LUND UNIVERSITY

# User guide for developed DLLs for real fluid property models in IPSEpro process simulator

Mondejar, Maria

2015

# User guide for developed DLLs for real fluid property models in IPSEpro process simulator

**2015**

**Maria E. Mondejar**
Division of Thermal Power Engineering
Department of Energy Sciences

# Contents

# 1. What is in this guide?

This guide is intended for beginner users of IPSEpro process simulation software [1] and contains a brief introduction on the development of **dynamic link libraries** (**DLL**) with external functions to be used in IPSEpro, as well as a short explanation on how to use the DLLs created at the Division of Thermal Power Engineering at Lund University, for the use of real fluid property estimation.

The DLLs presented here are enumerated as follows:

- **Binary.DLL:** a DLL for property estimation of binary fluid mixtures, by using Refprop [2] subroutines.
- **Multi.DLL:** a DLL for property estimation of multicomponent fluid mixtures, by using Refprop subroutines.
- **SeaWater.DLL:** a DLL that contains correlations for property estimation of seawater.

Detailed information on the development of DLLs for IPSEpro can be found at the section 11 of the user manual for the **Model Development Kit** of IPSEpro, and other references mentioned in section 5. However, this guide provides a quick start guide for those who want to use or modify these DLLs, with general tips based on my experience on their development.

## 2. How to create a DLL for IPSEpro

As already mentioned, the detailed instructions for the development of dynamic link libraries (DLL) for their use in IPSEpro can be found in the section 11 of the user manual for the Model Development Kit of IPSEpro. Only a basic description is given here.

The instructions in this user guide are based on my experience and refer to the development of DLL using Microsoft Visual Studio 10 and C++ as the programming language. The version of IPSEpro was 5.1, and that of Refprop was 9.1.

### 2.1. Steps to follow to develop a DLL

1. Start a DLL Project in Visual Studio.

Go to: `File > New > Project > Visual C++ > Win 32 Project > Name and location of the project > Next > Select DLL > Finish`

After this, you should see something similar to Figure 1.

