ViCANdo

JAVASCRIPT & QML EXTENSION

TABLE OF CONTENTS

Chapter

1

1. JAVASCRIPT EXTENSION

ViCANdo can be extended with custom functionality, in the form of Scriptlets. Script components are written in JavaScript and a Scriptlet can be started on demand, or by a Trigger, configured to start a Scriptlet.

1.1 ABOUT JAVASCRIPT

JavaScript (JS) is an interpreted computer programming language. It was originally implemented as part of web browsers so that client-side scripts could interact with the user, control the browser, communicate asynchronously, and alter the document content that was displayed. More recently, however, it has become common in both game development and the creation of desktop applications.

JavaScript is a scripting language that is dynamic, is type safe, and has first-class functions. Its syntax was influenced by the language C. JavaScript copies many names and naming conventions from Java, but the two languages are otherwise unrelated and have very different semantics. The key design principles within JavaScript are taken from the self and Scheme programming languages. It is a multi-paradigm language, supporting object-oriented, imperative, and functional programming styles.

JavaScript's use in applications outside of web pages—for example, in PDF documents, site-specific browsers, and desktop widgets—is also significant. Newer and faster JavaScript VMs and frameworks built upon them (notably Node.js) have also increased the popularity of JavaScript for server-side web applications.

JavaScript was formalized in the ECMAScript language standard and is primarily used as part of a web browser (client-side JavaScript). This enables programmatic access to computational objects within a host environment.

EXAMPLES

HELLO WORLD

```
project.logMessage ("Hello World!")
```

FACTORIAL

```
function factorial (n) {
    if {n === 0} {
     return 1;
}
return n * factorial {n - 1};
}
```

1.2 GLOBAL OBJECTS

NAME	DESCRIPTION
Project	Access project resources
Sources []	Array of sources available in project
Presenter []	Array of presenters available in project
Scriptlet []	Array of Scriptlet available in project
Trigger []	Array of triggers available in project
dbc []	Array of DBC's available in project

GLOBAL CONSTANTS

```
var Idle          =  0x00  // Project is idle
var Preparing     =  0x01  // waiting for enabled sources to be activated
var Armed         = 0x02  // All enabled sources are now active
var Recording     = 0x03 // Recording data from all enabled sources into a new session
var Playing       = 0x04 // Playing a previously recorded session
var Pause         = 0x05 // Playback or record has been paused
var Rewind        = 0x06 // Rewind will happen when a session has been played to its end
point
                          // it will start from the beginning on next
Play
```

1.2.2 GLOBAL FUNCTIONS

PROTOTYPE	DESCRIPTION
delay(time_in_ms)	Wait for specified time in milli-seconds
includeScript(relative_script_path)	Include and parse another script file into this script context

INCLUDE SCRIPT FUNCTION

Provides the possibility to include another script file into the current script context. Now, it's not pre-processor inclusion, the script will be evaluated in the current context, the variable will depend from where includeScript is called. A script will only be included once, if includeScript is called more than one time including the same script file, it will have no effect.

EXAMPLE

```
// hello.js

Project.log (“Hello World from hello.js”)
IncludeScript (“hello_inc.js”)

// Call function defined in hello_inc.js
Hello()

//hello_inc.js
Project.log(“Hello from hello_inc.js”)

// define a callable function
this. Hello – function {}
{
Project.log(“Hello from hello{} function”}
}
```

1.3 TIMER OBJECT

PROTOTYPE	DESCRIPTION
integer elapsedInMs()	Elapsed time in milli-seconds since the timer was started
integer elapsedInUs()	Elapsed time in micro-seconds since the timer was started
restart ()	Restart the timer

```
var timer1 = Timer {}
// do something ....
var elapsed_time_in_us – timer1.elapsedInUs {}
```

1.3.1 TIMER WITH A CALLBACK

PROTOTYPE	DESCRIPTION
Timer (<JavaScript-function>)	Create a new timer with a given JavaScript function that will be invoked on timeout
Start(timeout_in_ms)	Start timer with specified timeout in milli-seconds
Stop()	Stop the timer
singleShot=true/false	Single shot property, set to true and timer will only expire 1 time after start()

NOTE: All timers will be automatically stopped after script has finished

Example using a periodic timer:

```
var counter = 0
var t1 = new Timer {function{} {
      project.log("Timer callback" + counter)
      counter ++
})

T1.start (1000)
```

Example using a single-shot timer:

```
var t1 = new Timer (function{} {
        Project.log ("Timer has expired")
})
T1.singleshot = true
T1.start (1000)
```

1.4 ALL OBJECTS

Methods (available on all kind of sources)

PROTOTYPE	DESCRIPTION
logMessage(string message)	Log a text message to the project console

