VECTOR

CAPL Scripting Quickstart

CAPL (Communciation Access Programming Language)

For CANalyzer and CANoe

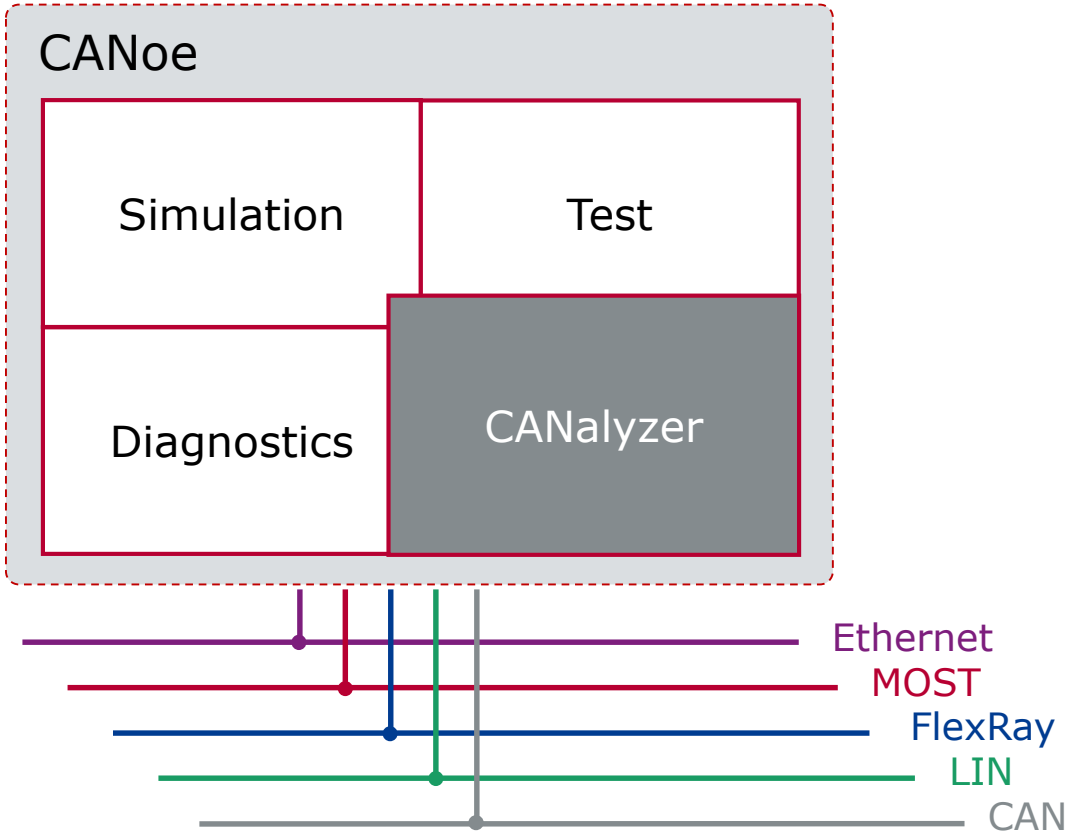
V0.1 | 2019-04-17

Agenda

- **Before Getting Started**
- Visual Sequencer (GUI Based Programming)
- Brief Introduction to CAPL
- Panel Creation and CAPL
- Additional Information
- Contact Information

