



## **D5.2 Modelling of individual component models of the overall RESTORE system and its techno-economic simulation (V2)**



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101036766.

| <b>PROJECT INFORMATION SHEET</b> |   |
|----------------------------------|---|
| <b>Project Acronym</b>           | RESTORE   |
| <b>Project Full Title</b>        | Renewable Energy based seasonal Storage Technology in Order to Raise Environmental sustainability of DHC  |
| <b>Grant Agreement</b>           | 101036766   |
| <b>Call Identifier</b>           | H2020-LC-GD-2020-1  |
| <b>Topic</b>                     | Innovative land-based and offshore renewable energy technologies and their integration into the energy system   |
| <b>Project Duration</b>          | 48 months (October 2021 – September 2025)   |
| <b>Project Website</b>           | <a href="http://www.restore-dhc.eu">www.restore-dhc.eu</a>  |
| <b>Disclaimer</b>                | The sole responsibility for the content of this document lies with the authors. It does not necessarily reflect the opinion of the funding authorities. The funding authorities are not responsible for any use that may be made of the information contained herein. |

| <b>DELIVERABLE INFORMATION SHEET</b> |  |
|--------------------------------------|--|
| <b>Number</b>                        | Deliverable D5.2   |
| <b>Full Title</b>                    | D5.2 Modelling of individual components of the overall RESTORE system and its techno-economic simulation (V2 final)  |
| <b>Related WP</b>                    | WP5 (RESTORE EU-wide Replication in Virtual Representation on Real Use-Cases)  |
| <b>Related Tasks</b>                 | Task 5.1 - Modelling of individual components of the overall RESTORE system.<br>Task 5.2 - RESTORE techno-economic modelling of the integrated system  |
| <b>Lead Beneficiary</b>              | SIMTECH (SIM)  |
| <b>Author(s)</b>                     | Stefan Bergmann (SIM) – <a href="mailto:s.bergmann@simtechnology.com">s.bergmann@simtechnology.com</a><br>Fatima Dargam (SIM) - <a href="mailto:f.dargam@simtechnology.com">f.dargam@simtechnology.com</a> |
| <b>Contributor(s)</b>                | Javier Baigorri (CEN), Xabier Randez (CEN),<br>Francisco Cabello (CENER), Erhard Perz (SIM)  |
| <b>Reviewer(s)</b>                   | Francisco Cabello (CENER)  |
| <b>Dissemination level</b>           | Public   |
| <b>Due Date</b>                      | August 2025 (M47)  |
| <b>Submission Date</b>               | August 31 <sup>st</sup> , 2025   |
| <b>Status</b>                        | Version 1.0  |

| <b>QUALITY CONTROL ASSESSMENT SHEET</b> |             |                              |  |
|---|-------------|------------------------------|--|
| <b>ISSUE</b>                            | <b>DATE</b> | <b>COMMENT</b>               | <b>AUTHOR</b>                            |
| <b>V0.1</b>                             | 20/08/2025  | Draft_Version V0.1           | Stefan Bergmann (SIM)                    |
| <b>V0.2</b>                             | 27/08/2025  | Contribution to Chapter 4    | Francisco Cabello (CENER)                |
| <b>V0.3</b>                             | 27/08/2025  | Draft_Version V0.3           | Fatima Dargam (SIM)                      |
| <b>V0.4</b>                             | 28/08/2025  | Complete Version V0.4 Review | Erhard Perz (SIM)<br>Fatima Dargam (SIM) |
| <b>V0.5</b>                             | 29/08/2025  | Final Version Review         | Francisco Cabello (CENER)                |
| <b>V1.0</b>                             | 30/08/2025  | Submission to the EC         | Francisco Cabello (CENER)                |

## Summary

---

This document provides information about the modelling of individual components of the overall RESTORE system. Deliverable D5.2 is the result of the work carried out in task T5.2 (RESTORE techno-economic modelling of the integrated system, M9-M47), led by CENER, initiated in task T5.1 (Modelling of individual components of the overall RESTORE system, M7-M30), led by SIMTECH. The main purpose of this report is to describe the final version of the component models developed for the dedicated model library RESTORE\_Lib, and to describe the process level of the overall RESTORE model, based on the TCES and on the rORC technologies, with the economic modelling of the system.