PROJECT OBJECT

PROTOTYPE	DESCRIPTION
Log(string message)	Log a text message to the project console (just a short alias to logMessage function)
Int currentState()	Return the current state, may be idle, Preparing, Armed, Recording, Playing, Pause or Rewind
WaitForStateChange(timeout_in_ms)	Wait for State Change if no state change happen in specified timeout an exception is thrown
registerStateChangeCallback (callback)	Register a function callback that will be invoked when project state has changed
Activate()	Activate all enabled sources, next state will be Armed
startRecord()	Start recording. Can only be done in Armed state
Stop()	Stop all activity and go to Idle state
clearConsole()	Clear the ViCANdo text console
StoreValue(String key, Value, persistent=False)	Store a value within the project. The value will be permanent until the project is closed or the script engine is restarted. If persistent parameter, the value will be permanently stored within the project. Note: only string values can be permanently stored
removeValue(String key)	Remove a stored value
<Value> value(key)	Retrieve a stored value with store Value (String, key, value)
ClearStoredValues()	Remove all stored values

FUNCTIONS PROPERTIES

Project. Sources []	Array of sources available in project also available as a global object
Project. Presenter[]	Array of presenters available in project also available as a global object
Project. Scriptlet []	Array of Scriptlet available in project also available as a global object
Project. Trigger[]	Array of triggers available in project also available as a global object
Project. Dbc []	Array of DBC's available in project also available as a global object
Project. Directory	Absolute path to the location of the current project

RegisterStateChangeCallback function

The function registered will only be called when the script is running. After script has finished, no more callbacks are received. The callback function has two parameters previous_state and new_state that will be some of the constants Idle, Preparing, Armed, Recording, Playing, Pause or Rewind.

Example on how to use the registerStateChangeCallback function:

```
Project.registerStateChangeCallback{function(previous_state,new_state)
{
project.log("Previous state" + previous_state);
project.log("New state" + new_state);
})

Project.log("waiting for state change")
Project.waitForStatechange (5000)
```

OBJECT NAMES

Every component in the project tree (Source, Presenters, Scriptlets, etc.) can be named with an Object name. This is done through the Component Properties pane in ViCANdo. A component can be accessed from JavaScript by its object name, via the project object, project. <Object-name>.

Example using a CAN Source where its Object name is set to main_can_source:

```
// Send an extended CAN frame
Project.main_can_source.send(0x2500,new Array(10,20,30,40,50,60,70,80)}
```

QML PRESENTER OBJECT

PROTOTYPE	DESCRIPTION
SetProperty(string property_name,object value)	Set a property with given value in the QML component
Object property(string property_name)	Get the value for a specified property in the QML components

1.5 SYSTEM FUNCTIONS

Provides a collection of system functions.

PROTOTYPE	DESCRIPTION
System.executeProgram(program,argument_list)	Start a new program with the arguments given in argument list and wait for the process to finish. If the script engine is stopped before the program has finished the process will be brutally terminated. The return value from this function contains the process exit code the standard and error output as a string in a array as {<exit_code>,<standard_output>,< Standard-error>}

System.executeProgramDetached(program,argument_list)	Start a new process program with the arguments given in arguments list and spawn it in the background. The return result is the PID of the process
System.availableTextCodecs()	List all available text codecs by name. Return an array of string objects.

1.6 FILE SYSTEM FUNCTIONS

Provides a collection of functions for file I/O and basic file-system manipulation. All functions are provides by the fs object.

PROTOTYPE	DESCRIPTION
Fs.unlink(Path)	Remove/delete a give path on file-system
Fs.rename(old_path_new_path)	Rename file specified by old_path to new_Path
Boolean fs.FileExists(path)	Return true if the specified path is a that exists on the file-system
Boolean fs.openDirectory(path)	Return true if the specified path is a directory
Stream fs.codeFile(path, mode)	Open a file. Mode can be "r' for read only."w" only write, "rw" for read and write, "a" for append stream object
Fs.close(stream)	Close an open stream
String fs.currentDirectory()	Returns the absolute path of the process current working directory
String fs.homeDirectory()	Returns the absolute path of the user's home directory. This will differ depending on operating system
String fs.tempDirectory()	Returns the absolute path of the operating system's temporary directory
String fs.separator()	Returns directory separator used on the target system, "I "under Unix (including Mac OS X) and "\" under windows
Fs.chdir(path)	Change the current directory to path
Fs. Mkdir(path)	Create directory path relative to currentDirectory ()
Fs.rndir(path)	Remove path must be a directory, and its relative to currentDirectory()
String list fs.list (path)	Remove path must be a directory and its relative to currentDirectory()
Stringlist fs.listFiles(path)	Return a list of files and directories in path
Fileinfo fs.fileInfo(path)	Return a fileinfo object for the given path the Fileinfo object provides system independent file information

STREAM OBJECT

PROTOTYPE	DESCRIPTION
writeLine(string)	Write a line to stream, LF is added to the end
String readLine{timeout_in_ms = <infinite>}	Read a line from the stream. If not a complete line is read from in timeout_in_ms milli-seconds is aborted with an exception
Flash()	Flushes any buffered data to the stream
Boolean eof()	Return true if at end of file
setEncoding(codec_name)	Set the text codec to used when read and writing text from this stream. Use system. availableTextCodecs() list available text codecs

