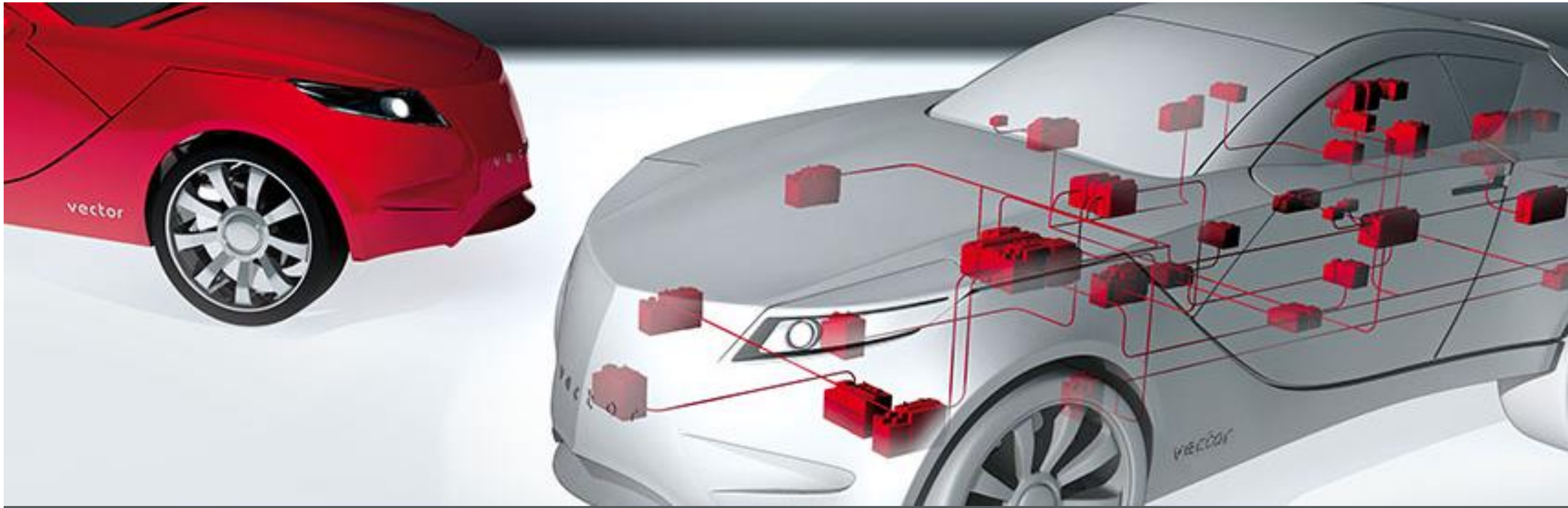
VECTOR

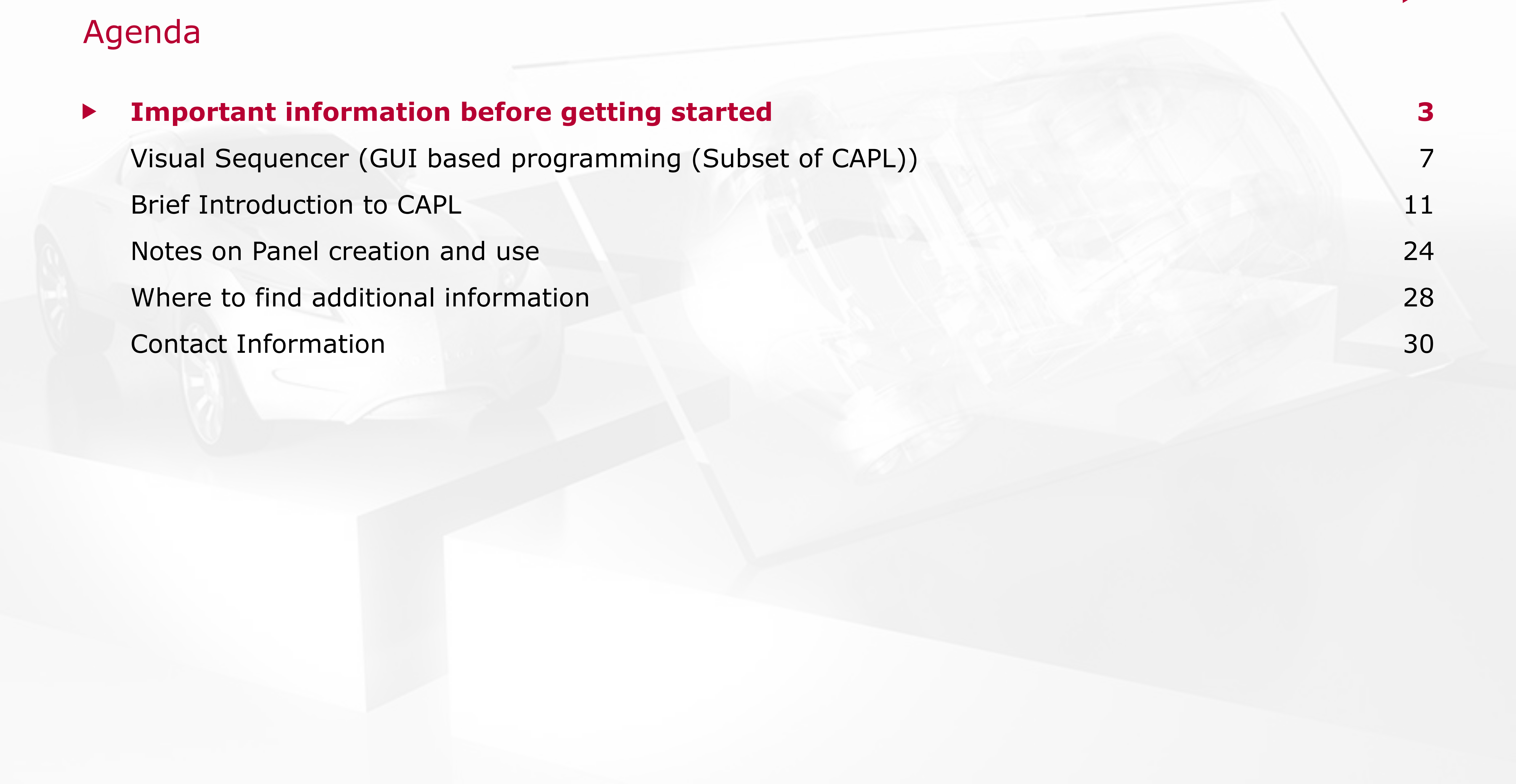


CAPL Scripting Quickstart

CAPL (Communication Access Programming Language) For CANalyzer and CANoe

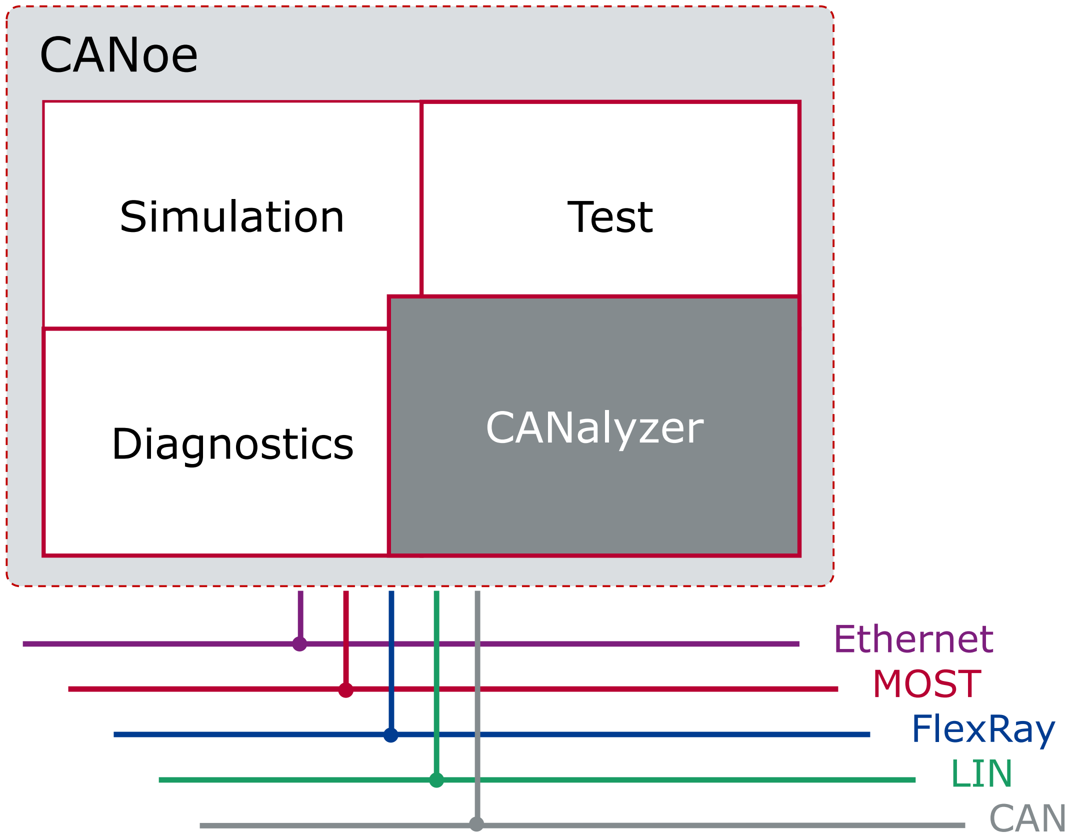
V1.01 | 2015-12-03

Agenda



CANalyzer versus CANoe

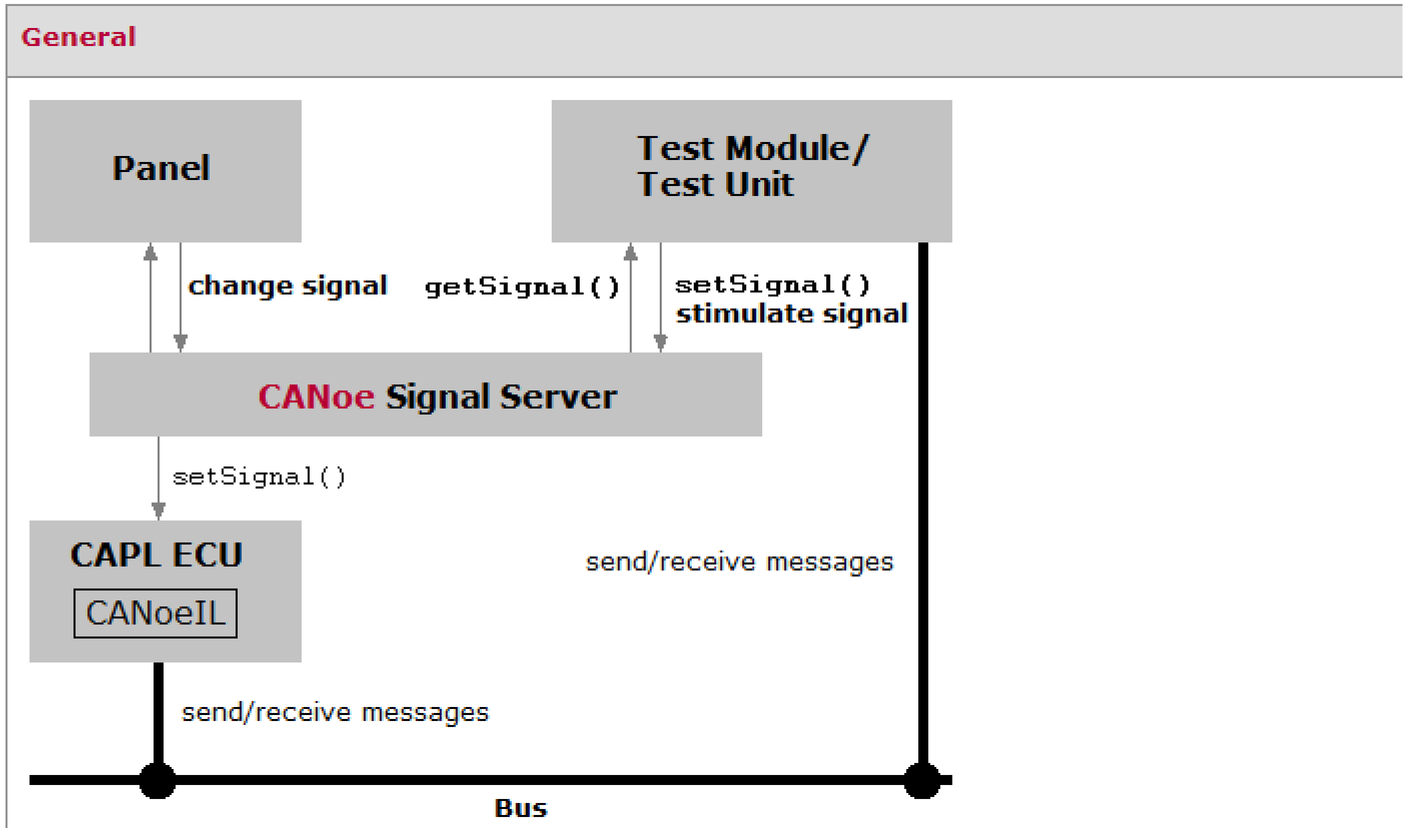
CANalyzer is wholly contained within CANoe



CANoe offers significant additional capability beyond CANalyzer to:

> Stimulate the network(s) with **Interaction Layer** knowledge
> Run automated tests and generate test reports
> Implement automated diagnostic tests

CANoe and the added value of the Interaction Layer



- The CANoe Interaction Layer (in short CANoeIL):
  - Provides a signal-oriented means of accessing the bus
  - Map signals to their appropriate send messages
  - Controls the sending of these messages as a function of the (OEM) Send Model
- Transmission of messages and signals is described based on attributes in the database
- CANoeIL models the transmission behavior at run-time using those attributes

Overview of CANalyzer variants

