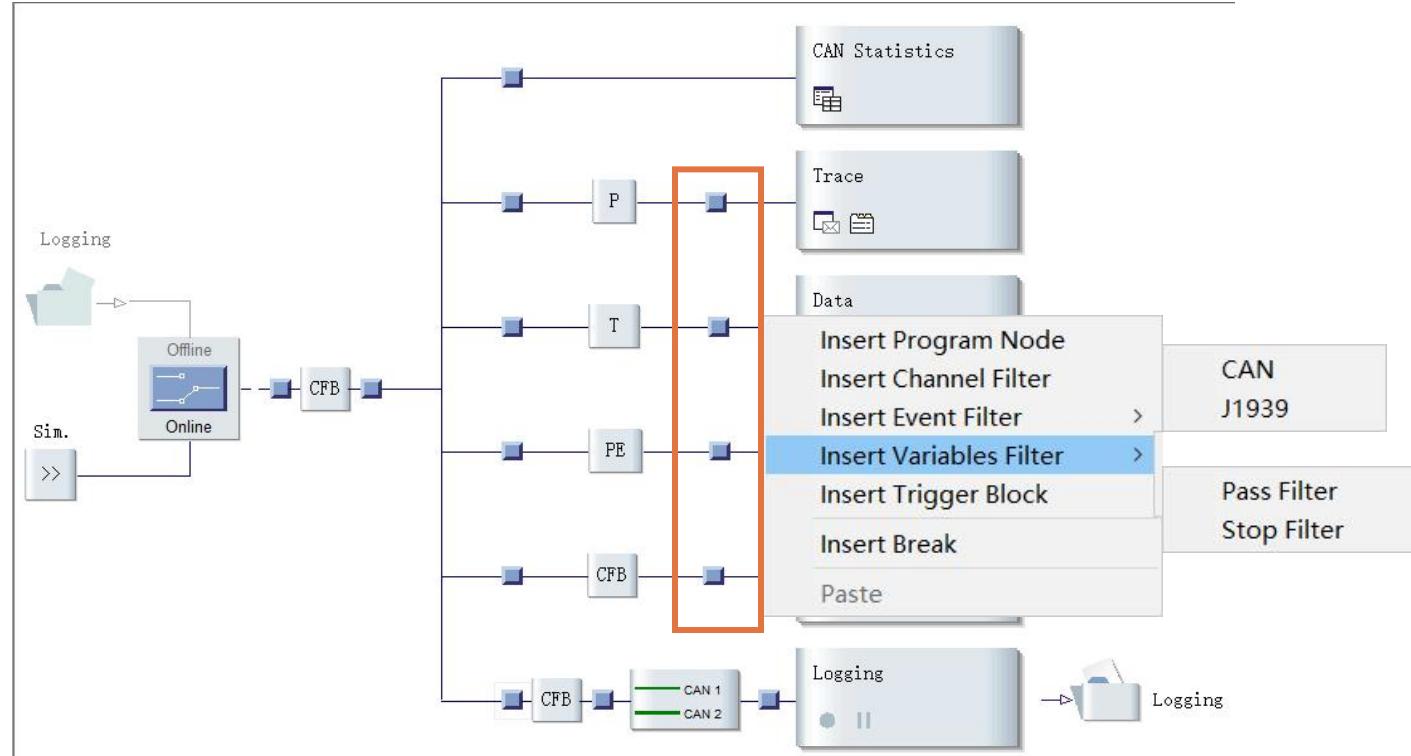
## CONTENTS



1. 概览
2. 工程创建
3. 报文发送
4. 分析窗口
5. 过滤功能模块
6. 数据记录
7. 离线数据分析
8. 系统变量环境变量
9. Panel设计
10. CAPL语言

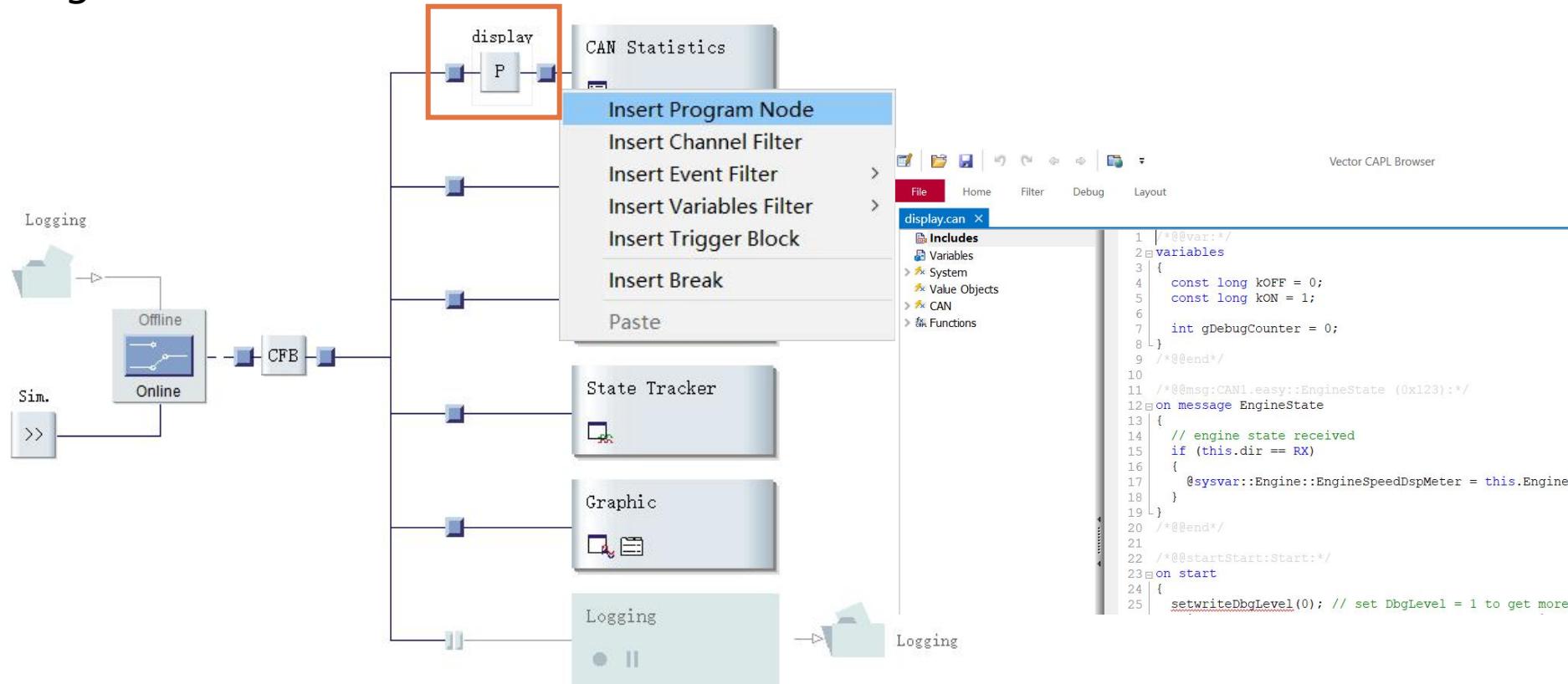
## &gt; 过滤功能设置

此功能属于分析功能，在Measurement Setup中设置



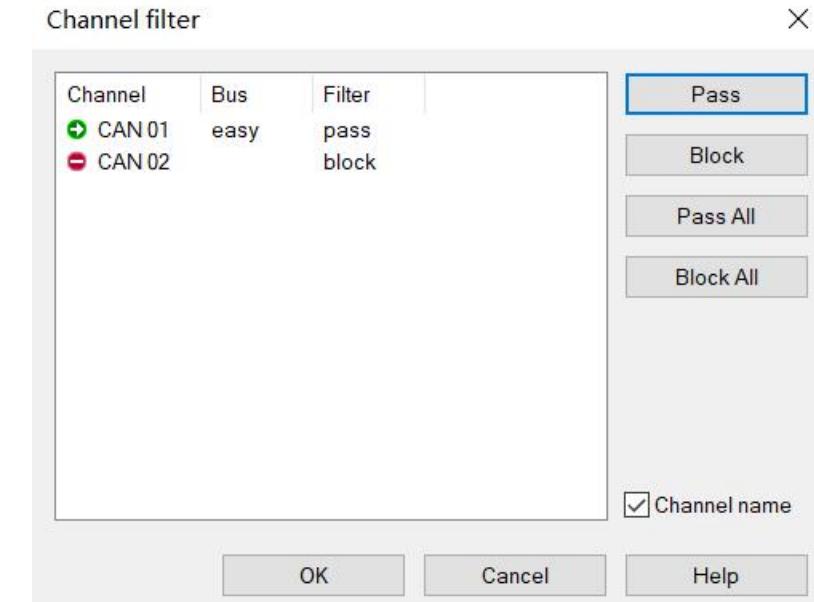
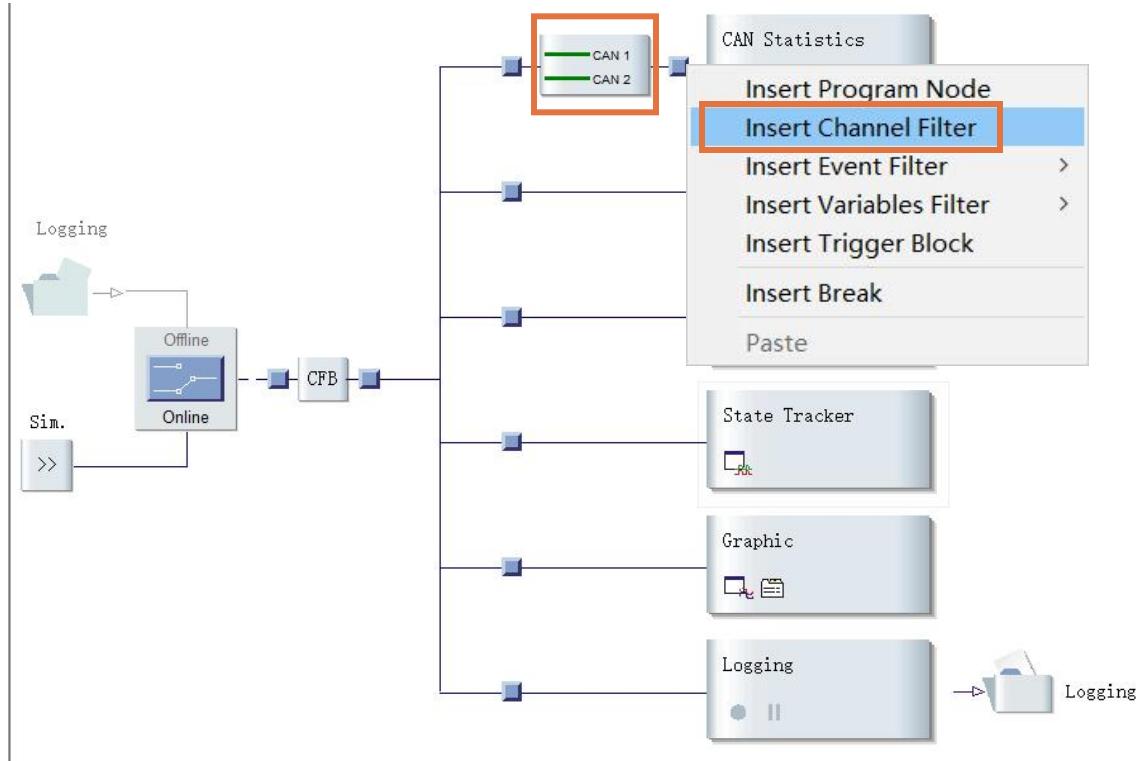
## &gt; 过滤功能设置

Program Node:



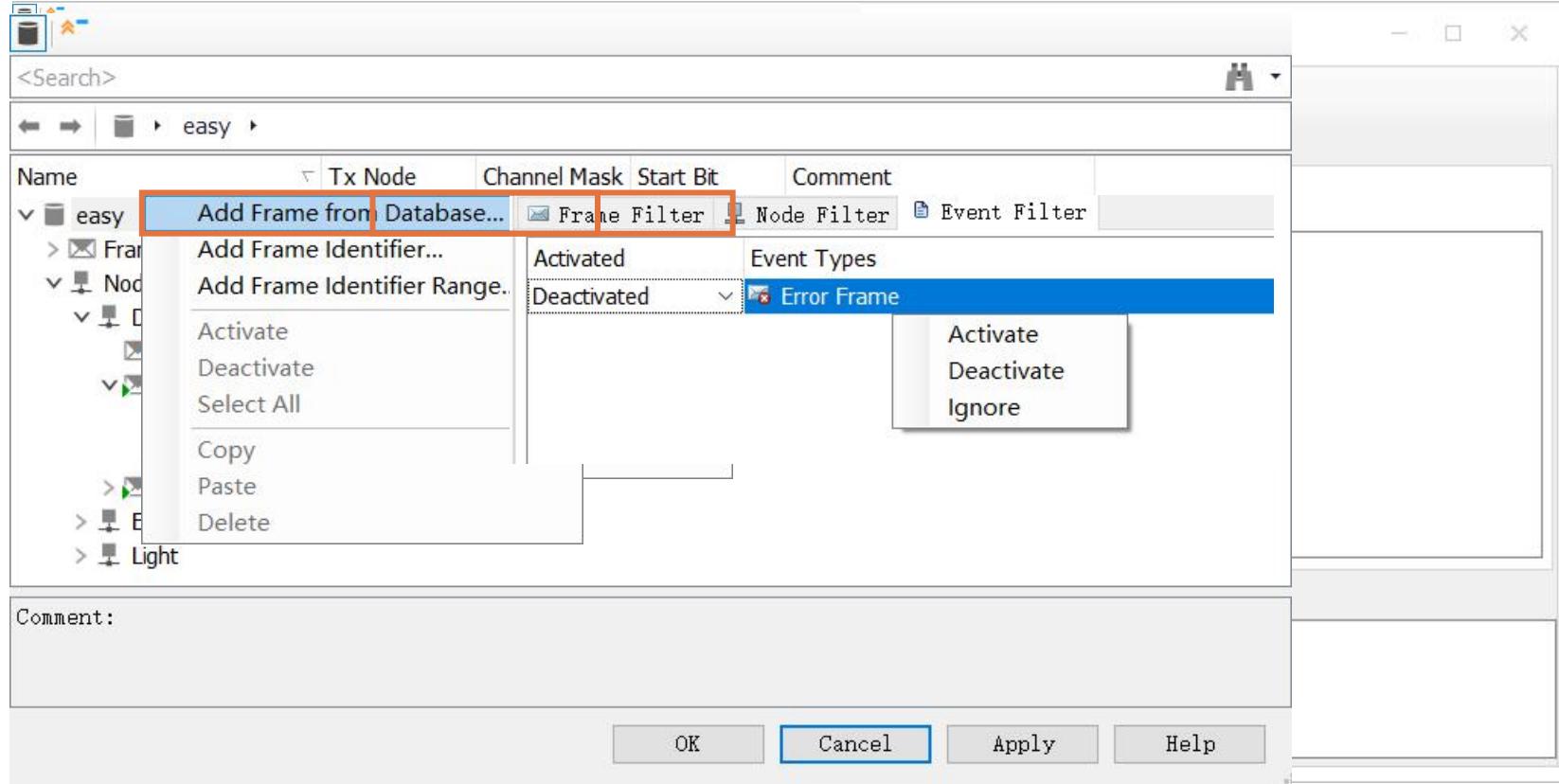
## &gt; 过滤功能设置

Channel filter:



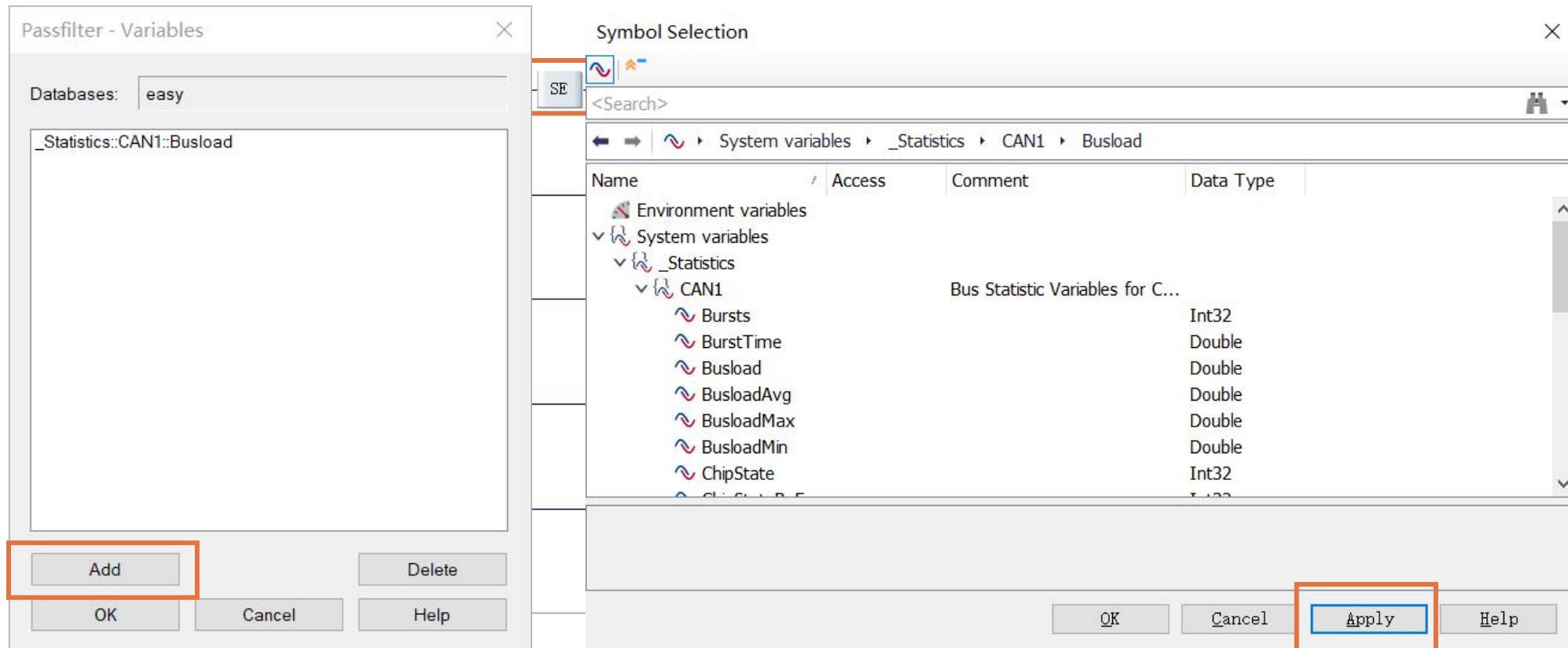
## &gt; 过滤功能设置

Event Filter:

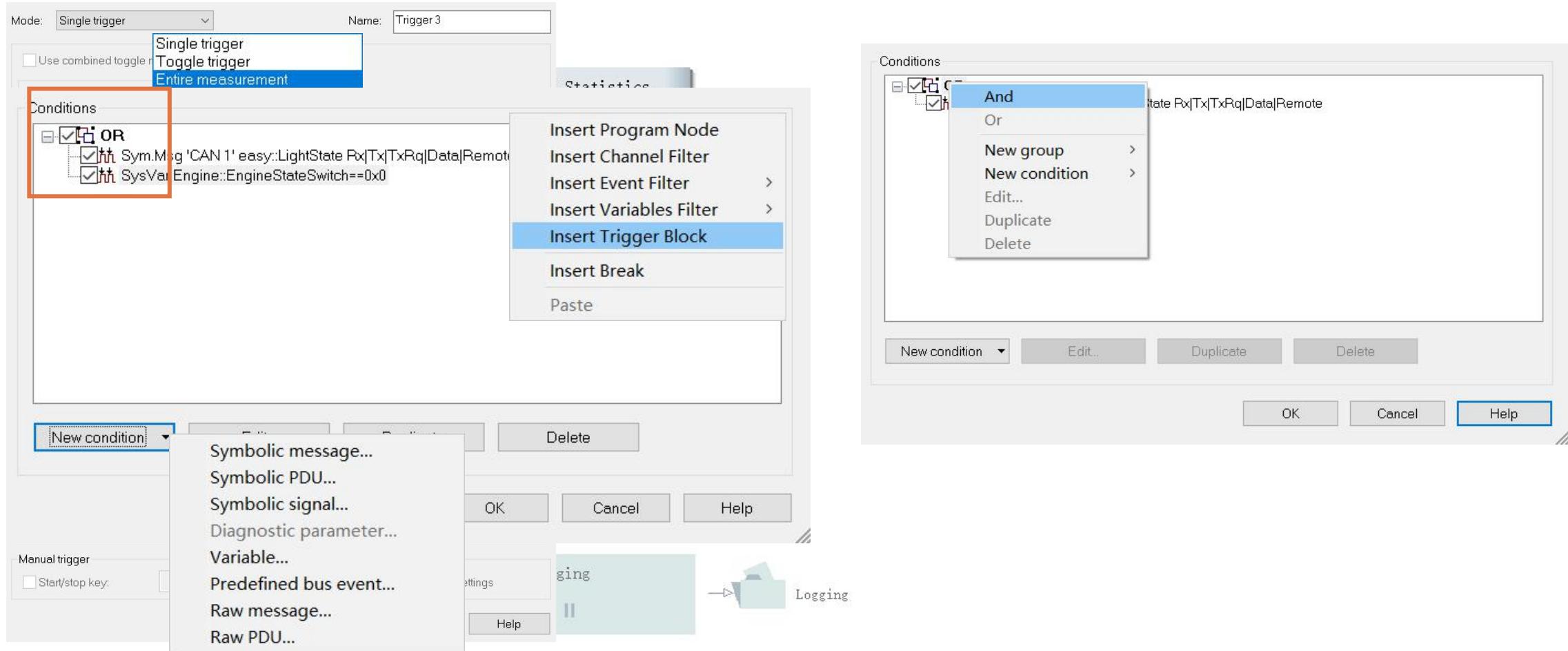


## &gt; 过滤功能设置

Variables Filter:



## &gt; 过滤功能设置



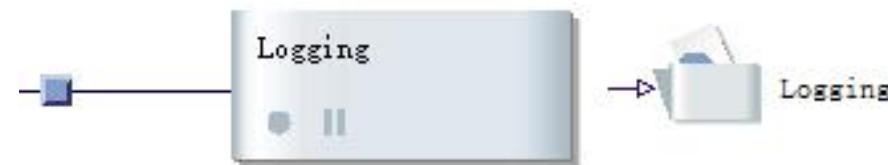
## CONTENTS



1. 概览
2. 工程创建
3. 报文发送
4. 分析窗口
5. 过滤功能模块
6. 数据记录
7. 离线数据分析
8. 系统变量环境变量
9. Panel设计
10. CAPL语言