FILE INFO OBJECT

PROPERTY	DESCRIPTION
path	Absolute path including the file name
Created	The data and time when the file was created
lastModified	The data and time when the file was last modified
lastRead	The data and time when the file was last read (accessed)
size	The size of the file in bytes
owner	The Owner of the file. On systems where files do not have owners or if an error occurs. It will contain an empty string
Group	The group of the file. On windows on systems where files do not have owners or if an error occurs it will contain an empty strings
directory	Set to true if this object points to a directory or to a symbolic link to a directory otherwise set to false
Executable	Set to true if the user can read the file; otherwise set to false
hidden	Set to true if this is a 'hidden' file; otherwise set to false
Readable	set to true if the user can write to the file; otherwise set to false
Writable	Set to true if the user can write to the file; otherwise set to false
symlink	Set to true if this object points to a symbolic link (or to a shortcut on windows) otherwise return false

Example creating a text file and writing some lines:

```
var out = fs.openfile("/tmp/test.txt","w")
out.writeLine("Test line 1")
out.writeLine("Test line 2")
out.writeLine("Test line 3")
out.writeLine("Test line 4")
fs.close(out)
```

Example reading some lines from a text file:

```
var f = fs.openFile("/tmp/test.txt","r")

var line_no = 1;
while (!f.eof()} {
  project.logMessage("line" + line_no + " " + f.readLine() };
  line_no ++;
}
Fs.close(f)
```

1.7 CAN SOURCE OBJECT

CONSTANTS

```
can.flag.Rtr                    = 0x00001
can.flag.Standard               = 0x00002
can.flag.Extended               = 0x00004
can.flag.Wakeup                 = 0x00008
can.flag.NError                 = 0x00010
can.flag.ErrorFrame             = 0x00020
can.flag.TxMsgAcknowledge       = 0x00040
can.flag.TxMsgRequest           = 0x00080
can.flag.ErrorMask              = 0x0ff00
can.flag.ErrorHWOverrun         = 0x00200
can.flag.ErrorSWOverrun         = 0x00400
can.flag.ErrorStuff             = 0x00800
can.flag.ErrorForm              = 0x01000
can.flag.ErrorCRC               = 0x02000
can.flag.ErrorBIT0              = 0x04000
can.flag.ErrorBIT1              = 0x08000
can.flag.Statistic              = 0x10000
```

METHODS

PROTOTYPE	DESCRIPTION
Send (unit id, byte [] data)	Send a CAN frame, extended is default
sendExtended(int id, byte[] data)	Send an extended CAN frame
sendStandard(int id, byte[] data)	Send a standard CAN frame
SendRemote(int id, byte[] data)	Send a remote CAN frame, default is extended
sendStandardRemote(int id, byte[] data)	Send a standard remote CAN frame
sendExtendedRemote(int id, byte[] data)	Send an extended remote CAN frame
Array receive(timeout_in_ms = <infinite>)	Receive a CAN frame, if nothing is received in timeout_in_ms script execution is aborted with an exception
Boot is Virtual()	Returns true if source is attached to virtual CAN interface

PROPERTIES

PROPERTIES	TYPE	DESCRIPTION
deviceList	String-array	List of current available CAN interfaces
device	Integer	The current device index
deviceName	integer	The name of the current CAN interface

Example using the CAN source object source [0] as a CAN source:

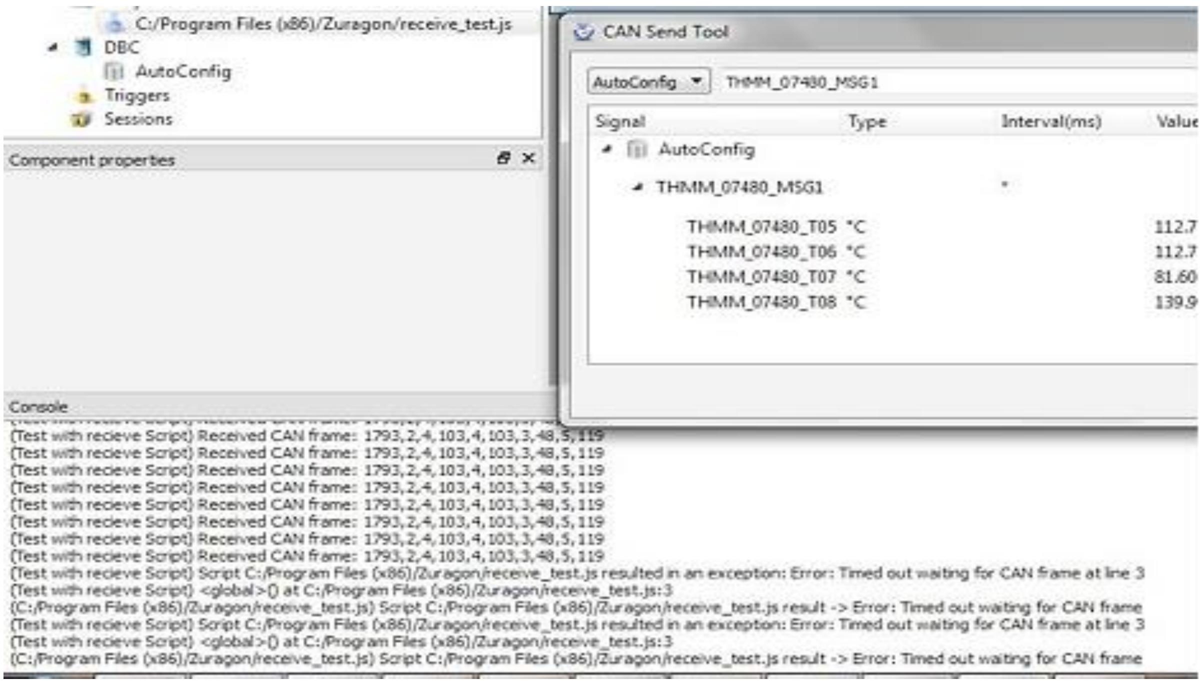
```
// Send an extended CAN 29bit frame
Source[0].send(0x2500, new Array(10,20,30,40,50,60,70,80));

// Send a standard CAN 11bit frame
Source[0].sendStandard (0x100, new Array(10,20,30,40,50,60,70,80));
```

Example of receiving a CAN frame:

```
var can_frame;
while (true) {
can_frame = source[0].receive(5000)
project.logMessage(“Received CAN frame: “ + can_frame)
}
```

This example is waiting to receive a CAN frame for 5000 ms If the CAN frame is not received on time, ViCANdo console window will print "Error: Timed out waiting for CAN frame
It looks like this in ViCANdo, where the received frames are displayed like this in the console window:



Signal	Type	Interval(ms)	Value
AutoConfig			
THMM_07480_MSG1		*	
THMM_07480_T05	°C		112.7
THMM_07480_T06	°C		112.7
THMM_07480_T07	°C		81.60
THMM_07480_T08	°C		139.9

```
(Test with recieve Script) Received CAN frame: 1793,2,4,103,4,103,3,48,5,119
(Test with recieve Script) Received CAN frame: 1793,2,4,103,4,103,3,48,5,119
(Test with recieve Script) Received CAN frame: 1793,2,4,103,4,103,3,48,5,119
(Test with recieve Script) Received CAN frame: 1793,2,4,103,4,103,3,48,5,119
(Test with recieve Script) Received CAN frame: 1793,2,4,103,4,103,3,48,5,119
(Test with recieve Script) Received CAN frame: 1793,2,4,103,4,103,3,48,5,119
(Test with recieve Script) Received CAN frame: 1793,2,4,103,4,103,3,48,5,119
(Test with recieve Script) Received CAN frame: 1793,2,4,103,4,103,3,48,5,119
(Test with recieve Script) Script C:/Program Files (x86)/Zuragon/receive_test.js resulted in an exception: Error: Timed out waiting for CAN frame at line 3
(Test with recieve Script) <global>() at C:/Program Files (x86)/Zuragon/receive_test.js:3
(C:/Program Files (x86)/Zuragon/receive_test.js) Script C:/Program Files (x86)/Zuragon/receive_test.js result -> Error: Timed out waiting for CAN frame
(Test with recieve Script) Script C:/Program Files (x86)/Zuragon/receive_test.js resulted in an exception: Error: Timed out waiting for CAN frame at line 3
(Test with recieve Script) <global>() at C:/Program Files (x86)/Zuragon/receive_test.js:3
(C:/Program Files (x86)/Zuragon/receive_test.js) Script C:/Program Files (x86)/Zuragon/receive_test.js result -> Error: Timed out waiting for CAN frame
```

1.7.1 J1939 SOURCE OBJECT

PROTOTYPE	DESCRIPTION
send(sa,da,priority,pgn,byte[] data)	Send J1939 message addressed to a node in the network
sendBAM((sa,da,priority,pgn,byte[] data)	Send a broadcast J1939 message
receive(timeout_in_ms = ,infinite)	Receive a J1939 message, if nothing is received in timeout_in_ms script execution is aborted with an exception

