Exporting MOSAIC Models to Chemcad

Gregor Tolksdorf

February 9, 2016

1 Introduction

This is a guide on how to export self-implemented models from MOSAIC to a Chemcad flowsheet taking advantage of Chemcad's user added module capabilities. Knowledge of modeling in MOSAIC and the use of connectors and ports are pre-requisites. No knowledge of C++ language is required, but little knowledge on setting up a simulation in Chemcad is expected. Please uncheck the 'Hide extensions for known file types' option in the windows system folder options.

2 Prepare Visual Studio Project

If not already done, migrate the visual studio project delivered with Chemcad to the current version of visual studio installed on the PC to use. The original project files can be found in the Chemstations folder, e.g.
C:\Program Files\Chemstations\user added modules\usradd

It is recommended not to touch this folder, but to copy it into a location where the current user does not need administrator rights to modify and save the files, e.g.
C:\user\Documents\Visual Studio 2015\Projects\DemoChemcadUserADD

Double-click the file *usradd.dsp* to let Visual Studio migrate the project (accept the unidirectional upgrade). Configure the Visual Studio project with the settings given in the list (compare to figures 1 and 2.)

- All Configurations
  - General settings, output directory:

        * $(SolutionDir) $(Configuration)\
    - General settings, intermediate directory
        * $(Configuration)\
    - Linker, General, output file
        * $(OutDir)$(TargetName)$(TargetExt)
    - Linker, General, additional library directories
        * C:\Program Files\Chemstations\user added modules\libr
    - Linker, Input, additional dependencies
        * ccxdll.lib;ccxdlg.lib; *(remove the path in front of the two existing libraries!)*

- Debug configuration
    - C/C++, Code generation: runtime library
        * Multithreaded-Debug-DLL
    - C/C++, Code generation: function level linking
        * Yes (/Gy)

- Release configuration
    - C/C++ Code generation: runtime library
        * Multithreaded-DLL

3 Create a MOSAIC Model

The first step is to create a MOSAIC model as one normally would. In this example, an isobar membrane separation unit without heat balance is to be simulated based on a MESHI equation system as described below:

$$F^F \cdot x_i^F = F^R \cdot x_i^R + F^P \cdot x_i^P \tag{1}$$

$$F^P = \theta \cdot F^R \tag{2}$$

$$\alpha = \frac{x_{i=1}^P}{x_{i=2}^P} \cdot \frac{x_{i=2}^R}{x_{i=1}^R} \tag{3}$$

