

ISOBUS Workshop

Example Application:

Tutorial for ISO-AgLib

- ▶ **Documentation Structure of ISO_{AgLib} Version 2.1**
- ▶ **Setup of Development Environment**
- ▶ **Virtual CAN BUS with CAN_SERVER**
- ▶ **Getting Started – “Hello World” in ISOBUS World**
- ▶ **Accessing Part7 Application Layer PGN**
- ▶ **VT-Client Support for GUI Development**
- ▶ **Application Integration**
- ▶ **Outlook**

▶ **ISO_{AgLib} Overview with latest news**

- <http://www.isoaglib.org/index.html#LatestNews>

▶ **Download & Installation**

- package download - <http://www.isoaglib.org/PageDownload.html#HowtoDownload>
- SVN repository access - <http://www.isoaglib.org/PageDownload.html#HowtoAccessRepository>
- Installation with compiler preparation, project file setup and usage of main tools <http://www.isoaglib.org/PageInstallation.html>

▶ **Developer Information**

- Getting started with ISO_{AgLib} with Hands-On Steps, Documentation Overview Pages and Getting Help - <http://www.isoaglib.org/PageDeveloperInfo.html>
- Changelog - <http://www.isoaglib.org/PageChanges.html.html>

▶ **Technical Information**

- General Information - <http://www.isoaglib.org/PageTechInfo.html#InfGeneralInformation>
- Structural Overview - <http://www.isoaglib.org/PageTechInfo.html#StructuralOverview>

▶ **Licensing - <http://www.isoaglib.org/PageLicensing.html>**

▶ **Contact & Support - <http://www.isoaglib.org/PageContactAndSupport.html>**

▶ Accessing SVN Repository

- Description at *Download & Installation* → *Accessing the Repository*
- <http://www.isoaglib.org/PageDownload.html#HowtoAccessRepository>
- Windows SVN Tool: TortoiseSVN (<http://tortoisesvn.tigris.org/>)
- HEAD of Development:
<http://projects.osb-ag.de/svn/OSB/IsoAgLib/IsoAgLib>
- Maintained Releases:
<http://www.isoaglib.org/svn/OSB/IsoAgLib/releases/maintained/>
- Snapshot Releases:
<http://www.isoaglib.org/svn/OSB/IsoAgLib/releases/snapshot/>

▶ update_makefile.sh

- Description at *Download & Installation* → *Installation* → *Project File Creation* –
<http://www.isoaglib.org/PageInstallation.html#HowtoCreateProjectFiles>
- Supports easy project setup
- Needs MSYS (<http://www.mingw.org/msys.shtml>) for Windows hosts
- Interacts best with Dev-C++ (<http://www.bloodshed.net/dev/devcpp.html>)
- Typical call: `./update_makefile.sh –target-system=pc_win32 –pc-can-driver=socket_server conf_3_0_VirtualTerminalIso`

▶ Understanding and Adapting Project Independent Settings

- Basic Hardware Setup of Target System (e.g. count of CAN ports)
- Build System Configuration for Cross Compilers (like TASKING EDE)
- Build System Configuration for Win32 (like Visual C++ and Dev-C++)
- Build System Configuration for gcc based cross compilers and the like

▶ vt2iso

- Description at *Download & Installation* → *Installation* → *Installing vt2iso* - <http://www.isoaglib.org/PageInstallation.html#InstallIsoToolVt2Iso>
- Normally the EXE of a current version is directly supplied by SVN
- Individual Recompile with Dev-C++
 - Install Xerces-C (<http://devpaks.org/show.php?devpak=41>), *FreeImage* and *ImageLib* with the package manager of Dev-C++
 - Open project file in [IsoAgLib/library/xgpl_src/build/vt2iso/vt2Iso.dev](#)
 - Start project compile
 - Copy xerces-c_2_5_0.dll and FreeImage.dll to system PATH (I.e. so that vt2iso.exe finds them)