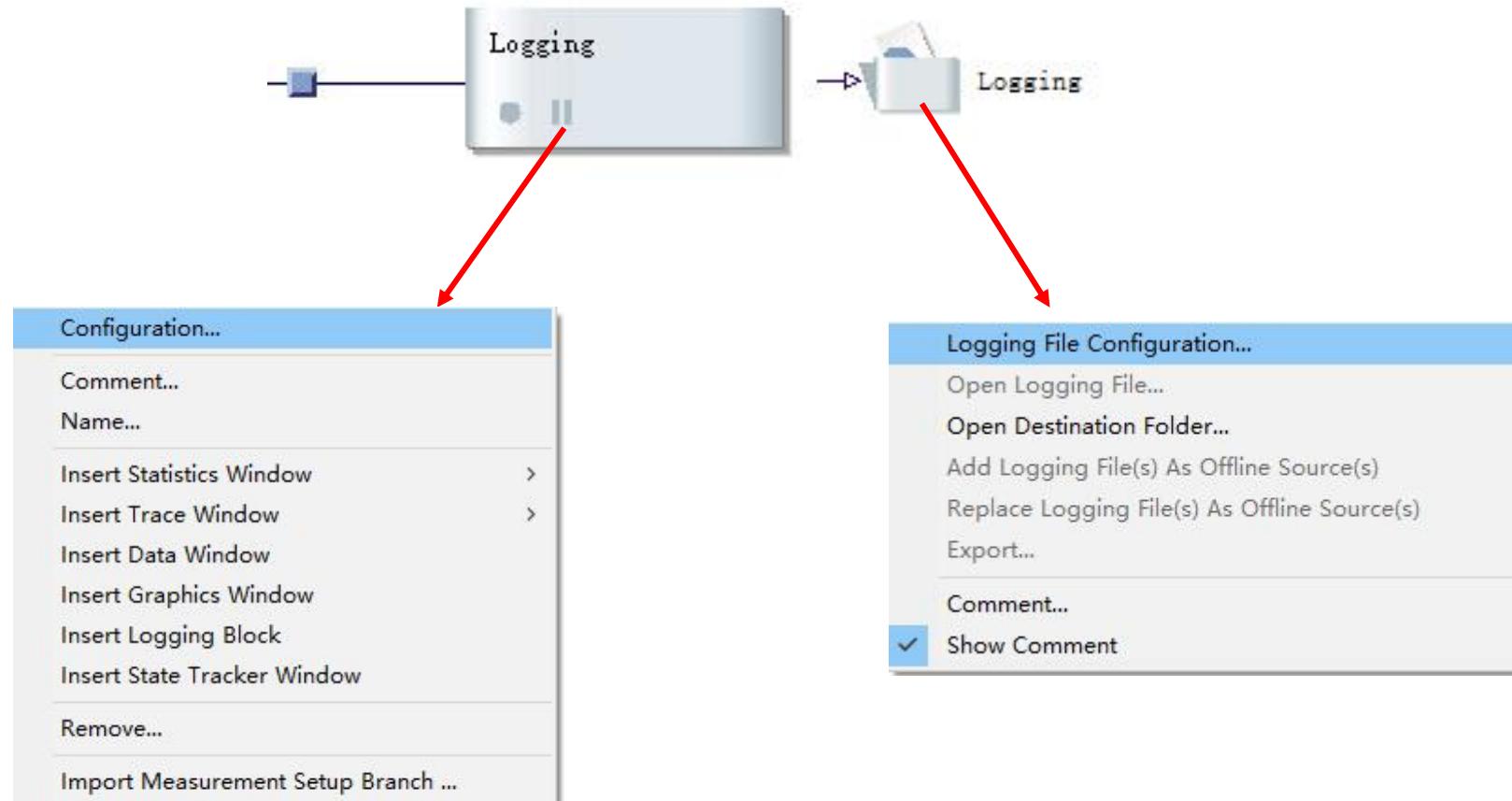
## 报文类记录格式

- 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

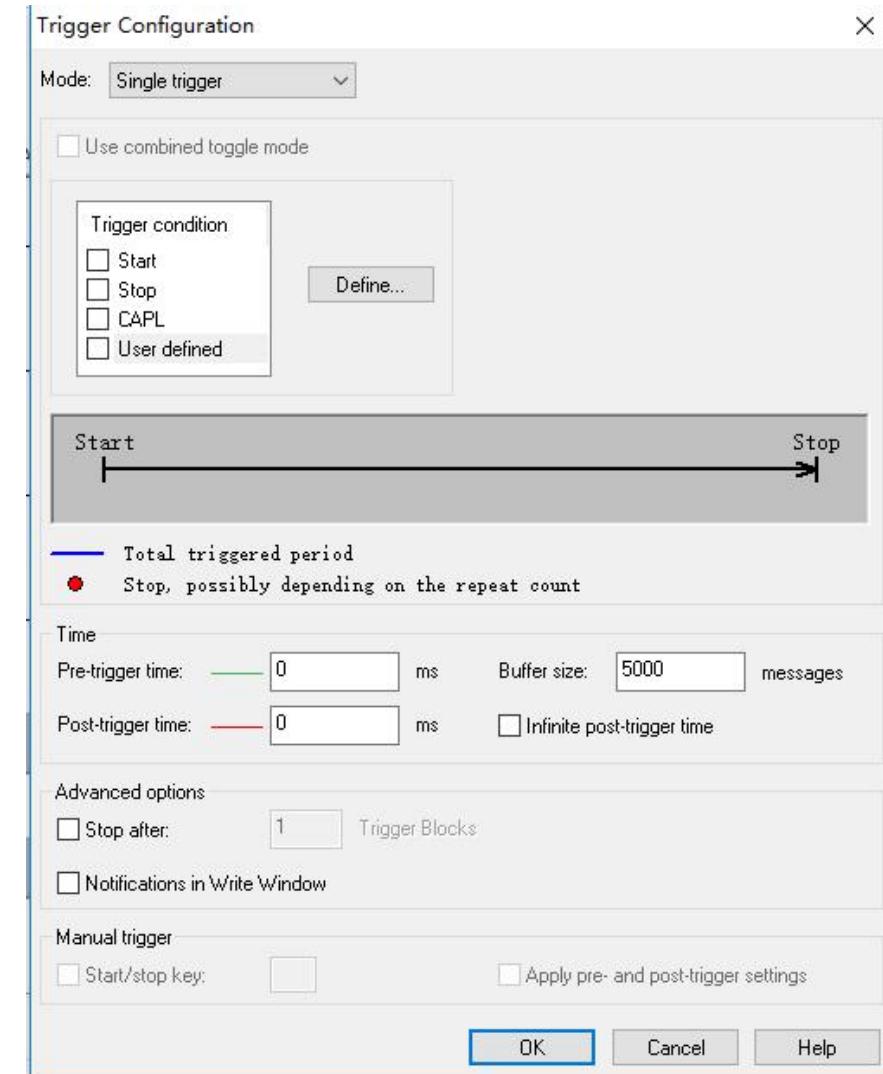
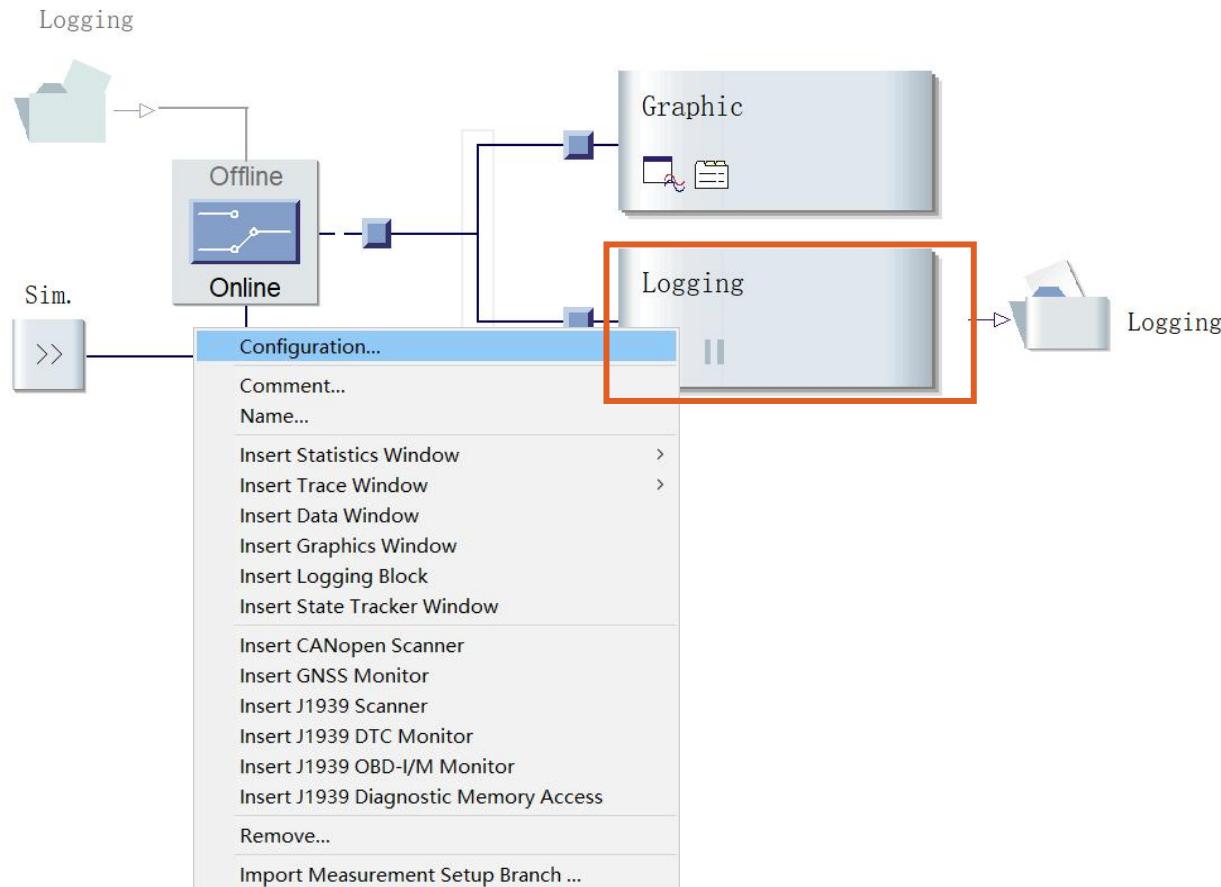