$$\sum_{i=1}^{NC} x_i^P = 1 \tag{4}$$

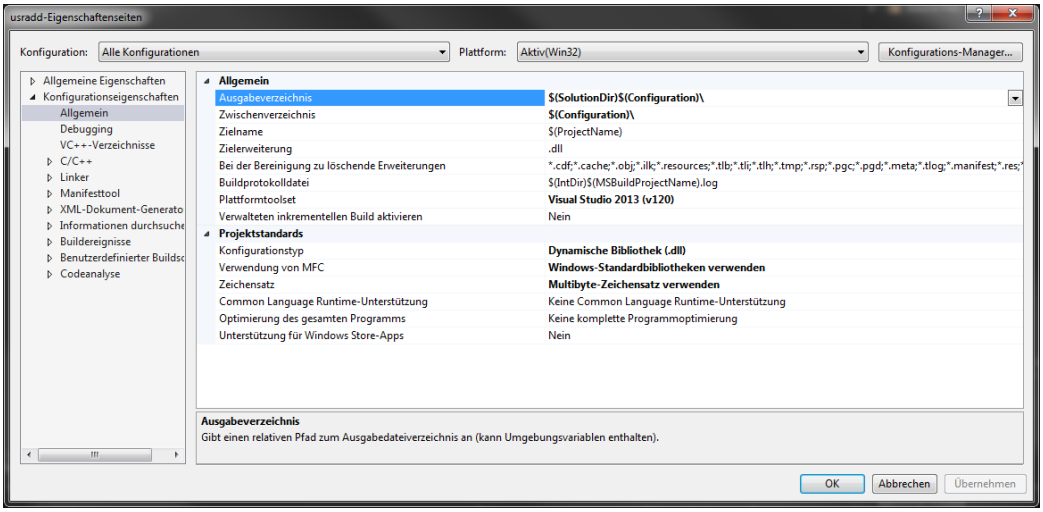


Figure 1: All configurations general settings

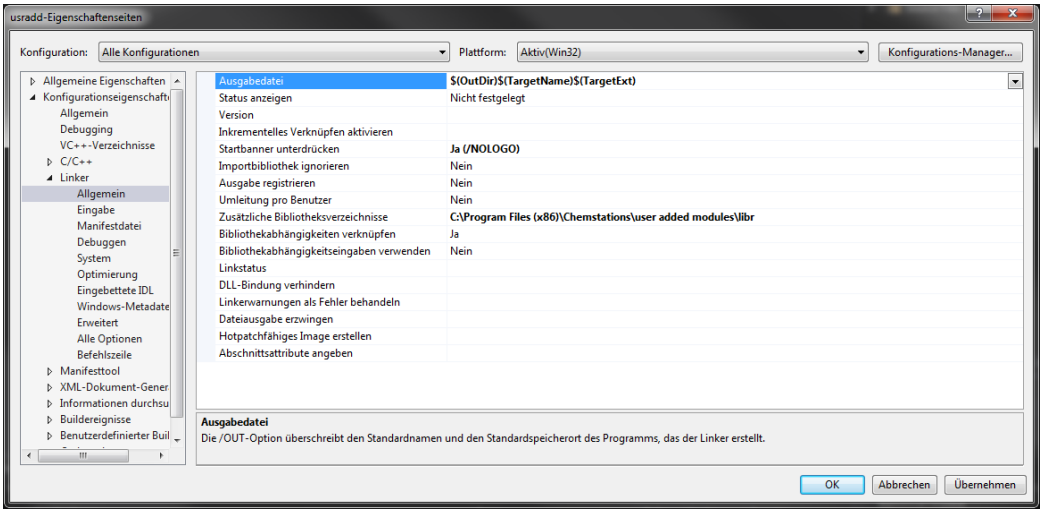


Figure 2: All configurations general linker settings

$$\sum_{i=1}^{NC} x_i^R = 1 \tag{5}$$

$$T^P = T^F \tag{6}$$

$$T^R = T^F \tag{7}$$

$$p^P = p^F \tag{8}$$

$$p^R = p^F \tag{9}$$

Create a parameter list containing the variables α and θ, and add it to

Table 1: Variable matching for the ports

Super Naming	Sub Naming feed	Sub Naming permeate	Sub Naming retentate
F	F^F	F^P	F^R
T	T^F	T^P	T^R
p	p^F	p^P	p^R
x_i	x_i^F	x_i^P	x_i^R

the equations where they occur. The variables of the parameter list will in the end be accesible via graphical user interface in the Chemcad user added module (UAM).

4 Introduce Ports

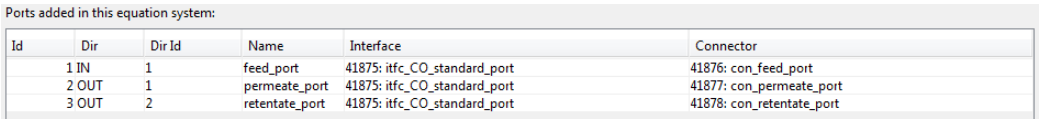
Ports are necessary to allow the model to be connected to streams. One inlet port (feed) and two outlet ports (permeate, retentate) are necessary. Every port contains a connector to translate the variable names, and an interface naming the required variables for the stream.
The interface containing the necessary variables has already been implemented an can be seen in figure 4.

Variable Naming	Name	Dim	Engin. Unit	In/Out
F	CO total molar flow	Scalar	mol/s	In/Out
N_i	CO component molar flow	Vector	mol/s	In/Out
T	CO temperature	Scalar	K	In/Out
h	CO molar enthalpy	Scalar	J/s	In/Out
p	CO pressure	Scalar	Pa	In/Out
x_i	CO overall mole fraction	Vector	mol/mol	In/Out

Figure 3: Interface for Chemcad Stream Connection

Connectors need to be created following the nomenclature matching as stated in table 1. It is not necessary to translate variables for molar volume as these variables are not included in the model.

Now the inlet and outlet ports can be created by selecting the direction and loading the interface and the connectors.



Ports added in this equation system:

Id	Dir	Dir Id	Name	Interface	Connector
1	IN	1	feed_port	41875: itfc_CO_standard_port	41876: con_feed_port
2	OUT	1	permeate_port	41875: itfc_CO_standard_port	41877: con_permeate_port
3	OUT	2	retentate_port	41875: itfc_CO_standard_port	41878: con_retentate_port

Figure 4: Ports for Chemcad Stream Connection

5 Code Generation

Open the newly created equation system in the *Evaluate/Simulation* tab and check that it contains the three ports (feed_port, permeate_port, retentate_port). Select the value (*Indexing* tab) for the index i (number of components) as 2 and confirm. At this point the evaluation should contain 10 equations.