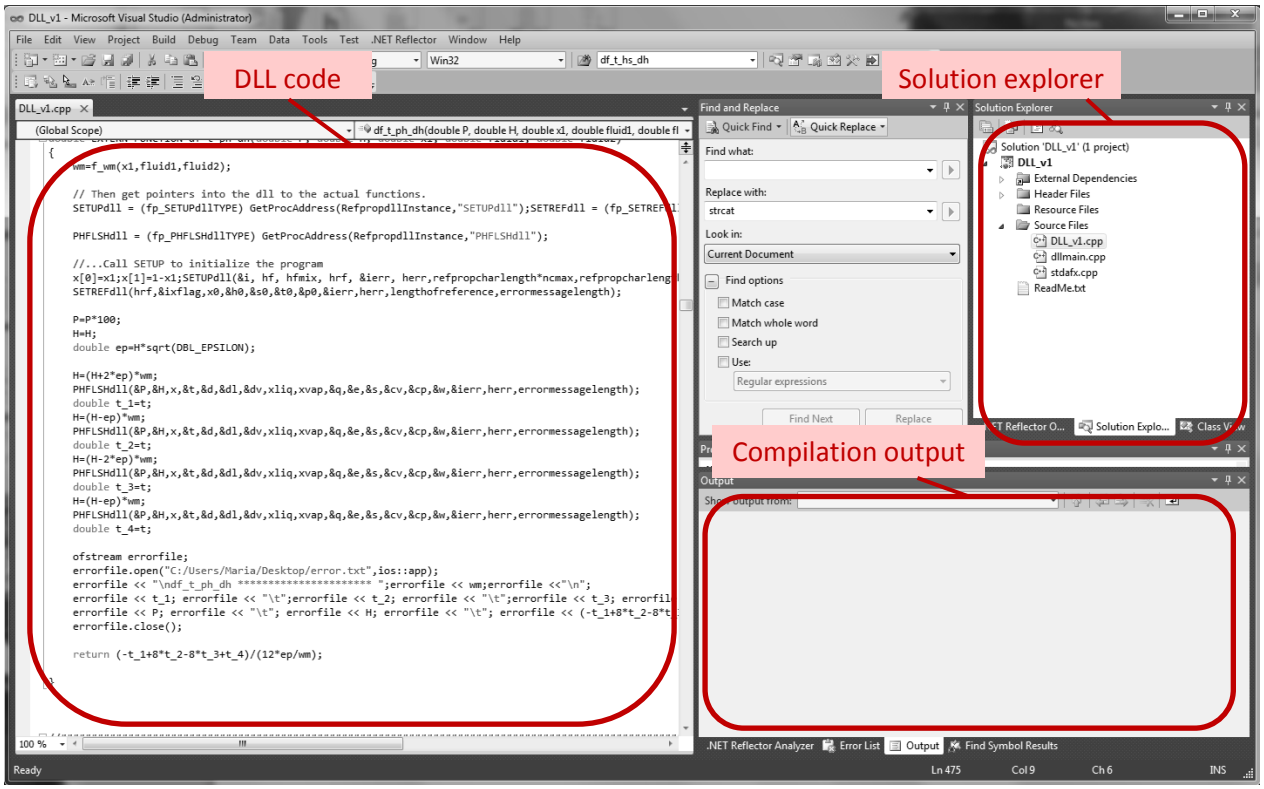


*Figure 1. General view of the Microsoft Visual Studio panel.*

2. Header files (.h) are generated automatically, as well as **dllmain.cpp** and **stdafx.cpp** (they can be found in the solution explorer). The code referring to the function must be inserted in **name_of_dll.cpp**

3. The following property must be changed to avoid compilation problems. In the solution explorer:

`right click on the project name > Properties > General > Character Set > Set to Multi-Byte Character Set`

4. Write the code of the external functions:

The external functions must have the specific format shown in Figure 2. Because the solving process of IPSEpro is based on the Newton's method, it is necessary to define, for each function, the partial derivatives for each of the input variables. Figure 2 shows the general required format of the functions.

```
#include <windows.h>
#include <stdio.h>
#include "stdafx.h"

using namespace std;

#define EXTERN_FUNCTION __declspec(dllexport) _stdcall

// Function declarations
extern "C"
{
double EXTERN_FUNCTION name_of_function(variable_type variable_name, ...);
double EXTERN_FUNCTION partial_derivative(variable_type variable_name,..);
...
}

double EXTERN_FUNCTION name_of_function (variable_type variable_name, ...)
{
function_operations;
return function_output;
}
```

Include always

to export functions

For each function, the partial derivatives can be defined

*Figure 2. General required format of the functions.*

ⓘ It is important to notice that IPSEpro only accepts input and output variables of type *double*.

ⓘ In certain cases, not defining the partial derivatives of a function can be valid. This occurs when the variables are set as parameters, and it is not expected to use the IPSEpro solver to estimate their value (e.g. if the variable of a function is the composition of a mixture, this parameter is always set, and therefore the partial derivatives are not needed.

5. When the code of the DLL is ready, build it:

```
Build > Build solution
```

The result of the compilation will be shown in the compilation output window, and should look like this:

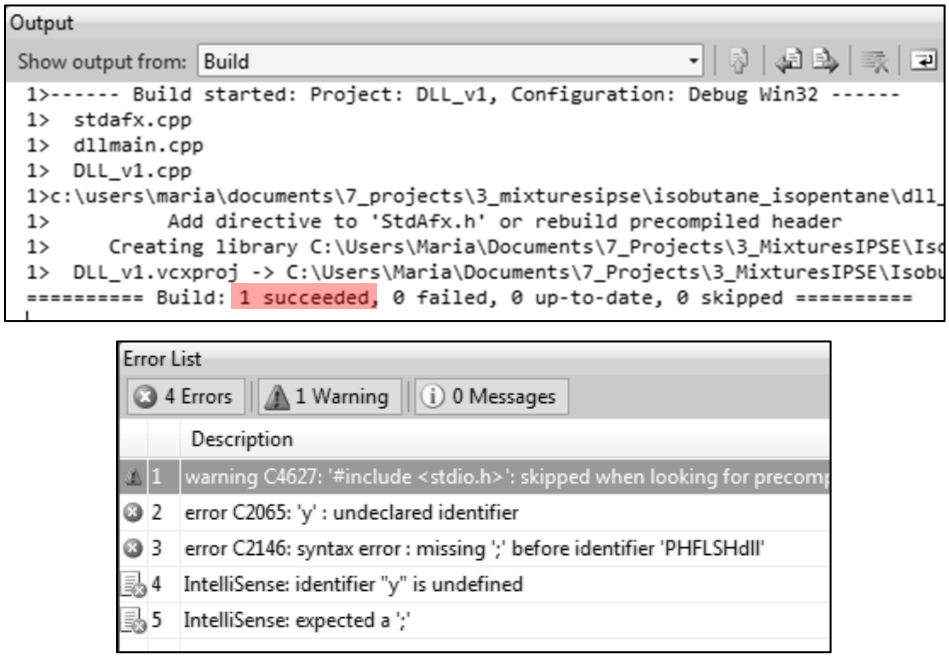


*Figure 3. Result from the compilation of a DLL project: a) successful compilation; b) unsuccessful compilation.*

6. If the compilation is successful, the DLL can be found in the projects folder, under the subfolder 'Debug'.

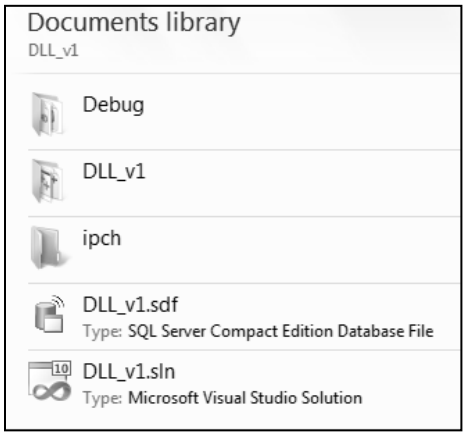


*Figure 4. DLL project folder content. The compiled DLL is found in the subfolder 'Debug'.*

7. If the compilation is not successful, the compilation output window will show a list with the compilation errors and their location in the code. This list is not 100% reliable but it can serve as a guide for debugging the code. Typical errors that provoke the compilation to fail are:

- The declared number of functions does not correspond with the number of functions defined.
- There is a mismatch between the number of inputs in the declaration, and in the definition of a function.
- Spelling errors.
- Lack of necessary headers (e.g. Refprop.h, windows.h...).

## 2.2. Create a DLL to call the Refprop DLL subroutines

In order to be able to call the Refprop subroutines from our DLL some modifications on the syntaxes are needed. The necessary files for this can be downloaded at the **official web page of Refprop NIST**:

http://www.boulder.nist.gov/div838/theory/refprop/LINKING/Linking.htm#CApplications

First, the file **refprop2.h** must be copied in the same folder where you have the C++ code. Then the file **refprop.dll** (from the Refprop installation folder) must be copied in the same path where the IPSEpro library will be allocated.

The function code must now include the call to the Refprop DLL:

```
#include <windows.h>
#include <stdio.h>
#include "stdafx.h"
#include "refprop2.h"

using namespace std;
```

*Figure 5. General required format of the functions, including call to Refprop subroutines.*

The file **EX_C2.c** from the package downloaded at the Refprop website contains a code example to call the Refprop subroutines. The DLL with functions for IPSEpro can be developed based on this example. Most of the information needed for the modification of the example is given in the comments, but a short key is provided here:

| | |
|---|---|
| **long i** | number of components in a mixture |
| **double x** | array with components molar fractions |
| **char hf** | string with name of the mixture components (eg. "isobutan.fld\|ipentane.fld"); |
| **char hrf** | reference state ("NBP", "DEF", ...) |
| **char hfmix** | mixture model (i.e. "hmx.bnc") |