# 数据记录

## 记录模块

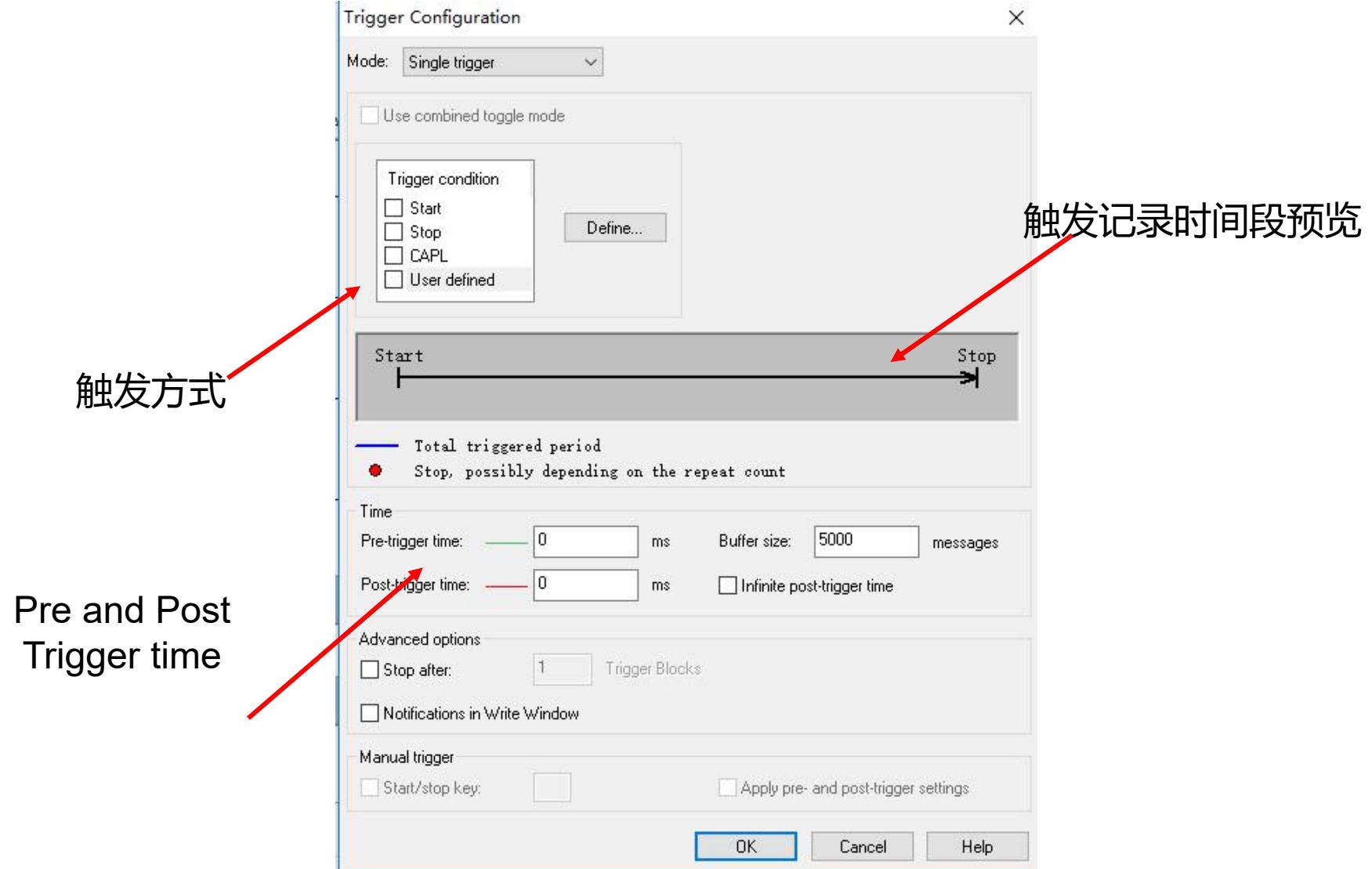


## &gt; 记录模块



# 数据记录

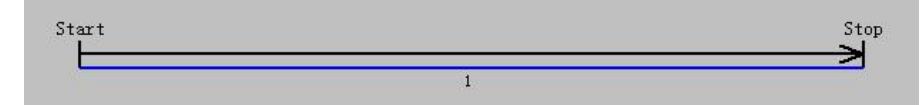
## 记录模块



### > 记录模块

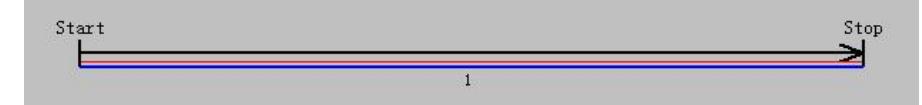
Entire Measurement

完整测量：从开始测量到结束记录完整数据



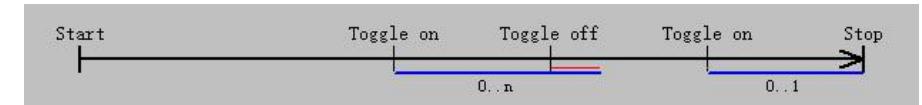
Single Trigger

单触发：一个触发条件，记录触发点前后数据



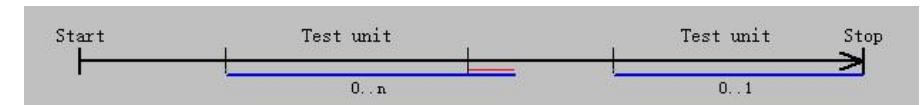
Toggle Trigger

组合触发：记录两个触发条件中间数据



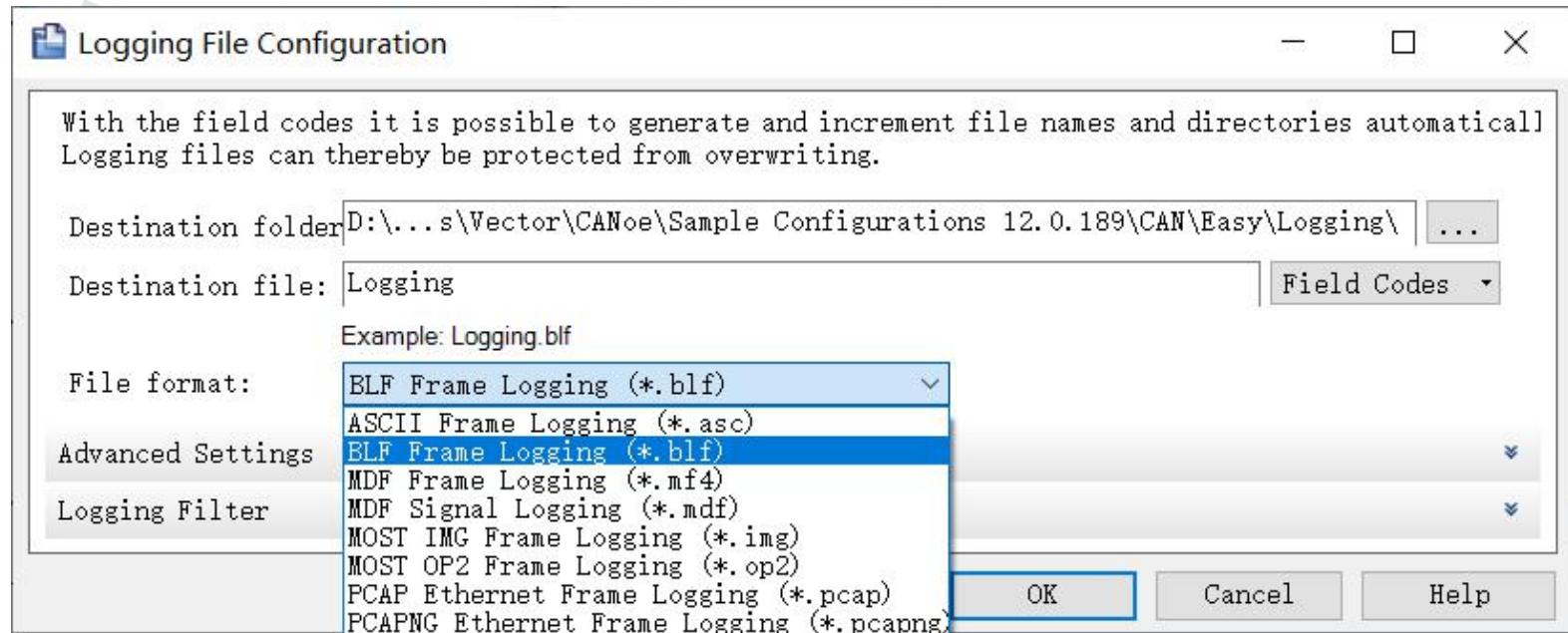
Test Trigger

测试触发：记录测试单元数据



### > 记录模块

Logging



## &gt; 记录模块

## CAPL Trigger:

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
on message 100
stopLogging();
// stops all Logging Blocks
write("Logging start");
startLogging( "Logging 1");
// starts the Logging Block "Logging 1"
stopLogging( "Logging 1");
// stops the Logging Block "Logging 1" after 1000ms
}
startLogging( "Logging 1", 2000);
// starts the Logging Block "Logging 1" with pre-trigger time 2000
on timer logging
milliseconds.
{
stopLogging( "Logging 1", 1000);
// stops the Logging Block "Logging 1" with post-trigger time 1000
trigger(); // Stop logging
milliseconds.
```

## CONTENTS



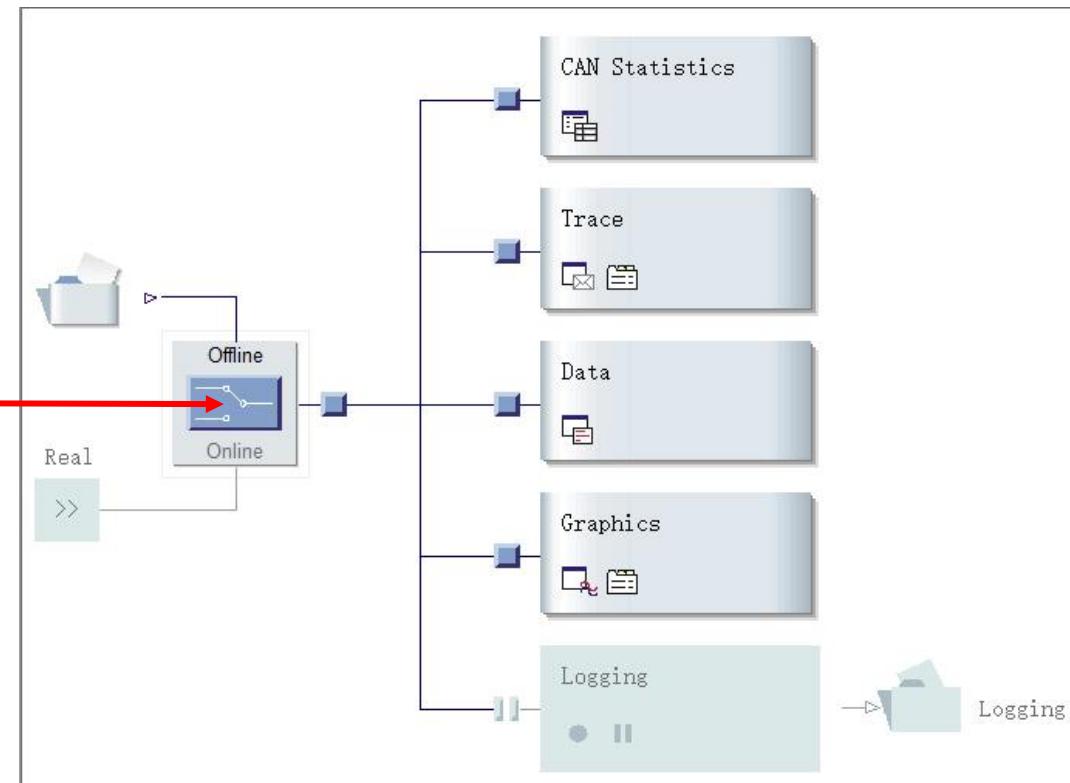
1. 概览
2. 工程创建
3. 报文发送
4. 分析窗口
5. 过滤功能模块
6. 数据记录
7. 离线数据分析
8. 系统变量环境变量
9. Panel设计
10. CAPL语言

## Offline Mode



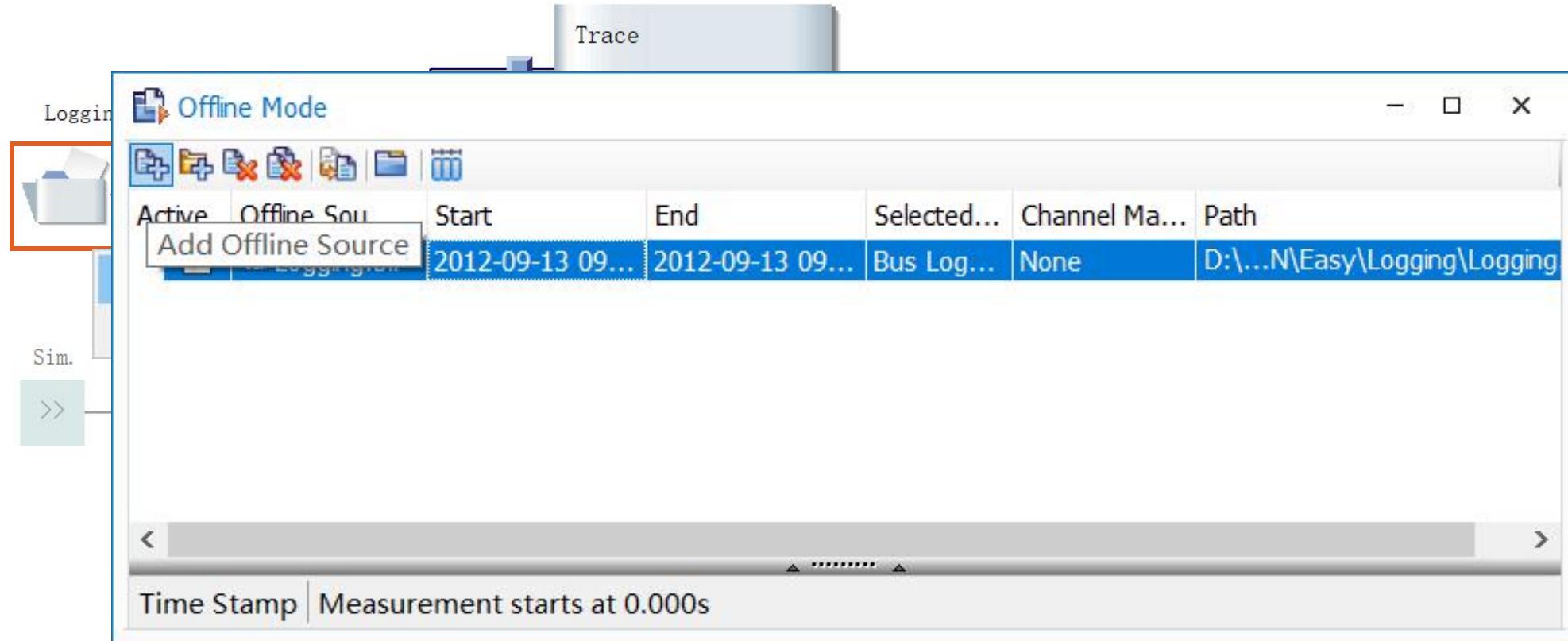
- Offline模式下回放记录文件
- 分析记录的数据
- 无需连接真实总线
- 可在分析窗口中分析数据

双击切换  
Online/Offline



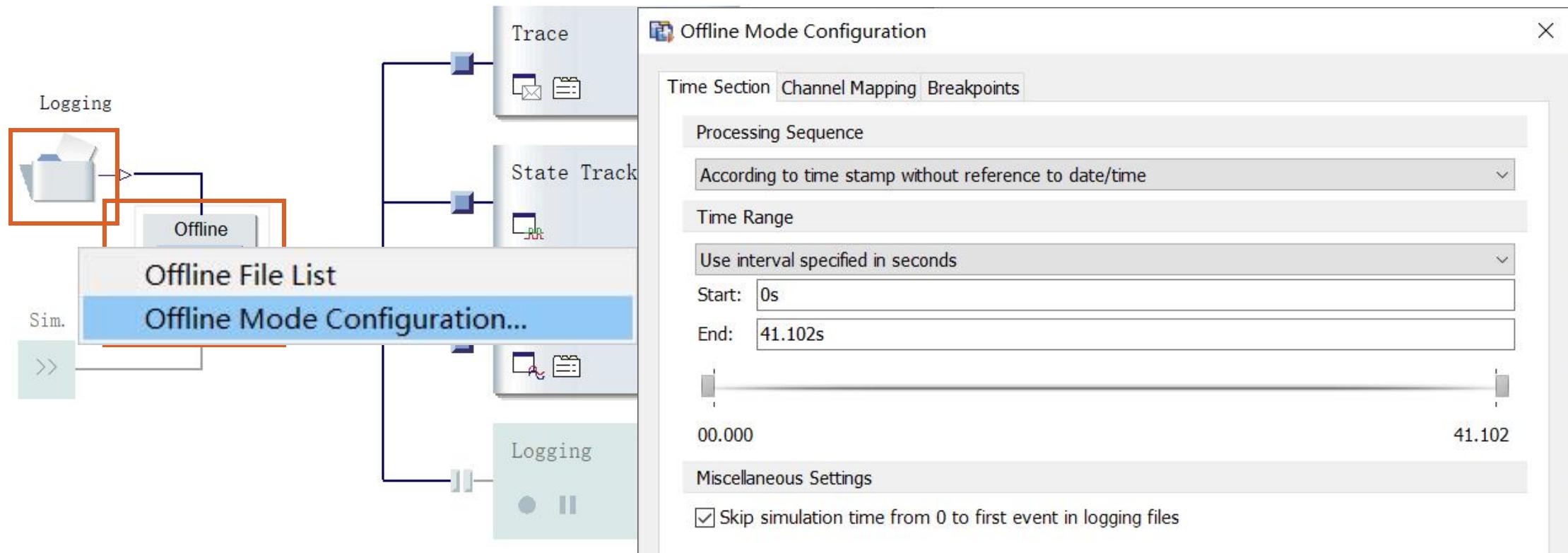
## &gt; 离线模式

在Measurement Setup窗口进行离线模式的切换：



## &gt; 离线模式

在Measurement Setup窗口进行离线模式的切换：



## > 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。

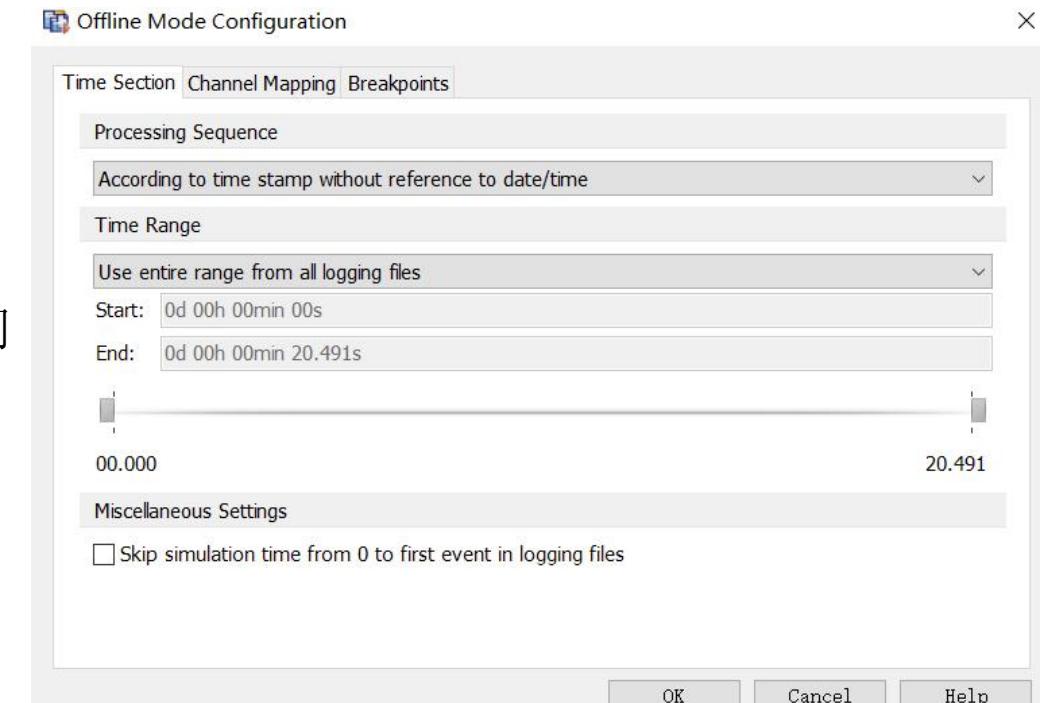
CANoe录制的ASC文件格式：

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



## &gt; 离线模式

文件分析时可以映射文件与CANoe的通道

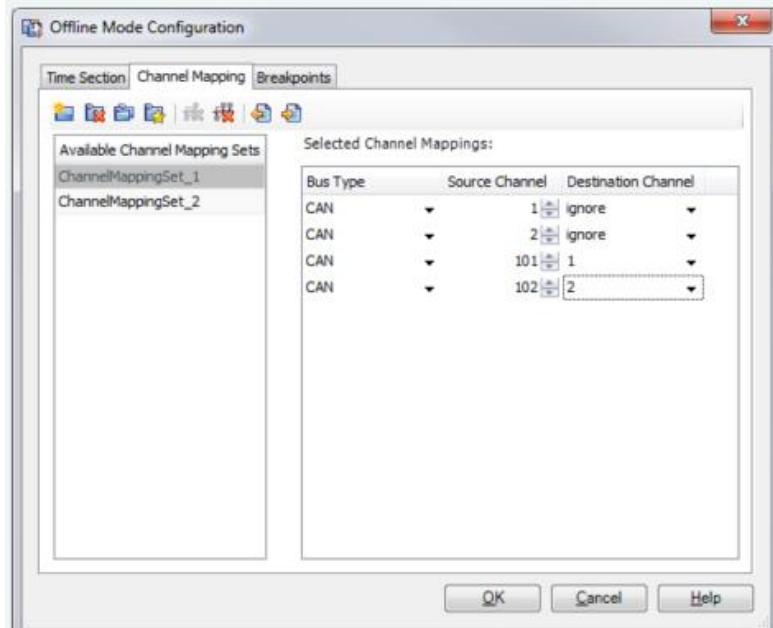
**Example**

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.

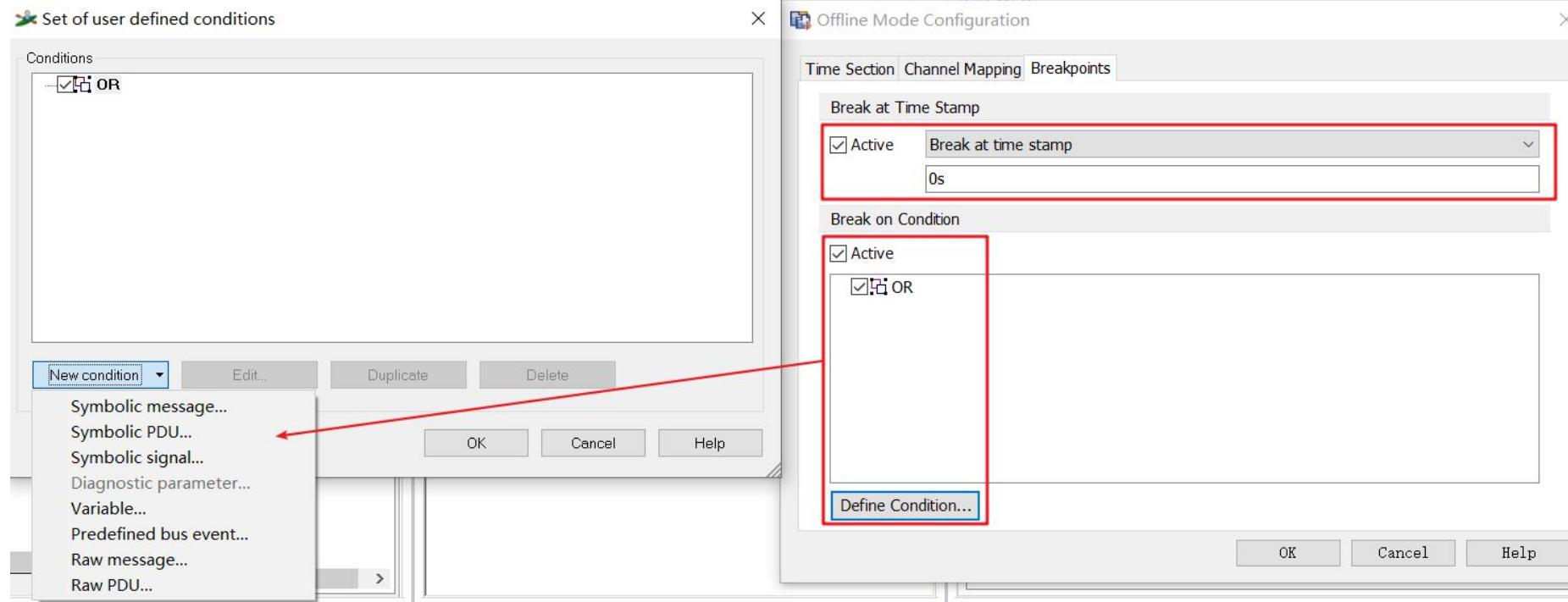
In this case, the dialog looks as follows:



## &gt; 离线模式

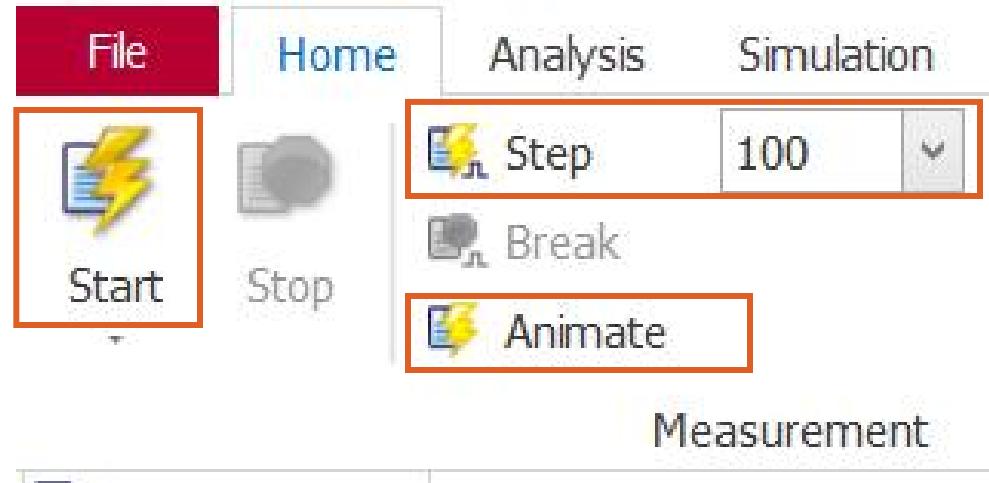
## 断点条件设置：

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。



## &gt; 数据回放

数据回放形式：



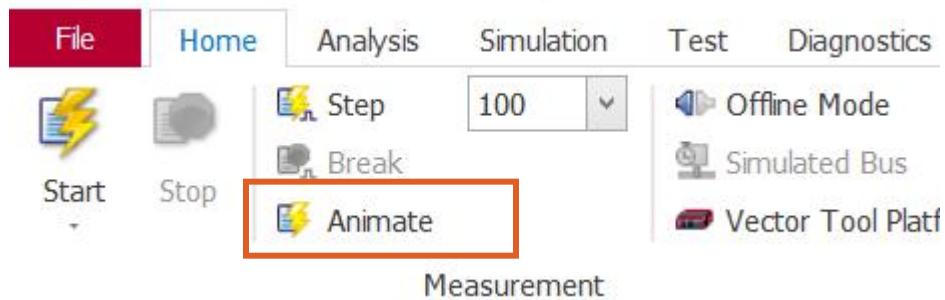
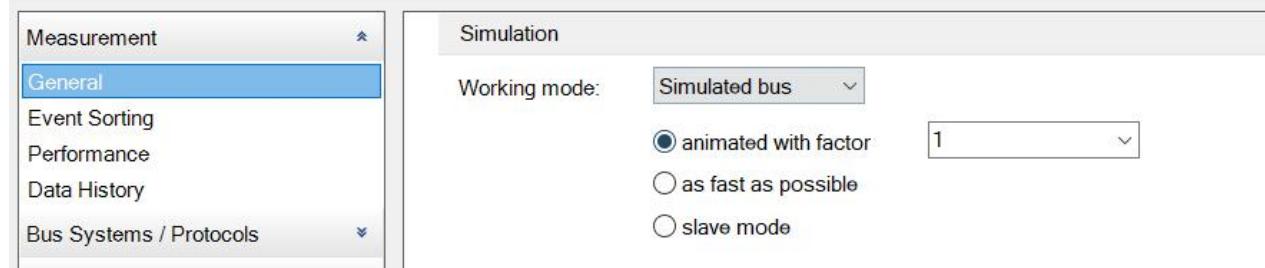
Start 和 Stop：整个文件回放

Step：步进回放，可以设置步进的时间

Animate：事件回放，默认

## &gt; Animate

CANoe Options

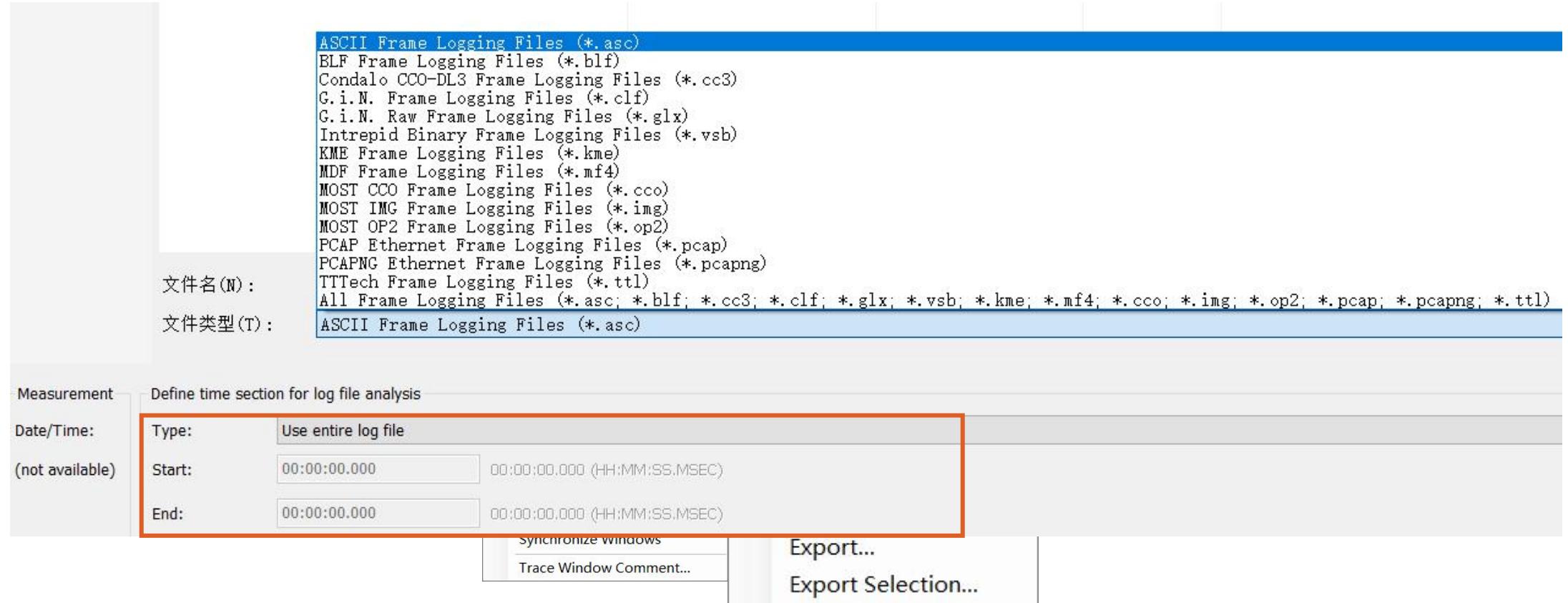