In the *Info* tab, provide user-defined names for the different namespaces, e.g. e0==membrane, e0p1==feed, e0p2==permeate, e0p3==retentate.

In the *Variable Specification* tab, select all inlet stream's quantities (flow, temperature, pressure, and composition) as design variables. You can identify them by looking at the namespaces (e.g. e0p1==feed). Give proper starting values for the flow, composition, temperature and pressure of the outlet stream, save that specification as a variable specification list. In the *Parameter Specification* tab, you can set values for the parameters that will later on be accessible via the user interface in Chemcad. Save that specification as a variable specification list.

An example of the problem specification is given in the tables 2 and 3.

Save that evaluation (description!) before continuing. In the *evaluation* tab select 'C++ Kinsol ChemCad' in the (predefined) language specification, choose appropriate code generation settings (solver strategy, id of the ADD module, and Chemcad version), and press generate code. View the code and export it to a folder where you have write access. The following files are created during export:

- $ADDX.lab
- ADDX.cpp
- ADDX.map
- UserInterfaceInfo_LABfile.txt
- UserInterfaceInfo_MYfile.txt

The X is replaced by the id of the ADD module specified in the code generation settings. Read the txt files and modify the lab file accordingly (it is necessary to set the units of the parameters in case they are not dimensionless).

Table 2: Iteration Variables

Variable	Initial Value	Lower Bound	Upper Bound	Unit
F^P	38	0	1.0E09	mol/s
T^P	295	0	1.0E09	K
p^P	1e5	0	1E09	Pa
$x^P_{i=1}$	0.994	0	1	mol/mol
$x^P_{i=2}$	0.006	0	1	mol/mol
F^R	62	0	1.0E09	mol/s
T^R	295	0	1.0E09	K
p^R	1e5	0	1E09	Pa
$x^R_{i=1}$	0.7	0	1	mol/mol
$x^R_{i=2}$	0.3	0	1	mol/mol

Table 3: Design Variables and Parameters

Variable	Value	Unit
F^F	100	mol/s
T^F	295	K
p^F	1e5	Pa
$x^F_{i=1}$	0.85	mol/mol
$x^F_{i=2}$	0.15	mol/mol
θ	0.6	-
α	70	-

6 Chemcad User Form (MY-file)

In order to give the user the opportunity to enter values for the parameters in Chemcad, it is necessary to build a user form. This is done with *ScrBuild.exe* that can be found in the Chemcad folder, e.g.
C:\Program Files\Chemstations\CHEMCAD\ScrBuild.exe

For a minimal configuration start ScrBuild.exe, open a new file, and save it as ADDX.my in the same directory as the ADDX.lab and ADDX.map files. Add two *Buttons* and two *EditDblBoxes* to the form and assign the values and types as indicated in figure 5. Feel free to add text fields (labels) and *text variables* (e.g. linked to the unit operation id) as you (and the future user of your form) like.