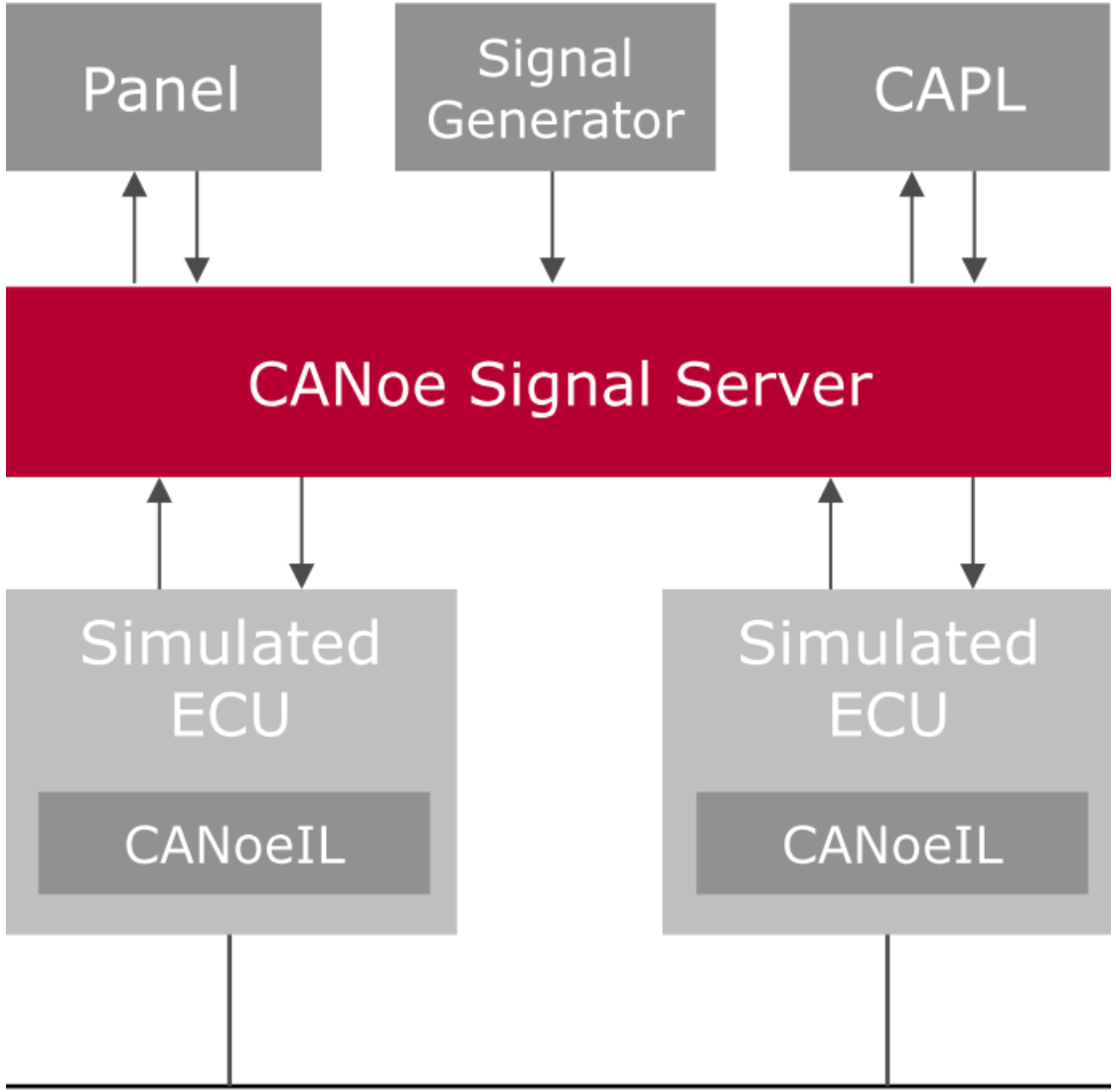
Difference between CANalyzer and CANoe

- 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 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



CAPL Support

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

- Before Getting Started
- **Visual Sequencer (GUI Based Programming)**
- Brief Introduction to CAPL
- Panel Creation and CAPL
- Additional Information
- Contact Information

General

- Available in both CANalyzer PRO and EXP as well as CANoe
  - Intended to allow some automation within the EXP variant

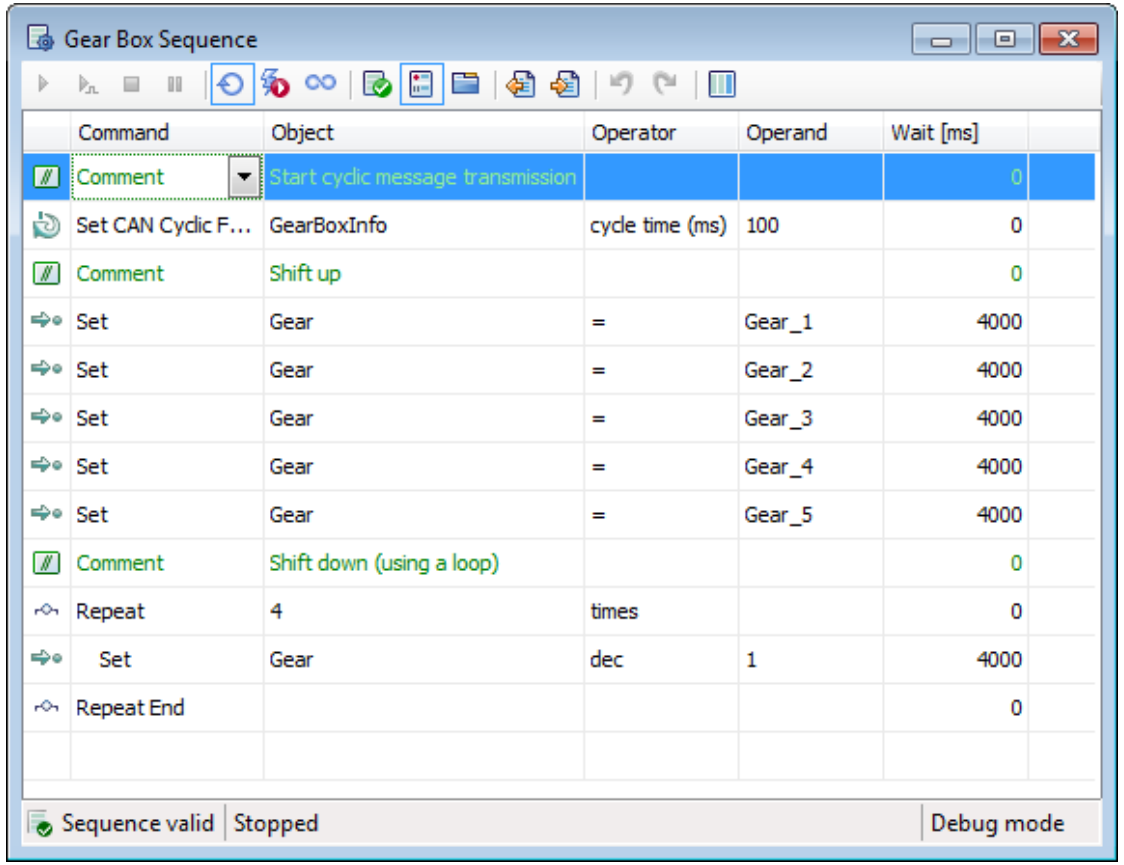
- The Visual Sequencer allows you to create *automated command sequences* with the purpose of
  - Stimulating the network
  - Controlling applications