CAPL is available in CANalyzer PRO and all versions of CANoe

CANalyzer is available in three different variants:

- PRO: Professional variant: full functionality
- EXP: Expert variant: supports all applications up to complex analysis of heterogeneous systems; does not support CAPL programs
- FUN: Fundamental variant: simple applications, does not support CAPL, diagnostic tester and panels

Detailed information about the variants of CANalyzer is available at our website: *http://www.vector.com/vi_canalyzer_variants_en.html*

Agenda

General

Available in both CANalyzer PRO and EXP

- Intended to allow some automation within the EXP variant

The Visual Sequencer allows you to create *automated command sequences* with the purpose of

- Stimulating the network
- Controlling applications

In order to *structure* the individual steps, loops and conditional command blocks can be used, such as

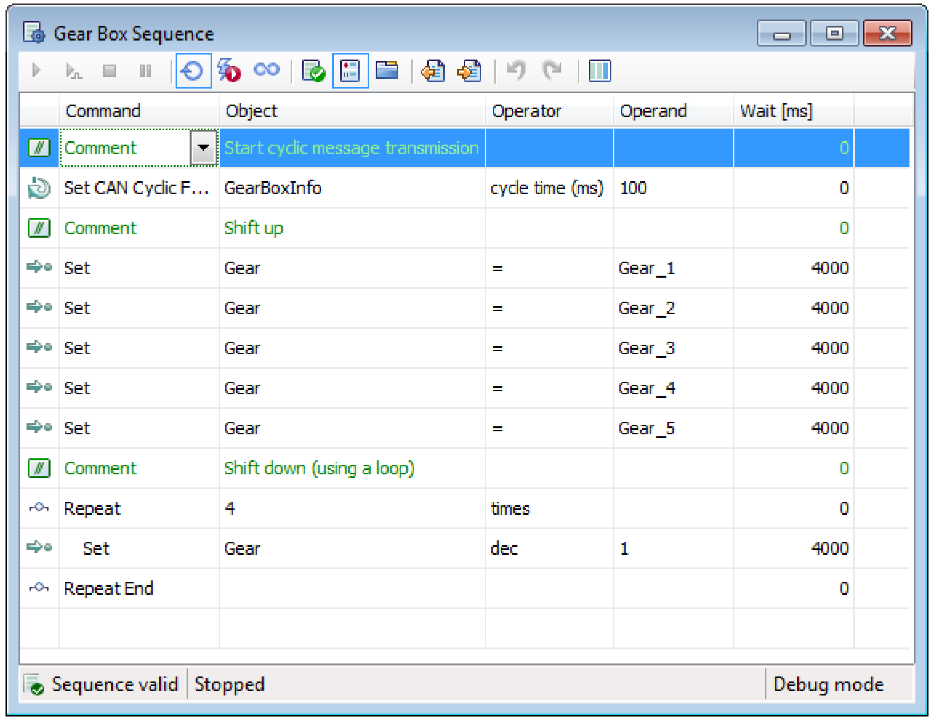
- **`if`**, **`else if`**, **`end if`**

Each sequence is shown in a *separate window*, and can be edited at any time, even while a measurement is running.