```
CAN.ini - 记事本
[OFFLINE]
AnimationDelay=300

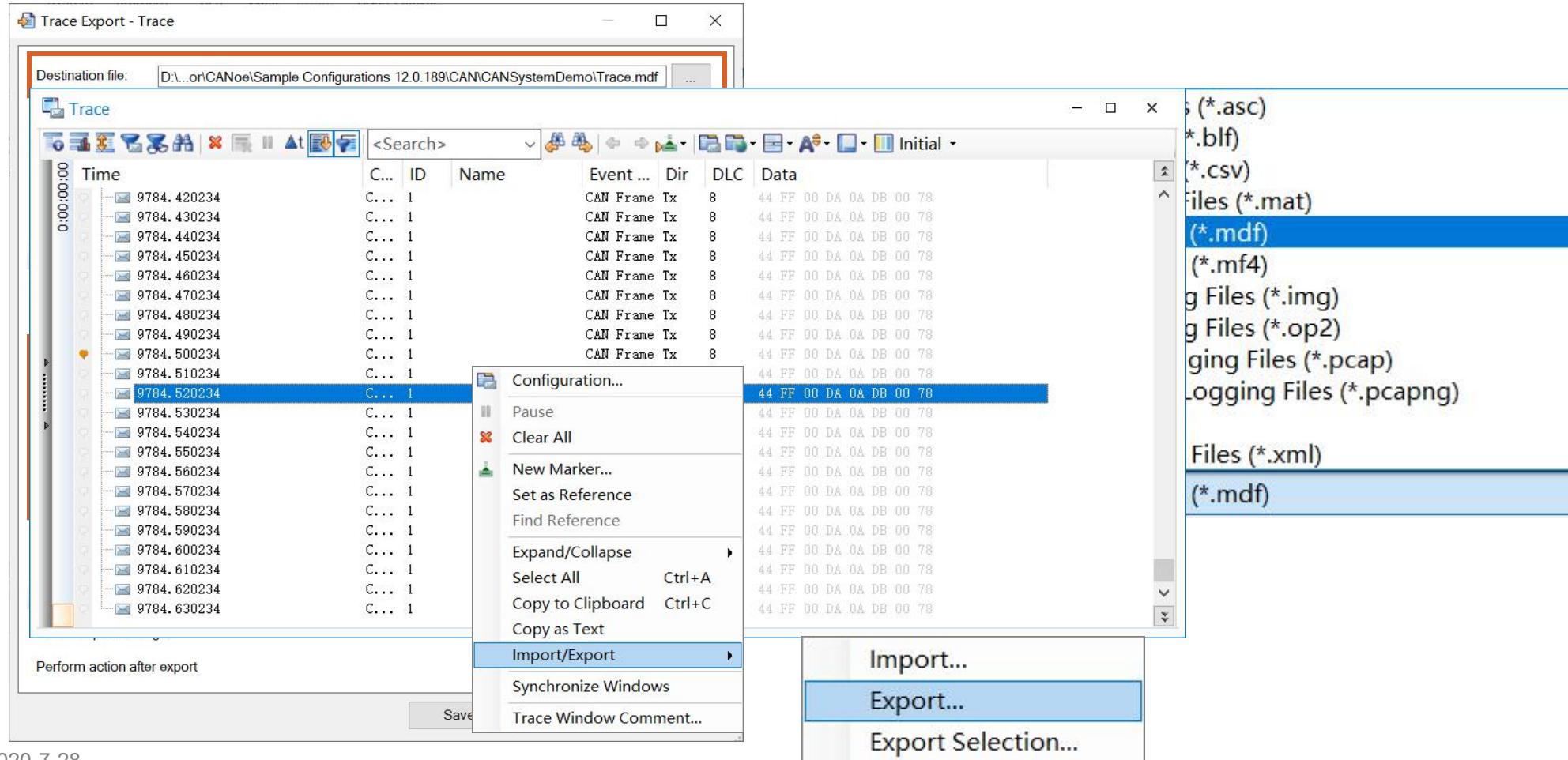
[CANalyzerUserDefinedLayout]