- 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.

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



See Sample Configuration: CAN System Demo (CANsystemdemo.cfg)

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

Write

Source	Message

XML Tester

Test Cases | Test Observer

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	?	

00:00:00

Engine Tester

Test Cases | Test Observer

Test Case Name	Verdict	Runtime
Engine Tester		

Agenda

- Before Getting Started
- Visual Sequencer (GUI Based Programming)
- **Brief Introduction to CAPL**
- Panel Creation and CAPL
- Additional Information
- Contact Information

General

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

- Network node modules
- Special evaluation programs for individual applications

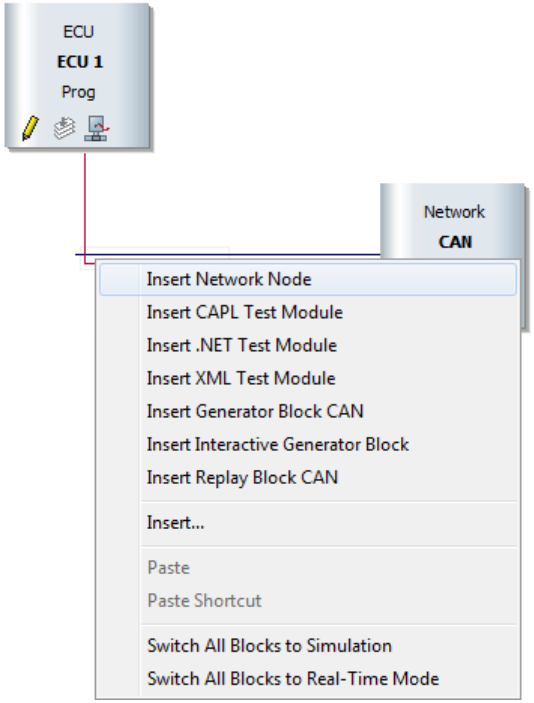
Some CAPL characteristics:

- 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

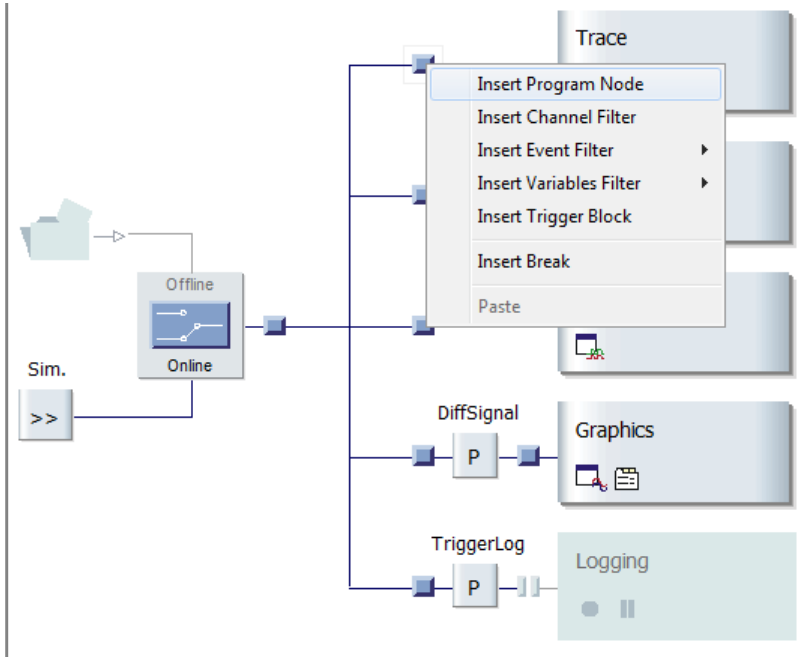
CANoe

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

Simulation Setup



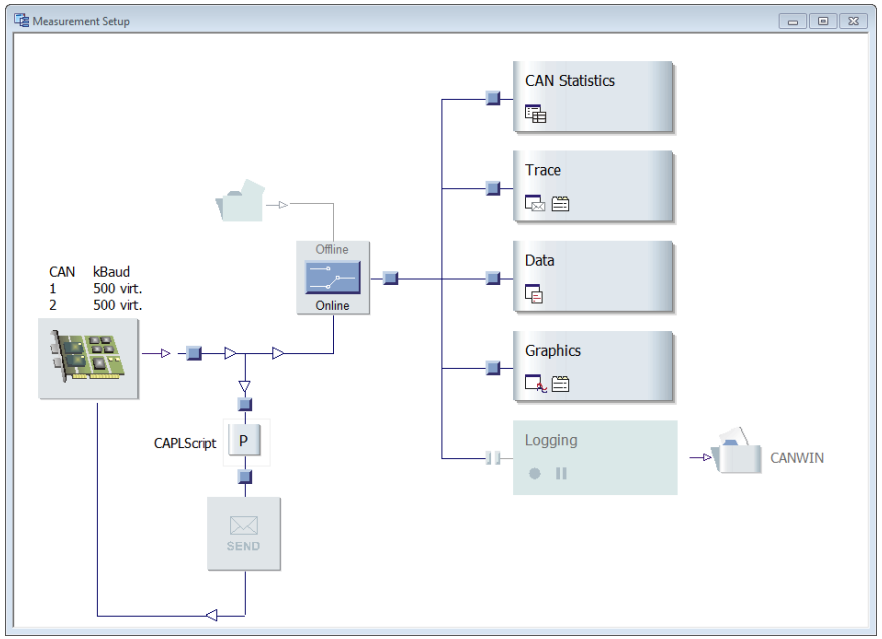
Measurement Setup



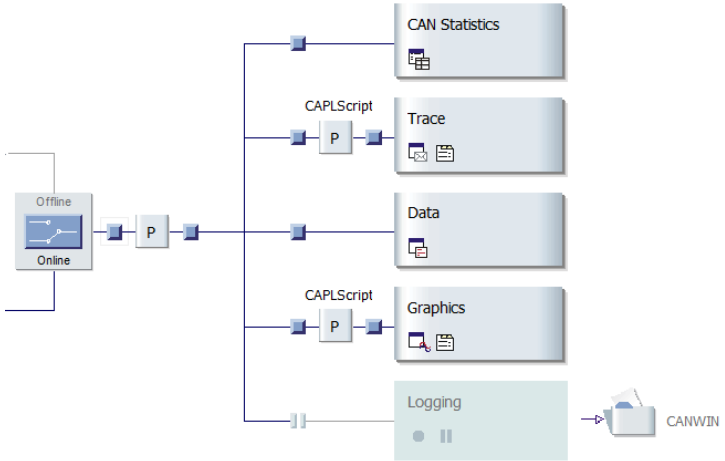
CANalyzer

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

Send Loop of the Measurement Setup



Analysis Branches



CAPL Browser

Vector CAPL Browser

File Edit View Filter Compiler Tools Window Help

motbus.can

- Includes
- Variables
- Event Handlers
  - on key '+'
  - on key '-'
  - on key 'i'
  - on key 'a'
  - on key 'e'
  - on timer tABSMsgTimer
  - on timer tEngMsgTimer
  - on errorFrame
  - on start
  - on timer tEngSpeedTimer
  - on timer tEngTempTimer
  - on timer tWheelInfoTimer
  - on timer tWheelInfoTimerIEEE
  - on message CAN1. GearBoxInfo (0x...
  - on message *
- Functions

Browser Tree
(List of Events)

Editor

```
121  /*@@end*/
122
123  /*@@timer:tEngSpeedTimer:*/
124  on timer tEngSpeedTimer
125  {
126    double inc     = 77.0;
127    int    del     = 10;
128    int    timeInc = 1;
129    int    signalIncreased;
130    double engineSpeed;
131    double wheelSpeed;
132
133    setTimer(tEngSpeedTimer, del);
134    del += timeInc;
135
136    ////////////////////////////////////////////////////
137    /////// set signal EngineSpeed ////////////////////
138    if(engineSpeed > 6000.)
139    {
140      signalIncreased = 0;
141      del = 10;
142      timeInc = 1;
143      gMsgEngMsg.IdleRunning = EngineData.IdleRunning::Id
```