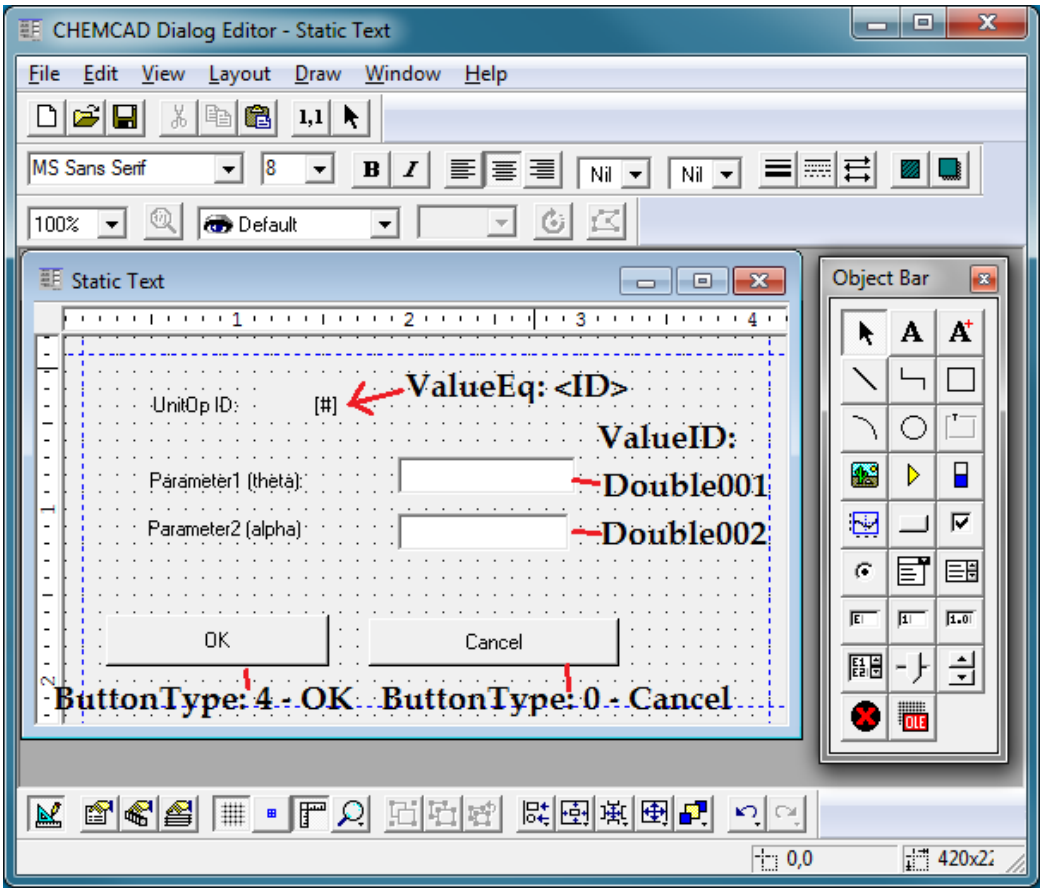


Figure 5: Simple user form with ScrBuild

7 Compile the usradd.dll

Copy the *ADDX.cpp* that has been generated and exported with MOSAIC into the src directory of the Visual Studio project before you start Visual Studio with a double-click on the project-file *usradd.vcxproj*. Depending on which solver is used inside the simulation, additional header files and libraries have to be included to get rid of compiler and linker errors. The source code generated by MOSAIC should contain hints which paths and files have to be entered for *All Configurations* in the usradd project properties. These are only hints, because the paths depend on the machine you are actually working on.
For Sundial's Kinsol it may be something like
C:\Program Files\sundials\sundials-2.6.2-install\include
for the header,
C:\Program Files\sundials\sundials-2.6.2-install\lib
for the linker library path, as well as *sundials_kinsol.lib* and *sundials_nvecserial.lib* as additional libraries. Please navigate through your file system to get the exact path information.

Set the *Debug* configuration active and select *Rebuild useradd* in the Visual Studio menu. Warnings regarding the conversion of double into float will occur in *addpipe.cpp* and *addk.cpp* and can be ignored for this exercise. You will see a message like *Rebuild All successful* in the Visual Studio status bar if the *usradd.dll* was successfully created, otherwise "Rebuild All failed". You can check the successful build process by navigating to the Visual Studio project directory and taking a look into the Debug subfolder where *usradd.dll* and *usradd.pdb* should be found now (compare to figure 6).

7.1 Some linker error handling

If you see a warning containing basically all KINSol commands (i.e. KINCreate etc) of the ADDX.cpp source combined with a message like "could not find external symbol __declspec(dllimport)", the sundials libraries have to be compiled again. Go to the sundials-build directory (e.g. sundials-2.6.2-build_include_sundials) and open sundials_config.h.

At the end of this file change the code to

```
#ifdef BUILD_SUNDIALS_LIBRARY
#define SUNDIALS_EXPORT __declspec(dllexport)
#else
//#define SUNDIALS_EXPORT __declspec(dllimport)
#define SUNDIALS_EXPORT
```

#endif

Run *msbuild ALL_BUILD.vcxproj* and *msbuild INSTALL.vcxproj* on the command line (Visual Studio-Tools) to compile the sundials libraries again.