▶ proc2iso

- <http://www.isoaglib.org/XMLProcSpec.html>
- Normally the EXE of a current version is directly supplied by SVN
- Individual Recompile with Dev-C++
 - Install Xerces-C (see above)
 - Open project file in [IsoAgLib/library/xgpl_src/build/proc2iso/proc2Iso.dev](#)

▶ **Support for Windows and Linux**

▶ **Main Target:**

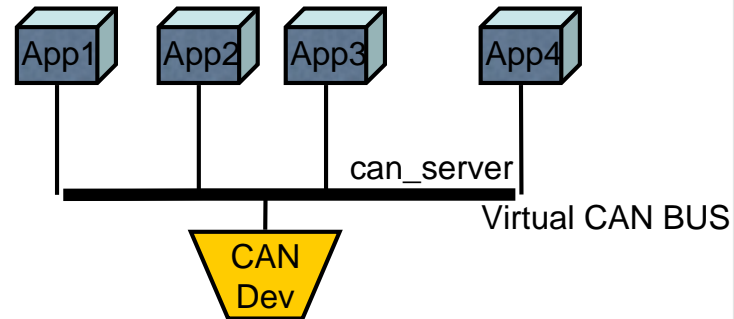
- Several applications communicate with each other and shared CAN device
- Separate application from CAN device specific details

▶ **Features:**

- Pure virtual operation without any real CAN device supported
- Provide optional shortcut for targeted messages between internal applications
- Provide file logging and replay of CAN log

▶ **Build:**

- Description at *Download & Installation* → *Installation* → *Project File Creation*
- <http://www.isoaglib.org/PageAdditionalInfo.html#CanServerApp>
- Optional, as current EXE versions are available from SVN
- Project file from [IsoAgLib/examples/compiler_projects/Dev-C++/canserver](#) and [IsoAgLib/examples/compiler_projects/VC8/canserver/](#)



▶ **Functional Overview of Tutorial 0_3_LookupIso**

- http://www.isoaglib.org/d2/dc6/0_3_LookupIso_8cpp-example.html
- Perform Address Claim of Local Ident (ECU)
- Lookup information about remote network nodes

▶ **Source Code Overview**

- <http://www.isoaglib.org/NetworkMgmtPage.html>
- Network management at
library/xgpl_src/IsoAgLib/comm/Part5_NetworkManagement/
- CAN access with library/xgpl_src/IsoAgLib/comm/Part3_DataLink/ and
library/xgpl_src/IsoAgLib/driver/can/
- Example code at
examples/src/Tutorials/0_NetworkManagement/0_3_LookupIso/
- Feature activation in project config file with “PRJ_ISO11783=1”
- Main classes: ildentItem_c, ilsoMonitor_c, iScheduler_c, iCanIo_c,
iSystem_c

▶ **Targets of ISO 11783 Part 7 Management**

- Management of ISO 11783 Part 7 (full)
- Individual class per PGN
- Time decoupling: store value updates from TECU for time independent value access from main application
- Perform automatic periodic send where needed

▶ **Available Handler Classes (configurable as implement or tractor)**

- **iTracGeneral_c** for tractor data like hitch, link force, Power State, TECU Language
- **iTracMove_c** for tractor driving information like speed
- **iTracPto_c** for tractor PTO information
- **iTimePosGps_c** for NMEA2000 access and calendar information
- Next classes are only available for registered users:
 - **iTracMoveSetpoint_c** handler for implement controls tractor speed
 - **iTracPtoSetpoint_c** handler for implement controls tractor PTO
 - **iTracLight_c** for management of lighting information
 - **iTracAux_c** for control of tractor AUX elements
 - **iTracCert_c** for certification handling
 - **iTracGuidance_c / iTracGuidanceCommand_c** for guidance control

- ▶ **Functional Overview of Tutorials 1_0_ReadIso and 1_2_WriteIso**
 - 1_0_ReadIso.cpp represents typical implement which listens to information from tractor and other information providers
 - http://www.isoaglib.org/d3/d3d/1_0_ReadIso_8cpp-example.html
 - 1_2_WriteIso.cpp represents tractor, which sends information on ISOBUS
 - http://www.isoaglib.org/d0/de1/1_2_WriteIso_8cpp-example.html