Symbols

Filtered by:

Network Symbols

Enter a search term

Display details

- motbus
  - Signals
  - Messages
    - ABSdata
    - EngineData
    - EngSpeedContr
      - Test
    - GearBoxInfo
      - EcoMode
      - Gear
      - ShiftRequest
    - WheelInfo
    - WheelInfoIEEE

Symbols and Functions

Comment:

CAPL Functions | Symbols

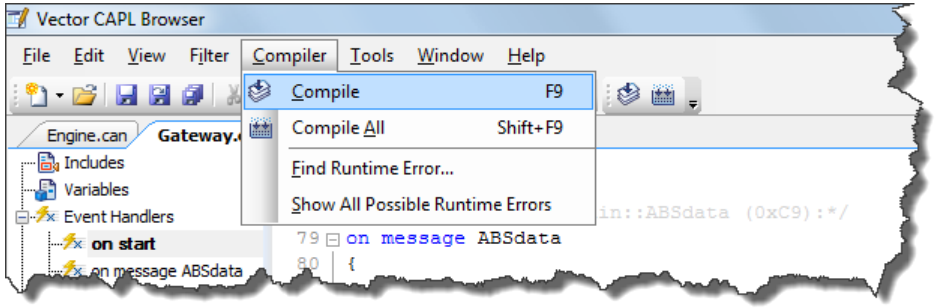
Output

Compiler Messages

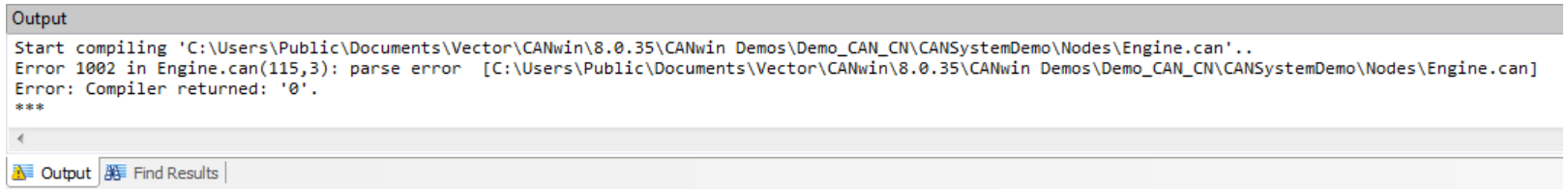
Ln 124 Col 24 INS

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:



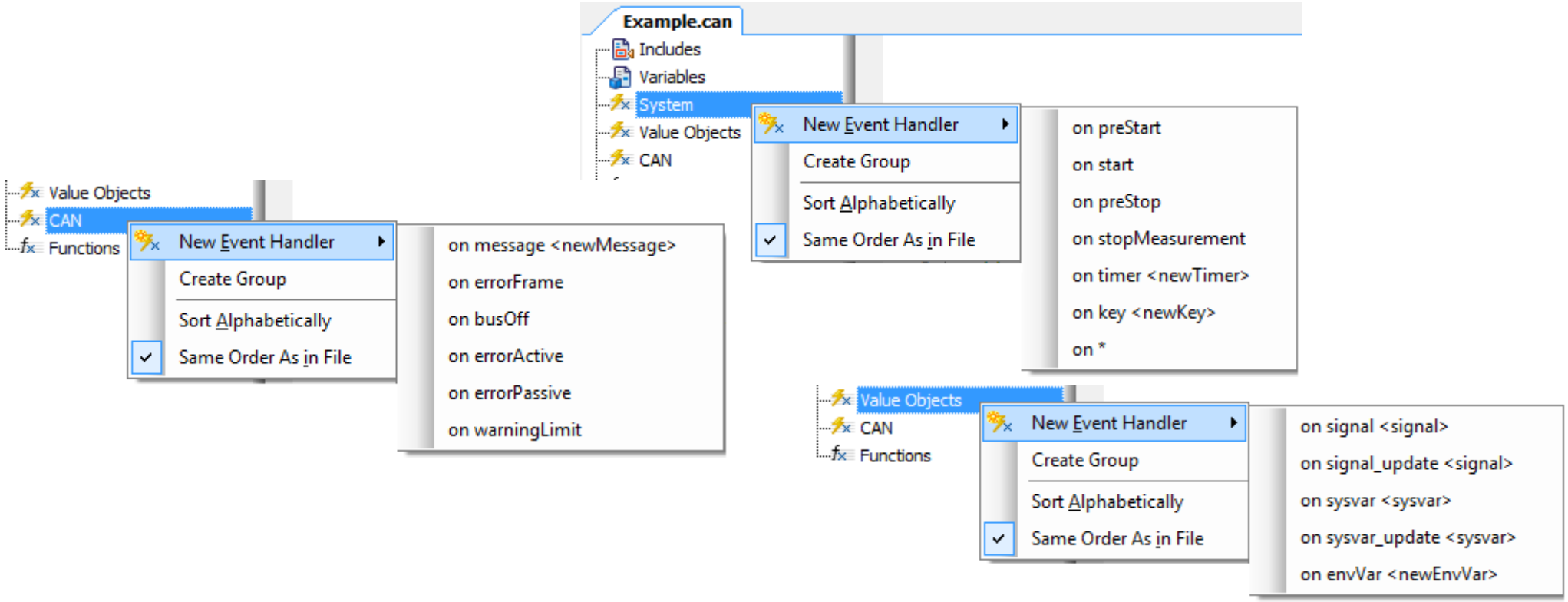
When you double-click the error description, the cursor in the *Text Editor* automatically jumps to the point in the source code, where the error originated.