1.7.2 ISO15765 SOURCE OBJECT

BASIC CONCEPTS AND ABBREVIATIONS

TYPE	DESCRIPTION
PCI	Protocol Control Information. May be single-Frame, First-Frame, Flow-Control or Consecutive-Frame. Find more details in the ISO15765 specification
Data	The payload of an ISO15765 message. Find more details in the ISO15765 specification
M Type	May be diagnostics or remote diagnostics. The parameter Mtype shall be used to identify the type and range of address information parameters included in a service call. This part of ISO 15765 specifies a range of two values for this parameter. The intention is that users of the documents can extended the range of values by specifying other types and combination of address information parameter to be used with the network layer protocol specified in this document. For each such new range of address information a new value for the Mtype parameter shall be specified to identify the new address information. - If Mtype = diagnostics then the address information AI shall consist of the parameters SA,TA and TAtype - If Mtype = remote diagnostics then the parameters AI shall consist of the parameters SA, TA, TAtype and AE
AI	These parameters refers to addressing information. As a whole, the AI parameters are used to identify the source address (SA), target address (TA) of message senders and recipients as well as the communication model for the message (TAType) and the optional address extension (AE)
SA	Network Source Address, 1 byte unsigned integer value range 00-FF hex. The SA parameter shall be used to encode the sending network layers protocol entity
TA	Network Source Address, 1 byte unsigned integer value range 00-FF hex. The TA parameter shall be used to encode the sending network layers protocol entity
TAType	Network Target Address type physical or functional. Physical addressing (1-1 communication) shall be supported
AE	Network Address Extension, 1 byte unsigned integer value, range 00-FF Hex. The AE parameter is used to extend the available address range for large networks, and tonencode both sending and receiving network layers entities of subnets other than the local network where the communication takes place. AE is only part of the addressing information if Mtype is set to remote diagnostics

ADDRESSING MODES

Normal addressing-

For each combination of SA, TA, TAtype and Mtype, a unique CAN identifier is assigned. PCI and Data is placed within the CAN frame data. For this mode an addressing map must be defined, see <iso15765-source>.setAddressMap (address_map)

Fixed addressing-

Normal fixed addressing is a sub format of normal addressing where the mapping of the address information into the CAN identifier is further defined. In the general case of normal addressing, described above, the correspondence between AI and the CAN identifier is left open. For normal fixed addressing, only 29 bit CAN identifiers are allowed.

Extended addressing-

For each combination of SA, TAtype and Mtype, a unique CAN identifier is assigned. TA is placed in the first data byte of the CAN frame data. PCI and Data is placed in the remaining bytes of the CAN frame data field. For this mode an addressing map must be defined, see <iso15765-source>.setAddressMap (address_map)

Mixed addressing-

Mixed addressing is the addressing format to be used if Mtype is set to remote diagnostics.

29 bit CAN identifier-

The address information (AI) is in the 29 bit CAN identifier, and the first CAN frame data byte shall be the AE.

11 bit CAN identifier-

For each combination of SA, TA and TAtype a unique CAN identifier is assigned. AE is placed in the first data byte of the CAN frame data. PCI and Data is placed in the remaining bytes of the CAN frame data field. For this mode an addressing map must be defined, see <iso15765-source>.setAddressMap (address_map)

CONSTANTS

```
tp.iso15765.NormalAddressMode         = 0x0;
tp.iso15765.ExtendedAddressMode       = 0x1;
tp.iso15765.FixedAddressMode          = 0x2;
tp.iso15765.mixedAddressMode          = 0x3;
tp.iso15765.Physical                  = 0x0;
tp.iso15765.functional                = 0x1;
tp.iso15765.ExtAddrFlag               = 0x20000
tp.iso15765.UnknownTypeFlag           = 0x40000
```

METHODS

PROTOTYPE	DESCRIPTION
setAddressMap(address_map)	Set the address map, used for Normal, Extended and Mixed addressing (using 11bit CAN identifier), see examples below
Send(sa,ta,ta_type.id.dat)	Send a ISO 15765 message to from SA addressed to TA
sendNormalRaw(can_request_id.can_response_id.data)	Send an ISO 15765 message specifying the request and response CAN ID. Note: this function can only be used when in Normal addressing mode. When sending function or single frame messages can_response_id is allowed to be set to null
Receive(timeout_in_ms = <infinite>)	Receive an ISO15765 message if nothing is received in timeout_in_ms script execution is aborted with an exception. On successful receive of a message an array is returned in the following format; {<CAN-Id>,<SA>,<TA>,(TAType>,<Flags>,<data>} If address mode is MixedAddressMode the array will also contain AE, Format:{<CAN-ID>,<SA>,<TA>,<TAType>,<AE>,<Flags>,<Data>}

DEFINING AN ADDRESS MAP

For Normal, Extended, and Mixed address mode (with 11bit CAN identifiers) an address map must be defined. A unique CAN identifier is defined for each combination of SA, TA and TAType. Note that for extended address mode only SA and TAType is used.