Task T5.2 developed the project required models for system level simulation (yearly performance simulations, covering off-design performance of cycles and constraints due to transient behaviour) and for techno-economic analysis. The RESTORE sub-systems, with TCES and rORC technologies, were modelled using the customized RESTORE\_Lib model library created in T5.1, enhanced with additional economic parameters, that are cost-correlations of the investment costs for the components. The systems were tested and optimized within the process simulation environment within the RESTORE virtual tool, powered by the IPSE GO web-platform.

The economic model considered in the project is highly flexible, allowing the user to modify a high number of parameters that are considered for the economic simulations. In addition, the model receives information from the technical simulation to show as results relevant economic variables such as the Net Present Value, the Internal Return Rate and the Levelized Cost of Storage for Electricity and Heat.

The information provided in this document builds upon collaboration between SIMTECH, CENER, TU-WIEN, and POL, as RESTORE partners involved in task T5.2. The development of task WP5-T5.2 received input from WP1-T1.4 & T1.7, WP2, WP3, WP4, and from WP5-T5.1, T5.4, T5.5.

As part of RESTORE project's participation in the "Open Research Data Pilot", Deliverable D5.2, as a public dissemination-level document, will be made available in the RESTORE open-access research data repository within the Zenodo RESTORE Community (<https://zenodo.org/communities/101036766/?page=1&size=20>), for further reference and dissemination.

## Table of Contents

---

|  |    |
|--|----|
| 1. Introduction.....                               | 8  |
| 2. Component Models Contained in RESTORE_Lib ..... | 10 |
| 2.1. Available units .....                         | 10 |
| 2.2. Available connections.....                    | 13 |
| 2.3. Available global objects.....                 | 14 |
| 3. Component Model Details .....                   | 15 |
| 3.1. OR_wall_htex.....                             | 15 |
| 3.1.1. OR_wall_htex .....                          | 15 |
| 3.2. RESTORE_EA .....                              | 17 |
| 3.2.1. RESTORE_EA .....                            | 18 |
| 3.3. TCM_CD_Transformer.....                       | 18 |
| 3.3.1. TCM_CD_Transformer .....                    | 18 |
| 3.4. TCM_Connector .....                           | 20 |
| 3.4.1. TCM_Connector .....                         | 20 |
| 3.4.2. TCM_Connector_Oil_TCM .....                 | 21 |
| 3.5. TCM_Heat_sink.....                            | 22 |
| 3.5.1. TCM_Heat_sink.....                          | 23 |
| 3.6. TCM_Heat_source .....                         | 23 |
| 3.6.1. TCM_Heat_source .....                       | 24 |
| 3.7. TCM_Htex .....                                | 25 |
| 3.7.1. TCM_Htex.....                               | 26 |
| 3.8. TCM_Mixer.....                                | 28 |
| 3.8.1. TCM_Mixer.....                              | 29 |
| 3.9. TCM_Pipe .....                                | 30 |
| 3.9.1. TCM_Pipe .....                              | 30 |
| 3.9.2. TCM_Pipe_pl.....                            | 31 |
| 3.10. TCM_Pump .....                               | 33 |
| 3.10.1. TCM_Pump .....                             | 33 |
| 3.11. TCM_Reactor_Charging .....                   | 34 |
| 3.11.1. TCM_Reactor_Charging.....                  | 35 |
| 3.12. TCM_Reactor_Discharging .....                | 41 |
| 3.12.1. TCM_Reactor_Discharging .....              | 42 |
| 3.13. TCM_Separator.....                           | 49 |

|                                       |    |
|---------------------------------------|----|
| 3.13.1. TCM_Separator.....            | 49 |
| 3.14. TCM_Sink.....                   | 51 |
| 3.14.1. TCM_Sink.....                 | 51 |
| 3.15. TCM_Source.....                 | 51 |
| 3.15.1. TCM_Source.....               | 51 |
| 3.16. TCM_Splitter.....               | 52 |
| 3.16.1. TCM_Splitter.....             | 52 |
| 3.17. TCM_T_Htex.....                 | 53 |
| 3.17.1. TCM_T_Htex.....               | 55 |
| 3.18. TCM_Valve.....                  | 57 |
| 3.18.1. TCM_Valve.....                | 57 |
| 3.19. TCM_W_Separator.....            | 58 |
| 3.19.1. TCM_W_Separator.....          | 58 |
| 3.20. T_TCM_Htex.....                 | 59 |
| 3.20.1. T_TCM_Htex.....               | 60 |
| 3.21. T_wall_htex.....                | 62 |
| 3.21.1. T_wall_htex.....              | 63 |
| 3.22. electricity_meter.....          | 64 |
| 3.22.1. electricity_meter.....        | 64 |
| 3.22.2. electricity_meter_dyn.....    | 64 |
| 3.22.3. electricity_meter_period..... | 65 |
| 3.23. flow_meter.....                 | 66 |
| 3.23.1. flow_meter.....               | 66 |
| 3.23.2. flow_meter_dyn.....           | 66 |
| 3.23.3. flow_meter_period.....        | 67 |
| 3.24. heat_meter.....                 | 67 |
| 3.24.1. heat_meter.....               | 67 |
| 3.24.2. heat_meter_dyn.....           | 68 |
| 3.24.3. heat_meter_period.....        | 68 |
| 3.25. time_counter.....               | 68 |
| 3.25.1. time_counter_dyn.....         | 69 |
| 3.25.2. time_counter.....             | 69 |
| 3.26. wall_OR_htex.....               | 70 |
| 3.26.1. wall_OR_htex.....             | 70 |
| 3.27. wall_T_htex.....                | 74 |