- ▶ **Source Code Overview**
 - <http://www.isoaglib.org/AppLayerPage.html>
 - Application layer access at
library/xgpl_src/IsoAgLib/comm/Part7_ApplicationLayer/
 - Feature activation in project config file with “PRJ_BASE=1” or selective like “PRJ_TRACTOR_GENERAL=1”

- ▶ **Exemplary Source Code Snippets**
 - Define role with function calls like `getTracMoveInstance().config(&c_myIdent.isoName(), IsoAgLib::IdentModeTractor);`
 - Implement: Read ground based tractor speed with `getTracMoveInstance().speedReal()`
 - Tractor: Set new front draft with `getTracGenerallInstance().setHitchFrontDraft(250);`

▶ **Targets of ISO 11783 Part 6 Management**

- Abstract from details of handshaking, transport protocol and VT property negotiation
- One pool definition fits all VT types – on-the-fly adaptation
- Automatic scaling of sizes, positions and font-size
- Adaptation of colors either automatically or individually (e.g. turn all background colors to white for b&w terminal)
- Multi-Language support with language code table files and language dependent content (e.g. replace output string with bitmaps for some languages)
- Distribution of mask definition to multiple files allows to establish company and product line standards for styles (font attributes for some kinds of content), complete masks and mask content containers
- XML Syntax supports flexible extensions to steer terminal adaptation (e.g. provide different bitmaps based on color depth of terminal)
- Event handlers for simple reaction on operator input

▶ **Toolchain for editing of XML based pool definition**

- Direct Editing of XML-Syntax (esp. in combination with scripts) and vt2iso comparable to raw HTML design of websites
- **PoolEdit**
 - Open Source
 - From <http://automation.tkk.fi/Farmix/PoolEdit>
 - JAVA Application
 - Wizards for table and meter
- **OSB Graphic GUI-Builder**
 - Prototype spring 2008
 - Based on same VT engine as AGCO VT Server
 - Provides live execution of pool (input dialogs, macros) and real VT mode with upload of CAN based pools
 - Strong validation from VT engine (static and dynamic checks)
 - Direct support of all possibilities of language dependent pool content
 - Project file based control of editable and readonly XML-Files (e.g. for company global standard content and standard format)
 - Multiple selection with deep copy, group definition
 - Strict template based creation of single items or groups of items

► Documentation

- XML-Syntax – <http://www.isoaglib.org/XMLspec.html>
- Installation of tool vt2iso - <http://www.isoaglib.org/PageInstallation.html#InstallIsoToolVt2Iso>
- Transport protocol - <http://www.isoaglib.org/DataLinkPage.html>

► XML Pool Structure

- Detailed documentation in <http://www.isoaglib.org/XMLspec.html#XMLGeneralStructure>
- Objectpool definition in XML-File which must be the first in alphabetical order of all XML-Files for the pool (e.g. simpleVTIsoPool.xml-0_WorkingSetAndVariables)
 - `<?xml version="1.0" encoding="iso-8859-1"?>`
 - `<objectpool dimension='200' std_bitmap_path='bitmaps' fix_bitmap_path='bitmaps-vario240'>`
 - `<workingset name='MyWorkingSet' background_colour='white' selectable='yes' active_mask='MyDataMask1'>`
 - `<!-- WorkingSet Descriptor Object shall be included here -->`
 - `</workingset>`
 - `<!-- All other objects can be defined now... -->`
 - `</objectpool>`

▶ XML Pool Structure (continued)

- Data Mask content in other XML-Files
(e.g. simpleVTIsoPool.xml-1_DataMask1)
 - `<?xml version="1.0" encoding="iso-8859-1"?>`
 - `<objectpool> <!-- dimension and bitmap_path needn't (and mustn't) be defined again! -->`
 - `<datamask id='0815' name='MyDataMask1'
background_colour='white' soft_key_mask='MySoftKeyMask1' >`
 - `<container name='MyDataMask1Container1' pos_x='0'
pos_y='0' width='200' height='100'>`
 - `<!-- object included in the container following... -->`
 - `</container>`
 - `<!-- more objects in this datamask can be defined now. -->`
 - `</datamask>`
 - `<!-- still more objects in this objectpool can be defined now... -->`
 - `</objectpool>`