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

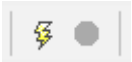
Adding Event Handlers



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.

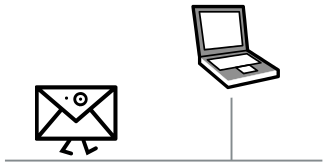
Important Event Handlers (CAN)

- 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");
}
```

VECTOR

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

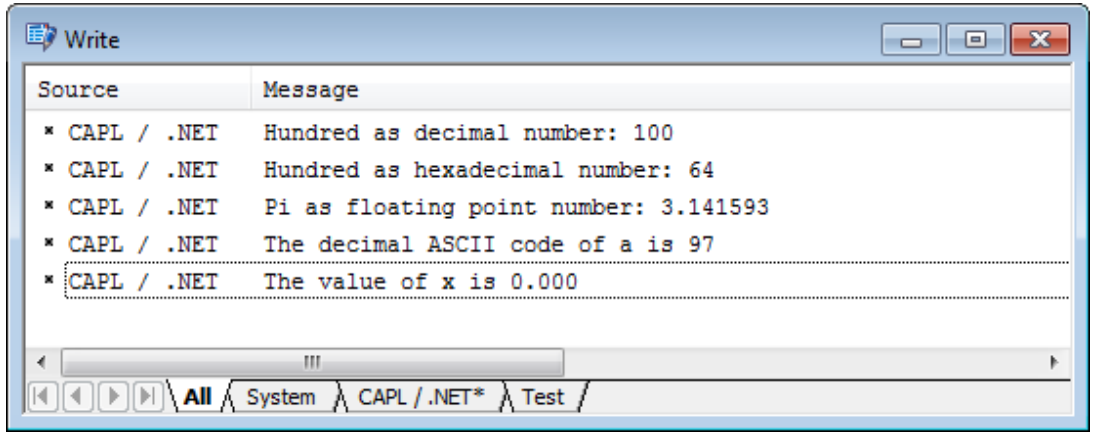
Type		Name	Bit	Note
Integers	Signed	`int`	16	
		`long`	32	
		`int64`	64	
	Unsigned	`byte`	8	
		`word`	16	
		`dword`	32	
		`qword`	64	
Floating point		`float`	64	Per IEEE
		`double`	64	Per IEEE
Single character		`char`	8	
Message variable	for CAN	`message`		for CAN messages
Time variables	for seconds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

Variables in CAPL

- CAPL code:

```
int i = 100;          // Declaration and initialization of an integer STATIC VARIABLES!!
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:



String Format Specifiers

Specifier	Description
"%ld","%d"	decimal display
"%lx","%x"	hexadecimal display
"%lX","%X"	hexadecimal display (upper case)
"%lu","%u"	unsigned display
"%lo","%o"	octal display
"%s"	display a string
"%g","%f"	floating point display. e.g. %5.3f means, 5 digits in total (decimal point inclusive) and 3 digits after the decimal point. 5 is the minimum of digits in this case.
"%c"	display a character
"%%"	display %-character
"%I64d","%lld"	decimal display of a 64 bit value
"%I64x","%llx”	hexadecimal display of a 64 bit value
"%I64X","%llX"	hexadecimal display of a 64 bit value (upper case)
"%I64u","%llu"	unsigned display of a 64 bit value
"%I64o","%llo"	octal display of a 64 bit value