|   |     |
|---|-----|
| 3.27.1. wall_T_htex .....   | 75  |
| 3.28. OR_Stream .....   | 75  |
| 3.28.1. OR_Stream .....   | 76  |
| 3.29. TCM_Stream .....  | 76  |
| 3.29.1. TCM_Stream .....  | 76  |
| 3.30. q_cond_trans.....   | 79  |
| 3.30.1. q_cond_trans.....   | 79  |
| 3.31. OR_Composition .....  | 79  |
| 3.31.1. OR_Composition .....  | 80  |
| 3.32. TCM .....   | 84  |
| 3.32.1. TCM .....   | 84  |
| 3.33. time_period.....  | 85  |
| 3.33.1. time_period.....  | 86  |
| 4. Economic modelling .....   | 87  |
| 4.1.1. The investment cost module.....  | 89  |
| 4.1.2. The Levelized Cost of Storage for Electricity .....                          | 92  |
| 4.1.3. The Levelized Cost of Heat .....   | 95  |
| 4.1.4. Example of application of the economic model to a RESTORE virtual use-case.. | 95  |
| 5. Conclusion.....  | 98  |
| 6. References .....   | 99  |
| Annex I. RESTORE Virtual Tool powered by IPSE GO .....                              | 101 |
| Annex II. RESTORE Reference Design Model.....                                       | 102 |
| Annex III. RESTORE_Lib Components .....   | 103 |

# 1. Introduction

This document focuses on the modelling of individual components of the overall RESTORE system, presenting the economic model for the RESTORE project [1].

Deliverable D5.2 is the result of the work carried out in task T5.2, concerning the “RESTORE techno-economic modelling of the integrated system”, initiated in task T5.1 which dealt with the “Modelling of individual components of the overall RESTORE system”, with the main goal of modelling, fine-tuning & testing all component-models of the overall RESTORE system, based on partners’ received information; as well as on the creation and maintenance of the customized Process Model Library (RESTORE\_Lib) for the project, used in the simulations of the systems IPSEpro and IPSE GO [2], [3].

D5.2 main purpose is to describe the final version of the component models developed for the dedicated model library RESTORE\_Lib, and to describe the process level of the overall RESTORE model, based on the TCES and on the rORC technologies, with the economic modelling of the system. Task T5.2 developed the project required models for system level simulation (yearly performance simulations, covering off-design performance of cycles and constraints due to transient behaviour) and for techno-economic analysis.

The task of modelling the individual components for the overall RESTORE system in T5.1 was directly responsible to feed input information for the development of the WP5 tasks T5.2 (Techno-Economic Modelling of the Integrated Systems), T5.3 (Web-Platform Adaptation for RESTORE Dynamic and Techno-economic Modelling to represent the Use-Cases), and T5.4 (Implementation, Optimization, Management & Validation of RESTORE Use-Cases using the Simulation Web Platform), with main input from the work packages WP2 and WP3, mostly related to the “TCES dedicated models for the reactor simulation”, and the “Dedicated models for the thermodynamic cycle and the dynamic behavior of RESTORE system”.

To model individual components of the overall RESTORE system and to be able to use them in system simulations, the model library RESTORE\_Lib has been developed in task T5.1 led by SIMTECH. D5.1 includes the initial results and software-models derived from task T5.1, with an overview of the library content. The RESTORE sub-systems, with TCES and rORC technologies, were modelled using the customized RESTORE\_Lib model library created in T5.1, enhanced with additional economic parameters, that are cost-correlations of the investment costs for the components. The systems were tested and optimized within the process simulation environment within the RESTORE virtual tool, powered by the IPSE GO web-platform.