Visual Sequencer (GUI based programming (Subset of CAPL))

Features

- Send messages (cyclically)
- Set signals/variables
- If, else, else if and repeat commands
- Wait commands
- Start/stop replay
- Write text or values to write window or file
- Graphical debug
- Auto complete for names

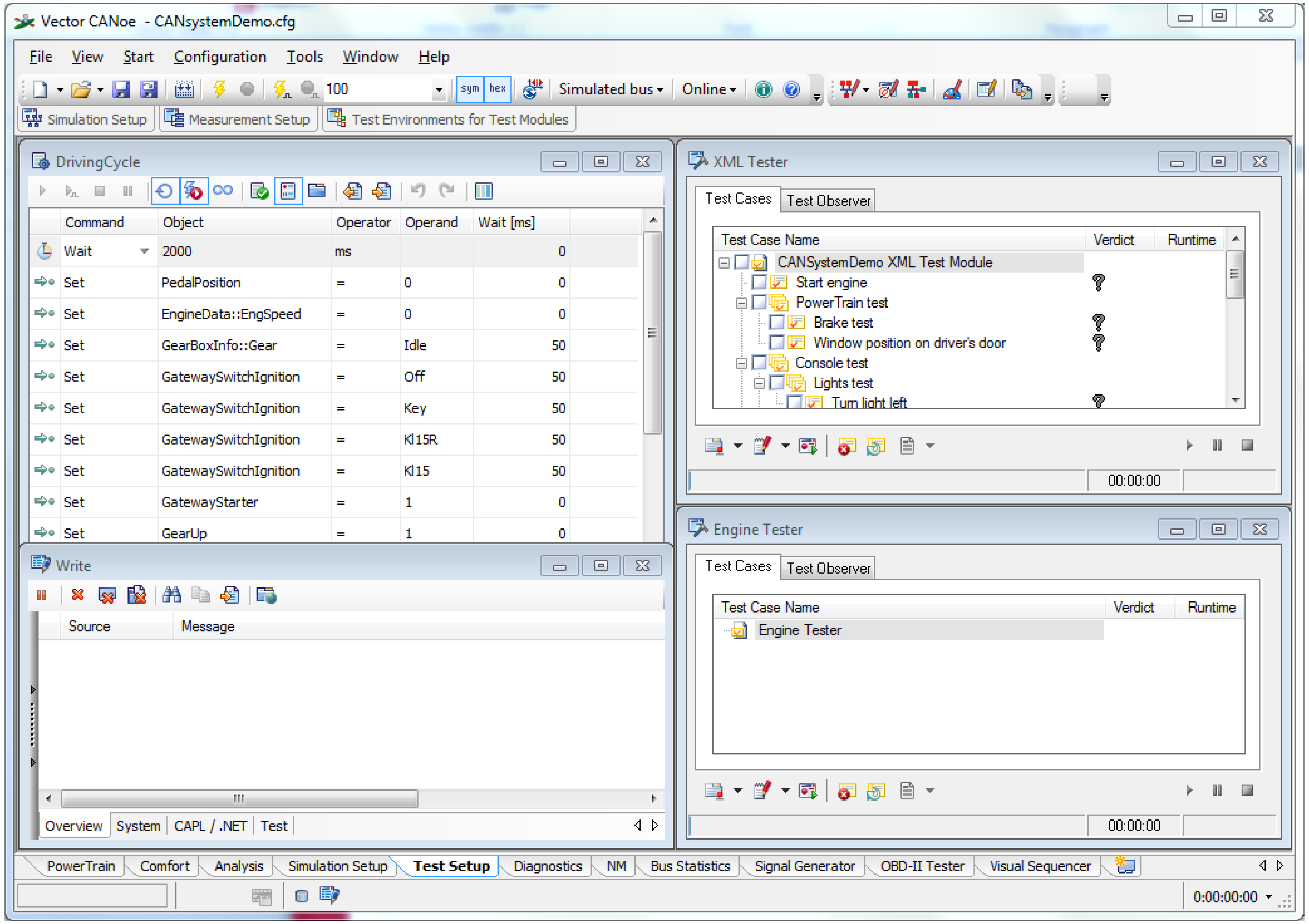


Command	Object	Operator	Operand	Wait [ms]
Comment	Start cyclic message transmission			0
Set CAN Cyclic F…	GearBoxInfo	cycle time (ms)	100	0
Comment	Shift up			0
Set	Gear	=	Gear_1	4000
Set	Gear	=	Gear_2	4000
Set	Gear	=	Gear_3	4000
Set	Gear	=	Gear_4	4000
Set	Gear	=	Gear_5	4000
Comment	Shift down (using a loop)			0
Repeat	4	times		0
Set	Gear	dec	1	4000
Repeat End				0

Sequence valid | Stopped | Debug mode

Visual Sequencer (GUI based programming (Subset of CAPL))

See the CANsystemdemo.cfg included with your installation



Vector CANoe - CANsystemDemo.cfg
File View Start Configuration Tools Window Help
100
sym hex
Simulated bus
Online
Simulation Setup
Measurement Setup
Test Environments for Test Modules
DrivingCycle
XML Tester
Test Cases
Test Observer
00:00:00
Engine Tester
Write
Source
Message
Overview System CAPL / .NET Test
PowerTrain Comfort Analysis Simulation Setup Test Setup Diagnostics NM Bus Statistics Signal Generator OBD-II Tester Visual Sequencer
0:00:00:00

DrivingCycle

Command	Object	Operator	Operand	Wait [ms]
Wait	2000	ms		0
Set	PedalPosition	=	0	0
Set	EngineData::EngSpeed	=	0	0
Set	GearBoxInfo::Gear	=	Idle	50
Set	GatewaySwitchIgnition	=	Off	50
Set	GatewaySwitchIgnition	=	Key	50
Set	GatewaySwitchIgnition	=	Kl15R	50
Set	GatewaySwitchIgnition	=	Kl15	50
Set	GatewayStarter	=	1	0
Set	GearUp	=	1	0

XML Tester

Test Case Name	Verdict	Runtime
CANSystemDemo XML Test Module		
Start engine	?	
PowerTrain test		
Brake test	?	
Window position on driver's door	?	
Console test		
Lights test		
Turn light left	?	

Engine Tester

Test Case Name	Verdict	Runtime
Engine Tester		

Agenda

General

Functional blocks based on CAPL (Communication Access Programming Language) can be created to program

- Network node modules
- Special evaluation programs for individual applications

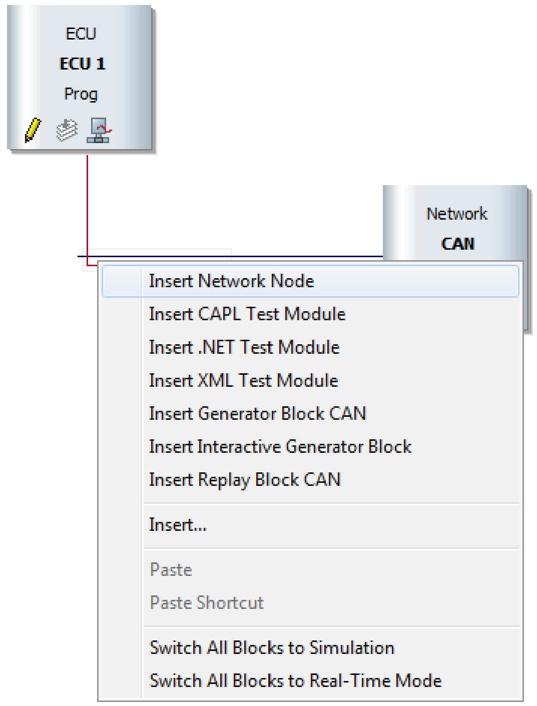
Some CAPL characteristics:

- C-like programming language
- **Event based**, not interrupt driven
- CAPL programs are created using an integrated development environment called the CAPL Browser
- Direct access to signals, system variables and diagnostic parameters
- Able to link user created DLLs

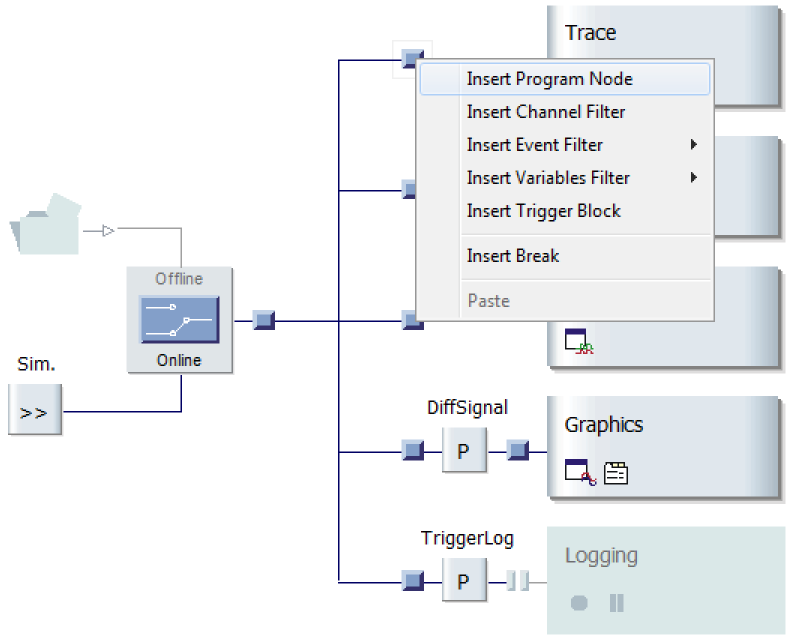
Field of Application CANoe

- Creating and extending simulations
- Implementing functions for analysis in the measurement setup