Example defining an address map having 3 nodes in the network, with addresses 1, 2 and 5:

```
var address_map = [ { id:0x242, sa:5, ta:1,
ta_type:tp.iso15765.Physical },
                    { id:0x243, sa:5, ta:2,
ta_type:tp.iso15765.Physical },
                    { id:0x542, sa:1, ta:2,
ta_type:tp.iso15765.Physical },
                    { id:0x543, sa:2, ta:2,
ta_type:tp.iso15765.Physical },
                    { id:0x643, sa:1, ta:2,
ta_type:tp.iso15765.Physical },
                    { id:0x843, sa:2, ta:2,
ta_type:tp.iso15765.Physical } ]

project.iso_source.setAddressMap (address_map)
```

EXAMPLE SENDING A SINGLE-FRAME

```
// SA 1 TA 2 TAType Physical
iso_source.send(1,2, tp.iso15765.Physical, [22,23,24,25,26,27]);
```

EXAMPLE SENDING A MULTI-FRAME

```
var packet =[] ;
    for ( i=0; i<33; ++i) {
      packet.push(i)
}
// SA TA 5 TAType Physical
iso_source.send(2,5,tp.iso15765.Physical,packet);
```

1.8 LIN SOURCE OBJECT

CONSTANTS

```
lin.flag.ParityError            = 0x00001 // Rx: parity error(the identifier)
lin.flag.ChecksumError          = 0x00002 // Rx: checksum error
lin.flag.NoData                 = 0x00004 // Rx: header only
lin.flag.BitError                = 0x00008 // Tx: transmitted 1, got 0 or vice versa
lin.flag.TxSlaveResponse         = 0x00010 // Rx: echo of a slave response we transmitted
lin.flag.ClassicChecksum         = 0x00020 // Rx or Tx
lin.flag.TxMsgAcknowledge        = 0x00040 // Tx message acknowledge
lin.flag.TxMsgRequest            = 0x00080 // Tx message request
lin.flag.ErrorHWOverrun          = 0x00200 // Rx: LIN interface overrun
lin.flag.ErrorSWOverrun          = 0x00400 // Rx: receive queue overrun
lin.flag.SynchError              = 0x00800 // Synch error
lin.flag.WakeUp                  = 0x01000 // Awake up frame was received
```

METHODS

PROTOTYPE	DESCRIPTION
Send(unit id, byte[] data)	Send data on the LIN bus
Array receive (timeout_in_ms = <infinite>)	Receive data on the LIN bus, if nothing is received in timeout_in_ms script execution is aborted with an exception
SetSlaveResponse(uint id, byte[] data)	Only for slave. Set or update a message response data for a specified ID
clearslaveResponse(unit id)	Only for slave. Clear a message response
clearslaveResponse()	Only for slave. Clear all message response
sendMasterRequest(uint id)	For master node, request a
sendWakeUp()	Send a wake-up frame

LIN MASTER AND SLAVE EXAMPLE

```
var master = project.master_lin_channel
var slave = project.slave_lin_channel

slave.setSlaveResponse(10, [1,2,3,4,5,6,7,8])
slave.setSlaveResponse(11, [10,20,30,40,50,60,70,80])
slave.setSlaveResponse(12, [11,21,31,41,51,61,71,81])
slave.setSlaveResponse(13, [12,22,32,42,52,62,72,82])
slave.setSlaveResponse(14, [13,23,33,43,53,63,73,83])
slave.setSlaveResponse(15, [14,24,34,44,54,64,74,84])

project.log("send master request")
for (var i = 10; i <=15; ++i )
{
    Master. SendMasterRequest(i)
    var response
    do {
        response +slave.receive (100)
         } while (response [0] ! = i)
     Project.log ("Slave response: " + response)
}
```

Chapter

2

2. QML_EXTENSION

ViCANdo can be extended with custom functionality in form of QML presenters. To use a QML component in ViCANdo, from the Presenter menu select QML Presenter. Choose QML source.

NOTE: that the project only has references to the external QML source files. If project is moved to another computer, the QML sources must also be available on the other computer at the same location, in order for the project to work properly.

2.1 ABOUT QML

QML (Qt Meta Language or Qt Modelling Language) is a JavaScript-based, declarative language for designing user interface–centric applications. It is part of Qt Quick, the UI creation kit developed by Nokia within the Qt framework. QML is mainly used for mobile applications where touch input, fluid animations (60 FPS) and user experience are crucial. QML documents describe an object tree of elements. QML elements shipped with Qt are a sophisticated set of building blocks, graphical (e.g., rectangle, image) and behavioral (e.g., state, transition, animation). These elements can be combined to build components ranging in complexity from simple buttons and sliders, to complete internet-enabled programs.
QML elements can be augmented by standard JavaScript both inline and via included .js files. Elements can also be seamlessly integrated and extended by C++ components using the Qt framework.