In addition, this deliverable also presents the economic model considered in the project, which is a highly flexible model, allowing the user to modify a high number of parameters for performing the economic simulations. The economic model considered for RESTORE, developed in collaboration with CENER, receives information from the technical simulation to show as results relevant economic variables such as the Net Present Value, the Internal Return Rate and the Levelized Cost of Storage for Electricity and Heat.

The information provided in this document builds upon collaboration between SIMTECH, CENER, TU-WIEN, and POL, as RESTORE partners involved in task T5.2. The development

of task WP5-T5.2 received input from WP1-T1.4 & T1.7, WP2, WP3, WP4, and from WP5-T5.1, T5.3, T5.4, T5.5. Besides D5.2's central scope on the component models development using SIMTECH's simulation tool [2] [3] to build the customized model library for RESTORE, it also considered the outcomes published from WP1, WP2, and WP3 (Deliverables: D1.1 - Report on Requirements and Specifications of the Overall Concept [4]; D1.4 on the specifications of RESTORE Use-Cases and Models [5]; D2.3 - Report on TCES Task D2.3 - Small-scale (1-2kW) TCES reactor tested and optimized [6]; D2.4 - Design report of the 30 kW/150 kWh TCES reactor [7]; D2.5 - Dedicated models for the reactor simulation [8]; D3.1 - Numerical model for HPORC systems optimization and application to different Test Cases [9]; D5.10 - RESTORE Replication Strategy V1 [10]; the WP5-T5.4 RESTORE Use-Cases Modelling Summary [11]; D5.4 - Implementation and validation of RESTORE Use-Case I: Residential / Industrial DH with Biomass and Solar Collectors [12]; D5.5 - Implementation and validation of RESTORE Use-Case II: Integration of different heat sources in DH of Cement Industry [13]; D5.6 - Implementation and validation of RESTORE Use-Case III: Integration of different heat sources in DH of Paper Mills Industry [14]; D5.7 - Implementation and validation of RESTORE Use-Case IV: Integration of different heat sources in DH of Steel-working industry [15]; D5.8 - Implementation and validation of RESTORE Use-Case V: District heating with Geothermal Technology [16]; and D5.9 - Implementation and validation of RESTORE Use-Case VI: Small-scale DHC network of Politecnico di Milano campus [17].

Besides this Introduction (Chapter 1), deliverable D5.2 is structured in the following way:

- Chapter 2 provides an overview of the updated models contained in the customized model library created for the RESTORE system, the RESTORE\_Lib.
- Chapter 3 presents details of the component models that were specifically developed for the RESTORE project to represent the overall RESTORE system.
- Chapter 4 explains the final economic model that was implemented for RESTORE.
- Chapter 5 presents the conclusion of the work reported in D5.2.
- Chapter 6 lists the references that the work reported in D5.2 was based upon.

## 2. Component Models Contained in RESTORE\_Lib

This chapter presents the component models, which are contained in the model library RESTORE\_Lib in its final version 1.1. These models have been developed using the IPSEpro Model Development Kit (MDK) [2].

The tables presented in this chapter describe the component models, which are either RESTORE specific developments, models which are adapted from general IPSEpro / IPSE GO models, or strictly general models shared with other IPSEpro / IPSE GO model libraries.

### 2.1. Available units

Table 1: Available Units

| Unit Name      | Description  | Type    |
|----------------|--|---------|
| G_OR_Htex      | heat exchanger for transfer from gas on hot side to organic fluid on cold side | adapted |
| G_Pipe         | pipe for gas streams   | general |
| G_Sink         | sink for a gas stream  | general |
| G_Source       | source for a gas stream  | general |
| OR_Boiler      | simple boiler model for ORC fluids   | adapted |
| OR_Compressor  | compressor for ORC fluids  | adapted |
| OR_Condenser   | condenser for ORC fluids, water cooled   | adapted |
| OR_Condenser_a | condenser for ORC fluids, air cooled, dry                                      | adapted |
| OR_Connector   | connector for ORC streams to be used in closed loops                           | adapted |
| OR_Expander    | expander for ORC fluids  | adapted |
| OR_G_Htex      | heat exchanger for transfer from ORC fluid on hot side to gas on cold side     | adapted |
| OR_Heat_sink   | heat sink for usage with OR streams  | adapted |
| OR_Heat_source | heat source for usage with OR streams  | adapted |
| OR_Htex        | general purpose heat exchanger for ORC fluids                                  | adapted |
| OR_Mixer       | mixer for ORC streams  | adapted |
| OR_Pipe        | pipe for ORC fluids  | adapted |
| OR_Properties  | calculation and display of physical properties of OR fluids                    | adapted |
| OR_Pump        | pump for ORC fluids  | adapted |
| OR_Separator   | vapour-liquid separator for ORC fluids   | adapted |
| OR_Sink        | sink for an ORC stream   | adapted |