▶ Event Handlers

- Detailed documentation in <http://www.isoaglib.org/XMLspec.html#vt2isoImplementation>
- Direct Mode with standalone pool handling class
- Derived Mode with application scope event handler class being derived from pool handling class
- Derived Mode allows to GUI state information and other internal information with event handler functions

- **Necessary event handlers**
 - void eventKeyCode ()
 - void eventNumericValue ()
 - void eventStringValue ()
 - void eventObjectPoolUploadedSuccessfully ()
 - void eventEnterSafeState ()

- **Optional event handlers (Selection)**
 - virtual void eventPointingEvent ()
 - virtual uint8_t convertColour ()
 - virtual void eventVtStatusMsg ()

▶ Multi Language Support

- Workingset definition with list of supported languages

```
<workingset name='MyMultiLanguageWorkingSet' background_colour='white'  
selectable='yes' active_mask='MyDataMask1'>  
  <!-- WorkingSet Descriptor Object shall be included here -->  
  <language code='en' /> <!-- First language is the default language! -->  
  <language code='de' />  
  <language code='it' />  
</workingset>
```

- Object definition with language dependent alternatives

```
<stringvariable name='StrNone' language="en" value='none selected!' />  
<stringvariable name='StrNone' language="de" value='nichts  
ausgew&auml;hlt' />  
<stringvariable name='StrNone' language="it" value='niente selezionato!' />  
  
<stringvariable name='StrRed' language="en+it" value="red" />  
<stringvariable name='StrRed' language="de" value="rot" />  
  
<stringvariable name='StrBlue' language="*" value="blue" />
```

▶ **Multi Language Support (continued)**

- Usage of external language files

simpleVTIsoPool.xml-2 MultiLanguageDataMask1:

```
<stringvariable name='StrYellow' language='en' value='yellow' />
```

```
<stringvariable name='StrYellow' language='de+it' />
```

simpleVTIsoPool.values.en.txt:

StrYellow, „yellow,„

simpleVTIsoPool.values.de.txt:

StrYellow, „gelb,„

▶ Pool Design Dependent VT-Client Module Compilation

- Documentation at <http://www.isoaglib.org/PageDeveloperInfo.html#VtClientRomReduction>
- Avoid inclusion of VT-Client vtobject modules which are not used for the pool, which can reduce Firmware size
- Vt2iso creates file “IsoTerminalObjectSelection.inc” with #define list of all used vtobject types
- Connect this to project with config file setting
//define relative path from first source path to the mask definition directory
PRJ_ISO_TERMINAL_OBJECT_SELECTION1="MaskDefinition“
- Independent of update_makefile.sh
#ifndef PRJ_ISO_TERMINAL_OBJECT_SELECTION1
 #define PRJ_ISO_TERMINAL_OBJECT_SELECTION1 MaskDefinition
#endif
- Up to four pools are supported with this approach
- If PRJ_ISO_TERMINAL_OBJECT_SELECTION[1|2|3|4] is not defined, all vtobjects are compiled into the project

▶ Tutorial 3_0_VirtualTerminalIso

- Documentation at http://www.isoaglib.org/d5/de2/3_0_VirtualTerminalIso_8cpp-example.html
- Source at examples/src/Tutorials/3_VirtualTerminal_Client/3_0_VirtualTerminalIso/
- Demonstrate typical set of event handlers
- Demonstrate partial pool update

```
static iVtClientServerCommunication_c* spc_tut30csc;
iVtObject_c* arrpc_vtObjectsToUpdate[] =
{&iVtObjectValSpeed, &iVtObjectValAccel, &iVtObjectBigLogo};
spc_tut30csc->sendCommandUpdateObjectPool
(arrpc_vtObjectsToUpdate,
sizeof(arrpc_vtObjectsToUpdate)/sizeof(iVtObject_c*));
```