Simulation Setup



Measurement Setup

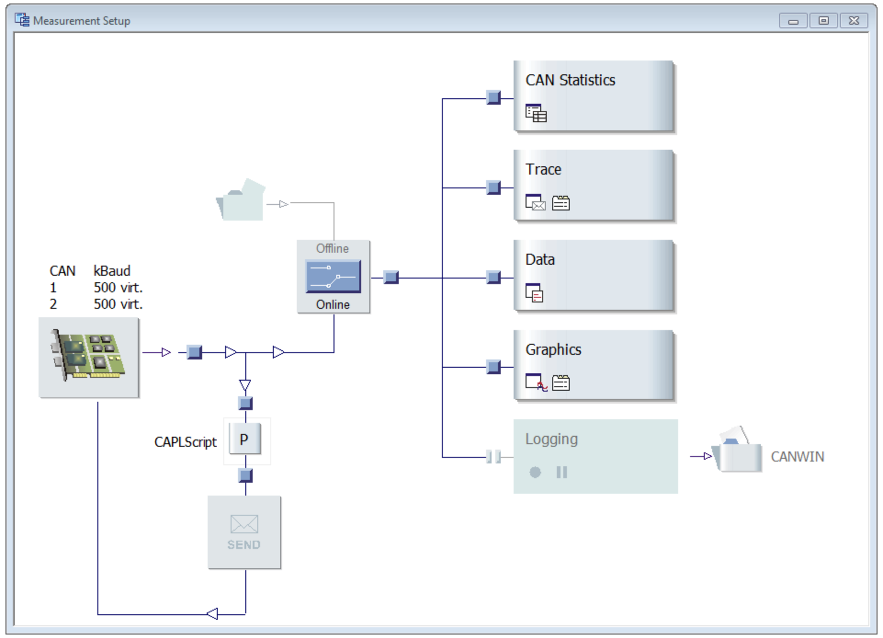


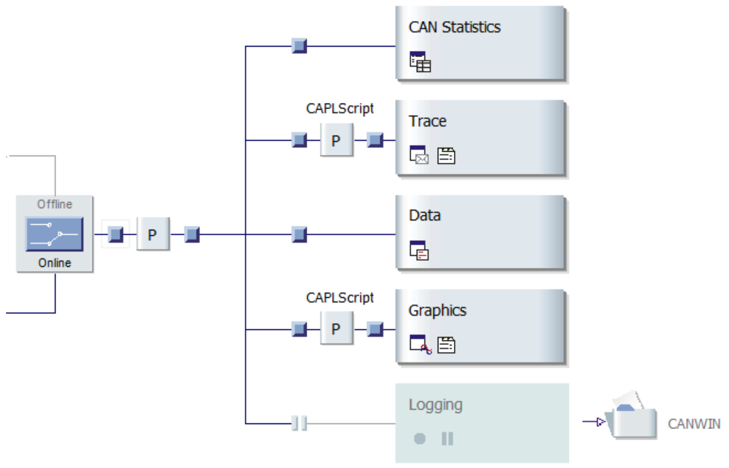
Field of Application CANalyzer

- Creating simulations or reactive scripts
- Implementing functions for analysis in the measurement setup

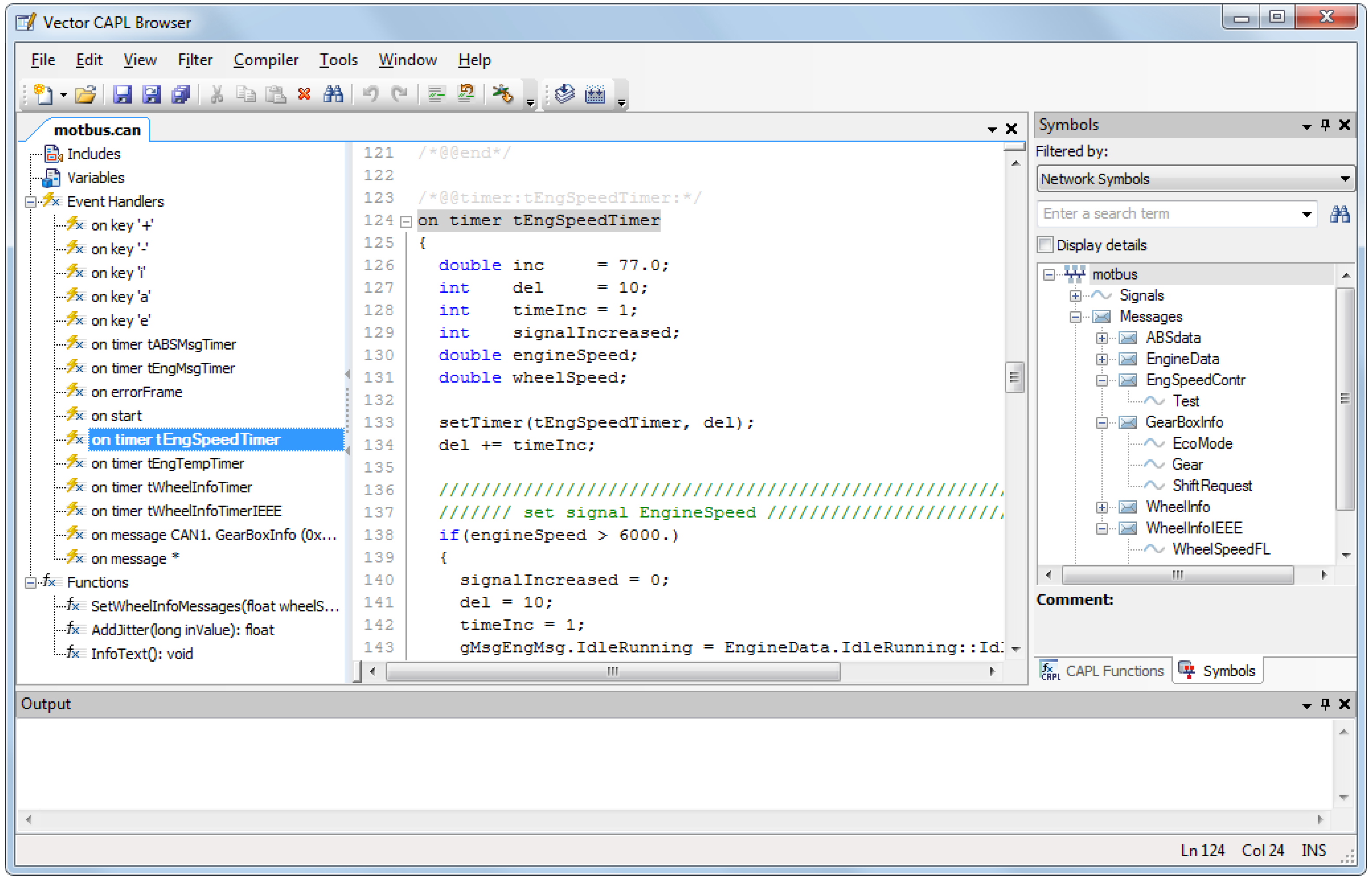
Send Loop of the Measurement Setup

Analysis Branches





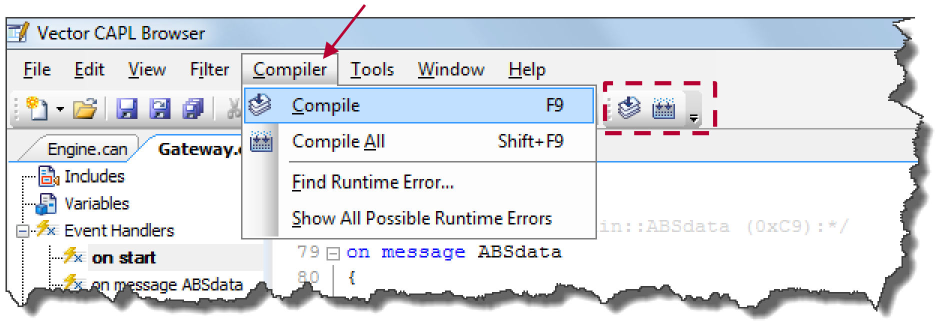
CAPL Browser



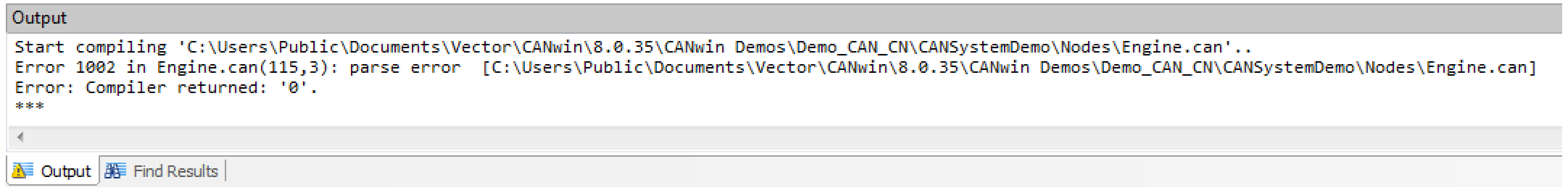
```
/*@@end*/

/*@@timer:tEngSpeedTimer:*/
on timer tEngSpeedTimer
{
  double inc     = 77.0;
  int    del     = 10;
  int    timeInc = 1;
  int    signalIncreased;
  double engineSpeed;
  double wheelSpeed;

  setTimer(tEngSpeedTimer, del);
  del += timeInc;

  ///////////////////////////////////////////////////////
  /////// set signal EngineSpeed ///////////////////////
  if(engineSpeed > 6000.)
  {
    signalIncreased = 0;
    del = 10;
    timeInc = 1;
    gMsgEngMsg.IdleRunning = EngineData.IdleRunning::Id
```