The main characteristics of the code are the following:

- The file **refprop2.h** contains the headers to all the Refprop subroutines, as well as their structure (number and type of input variables). As all the variables from IPSEpro are of type double, it is necessary to create functions in the DLL that 'convert' those into other types, if needed. This was done in the DLL developed for mixtures, were each fluid is selected following a reference number that can be found in the Annex I.
- The description of the Refprop subroutines, and each of their input variables, can be found in the file **MANUAL.txt** of the folder 'fortran' of the Refprop installation folder.
- The syntaxes to call a Refprop subroutine from the C++ code follows the following structure:

```
SETUPdll = (fp_SETUPdllTYPE) GetProcAddress(RefpropdllInstance,"SETUPdll");
//Initializes the program
SETREFdll = (fp_SETREFdllTYPE) GetProcAddress(RefpropdllInstance,"SETREFdll");
//Initializes the reference state for mixtures
PHFLSHdll = (fp_PHFLSHdllTYPE) GetProcAddress(RefpropdllInstance,"PHFLSHdll");
//Get address of subroutine PHFLSH
SETUPdll(&i, hf, hfmix, hrf, &ierr,herr,refpropcharlength*ncmax,refpropcharlength,
lengthofreference,errormessagelength);
//Calls setup with the defined values for i, hf, hfmix, hrf
SETREFdll(hrf,&ixflag,x0,&h0,&s0,&t0,&p0,&ierr,herr,lengthofreference,errormessage
length);
//Calls setref with the defined values for hrf,x0,h0,s0,t0,p0 (reference state values)
PHFLSHdll(&P,&H,x,&t,&d,&dl,&dv,xliq,xvap,&q,&e,&s,&cv,&cp,&w,&ierr,herr,errormess
agelength);
//Calls the subroutine to calculate (in this case) the state conditions with pressure and
//enthalpy as inputs
```

*Figure 6. Syntaxes to call the Refprop subroutines from C++ code (found at EX_C2.c).*

ⓘ By default, the reference state for the calculation of mixtures properties was set to NBP (normal boiling point). In order to change it, this should be made in the source code of the DLL.

## 2.3. Check the DLL compilation

In order to test the compilation of the DLL before calling it from IPSEpro, open the command prompt:

`Start > All Programs > Microsoft Visual Studio > Visual Studio Tools > Visual Studio command prompt >` go to the dll folder and write:

```
dumpbin /exports NameOfDLL.dll
```

The output should look as in Figure 7.