Operators

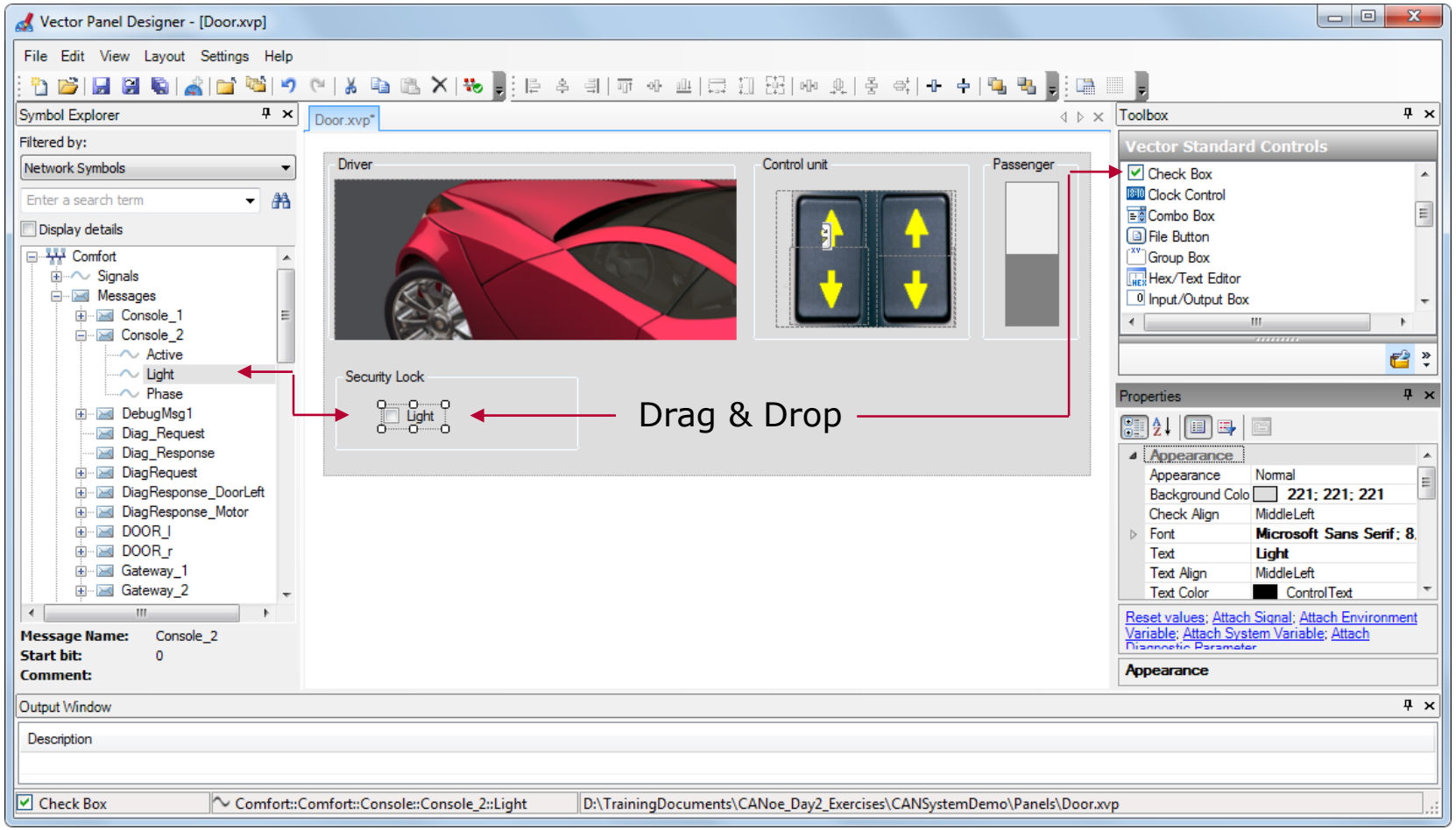
Operator	Description	Example
+ -	Addition, subtraction	-
* /	Multiplication, division	-
++ --	Increment or decrement by 1	a++; // increments a by 1
%	Modulo division (returns integer remainder of a division)	a = 4 % 3; // a is 1
< <=	Less than; less than or equal to	returns TRUE or FALSE
> >=	Greater 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

- Before Getting Started
- Visual Sequencer (GUI Based Programming)
- Brief Introduction to CAPL
- **Panel Creation and CAPL**
- Additional Information
- Contact Information