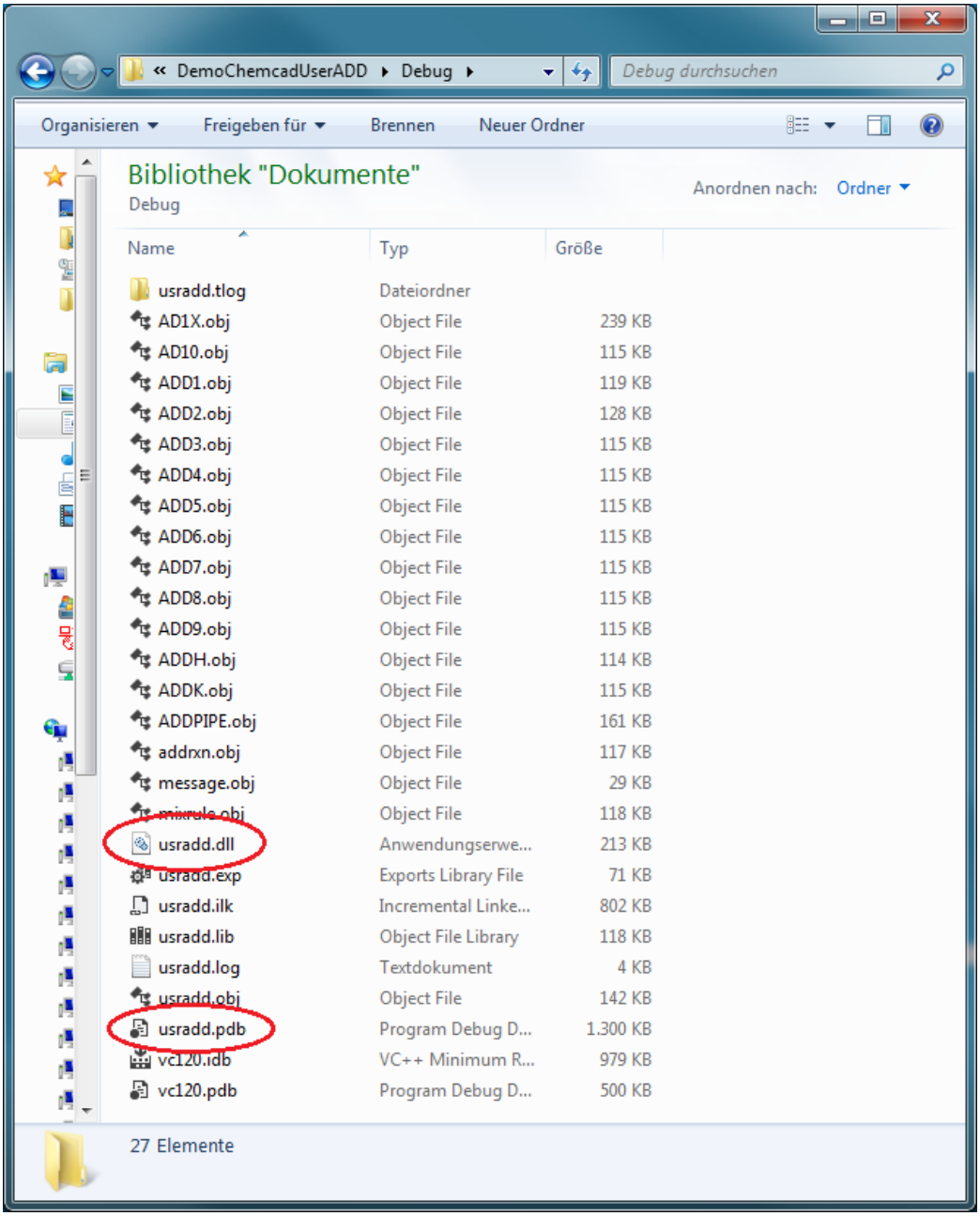


Figure 6: Debug directory after build of usradd project

8 Gather files in the CHEMCAD directory

Go to the Chemcad directory and make a backup of the *usradd.dll* before proceeding. In order to make the customized model available as a UAM in Chemcad, the files generated with

- MOSAIC (ADDX.map, $ADDX.lab),
- ScrBuild (ADDX.my), and
- Visual Studio (usradd.dll, usradd.pdb)

have to be copied into the Chemcad directory. You will need administrator privileges to do so.

9 Run and debug the UAM

Start Chemcad and let it find a license. Start Visual Studio by double-clicking the project-file *usradd.vcxproj*. Press Ctrl+Alt+P to debug the UAM by attaching the debugger to the Chemcad process (e.g. CC6.exe). In Visual Studio 2013 the status bar turns orange to indicate the attachment. Set a breakpoint to pause the execution of the UAM when hitting the breakpoint. Change to Chemcad and build a flowsheet including your UAM. Set the parameter values by double-clicking on the unit operation. The user form created with ScrBuild will be displayed to enter the parameter values. When all inlets of the UAM are specified, right-click the UAM and select *Run This UnitOp*. When the breakpoint is hit, Visual Studio will pop up and you can check the variable values by hovering the mouse over the respective variables in the editor. You may want to use the keyboard to step through the program (e.g. Shift+F8).