Compiling

- In order to generate an executable program file from a CAPL program, the program must be compiled with the CAPL compiler:



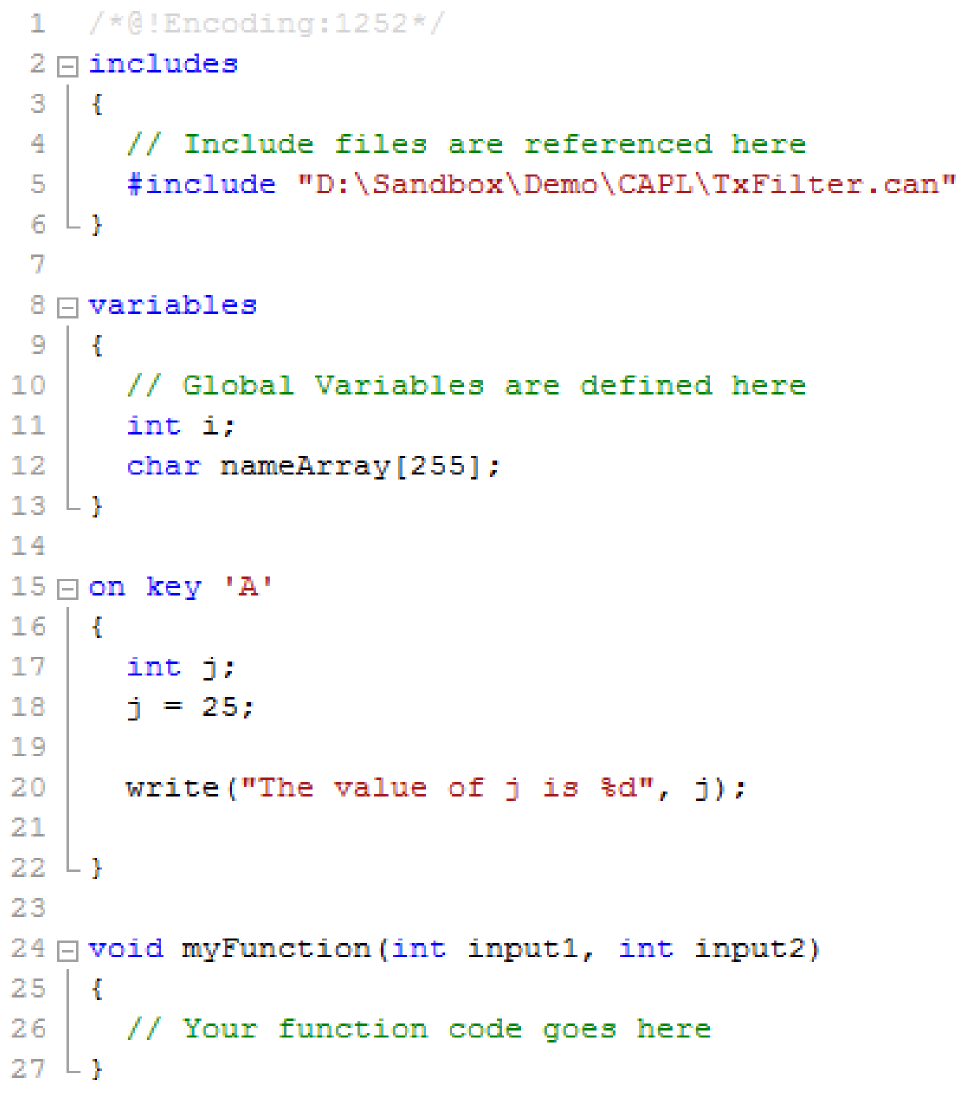
- Error messages are shown in the lower Message Window:



```
Start compiling 'C:\Users\Public\Documents\Vector\CANwin\8.0.35\CANwin Demos\Demo_CAN_CN\CANSystemDemo\Nodes\Engine.can'..
Error 1002 in Engine.can(115,3): parse error  [C:\Users\Public\Documents\Vector\CANwin\8.0.35\CANwin Demos\Demo_CAN_CN\CANSystemDemo\Nodes\Engine.can]
Error: Compiler returned: '0'.
***
```

When you double-click the error description, the cursor in the *Text Editor* automatically jumps to the point in the source code, where the error originated.

Examining a CAPL program



```
/*@!Encoding:1252*/
includes
{
  // Include files are referenced here
  #include "D:\Sandbox\Demo\CAPL\TxFilter.can"
}

variables
{
  // Global Variables are defined here
  int i;
  char nameArray[255];
}

on key 'A'
{
  int j;
  j = 25;

  write("The value of j is %d", j);

}

void myFunction(int input1, int input2)
{
  // Your function code goes here
}
```

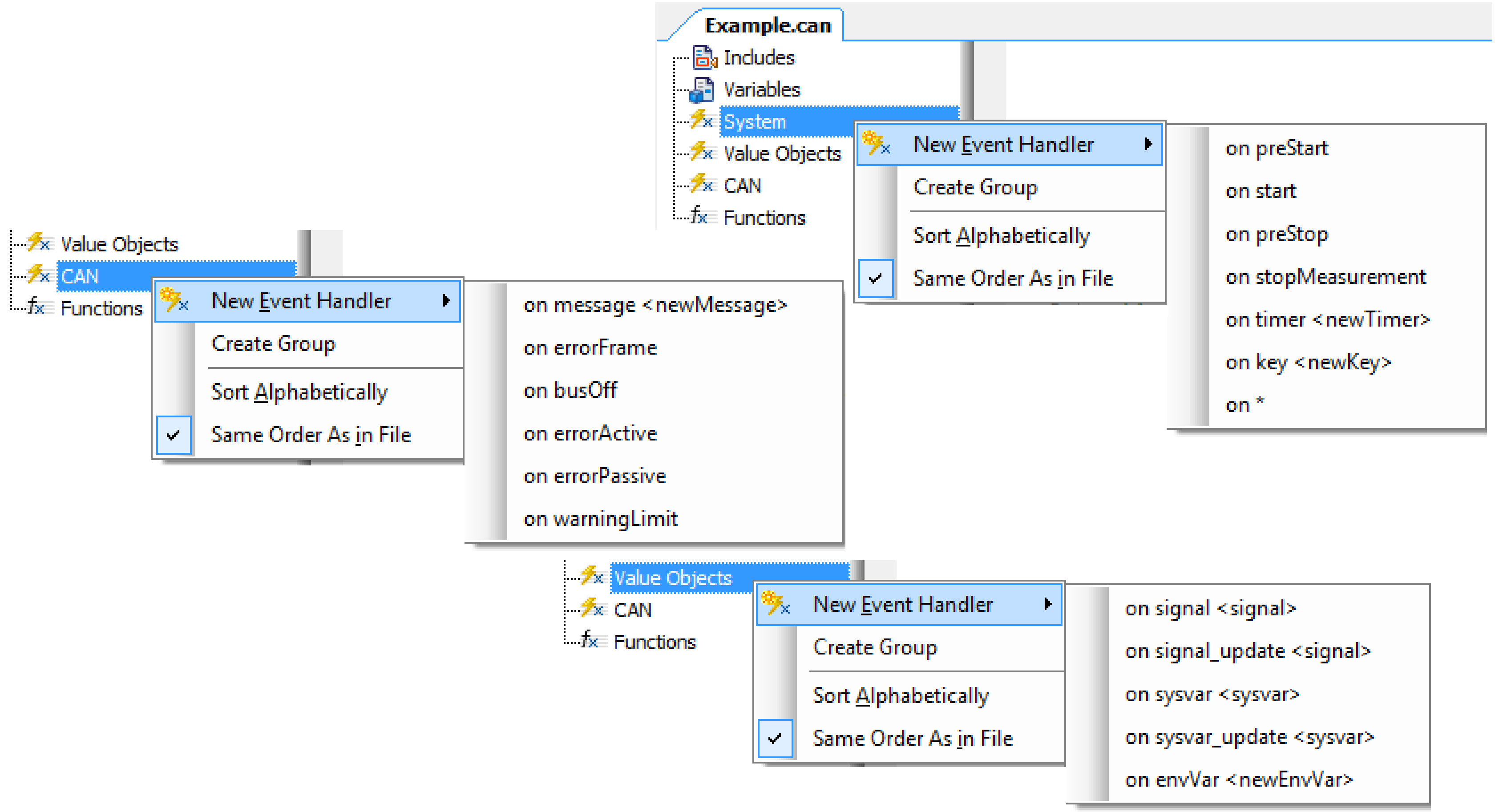
Additional CAPL files that contain generic code that can be reused in other programs

Variables defined here are accessible throughout the CAPL program

Multiple pre-defined event handlers exist for your use within CAPL. The code in this handler will only be executed when the event occurs.