[GUI]
LimitUserObjects=8000
```

## &gt; Trace窗口导入

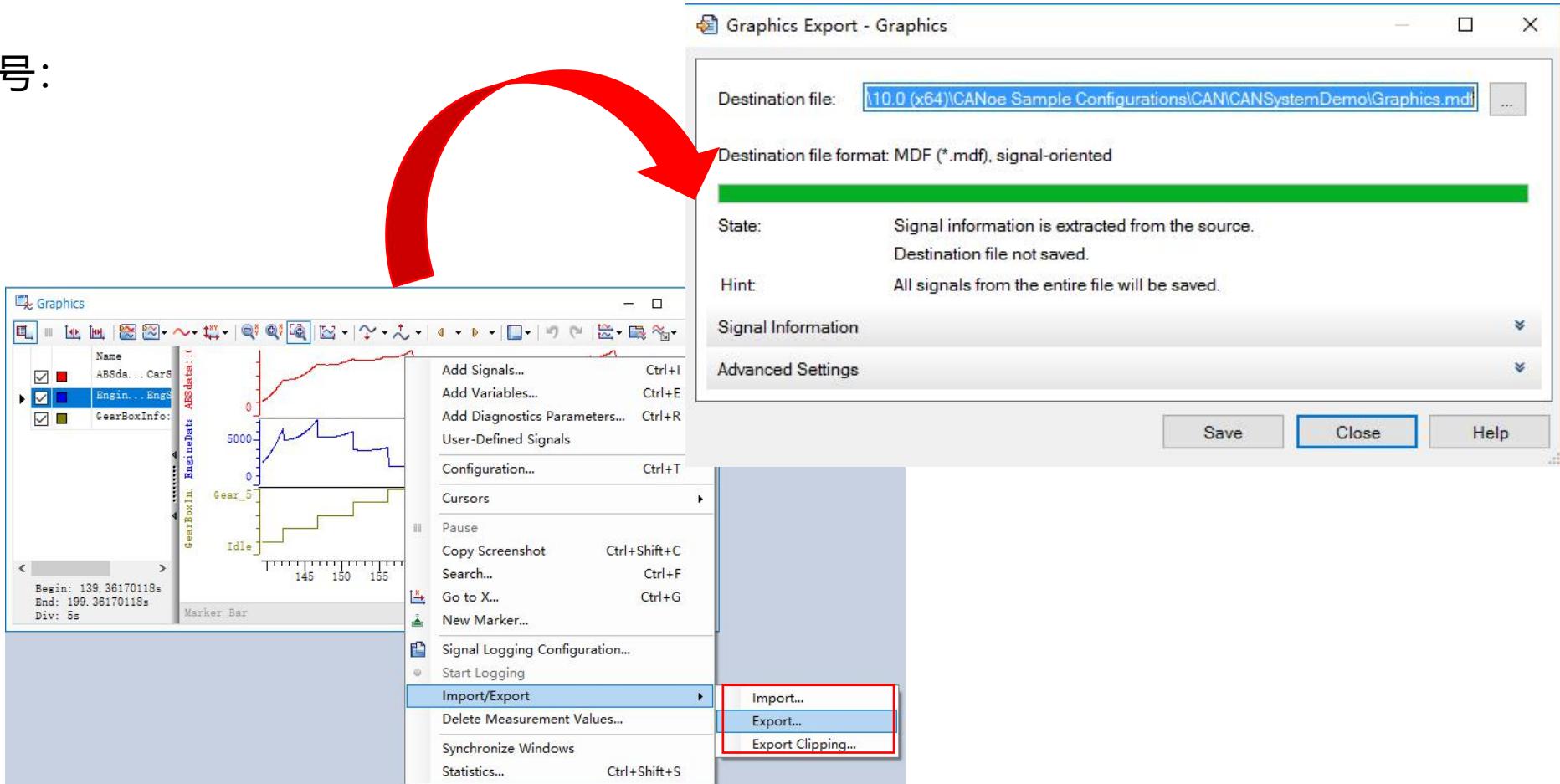


## &gt; Trace窗口导出



## &gt; Graphic 窗口导入导出信号

添加信号:

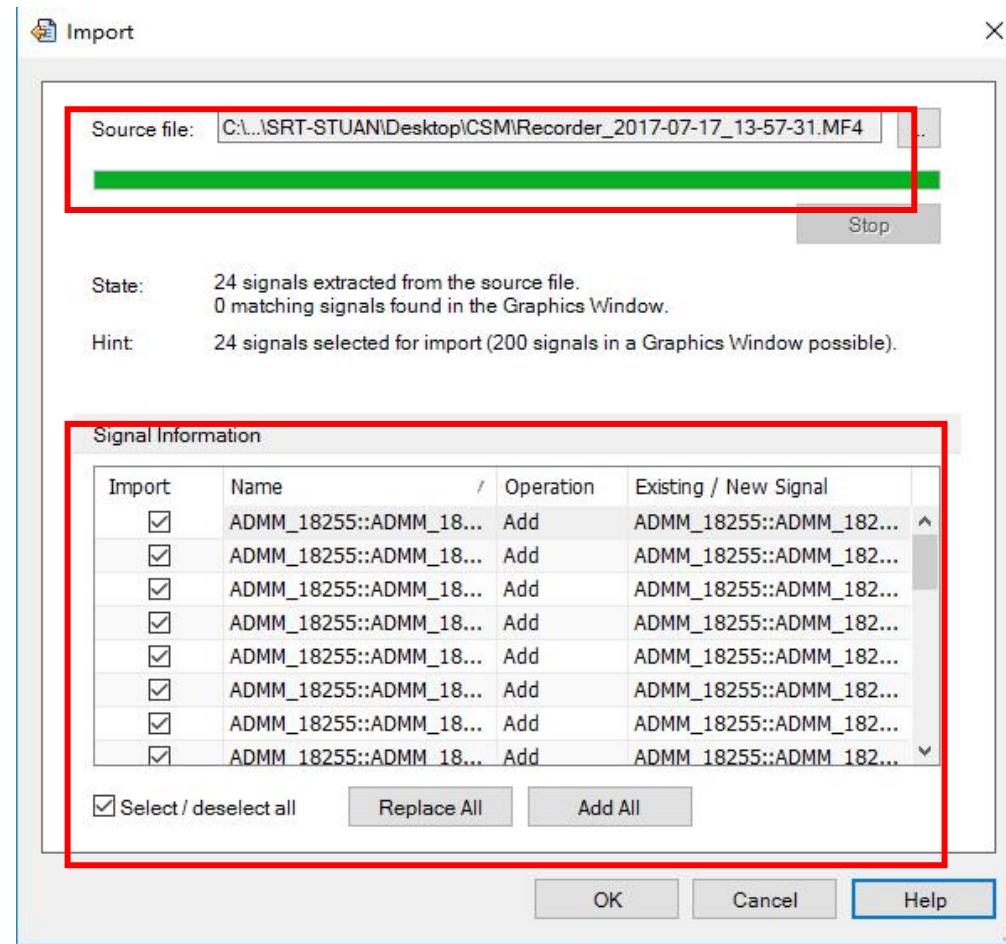


## &gt; Graphic 窗口导入信号

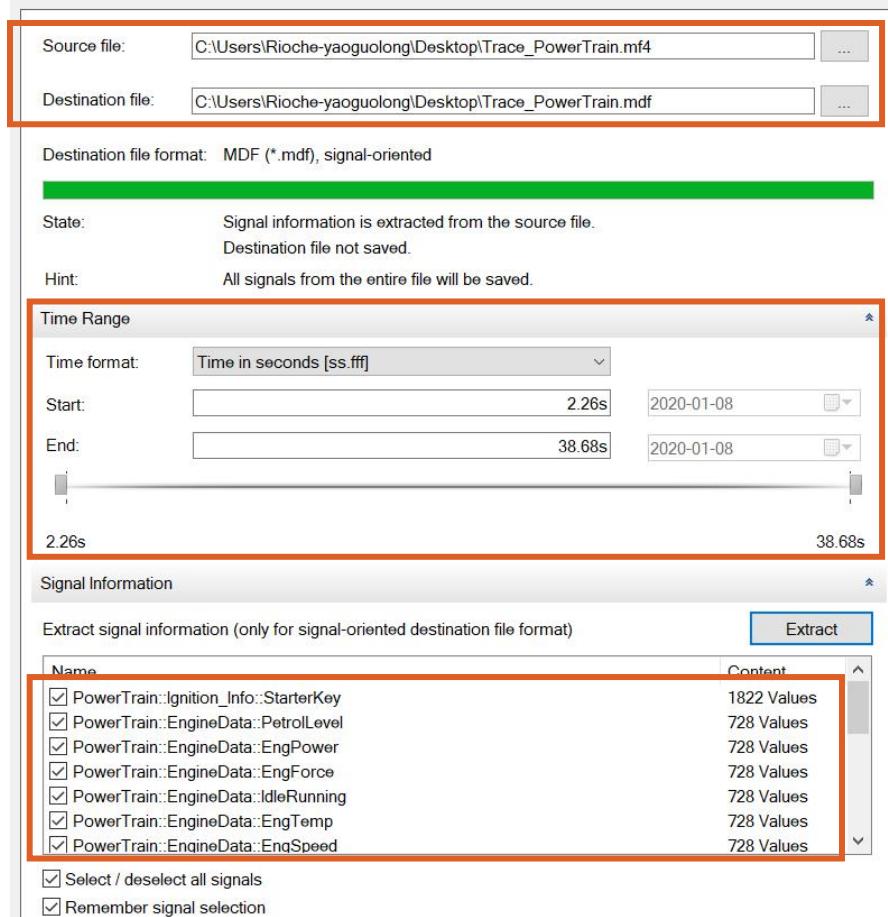
添加信号：  
源文件

文件中信号状态

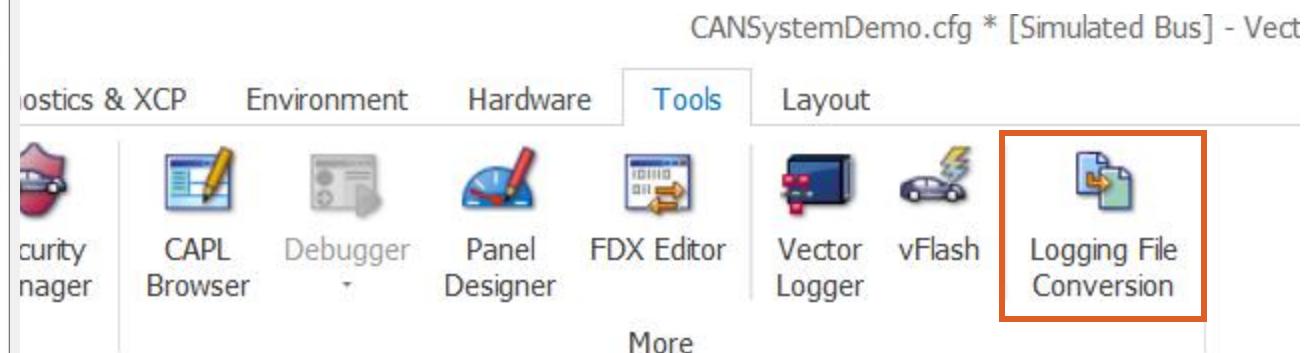
信号列表



## &gt; 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

## CONTENTS



1. 概览
2. 工程创建
3. 报文发送
4. 分析窗口
5. 过滤功能模块
6. 数据记录
7. 离线数据分析
- 8. 系统变量环境变量**
9. Panel设计
10. CAPL语言



> 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



>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

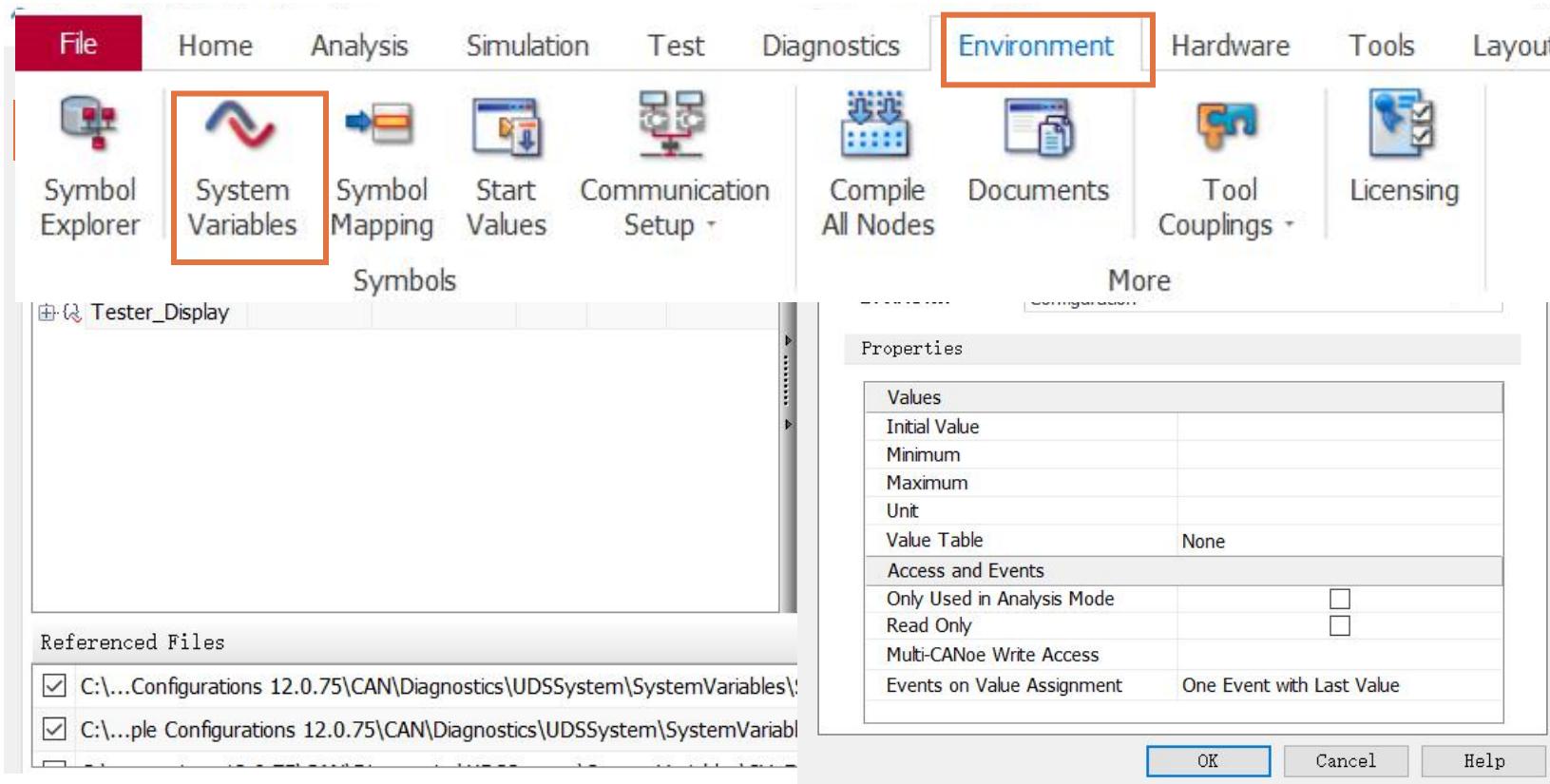
- >独立于总线的信息交互变量
- >存储位置  
CFG/XML (CANoe的工程文件)
- >可导出 XML 文件

## 系统变量创建



## &gt; 系统和环境变量

Environment | System Variables.....



# 系统变量初始值



## > 系统和环境变量

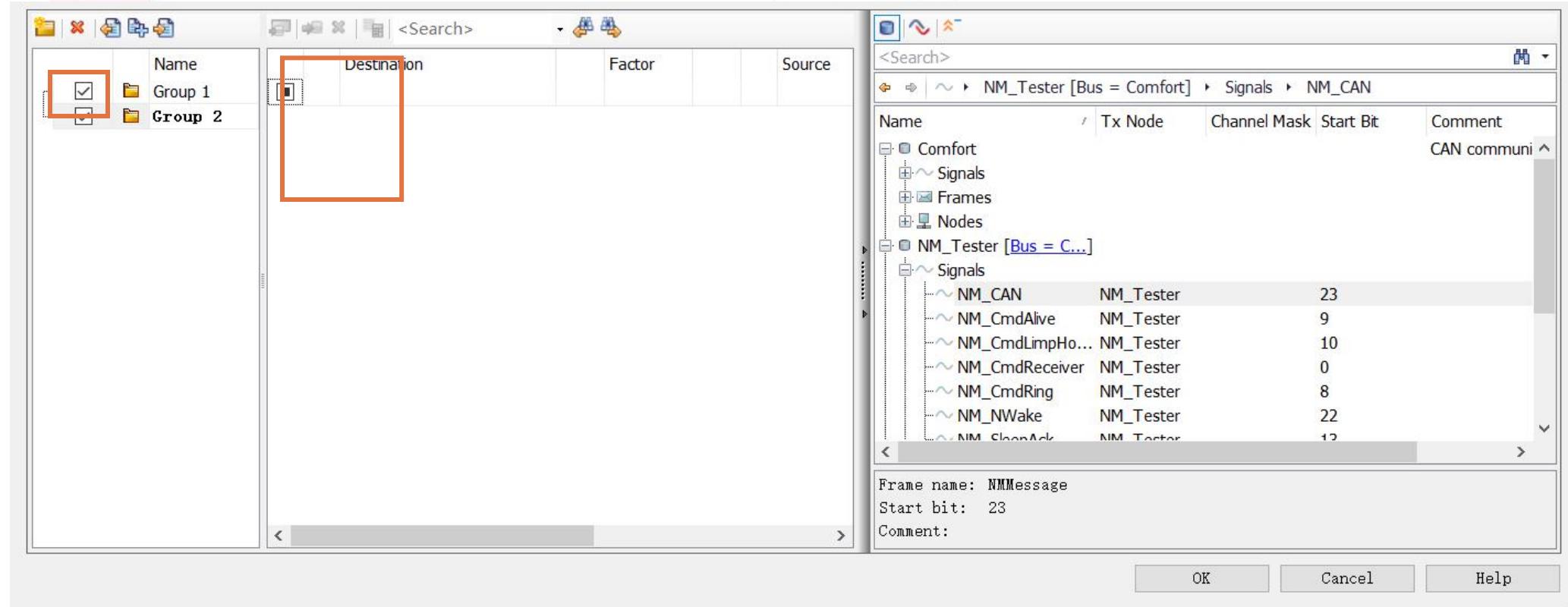
Environment | System Variables.....

The screenshot shows a software interface with a menu bar at the top. The 'Environment' tab is highlighted with a red box. Below the menu, there is a toolbar with several icons, and the 'Start Values' icon is also highlighted with a red box. A dialog box titled 'Start Values' is open, showing a table with one row labeled 'Start values group1'. The table has columns for Variable / Signal, Physical..., and Comment.

	Variable / Signal	Physical...	Comment
<input checked="" type="checkbox"/>	Start values group1		

## &gt; 系统和环境变量

Environment | System Variables.....



The screenshot shows a software interface for managing system variables and CAN signals. On the left, there is a table with columns: Name, Destination, Factor, and Source. The 'Name' column contains entries like 'Group 1' and 'Group 2', with the first one checked. The 'Destination' column is empty. On the right, there is a detailed view of CAN signals for the 'NM\_Tester [Bus = Comfort]' node. The tree view shows 'Comfort', 'NM\_Tester [Bus = C...]', and 'Signals'. The 'Signals' table lists the following data:

Name	Tx Node	Channel Mask	Start Bit	Comment
NM_CAN	NM_Tester		23	CAN commun
NM_CmdAlive	NM_Tester		9	
NM_CmdLimpHo...	NM_Tester		10	
NM_CmdReceiver	NM_Tester		0	
NM_CmdRing	NM_Tester		8	
NM_NWake	NM_Tester		22	
NM_SleepAck	NM_Tester		12	

At the bottom, there is a message box with the following text:

Frame name: NMMessage  
Start bit: 23  
Comment:

Buttons at the bottom right: OK, Cancel, Help.



## &gt; 系统和环境变量

Environment | System Variables.....

Communication Setup    Compile All Nodes    Documents

**Configuration**

- Communication Object Editor...
- Bindings...

**Value Table Templates**

Template	Value	Description
On_Off	0	SD
Active_Inactive	1	eee
NM_AUTOSAR		
NM_OSEK		
Initial_Invalid		
TestModule_States		
<b>Template</b>		

OK Cancel Help

**New System Variable**

Namespace:	Lights
Name:	
Comment:	
Data type:	Integer (32 Bit signed)
Location:	Configuration

**Properties**

Values	
Initial Value	0
Minimum	0
Maximum	1
Unit	
<b>Value Table</b>	Template
Name	Template
Min/Max from the Value table	<input checked="" type="checkbox"/>
<b>Value Table Items</b>	
0	SD
1	eee

Access and Events

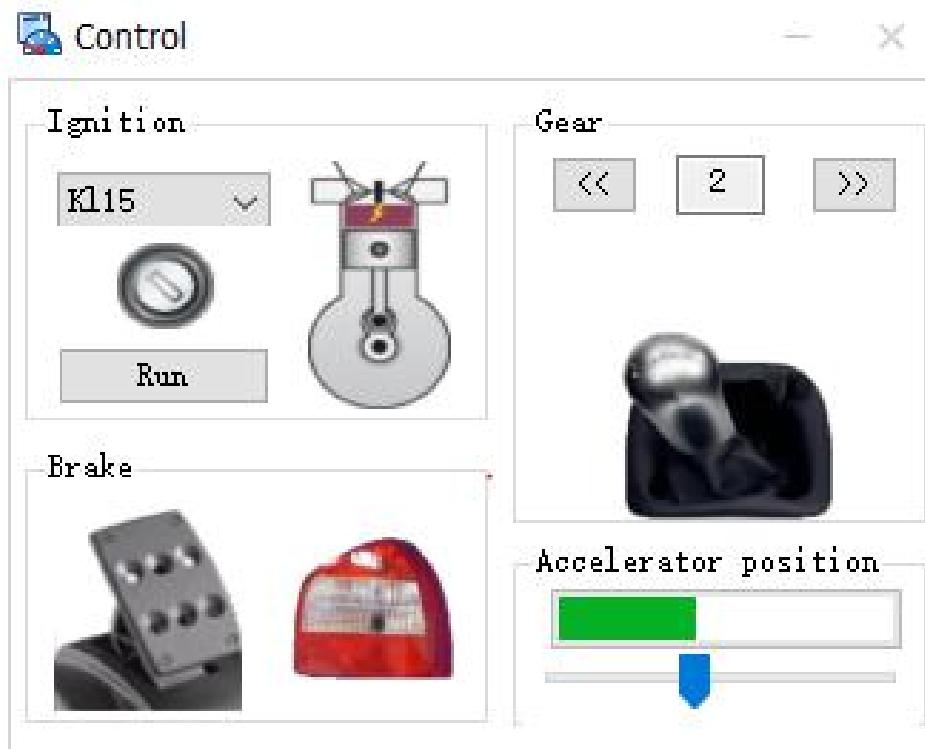
OK Cancel Help

## CONTENTS



1. 概览
2. 工程创建
3. 报文发送
4. 分析窗口
5. 过滤功能模块
6. 数据记录
7. 离线数据分析
8. 系统变量环境变量
9. Panel设计
10. CAPL语言

- > 创建交互界面
- > 显示相关信息
- > 输入控制参数





## 打开Panel Designer





Panel1.xvp - Panel Designer

File Home Panel

Paste Align Controls Bring to Front Send to Back Horizontal - Same Width Horizontally Check

Align Controls Vertical - Same Height Vertically Views

Vertical Both Both Center in Panel More

Symbol Explorer Panel1.xvp

Name

- Comfort
  - Signals