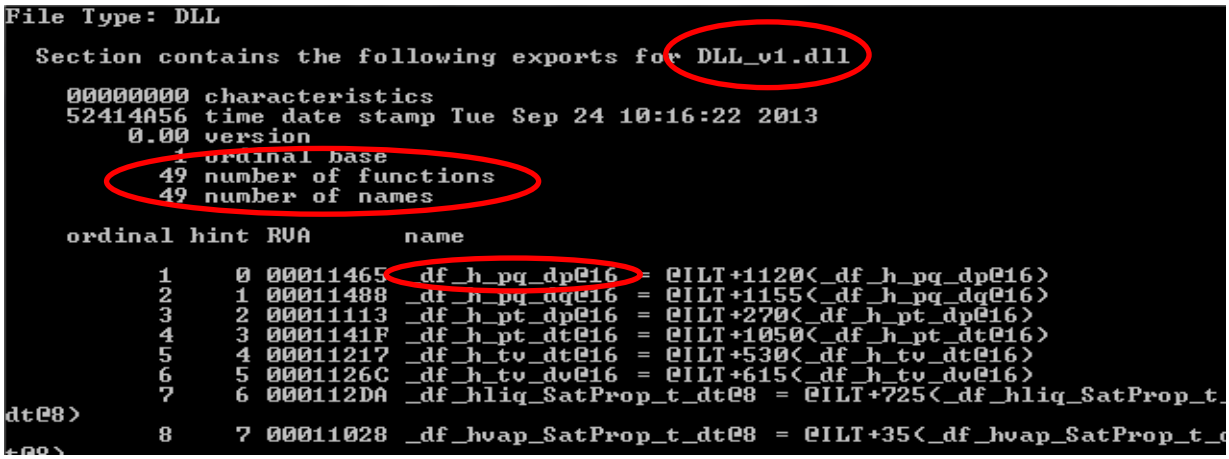


*Figure 7. Test for exported functions in the developed DLL.*

As it can be observed in Figure 7, the number of exported functions should agree the number of names (definitions). Each exported function appears as **_function@bitsnumber**, where the bits number corresponds to 8 times the number of input variables of type *double* (see IPSEpro manual for more information on this). This is an important step to check for the correct exportation of the functions.

# 3. Using the DLL from IPSEpro

## 3.1. Installing the DLL

The first step to use the developed DLL in IPSEpro is to copy it into the folder where the IPSEpro library files are. If the DLL calls the Refprop subroutines, the **refprop.DLL** should be also copied into this folder.

## 3.2. Calling the DLL functions from MDK

The DLL functions are called from the MDK. External functions can be used in **Connections**, **Global** variables and **Units**. New functions can be created from the MDK code by adding an **External Function**.

The first step to implement a new *fluid type* is to create a **Global** variable that contains all the functions exported by the DLL. Figure 8 shows the window for the definition of the external function. It is important to notice that the function and derivative names must be the same as the ones exported in the DLL. The inputs name can differ.

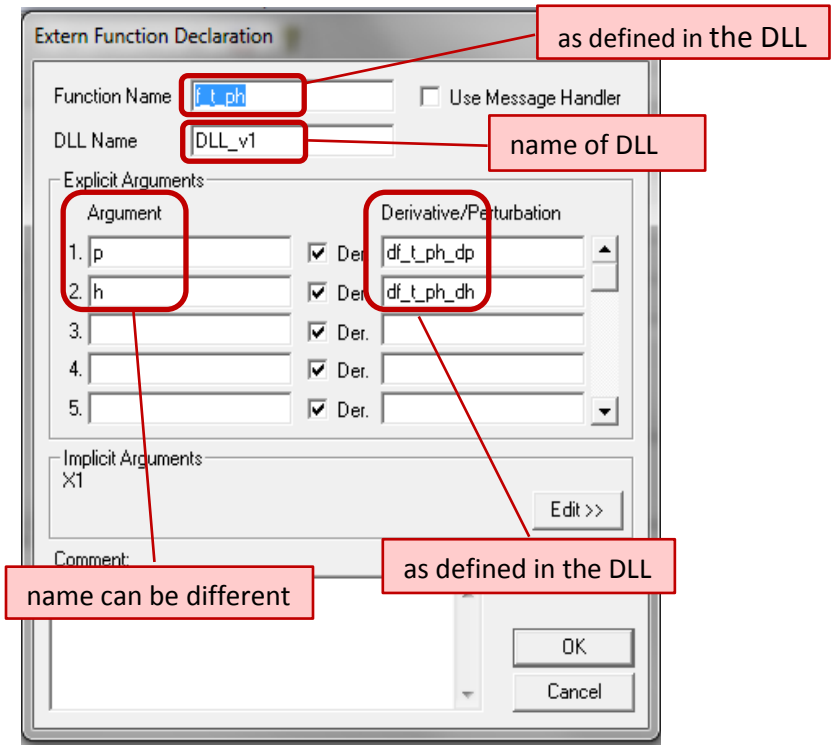


*Figure 8. Definition of external function in MDK.*

Secondly, new **Connection** and **Units** can be created, based on the new **Global**. It is important to notice that the same **Unit** can have connections from different **Globals**, which means that different property estimation models can be used for different streams on a same unit.

Typical errors that can appear at this stage are:

- No convergence of the DLL functions → Revise calculation in the source code
- Different name of function in MDK and DLL file
- Different number or type of variables for the functions

## 3.3. Usage of the developed DLLs

In this section the basic instructions to use the DLLs developed at the Division of Thermal Power Engineering are explained.