▶ Tutorial 3_2_VirtualTerminalMultilso

- Documentation at http://www.isoaglib.org/df/d4c/3_2_VirtualTerminalMultilso_8cpp-example.html
- Source at examples/src/Tutorials/3_VirtualTerminal_Client/3_2_VirtualTerminalMultiIso/
- Demonstrate parallel support of more than one <WSM,Pool> from one program

▶ **Tutorial 2_11_TaskcontrollerClientVirtualTerminalClient**

- Documentation at http://www.isoaglib.org/d0/dd3/2_11_TaskcontrollerClientVirtualTerminalClient_8cpp-example.html
- Source at examples/src/Tutorials/2_ProcessData/2_11_TaskcontrollerClientVirtualTerminalClient/
- Demonstrate ECU which acts both as VT-Client and TC-Client with one Working Set Master

▶ **Integrate Application Activities in ISO_{AgLib} Task Scope**

- Documentation at http://www.isoaglib.org/de/db5/7_0_SchedulerTaskNoCan_8cpp-example.html
- **Tutorials**
 - 7_0_SchedulerTaskNoCan
 - 7_1_SchedulerTaskCanSingleChannel
 - 7_2_SchedulerTaskCanMultipleChannels
- Define ISO_{AgLib} Scheduler triggered Task with deriving from IsoAgLib::iSchedulerTask_c

```
class MyTask_0 : public IsoAgLib::iSchedulerTask_c {
public:
    void init() {};
    virtual bool timeEvent( void ) { /* periodic activity */ return true;};
    virtual const char* getTaskName() const {return "MyTASK_0";};
    /** ... */
};
```

- Control retrigger time and period with
 - changeRetriggerTime (int32_t i32_nextRetriggerTime, int16_t ai16_newTimePeriod=-1)
 - changePeriod(int16_t ai16_newTimePeriod=-1)

▶ Integrate Application Activities in ISOAgLib Task Scope (cont.)

- ISOAgLib supports equidistant triggering of the tasks `timeEvent()` function
- I.e. average of `timeEvent()` calls per total time stays constant independent of jitter
- Tasks should allow as big as possible jitter, so that ISOAgLib can handle dynamic loads in a well defined way
 - Earliest execution time in relation to standard trigger time
→ ISOAgLib can trigger this task on IDLE state, so that next tasks get more time for their execution
 - Latest execution time in relation to standard trigger time
→ task before can get more execution time to fulfill its duties

▶ Integrate Application Activities in Separate Thread

- Documentation at http://www.isoaglib.org/da/d62/6_0_VtThreads_8cpp-example.html
- Tutorial 6_0_VtThreads
- Central mutex for ISO_{AgLib} access is managed in central scheduler
- Either ISO_{AgLib} periodic actions (as part of Scheduler_c::timeEvent()) can access internal data structures or the application
- Application should perform only short actions with acquired mutex, to avoid too long blocking of important periodic ISO_{AgLib} activities

- Try to acquire the central mutex:
Int i32_mutexResult =
IsoAgLib::getISchedulerInstance().tryAcquireResource();
- Wait until central mutex is free (use tryAcquireResource() first):
i32_mutexResult =
IsoAgLib::getISchedulerInstance().tryAcquireResource();
- Release central mutex:
IsoAgLib::getISchedulerInstance().releaseResource();

▶ Features for next 2.1.2 Release

- File Server Client integration
- Adaption of singleton pattern for compilers like IAR
- Windows CAN-Server for Sontheim (esp. with currently non working CANAS PCMCIA card)
- Some more documentation enhancements

▶ Feature ideas which could be developed on customer request

- Flexible mask contents dependent on further VT properties (comparable mechanism to language adaption):
 - Data Mask size
 - Amount of virtual softkeys
- Project file creation for IAR compiler with update_makefile.sh
- Enhancement of TC Client with on-the-fly pool adaptation corresponding to VT Client
 - Language and unit settings based selection of items during upload (comparable to flexible adaptation to VT language)
 - Interface for runtime edit, add, remove of Device Description items with partial pool update
- ...?