  - Frames
  - Nodes
- NM\_Tester [Bus = Comfort]
  - Signals
  - Frames
  - Nodes
- PowerTrain
  - Signals
  - Frames
  - Nodes

Comment: CAN communicati...

Panel设计区

Properties

Panel Panel Name Panel1

Output Window

Description

Comment: CAN communicati...

Toolbox Properties

Symbol Explorer

控件及属性

The screenshot shows the 'Panel Designer' application window titled 'Panel1.xvp'. The interface includes a top ribbon with File, Home, and Panel tabs, and various toolbars for editing and arranging controls. A central workspace is labeled 'Panel设计区' (Panel Design Area) and contains a large empty rectangular frame. To the left is the 'Symbol Explorer' pane, which lists hierarchical symbols under categories like Comfort, NM\_Tester, and PowerTrain. To the right is the 'Properties' pane, which shows the current panel is named 'Panel1'. The bottom of the window features an 'Output Window' and a 'Comment' field. Red boxes highlight the 'Symbol Explorer' and the 'Properties' pane.



### 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

# Panels 设计

## Panel设计



### > 控件

The screenshot shows the Panel Designer software interface with the following components:

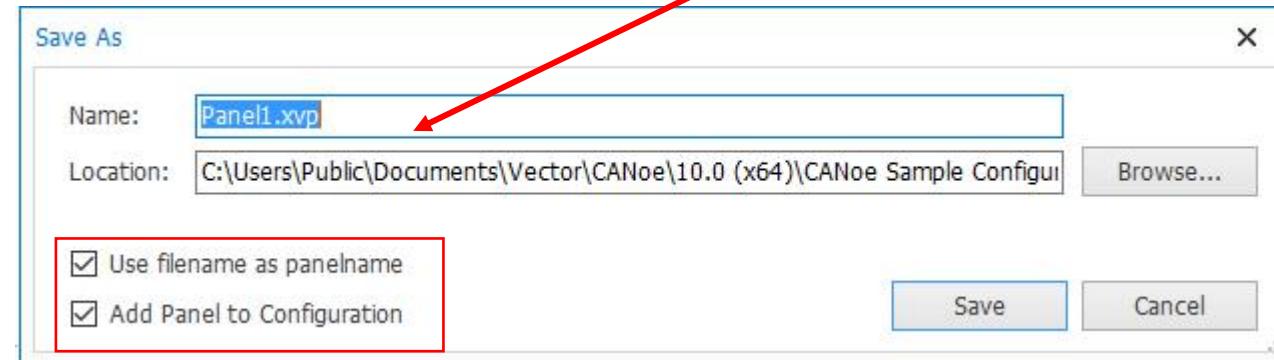
- Toolbox:** On the left, under "Vector Standard Controls", the "LCD Control" icon is highlighted with a red box.
- Symbol Explorer:** In the center-left, it shows a tree structure of signals. A specific entry, "Data3", is highlighted with a red box.
- Panel1.xvp:** The main workspace shows a digital display component with the value "00000.0".
- Properties:** On the right, the properties panel is open for the selected "LCD Control 1". It includes sections for Appearance, General, Layout, Number Of Digits, Symbol, and a bottom section with buttons for "Attach Signal...", "Attach Variable...", and "Attach Diagnostic Parameter...".

Red arrows point from the highlighted "LCD Control" in the Toolbox to the "Data3" entry in the Symbol Explorer, and from the "Data3" entry in the Symbol Explorer to the digital display component in the workspace.

> 保存格式\*. vxp

> 制作好的Panel可以在不同工程中复用

报文格式 \*. vxp

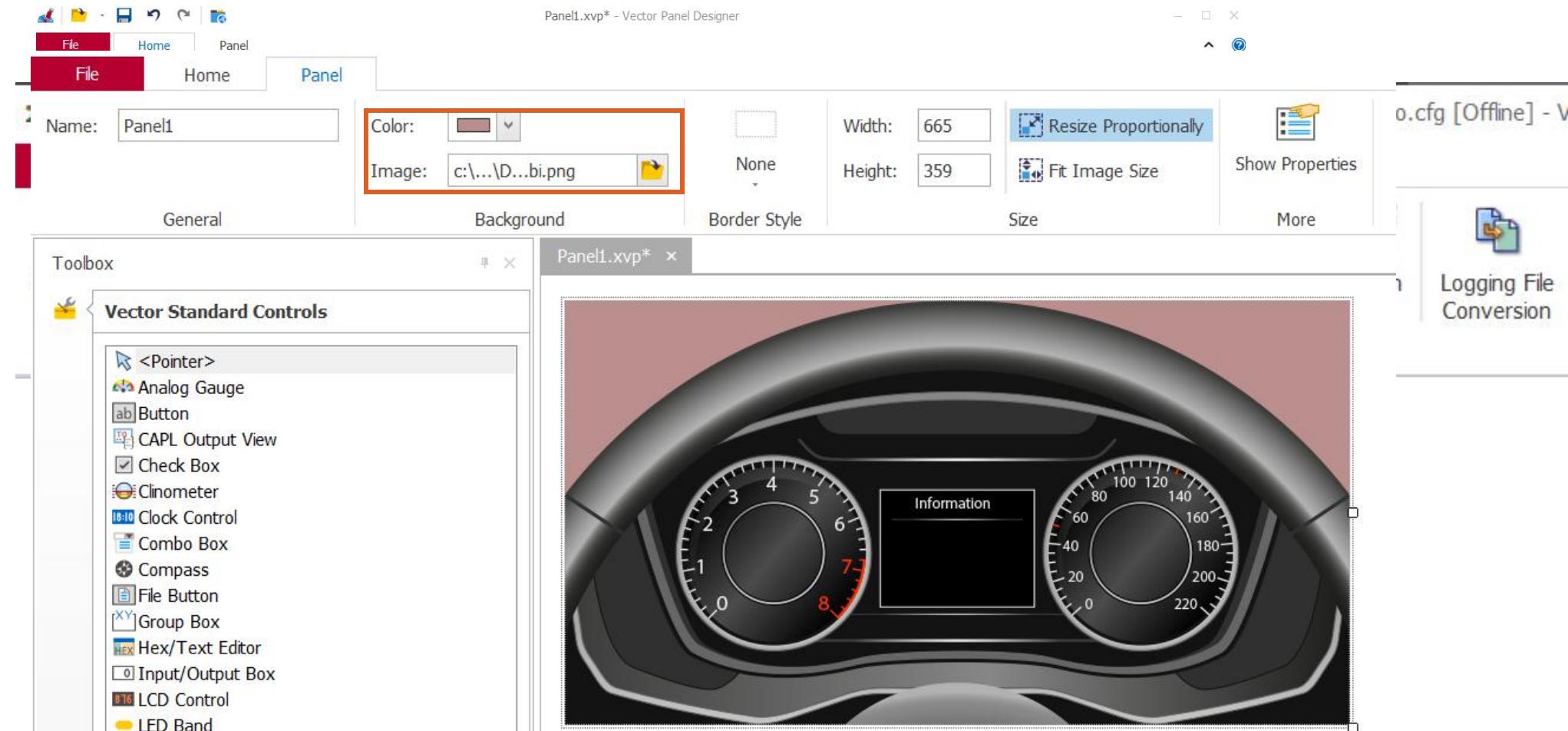


# Panels 设计

## Panel设计



### > 工具

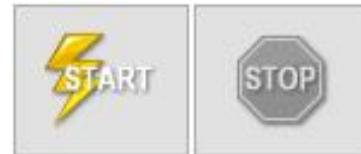


### > 控件简介

分类：显示与控制

Signalname:

输入输出



CANoe 开始/停止



状态显示



动作触发

## 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

Signalname:

- ◆ 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

Signalname:

- ◆ 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

## Panel 控件

Progress bar



- ◆ 在一定范围内显示进度

Clock Control



- ◆ 显示时钟， 默认为PC机时间

Track bar



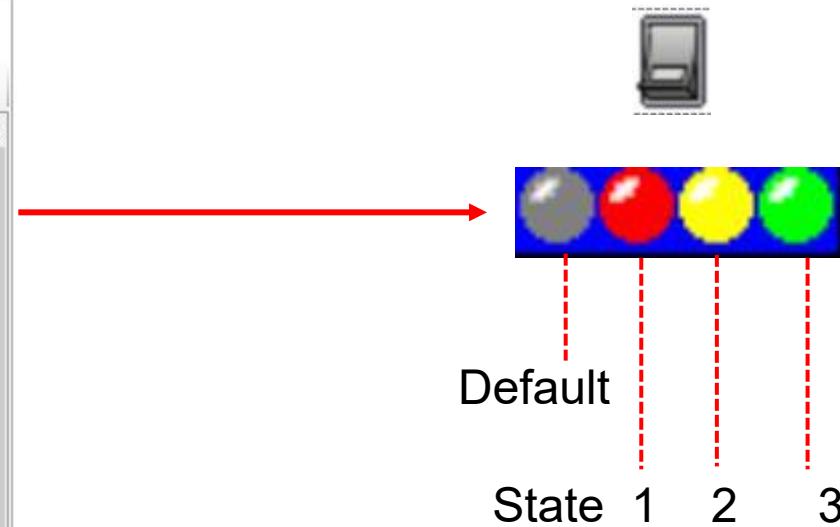
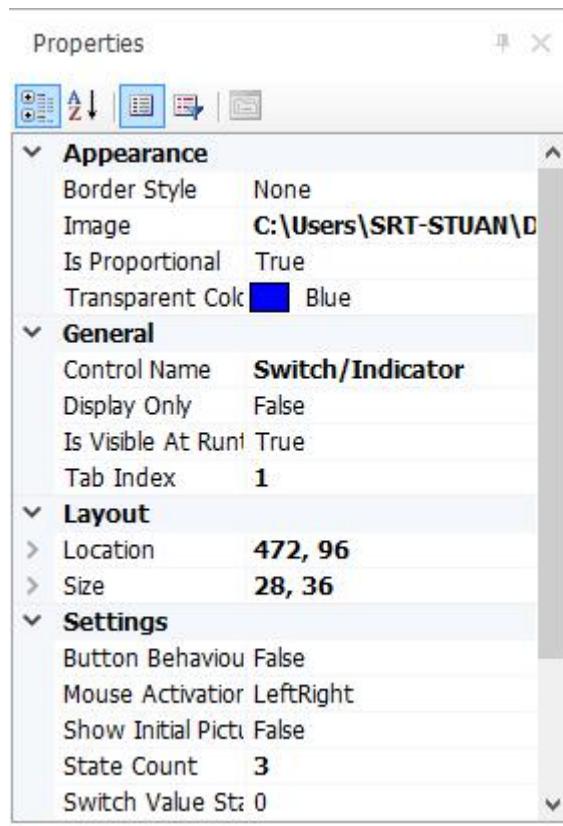
- ◆ 在一定范围内控制和显示元素

LCD Control



- ◆ 显示浮点型数字

## 位图设计控件显示界面



## Panel CAPL 接口函数

- SetControlBackColor
- SetControlForeColor
- SetControlColors
- SetDefaultControlColors
- [setControlVisibility](#)
- [openPanel](#)
- [closePanel](#)
- [enableControl](#)

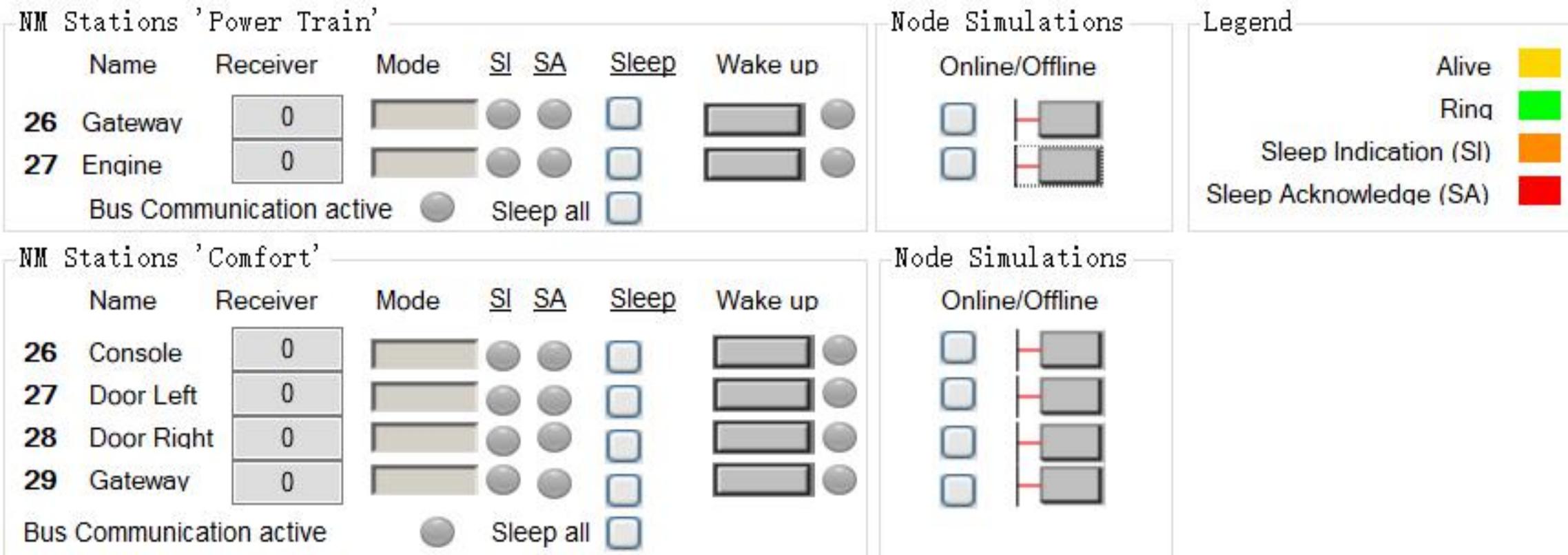
输出参数到控件中

[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。

## &gt; 概览



## CONTENTS



1. 概览
2. 工程创建
3. 报文发送
4. 分析窗口
5. 过滤功能模块
6. 数据记录
7. 离线数据分析
8. 系统变量环境变量
9. Panel设计
10. CAPL语言



## > 概述

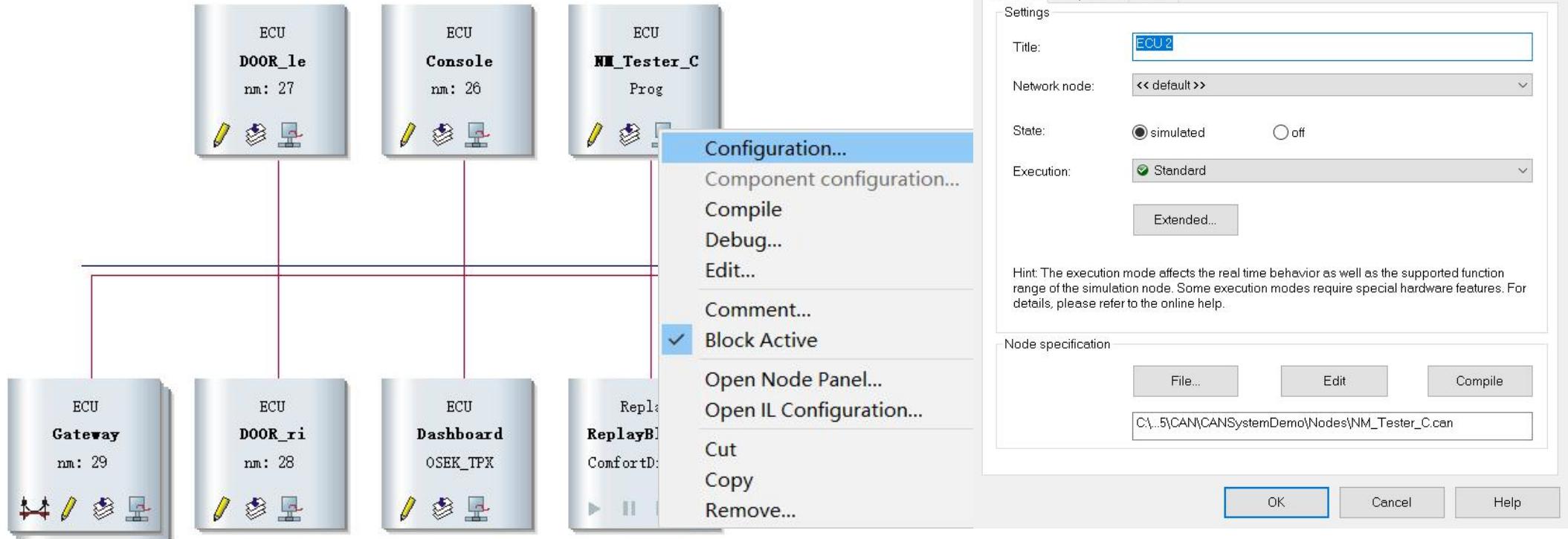
### 特征

- 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

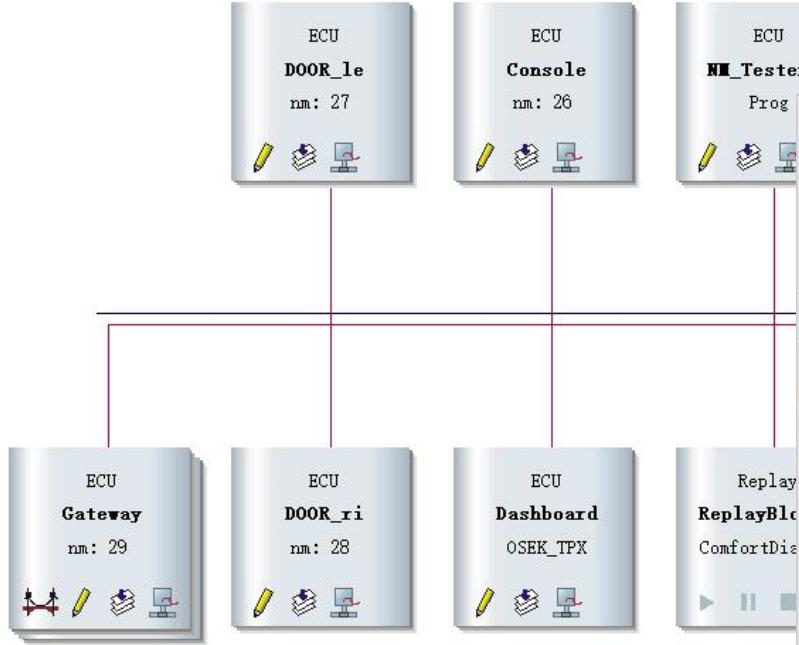
- 1, 节点仿真
- 2, 网络仿真
- 3, 节点测试

## &gt; 配置





## &gt; 配置



Context menu options for the NM\_Tester\_C ECU:

- Config
- Comp...
- Comp...
- Debug...
- Edit... (highlighted)
- Comm...
- Block...
- Open
- Open
- Cut
- Copy
- Remove

Vector CAPL Browser

**NM\_Tester.C.can X**