### DLL for property estimation of binary mixtures with Refprop (Binary.DLL)

A DLL for the property estimation of binary mixtures using the Refprop subroutines was developed and presented in [3], where a list of the provided functions is given. The way to define a binary mixture with this DLL is to create a **Global** variable with the new DLL functions and define its components and composition as shown in Figure 9.

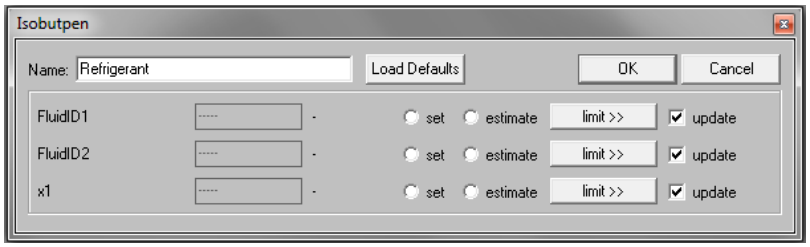


*Figure 9. Definition of the Global variable for binary mixtures.*

Here, **FluidID1** and **FluidID2** correspond to the reference numbers of the desired mixture components (see Annex I for the reference numbers of the fluids). Only pure components can be referred to here (which means that Refprop predefined mixtures and pseudo-pure fluids are not included). The variable **x1** refers to the molar fraction of the first component of the binary mixture. All of the parameters should be set.

ⓘ If a single component fluid is to be used instead, it is enough to set the variable x1 to 1.0.

ⓘ It is not possible to estimate any of the variables FluidID1, FluidID2 and x1, as it would lead to a calculation error.

ⓘ The Refprop limitations for the calculation of properties of certain mixtures apply also here. In the case of mixtures in which the mixture parameters are estimated, no warning message would be seen from IPSEpro.

### DLL for property estimation of multicomponent mixtures with Refprop (Multi.DLL)

The DLL for property estimation of multicomponent mixtures was developed in order to be able to define multicomponent mixtures of up to four components in a flexible way. Figure 10 shows the window for the definition of the multicomponent mixture composition:

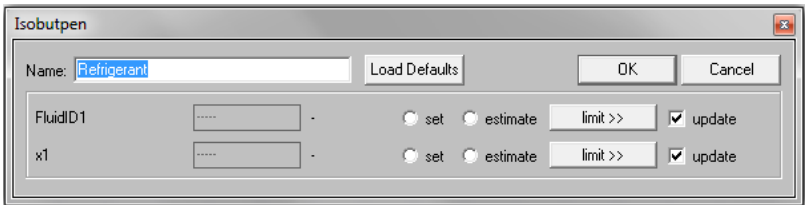


*Figure 10. Definition of the Global variable for multicomponent mixtures.*

The variable **FluidID1** has the following format: **WWWXXXYYYZZZ**, where WWW, XXX, YYY and ZZZ refer to the three-figure reference number of the fluids that the mixture contains. The variable **x1**, with format **wwwxxxyyyzzz**, refers to the decimal part of the molar fraction of each component (i.e. 0.www). For example, a ternary mixture of butane, propane and pentane with equal molar composition would be defined as:

```
FluidID1: 005070067      x1:  333333334
```

The limitation on the length of variables in IPSEpro, limits in turn the maximum number of components to four.

ⓘ If a single component fluid is to be used with this DLL, the FluidID1 variable must be defined as XXXXXX, where XXX is the three-figure reference number of the fluid, and x1 is 999001.

### DLL for the equation of state of seawater (SeaWater.DLL)

The DLL for the calculation of the properties of seawater is based on the equation of state of seawater of McDougall and Barker [4] and is only intended for the calculation of the enthalpy, entropy and density of the seawater, based on the temperature, pressure and salinity as input variables. Figure 11 shows the window for the definition of the seawater salinity (**sal**), which must be given in ppt (parts per thousand).