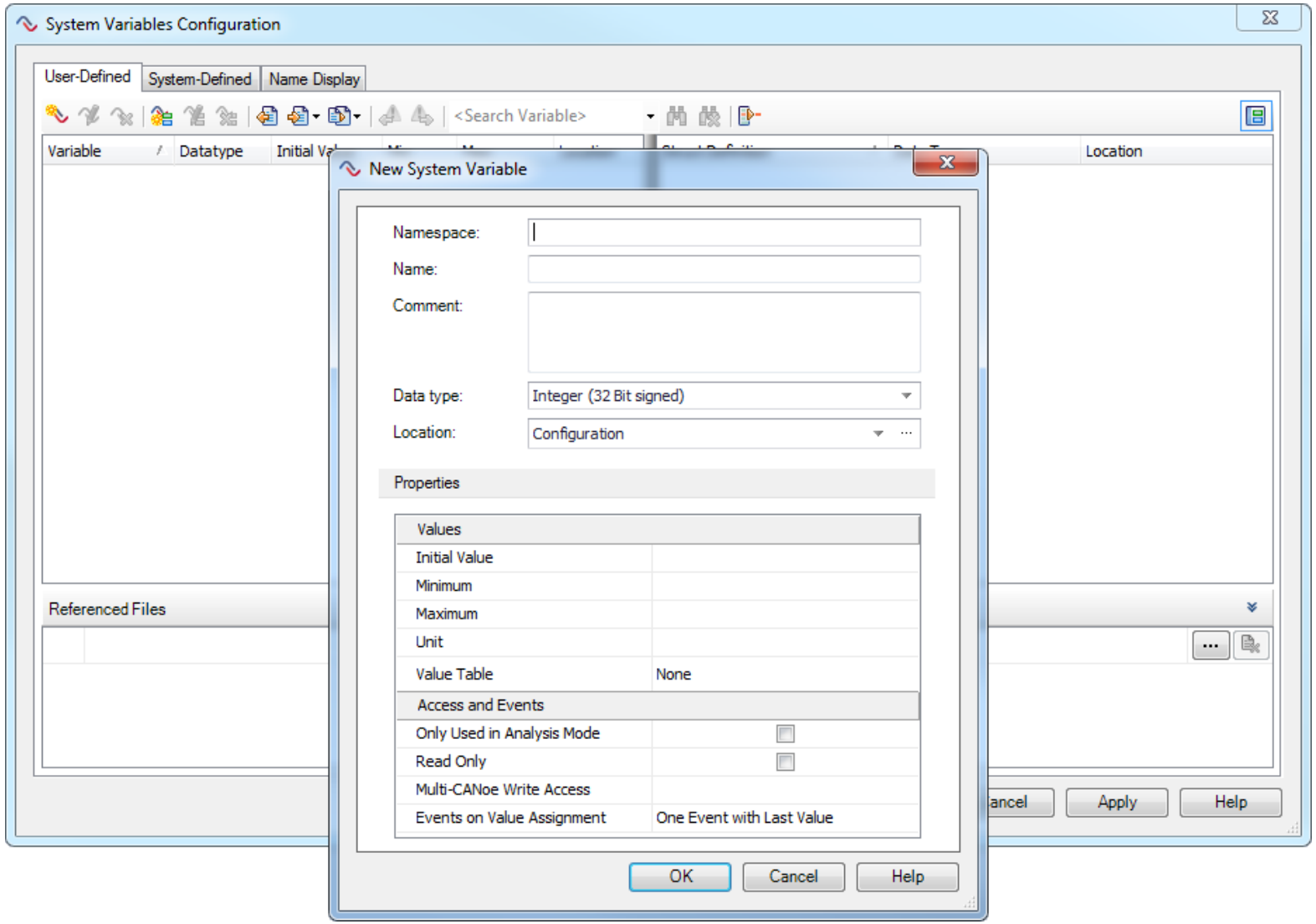
Creating a Panel

- A signal can be mapped to each display or control as simple as drag and drop
- CANalyzer display only controls for signals



Creating System Variables

- System Variables can be system defined or user defined.
- The variables can be created, saved, imported and exported.



Agenda

- Before Getting Started
- Visual Sequencer (GUI Based Programming)
- Brief Introduction to CAPL
- Panel Creation and CAPL
- **Additional Information**
- Contact Information

Online Help File

Vector | CANoe & Modules

Contents | Index | Search | Favorites

- CANoe
- Sample Configurations
- CANdb++
- CAPL Browser
- CAPL Introduction
  - Start Page
  - Glossary
  - CAPL Warnings
  - Basics
  - Structure of a CAPL Program
  - Compile a CAPL Program
  - Working with Databases
  - Access to Values/Signals in CAPL
  - Tips
- CAPL Functions
- Debugger
- Panel Designer
- FDX Editor
- Option CANoe .DiVa
- Vector Pool License Dialog
- Technical References

Topic search

CAPL Introduction

CANoe » 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

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

Direct Access to Values/Signals in CAPL

Direct Access to Variable Values

- 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 Function Parameters
- Heuristics for Signal Identification

Agenda

- Before Getting Started
- Visual Sequencer (GUI Based Programming)
- Brief Introduction to CAPL
- Panel Creation and CAPL
- Additional Information
- **Contact Information**

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

My contact info:

- Shawn.Kaschner@vector.com

Your questions are welcome!

Author:
Kaschner, Shawn
Vector North America