You can create your own functions (special handler) that contain related code to be executed frequently

Adding an Event Handler



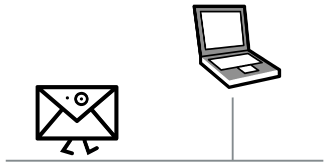
CAPL is a procedural language in which the execution of program blocks is controlled by events. These program blocks are referred to as event procedures.

Important Event Handlers

- Start of measurement

```
on Start
{
   write ("Start of CANoe");
}
```

- Message received



```
on message 0x123
{
   write ("CAN Message 123");
}
```

- Signal change

```
on signal sigTemp
{
   write ("Signal Temperature");
}
```

- Time event

```
on timer tmrCycle
{
   write ("within cycle");
}
```

- Key press

```
on key 'a'
{
   write ("Key >a< pressed");
}
```

On Key Procedures

```
on key 'a'           // React to press of ‘a’ key

on key ' '           // React to press of spacebar

on key 0x20          // React to press of spacebar

on key F1            // React to press of F1 key

on key ctrlF12       // React to press of Ctrl-F12

on key PageUp        // React to press of Page Up key

on key Home          // React to press of Home key

on key *             // React to any key press except…
```

Data types for CAN

Type		Name	Bit	Note
Integers	Signed	`int`	16	
		`long`	32	
		`int64`	64	
	Unsigned	`byte`	8	
		`word`	16	
		`dword`	32	
		`qword`	64	
Floating point		`float`	64	Per IEEE
		`double`	64	Per IEEE
Single character		`char`	8	
Message variable	for CAN	`message`		for CAN messages
Time variables	for seconds	`timer`		for Timer in s
	for milliseconds	`mstimer`		for Timer in ms

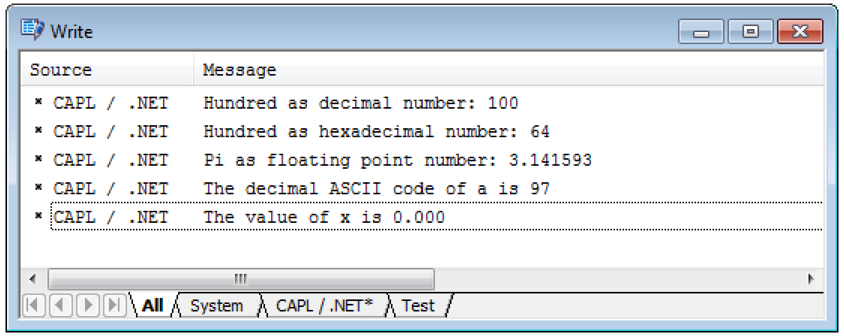
VECTOR

Variables in CAPL

- CAPL code:

```
int i = 100;          // Declaration and initialization of an integer
char ch = 'a';        // Declaration and initialization of a character
float x;              // Declaration of a floating point number

write ("Hundred as decimal number: %d", i);

write ("Hundred as hexadecimal number: %x", i);

write ("Pi as floating point number: %f", pi);

write ("The decimal ASCII code of %c is %d", ch, ch);

write ("The value of x is %f", x);
```

- Results:



```
Source          Message
CAPL / .NET     Hundred as decimal number: 100
CAPL / .NET     Hundred as hexadecimal number: 64
CAPL / .NET     Pi as floating point number: 3.141593
CAPL / .NET     The decimal ASCII code of a is 97
CAPL / .NET     The value of x is 0.000
```

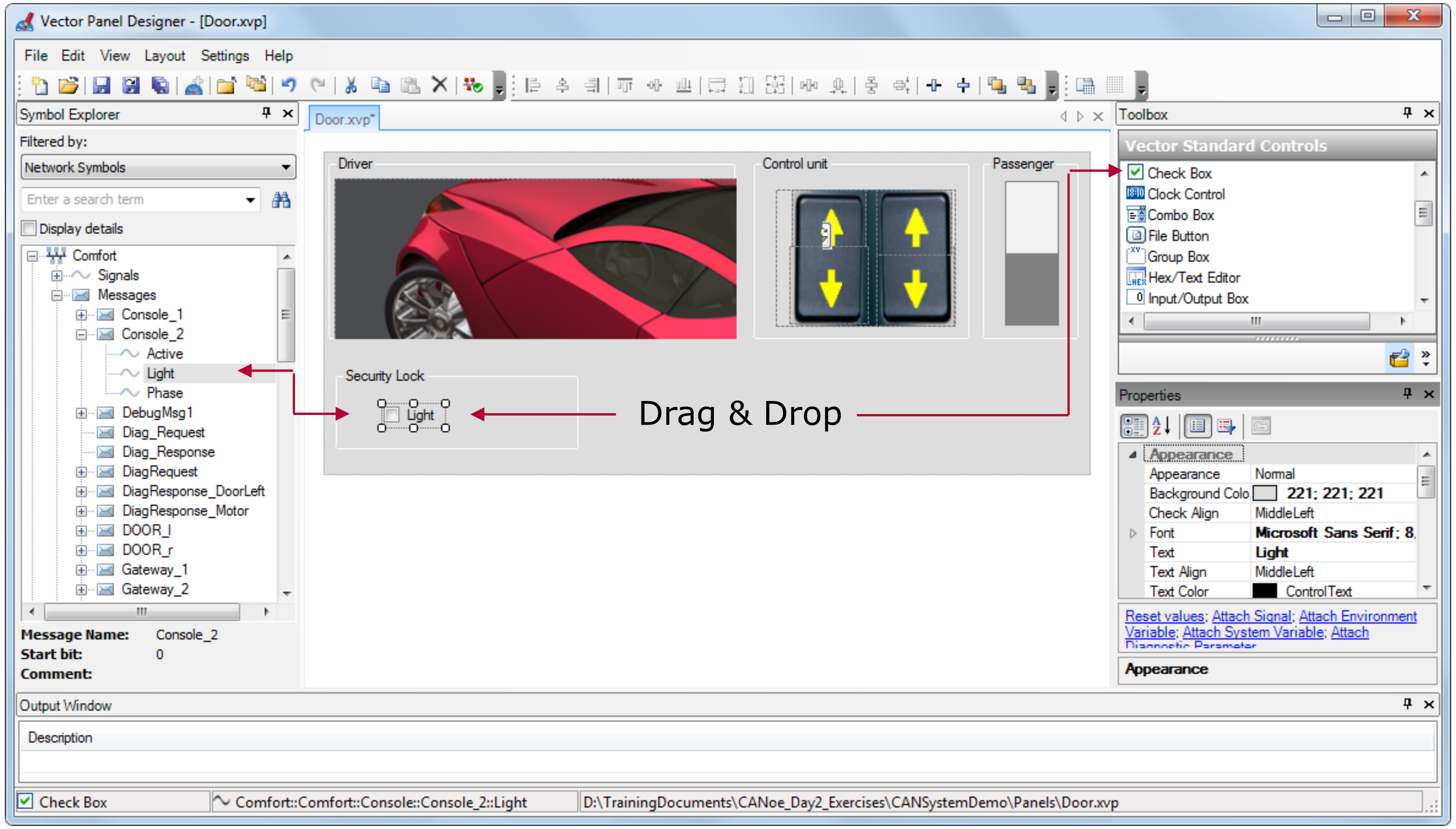
Operators

Operator	Description	Example
`+   -`	Addition, subtraction	-
`*   /`	Multiplication, division	-
`++  --`	Increment or decrement by 1	`a++;   // increments a by 1`
`%`	Modulo division (returns integer remainder of a division)	`a = 4 % 3;  // a is 1`
`<   <=`	Less than; less than or equal to	`returns TRUE or FALSE`
`>   >=`	Greater than; greater than or equal to	`returns TRUE or FALSE`
`==  !=`	Compare for equality or inequality	`returns TRUE or FALSE`
`&&`	Logic AND	`returns TRUE or FALSE`
`||`	Logic OR	`returns TRUE or FALSE`
`!`	Logic NOT	`changes TRUE to FALSE and vice versa`
`&`	Bitwise AND	`1 & 7  // yields 1 (0001 & 0111 → 0001)`
`|`	Bitwise OR	`1 | 7  // yields 7 (0001 | 0111 → 0111)`
`~`	Bitwise complement	`~1     // yields 14 (0001 → 1110)`
`^`	Bitwise exclusive OR (XOR)	`01^11 // ergibt 10`
`>>  <<`	Bit shift to right or left	`1 << 3 // yields 8 (0001 → 1000)`