HELLO WORLD EXAMPLE

A simple QML example that just displays Hello World with white text on a black background

```
import QtQuick 1.1

Rectangle {
Colour: “#000000”

Width: 100; height: 100

Text (
colour: “#FFFFFF”
anchors,fill: parent
horizontalAlignment : Text.AlignHCenter
verticalAlignment : Text.AlignVCenter
text: "Hello World"
}
}
```

2.2 BASIC QML ELEMENTS

Find out more at the Qt Project [1]

Item [2]	The item is the most basic of all visual items in QML
Rectangle [3]	The Rectangle items provides a filled rectangle with an optical border
Image[4]	The Image elements displays an image in a declarative user interface
Text[5]	The text item allows you to add formatted text to a scene
TextInput[6]	The TextInput item displays an editable line of text
TextEdit[7]	The TextEdit item displays multiple lines of editable formatted text
FocusScope[8]	The FocusScope object explicitly creates a focus scope
Component[9]	The component element encapsulates a QML component definition
MouseArea[10]	The MouseArea item enables simple mouse handling
Timer[11]	The Timer item triggers a handler at a specified interval

For a complete list of QML elements, please visit the QML Elements [12] page, from the Qt Projects page.

2.3 CONTEXT PROPERTIES

MAIN_SOURCE

This property provides the main source attached to the QML presenter.

SELF

This property provides a reference to its own presenter object.

METHODS

PROTOTYPE	DESCRIPTION
logMessage(string message)	Log a text message to the project console
Log(string message)	Same as logMessage method
forceReload()	Force reload of the QML component

PROJECT

This property provides project resources.

METHODS

PROTOTYPE	DESCRIPTION
logMessage(String Message)	Log a text Message to the project console
Log(String Message)	Same as logMessage method
Object Scriptlet(String name)	Get a reference to Scriptlet matching name
runScriptlet(String name)	Run a Scriptlet matching name
Bool isScriptletRunning (stringname)	Return true, if Scriptlet matching name is currently running
Object findObject(String Object_name)	Get a reference to a project component matching Object_name
clearConsole()	Clear the ViCANdo text console
StoreValue(String key, value, persistent = False)	Store a value within the project. The value will be permanent until the project is closed or the script-engine is restarted. If persistent parameter, the value will be permanently stored within the project. Note: only string values can be permanently stored
RemoveValue(String Key)	Remove a stored value
<value>value(key)	Retrieve a stored value with StoreValue(String key, value)
ClearStoredValues()	Remove all stored values
seekToPosition(time_in_ms)	Seek to position in the current selected session, time_in_ms is time given in micro-seconds

PROPERTIES

NAME	DESCRIPTION
Source[]	Array of sources available in project
Presenter[]	Array of presenters available in project
Scriptlet[]	Array of Scriptlets available in project
Trigger[]	Array of triggers available in project
Dbc[]	Array of DBCs available in project
projectDirectory	Absolute path to the location of the current project
currentSession	Current selected session objects, null if no session is selected

An example using the project property:

```
import QtQuick 1.1
Text {
Width: 100; height: 100
horizontalAlignment : Text.AlignHCenter
verticalAlignment : Text.AlignVCenter
text: “click on me”

MouseArea {
Anchors.fill: parent
onClicked: {
project.logMessage(“clicked on” + MouseX + “,” + MouseY);
}
}
}
```

SOURCE [ ]

This property provides an array of all sources available in project. Is mainly just an alias to project.Source [ ]

Methods (available on all kind of sources)

PROTOTYPE	DESCRIPTION
logMessage(string message)	Log a text message to the project console

CURRENT SESSION

This property provides access to the current selected session

Methods (available on all kinds of sources)

PROPERTY	DESCRIPTION
displayName	Session display name
CommentText	Session comment text
startDateTime	Session record date time
timeInUs	Current play/record position in micro-seconds
timeInMs	Current play/record position in milli-seconds
startTimeOffsetInUs	Start time offset of the session in micro-seconds >0 in case the session has been cropped
endTimeInUs	End time of the session in micro-seconds
Cropped	Set to true if the session has been cropped

2.4 CAN SOURCE SPECIFIC

Methods available on CAN sources

PROTOTYPE	DESCRIPTION
Send(uint id, byte[] data)	Send a CAN frame, extended is default
SendExtended(int id, byte[] data)	Send an extended CAN frame
sendStandard(int id, byte[] data)	Send a standard CAN frame
SendRemote(int id, byte[] data)	Send a remote CAN frame, default is extended
sendStandardRemote(int id, byte[] data)	Send a standard remote CAN frame
sendExtendedRemote(int id, byte[] data)	Send an extended remote CAN frame

Example that sends a CAN frame on mouse click:

```
import QtQuick 1.1
Text {
width: 100; height: 100
horizontalAlignment : Text.AlignHCenter
VerticalAlignment : Text.AlignVCenter
text: "click on me to send a CAN frame on source[0]"

MouseArea {
anchors.fill: parent
onClicked: {
source[0].send(100, [1,2,3,4,5,6,7,8])};
}
}
}
```

2.5 GPS SOURCE SPECIFIC

PROPERTIES AVAILABLE ON GPS SOURCES

NAME	DESCRIPTION
updateRate	Update rate in milli-seconds GPS events are dispatched changing this property will set the GPS device update rate, and the source internal update rate
minimalUpdateRate	Minimal update rate in milli-seconds that data events are dispatched. Data events will not be dispatched with lower time interval then what's set on this property
deviceUpdateRate	The GPS device update rate. When setting this property vendor specific NMEA command will sent to the GPS devices