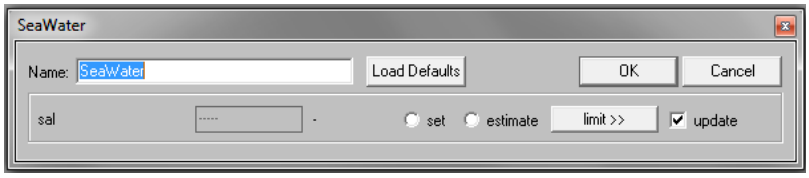


*Figure 11. Definition of the Global variable for sea water.*

### 3.4. Additional information

Each of the DLLs here presented here is provided with a library for their use. The components included in these libraries are limited to those needed in the simulation of waste heat recovery systems. However, extending the number of components does not represent any additional difficulty and can be done in the same way as with the standard libraries.

ⓘ For debugging reasons, the DLL was created so that each time one of the functions was called, the output results were logged in the following error file: "Desktop/error.txt". For this reason, it is important to clean periodically this file (e.g. from Matlab code:

```
ferror=fopen('C:\...\Desktop\error.txt','wt');
fclose(ferror);
```

# 4. Publications

The developed DLLs presented in this guide were used in the following works:

[1] Ahlgren, F., Mondejar, M.E., Genrup, M., Thern, M. Waste heat recovery in a cruise vessel in the Baltic Sea by using an organic Rankine cycle: a case study, in: Accepted in Turbo EXPO ASME 2015.

[2] Mondejar, M.E., Ahlgren, F., Thern, M., Genrup, M., 2015. Study of the on-route operation of a waste heat recovery system in a passenger vessel, Energy Procedia, 2015.

[3] Mondejar, M.E., Thern, M., Genrup, M., 2014. Aerodynamic considerations in the thermodynamic analysis of organic Rankine cycles, in: Proceedings of the ASME 2014 Power Conference. Baltimore, Maryland (USA).

## 5. Additional sources of information

Additional information with respect to the development and usage of DLL for IPSEpro can be found in the IPSEpro manuals:

- Section 11: Creating External Function DLLs, Model Development Kit

Other publications that present the development of DLLs for IPSEpro:

- Ji, X., Jonsson, M., Yan, J., Processes, E., Implementation of a real thermodynamic property model in IPSEpro.
- Thorbergsson, E., Grönstedt, T., Robinson, C., 2013. Integration of fluid thermodynamic and transport properties conceptual turbomachinery design, in: Proceedings of ASME Turbo Expo 2013: Turbine Technical Conference and Exposition. pp. 1–8.

Additional information on the usage of Refprop subroutines and their linking from different applications can be found at:

- http://www.boulder.nist.gov/div838/theory/refprop/LINKING/Linking.htm
- Fortran files of the Refprop installation folder

For additional support on the use or modification of the DLLs presented here contact **Maria E. Mondejar** at: maria.e.mondejar@hotmail.com

## 6. Acknowledgements

This work was developed with the funding from Lund University, and was accomplished thanks to Dr. Marcus Thern, for the support provided on the use of IPSEpro, Dr. Eric W. Lemmon for the support provided on the use of the Refprop subroutines for mixtures, and Dr. Egill Thorbergsson for the support on the development of DLLs.

# 7. References

[1] SimTech. IPSEpro Process Simulator 2013.

[2] Lemmon EW, Huber ML, McLinden MO. NIST Standard Reference Database 23: Reference Fluid Thermodynamic and Transport Properties-REFPROP, Version 9.1. Natl Inst Stand Technol 2013.

[3] Mondejar ME, Thern M. A new IPSEpro® library for the simulation of binary mixtures of real fluids in power cycle analysis. J Postdr Res 2014;2.

[4] McDougall TJ, Barker. PM. Getting started with TEOS-10 and the Gibbs Seawater (GSW) Oceanographic Toolbox. 2011.