```

File Home Filter Debug Layout
NM_Tester.C.can X
Includes
Variables
System
Value Objects
CAN
Diagnostics
Functions
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 mnNMMessage;
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         mnNMMessage = (message NMMessage)this;
66         gNMReceiver[iSgNr] = mnNMMessage.NM_CmdReceiver;
67     }
}

```

CAPL Functions

Enter a search term

Event Handlers

System Functions

User Defined Groups

NM\_Tester.C.can

J1939TestServiceLibraryNL.dll

Event Handlers

CAPL Functions Symbols

Ln 1 Col 1 INS



## &gt; CAPL编辑器

**事件列表**

**全局变量**

**对象与函数**

**事件步骤**  
本地变量也在此定义

**消息窗口**

```

1 /*@!Encoding:936*/
2 includes
3 {
4
5 }
6
7 variables
8 {
9     timer cycletime;
10 }
11
12 on start
13 {
14     write("CANoe Start");
15     setTimer(cycletime,1);
16     $FlashLight=0;
17 }
18 on timer cycletime
19 {
20     $FlashLight++;
21     setTimer(cycletime,1);
22     if($FlashLight>2)
23     {
24         $FlashLight=0;
25     }
26 }

```



## &gt; 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  
}
```



## &gt; CAPL 数据类型

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



## &gt; CAPL条件和循环

<b>Conditions</b>	<b>if</b>	<pre><b>if</b>      (a == 100) write ("Value is 100"); <b>else if</b> (a == 500) write ("Value is 500"); <b>else</b>           write ("Value is not 100 or 500");</pre>
	<b>switch</b>	<pre><b>switch</b> (a)</pre>
		<pre>{</pre>
		<pre>    <b>case</b> 100: write ("Value is 100"); <b>break</b>;</pre>
		<pre>    <b>case</b> 500: write ("Value is 500"); <b>break</b>;</pre>
		<pre>    <b>default</b>: write ("Value is not 100 or 500"); <b>break</b>;</pre>
		<pre>}</pre>
<b>Loops</b>	<b>for</b>	<pre><b>for</b> (a = 0; a &lt; 100; a++)</pre>
		<pre>{</pre>
		<pre>    write ("Value of %d", a);</pre>
		<pre>}</pre>
	<b>while</b>	<pre><b>while</b> (a &lt; 100)</pre>
		<pre>{</pre>
		<pre>    write ("Value of %d", a);</pre>
		<pre>    a++;</pre>
		<pre>}</pre>
	<b>do</b>	<pre><b>do</b></pre>
		<pre>{</pre>
	<b>do while</b>	<pre>    write ("Value of %d", a);</pre>
		<pre>    a++;</pre>
		<pre>} <b>while</b> (a &lt; 100)</pre>



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WarningLimt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

## &gt; 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
```



## &gt; 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

&gt;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

&gt;定时器事件

➤on timer myTimer //对myTimer 设定的时间到反应



## > 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 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 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 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 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 CAN Selectors

Type	Description	Value range
<b>BYTE(n)</b> <b>WORD(n)</b> <b>DWORD(n)</b> <b>QWORD(n)</b>	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
<b>ID</b>	Message identifier	0...0x7FF (0...2047), 0...0xFFFFFFFF (0...536870911)
<b>DLC</b>	Message length	0...8
<b>CAN</b>	Channel number	1...32
<b>DIR</b>	Direction of transmission	RX(0), TX(1), TXREQUEST(2)
<b>TIME</b>	Time stamp	Resolution 10 microseconds (CANcardXL)
<b>RTR</b>	Selector for distinguishing between Data and Remote frames	RTR = 0 → Data frame RTR = 1 → Remote frame
<b>MSGFLAGS</b>	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
<b>TYPE</b>	For more efficient evaluation → a combination of RTR and DIR	See Notes page



## &gt; 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");  
}
```

## ■ 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
}
```



## ■ PG

- pg TC1 pg\_tc1 = {  
    SA = 0x34,  
    DA = 0x24  
};  
output(pg\_tc1);



## ■ 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 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 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 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。



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);
```



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.



## Sending LIN messages

Output(linFrame msg)  
example

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



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);
```



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);  
}
```

## &gt; 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 message 101 {  
 float a = 0;  
 a = this.Signal1.phys;  
}`
- ✓ `on envVar Switch {  
 int val;  
 val = getvalue(this);  
 ...  
}`



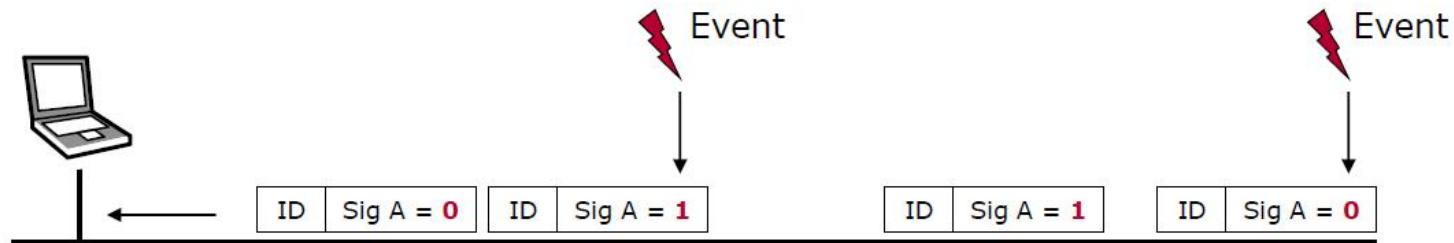
## > CAPL on envVar

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。

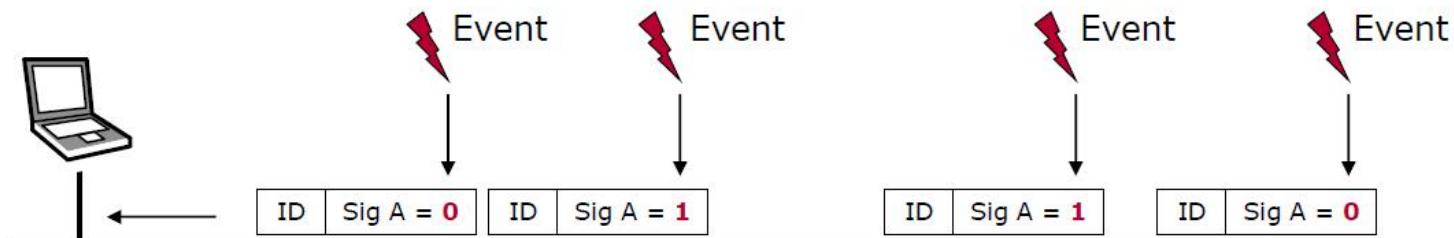
```
on envvar Switch {  
    message Controller msg; // 声明传输的CAN报文  
    msg.Stop = getvalue(this); // 读switch的值， // 将值赋给信号Stop  
    output(msg); // 输出报文到总线上  
}
```

## &gt; CAPL on signal on signal update

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



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



## &gt; 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