Agenda

Creating a Panel

A signal is mapped to each display or control:



Message Name: Console_2
Start bit: 0
Comment:

Appearance	
Appearance	Normal
Background Colo	221; 221; 221
Check Align	MiddleLeft
Font	Microsoft Sans Serif; 8,
Text	Light
Text Align	MiddleLeft
Text Color	ControlText

Reset values; Attach Signal; Attach Environment Variable; Attach System Variable; Attach Diagnostic Parameter

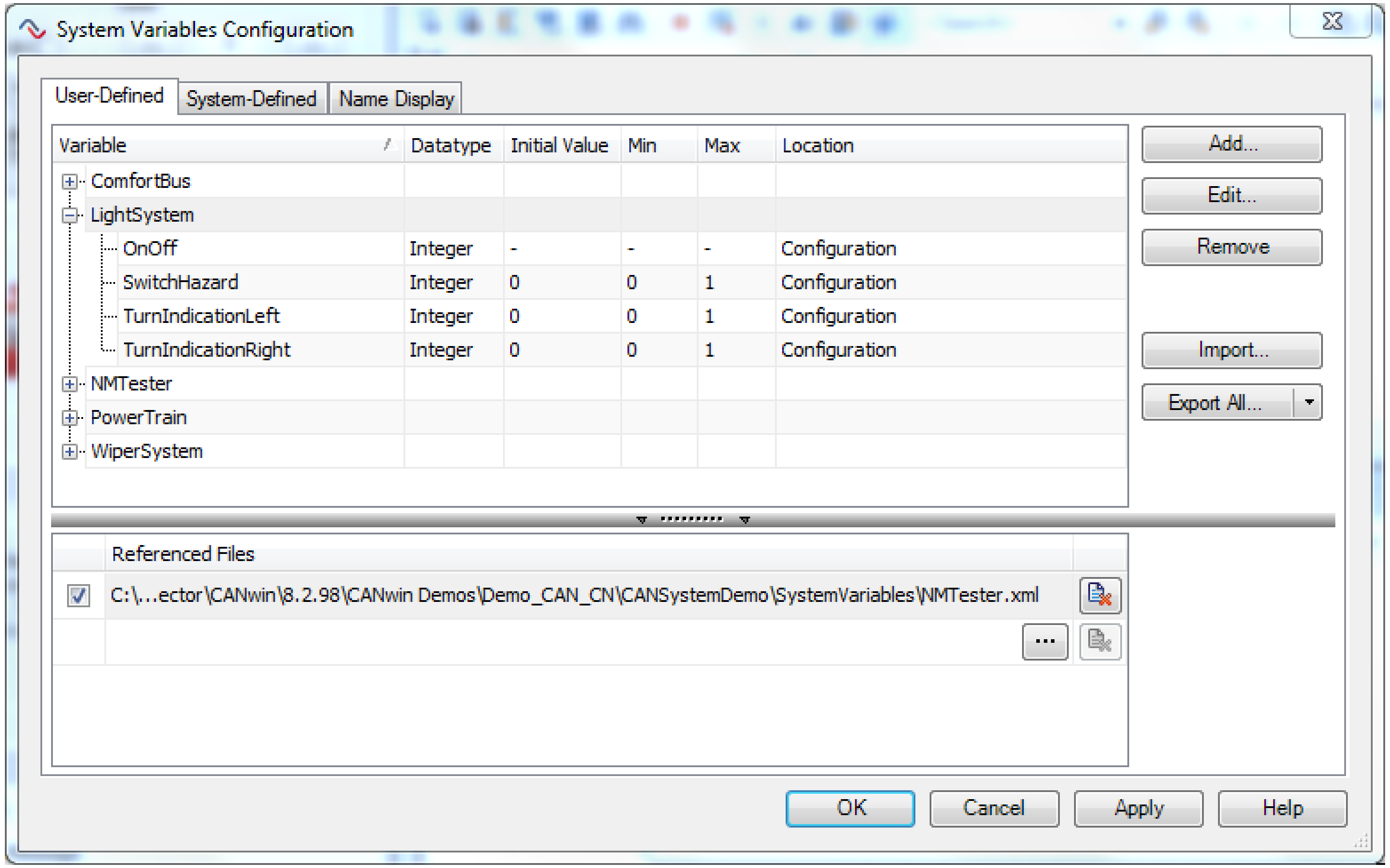
Appearance

Check Box | Comfort::Comfort::Console::Console_2::Light | D:\TrainingDocuments\CANoe_Day2_Exercises\CANSystemDemo\Panels\Door.xvp

VECTOR

Creating a System Variable for use with in a configuration

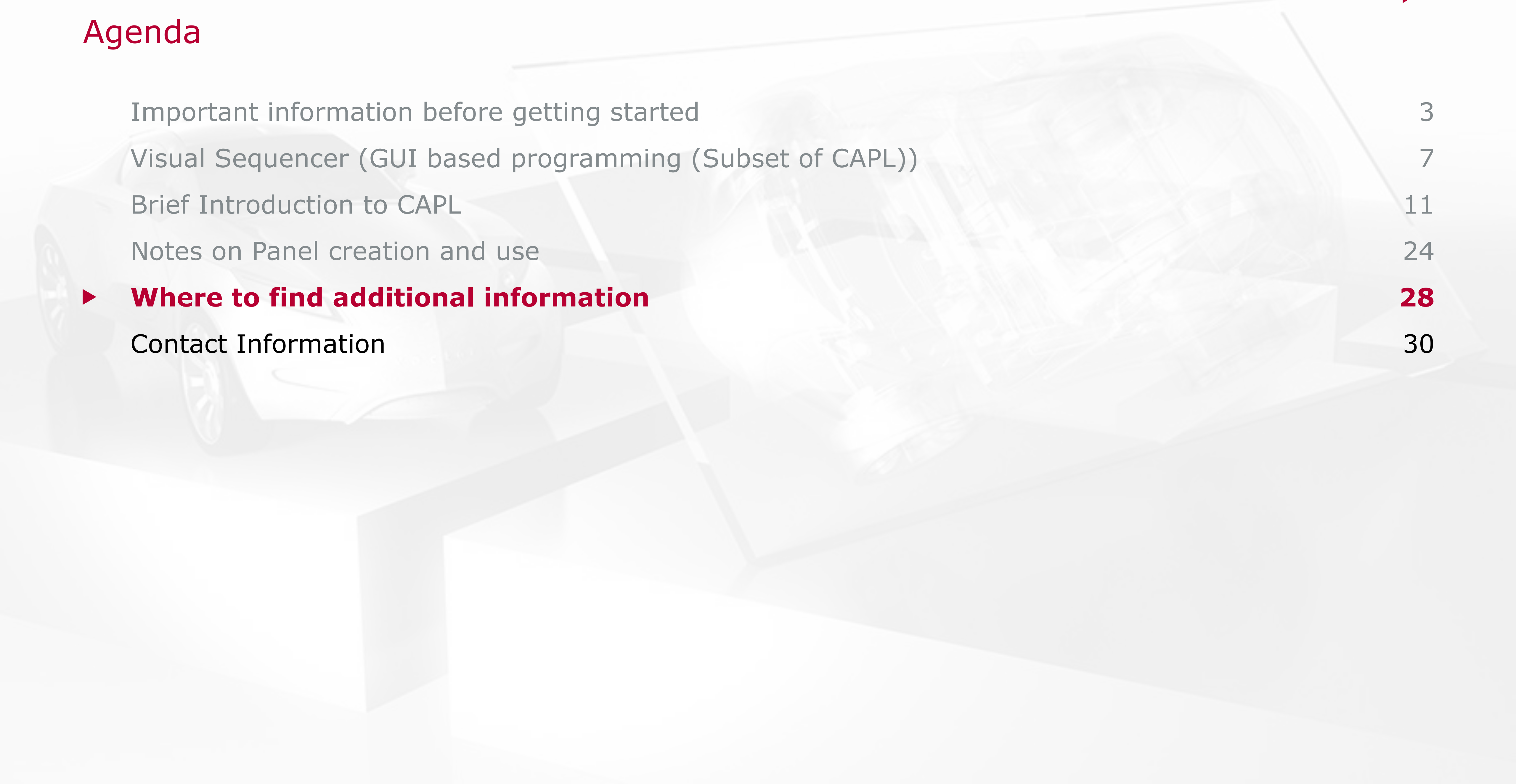
Signals can be automatically or user created, saved, exported, and imported via the Configuration|Systems Variables dialog:


System Variables Configuration

Variable	Datatype	Initial Value	Min	Max	Location
ComfortBus					
LightSystem					
OnOff	Integer	-	-	-	Configuration
SwitchHazard	Integer	0	0	1	Configuration
TurnIndicationLeft	Integer	0	0	1	Configuration
TurnIndicationRight	Integer	0	0	1	Configuration
NMTester					
PowerTrain					
WiperSystem					

	Referenced Files
✓	C:\...ector\CANwin\8.2.98\CANwin Demos\Demo_CAN_CN\CANSystemDemo\SystemVariables\NMTester.xml

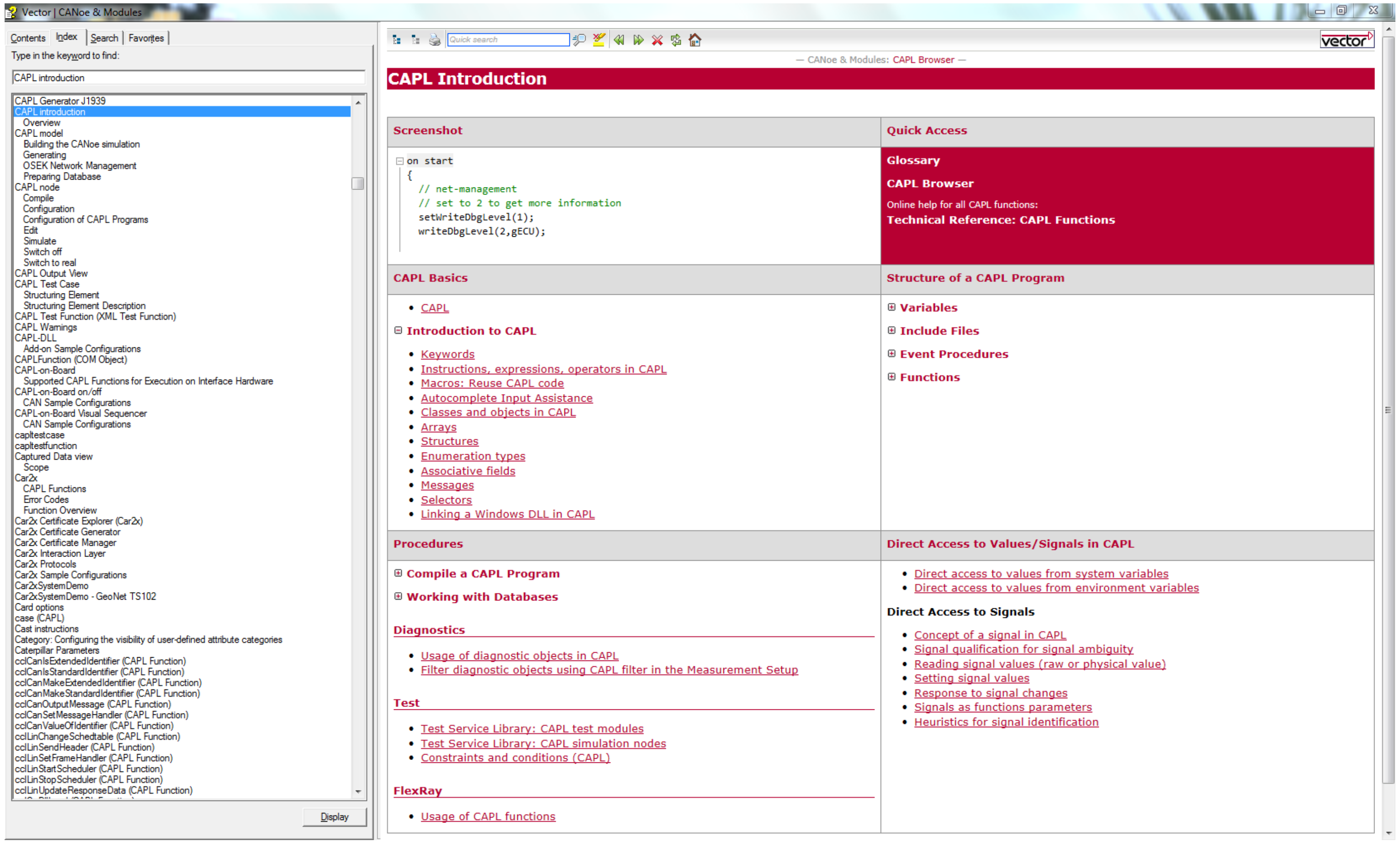
Agenda



Agenda

Where to find additional information

Online Help file: Use Index|CAPL Introduction



Vector | CANoe & Modules

Contents | Index | Search | Favorites

Type in the keyword to find:

CAPL introduction

CAPL Generator J1939
CAPL introduction
  Overview
CAPL model
  Building the CANoe simulation
  Generating
  OSEK Network Management
  Preparing Database
CAPL node
  Compile
  Configuration
  Configuration of CAPL Programs
  Edit
  Simulate
  Switch off
  Switch to real
CAPL Output View
CAPL Test Case
  Structuring Element
  Structuring Element Description
CAPL Test Function (XML Test Function)
CAPL Warnings
CAPL-DLL
  Add-on Sample Configurations
CAPLFunction (COM Object)
CAPL-on-Board
  Supported CAPL Functions for Execution on Interface Hardware
CAPL-on-Board on/off
  CAN Sample Configurations
CAPL-on-Board Visual Sequencer
  CAN Sample Configurations
capltestcase
capltestfunction
Captured Data view
  Scope
Car2x
  CAPL Functions
  Error Codes
  Function Overview
Car2x Certificate Explorer (Car2x)
Car2x Certificate Generator
Car2x Certificate Manager
Car2x Interaction Layer
Car2x Protocols
Car2x Sample Configurations
Car2xSystemDemo
Car2xSystemDemo - GeoNet TS102
Card options
case (CAPL)
Cast instructions
Category: Configuring the visibility of user-defined attribute categories
Caterpillar Parameters
cclCanIsExtendedIdentifier (CAPL Function)
cclCanIsStandardIdentifier (CAPL Function)
cclCanMakeExtendedIdentifier (CAPL Function)
cclCanMakeStandardIdentifier (CAPL Function)
cclCanOutputMessage (CAPL Function)
cclCanSetMessageHandler (CAPL Function)
cclCanValueOfIdentifier (CAPL Function)
cclLinChangeSchedtable (CAPL Function)
cclLinSendHeader (CAPL Function)
cclLinSetFrameHandler (CAPL Function)
cclLinStartScheduler (CAPL Function)
cclLinStopScheduler (CAPL Function)
cclLinUpdateResponseData (CAPL Function)

Display

Quick search

— CANoe & Modules: CAPL Browser —

CAPL Introduction

Screenshot

```
on start
{
  // net-management
  // set to 2 to get more information
  setWriteDbgLevel(1);
  writeDbgLevel(2,gECU);
```

Quick Access

Glossary

CAPL Browser

Online help for all CAPL functions:
Technical Reference: CAPL Functions

CAPL Basics

- CAPL

Introduction to CAPL

- Keywords
- Instructions, expressions, operators in CAPL
- Macros: Reuse CAPL code
- Autocomplete Input Assistance
- Classes and objects in CAPL
- Arrays
- Structures
- Enumeration types
- Associative fields
- Messages
- Selectors
- Linking a Windows DLL in CAPL

Structure of a CAPL Program

- **Variables**
- **Include Files**
- **Event Procedures**
- **Functions**

Procedures

- **Compile a CAPL Program**
- **Working with Databases**

Diagnostics

- Usage of diagnostic objects in CAPL
- Filter diagnostic objects using CAPL filter in the Measurement Setup

Test

- Test Service Library: CAPL test modules
- Test Service Library: CAPL simulation nodes
- Constraints and conditions (CAPL)

FlexRay

- Usage of CAPL functions

Direct Access to Values/Signals in CAPL

- Direct access to values from system variables
- Direct access to values from environment variables

Direct Access to Signals

- Concept of a signal in CAPL
- Signal qualification for signal ambiguity
- Reading signal values (raw or physical value)
- Setting signal values
- Response to signal changes
- Signals as functions parameters
- Heuristics for signal identification

Agenda

Looking for more information?

Visit our website:

> http://www.vector.com

Sign in for a Vector training class:

> http://www.vector.com/vi_training_en.html

Need help with Vector tools?

Contact our support team:

> (248) 449 – 9290 Option 2
> support@us.vector.com