CAPL Scripting Quickstart

CAPL (Communication Access Programming Language) For CANalyzer and CANoe
## Agenda

- **Important information before getting started**  
  Visual Sequencer (GUI based programming (Subset of CAPL))  
  Brief Introduction to CAPL  
  Notes on Panel creation and use  
  Where to find additional information  
  Contact Information

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Important information before getting started</td>
<td>3</td>
</tr>
<tr>
<td>Visual Sequencer (GUI based programming (Subset of CAPL))</td>
<td>7</td>
</tr>
<tr>
<td>Brief Introduction to CAPL</td>
<td>11</td>
</tr>
<tr>
<td>Notes on Panel creation and use</td>
<td>24</td>
</tr>
<tr>
<td>Where to find additional information</td>
<td>28</td>
</tr>
<tr>
<td>Contact Information</td>
<td>30</td>
</tr>
</tbody>
</table>
Important information before getting started

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
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](http://www.vector.com/vi_canalyzer_variants_en.html)
Agenda

Important information before getting started

- Visual Sequencer (GUI based programming (Subset of CAPL))
  - Brief Introduction to CAPL
  - Notes on Panel creation and use
  - Where to find additional information
  - Contact Information
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
See the CANsystemdemo.cfg included with your installation
Agenda

Important information before getting started 3
Visual Sequencer (GUI based programming (Subset of CAPL)) 7

Brief Introduction to CAPL 11
- Notes on Panel creation and use 24
- Where to find additional information 28
- Contact Information 30
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
Brief Introduction to CAPL

Field of Application CANoe

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

Simulation Setup

Measurement Setup
Brief Introduction to CAPL

Field of Application CANalyser

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

Send Loop of the Measurement Setup

Analysis Branches
Brief Introduction to CAPL

CAPL Browser
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

```c
/*@!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[235];
```

```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
}
```
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.
Brief Introduction to CAPL

Important Event Handlers

- **Start of measurement**
  ```plaintext
  on Start
  {
    write ("Start of CANoe");
  }
  ```

- **Message received**
  ```plaintext
  on message 0x123
  {
    write ("CAN Message 123");
  }
  ```

- **Signal change**
  ```plaintext
  on signal sigTemp
  {
    write ("Signal Temperature");
  }
  ```

- **Time event**
  ```plaintext
  on timer tmrCycle
  {
    write ("within cycle");
  }
  ```

- **Key press**
  ```plaintext
  on key 'a'
  {
    write ("Key >a< pressed");
  }
  ```
On Key Procedures

```plaintext
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...
```
<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Bit</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integers</td>
<td>Signed</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>int</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>long</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>int64</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td>Unsigned</td>
<td>byte</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>word</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>dword</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>qword</td>
<td>64</td>
</tr>
<tr>
<td>Floating point</td>
<td></td>
<td>float</td>
<td>64  Per IEEE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>double</td>
<td>64 Per IEEE</td>
</tr>
<tr>
<td>Single character</td>
<td></td>
<td>char</td>
<td>8</td>
</tr>
<tr>
<td>Message variable</td>
<td>for CAN</td>
<td>message</td>
<td>for CAN messages</td>
</tr>
<tr>
<td>Time variables</td>
<td>for seconds</td>
<td>timer</td>
<td>for Timer in s</td>
</tr>
<tr>
<td></td>
<td>for milliseconds</td>
<td>mstimer</td>
<td>for Timer in ms</td>
</tr>
</tbody>
</table>
Variables in CAPL

CAPL code:

```plaintext
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:

![Image of CAPL output]

- Hundred as decimal number: 100
- Hundred as hexadecimal number: 64
- Pi as floating point number: 3.141593
- The decimal ASCII code of a is 97
- The value of x is 0.000
## Brief Introduction to CAPL

### Operators

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

Important information before getting started 3
Visual Sequencer (GUI based programming (Subset of CAPL)) 7
Brief Introduction to CAPL 11

- **Notes on Panel creation and use** 24
Where to find additional information 28
Contact Information 30
Notes on Panel creation and use

Creating a Panel

A signal is mapped to each display or control:
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:
Agenda

Important information before getting started 3
Visual Sequencer (GUI based programming (Subset of CAPL)) 7
Brief Introduction to CAPL 11
Notes on Panel creation and use 24

- Where to find additional information 28
Contact Information 30
Agenda

Important information before getting started  3
Visual Sequencer (GUI based programming (Subset of CAPL))  7
Brief Introduction to CAPL  11
Notes on Panel creation and use  24

► Where to find additional information  28
Contact Information  30
Where to find additional information

Online Help file: Use Index | CAPL Introduction
Agenda

Important information before getting started 3
Visual Sequencer (GUI based programming (Subset of CAPL)) 7
Brief Introduction to CAPL 11
Notes on Panel creation and use 24
Where to find additional information 28

Contact Information 30
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