## Annex I. Reference number of the Refprop fluids in the IPSEpro DLL.

| | | | | | |
|---|---|---|---|---|---|
| acetone | 1 | mxylene | 41 | xenon | 81 |
| ammonia | 2 | md2m | 42 | novec649 | 82 |
| argon | 3 | md3m | 43 | r11 | 83 |
| benzene | 4 | md4m | 44 | r113 | 84 |
| butane | 5 | mdm | 45 | r114 | 85 |
| 1butene | 6 | methane | 46 | r115 | 86 |
| co2 | 7 | methanol | 47 | r116 | 87 |
| co | 8 | mlinolea | 48 | r12 | 88 |
| cos | 9 | mlinolen | 49 | r1216 | 89 |
| c2butene | 10 | moleate | 50 | r123 | 90 |
| cyclohex | 11 | mpalmita | 51 | r1233zd | 91 |
| cyclopen | 12 | mstearat | 52 | r1234yf | 92 |
| cyclopro | 13 | c1cc6 | 53 | r1234ze | 93 |
| d4 | 14 | mm | 54 | r124 | 94 |
| d5 | 15 | neon | 55 | r125 | 95 |
| d6 | 16 | neopentn | 56 | r13 | 96 |
| decane | 17 | nitrogen | 57 | r134a | 97 |
| d2 | 18 | nf3 | 58 | r14 | 98 |
| dee | 19 | n2o | 59 | r141b | 99 |
| dmc | 20 | nonane | 60 | r142b | 100 |
| dme | 21 | oxylene | 61 | r143a | 101 |
| c12 | 22 | octane | 62 | r152a | 102 |
| ethane | 23 | orthohyd | 63 | r161 | 103 |
| ethanol | 24 | oxygen | 64 | r21 | 104 |
| ebenzene | 25 | pxylene | 65 | r218 | 105 |
| ethylene | 26 | parahyd | 66 | r22 | 106 |
| fluorine | 27 | pentane | 67 | r227ea | 107 |
| d2o | 28 | c4f10 | 68 | r23 | 108 |
| helium | 29 | c5f12 | 69 | r236ea | 109 |
| heptane | 30 | propane | 70 | r236fa | 110 |
| hexane | 31 | c3cc6 | 71 | r245ca | 111 |
| hydrogen | 32 | propylen | 72 | r245fa | 112 |
| hcl | 33 | propyne | 73 | r32 | 113 |
| h2s | 34 | so2 | 74 | r365mfc | 114 |
| isobutan | 35 | sf6 | 75 | r40 | 115 |
| ibutene | 36 | toluene | 76 | r41 | 116 |
| ihexane | 37 | t2butene | 77 | rc318 | 117 |
| ioctane | 38 | cf3i | 78 | re143a | 118 |
| ipentane | 39 | c11 | 79 | re245cb2 | 119 |
| krypton | 40 | water | 80 | re245fa2 | 120 |
| | | | | re347mcc | 121 |

## Annex II. Code in Matlab to change the composition of the working fluid.

```
%% Change the working fluid components and composition
invoke(project,'Fluid','FluidID1',fluid_number_1)
invoke(project,'Fluid','FluidID2',fluid_number_2)
invoke(project,'Fluid','x1',molar mass of component 1)
```

```
%% Resultado
function value=Resultado(proj,obj,item)
    OBJ = invoke(proj,'findObject',obj);
    ITEM = invoke(OBJ,'findItem',0,item);
    value=invoke(ITEM,'resultValue');
end
```

```
%% Input
function Input(proj,obj,item,value)
    OBJ = invoke(proj,'findObject',obj);
    ITEM = invoke(OBJ,'findItem',0,item);
    invoke(ITEM,'inputValue',value);
end
```

If Refprop subroutines are to be used from Matlab it is necessary to have the following files on the same folder of the Matlab code location:

**refpropm.m**

**rp_proto.m**

**rp_proto64.m**

These files can be downloaded at the official web page of Refprop:

http://www.boulder.nist.gov/div838/theory/refprop/LINKING/Linking.htm#MatLabApplications

ⓘ It is important to notice that the composition of the mixtures must be specified in molar fractions from IPSEpro, but in mass fractions if using the Matlab calls.