METHODS AVAILABLE ON GPS SOURCES

PROTOTYPE	DESCRIPTION
sendNMEA(string nmea_line)	Send a NMEA sentence to the GPS device the starting $ and checksum and CR, LF at the end of the line is automatically added
sendRawNMEA(string raw_nmea_line)	Send a NMEA sentence the starting $ and checksum is not added, CR, LF is added at the end of the line before sent to the device

EXAMPLE THAT SENDS NMEA SENTENCE ON MOUSE CLICK:

```
import QtQuick 1.1
Text {
width: 100; height: 100
horizontalAlignment : Text.AlignHCenter
VerticalAlignment : Text.AlignVCenter
text: "click on me to send a NMEA sentence on main_source "

MouseArea {
anchors.fill:parent
onClicked: {
/*PMTK command that will set the update rate to 500ms*/

main_source.sendNMEA("PMTK220,500")
}
}
}
```

2.6 COMPONENTS

SOURCE EVENT LISTENER

Use this component to catch data events from a source.

PROPERTIES

NAME	DESCRIPTION
Source	The source to receive data events from use main_source or source{0…n}

Example receiving data events from a CAN source:

```
import QtQuick 1.1
import com.zuragon.ViCANdo 1.0

Text {
id: root_item
colour: "#0000FF"
width: 500; height: 50

horizontalAlignment : Text.AlignHCenter
VerticalAlignment : Text.AlignVCenter

text: "No data received yet"

SourceEventListener {
source: main_source
onDataRecived {
root_item.text = "CAN frame received time" + data. Time + "ID" +data.id + "flags" +data.
Flags +"data" + data.dat;
}
}
}
```

NMEA DATA LISTENER

Use this component to capture NMEA data from a GPS device

Properties

NAME	DESCRIPTION
Source	The GPS source to receive NMEA data from use main_source or source(0…n)
Filter	A simple text filter, to only receive NMEA lines matching filter

Example:

```
import QtQuick 1.1
import com.zuragon.ViCANdo 1.0

Text {
id: root_item
colour: "#0000FF"
width: 500; height: 50

horizontalAlignment : Text.AlignHCenter
VerticalAlignment : Text.AlignVCenter

text: "No NMEA data received yet"

NMEADataListener {
source: main_source
filter: "GPRMC"

onNmeaData: {
root_item.text = nmea_line
}
}
}
```

PROJECT STATE EVENT LISTENER

Use this component to capture state-changes.

Example:

```
import QtQuick 1.1
import com.zuragon.ViCANdo 1.0

Rectangle {
id: main
width: 700
height: 360

ProjectStateEventListener {
id: project_state_listener

onIdle: {
project_state.text = "Idle"
project.log("Project Idle" + time_in_us)
}

onPreparing; {
project_state_text = "Preparing"
project.log("Project Preparing" + time_in_us)
}

onArmed: {
project_state.text = "Armed"
project.log("Project Armed" + time_in_us)
}

onPlaying: {
project_state.text = "Playing"
project.log("Project Playing" + time_in_us)
}
}
}
```

Chapter 3

3. APPENDIX

3.1 REFERENCES -

- http:/ / qt-project. org/ doc/ qt-4. 8/ qmlbasicelements. html
- http:/ / qt-project. org/ doc/ qt-4. 8/ qml-item. html
- http:/ / qt-project. org/ doc/ qt-4. 8/ qml-rectangle. html
- http:/ / qt-project. org/ doc/ qt-4. 8/ qml-image. html
- http:/ / qt-project. org/ doc/ qt-4. 8/ qml-text. html
- http:/ / qt-project. org/ doc/ qt-4. 8/ qml-textinput. html
- http:/ / qt-project. org/ doc/ qt-4. 8/ qml-textedit. html
- http:/ / qt-project. org/ doc/ qt-4. 8/ qml-focusscope. html
- http:/ / qt-project. org/ doc/ qt-4. 8/ qml-component. html
- http:/ / qt-project. org/ doc/ qt-4. 8/ qml-mousearea. html
- http:/ / qt-project. org/ doc/ qt-4. 8/ qml-timer. html
- http:/ / qt-project. org/ doc/ qt-4. 8/ qdeclarativeelements. html

3.2 ALL SOURCES AND CONTRIBUTORS-

QML_extension –
Source: http://192.168.0.31/w/index.php?title=QML_extension Contributors: Benny, 1 anonymous edits
JavaScript extension –
Source: http://localhost/w/index.php?title=JavaScript_extension Contributors: Benny, Joachim, 12 anonymous edits

3.3 IMAGE SOURCES AND LICENSES-

File: Recieve_CAN_message.jpg
Source: http://localhost/w/index.php?title=File:Recieve_CAN_message.jpg License: unknown Contributors: Joachim