|                                |  |          |
|--------------------------------|--|----------|
| <b>OR_Source</b>               | source for an ORC stream   | adapted  |
| <b>OR_Splitter</b>             | splitter for ORC streams   | adapted  |
| <b>OR_T_Htex</b>               | heat exchanger for transfer from ORC fluid on hot side to thermofluid on cold side   | adapted  |
| <b>OR_Turbine</b>              | turbine for ORC fluids   | adapted  |
| <b>OR_Valve</b>                | valve for ORC fluid  | adapted  |
| <b>OR_W_Htex</b>               | heat exchanger for transfer from ORC fluid on hot side to water on cold side   | adapted  |
| <b>OR_Xprescription</b>        | prescription/calculation of vapor quality of an ORC fluid  | adapted  |
| <b>OR_wall_htex</b>            | heat exchanger with wall transferring heat from organic fluid (OR) to another side   | specific |
| <b>RESTORE_EA</b>              | component model supplying economic analysis information  | specific |
| <b>TCM_CD_Transformer</b>      | transformer to combine charging and discharging operations of TCM  | specific |
| <b>TCM_Connector</b>           | connector for TCM to be used in closed loops   | specific |
| <b>TCM_Heat_sink</b>           | heat sink for TCM  | specific |
| <b>TCM_Heat_source</b>         | heat source for TCM  | specific |
| <b>TCM_Htex</b>                | heat exchanger for transfer from TCM fluid on hot side to TCM fluid on cold side   | specific |
| <b>TCM_Mixer</b>               | mixer for TCM streams  | specific |
| <b>TCM_Pipe</b>                | Pipe for thermochemical material (TCM)   | specific |
| <b>TCM_Pump</b>                | pump for TCM fluids  | specific |
| <b>TCM_Reactor_Charging</b>    | Reactor for charging step of TCM. High temperature side transferring heat to the reactor is optional. Different heat delivering working fluids (OR_ or T_) can be connected. | specific |
| <b>TCM_Reactor_Discharging</b> | Reactor for discharging step of TCM. Low temperature side receiving heat from the reactor is optional. Different heat receiving working fluids (OR_ or T_) can be connected. | specific |
| <b>TCM_Separator</b>           | separator for TCM stream   | specific |
| <b>TCM_Sink</b>                | sink for a TCM stream  | specific |
| <b>TCM_Source</b>              | source for a TCM stream  | specific |
| <b>TCM_Splitter</b>            | splitter for TCM streams   | specific |

|                        |   |          |
|------------------------|---|----------|
| <b>TCM_T_Htex</b>      | heat exchanger for transfer from TCM fluid on hot side to thermofluid on cold side  | specific |
| <b>TCM_Valve</b>       | valve for TCM stream  | specific |
| <b>TCM_W_Separator</b> | separator for water from TCM stream   | specific |
| <b>T_Connector</b>     | connector for heat transfer fluids to be used in closed loops                       | general  |
| <b>T_Heat_sink</b>     | heat sink for heat transfer fluids  | general  |
| <b>T_Heat_source</b>   | heat source for heat transfer fluids  | general  |
| <b>T_Htex</b>          | general purpose heat exchanger for heat transfer fluids                             | general  |
| <b>T_Mixer</b>         | mixer for heat transfer fluid streams   | general  |
| <b>T_OR_Htex</b>       | heat exchanger for transfer from thermofluid on hot side to ORC fluids on cold side | adapted  |
| <b>T_Pipe</b>          | pipe for heat transfer fluids   | general  |
| <b>T_Pump</b>          | pump for heat transfer fluids   | general  |
| <b>T_Sink</b>          | sink for a heat transfer fluid stream   | general  |
| <b>T_Source</b>        | source for a heat transfer fluid  | general  |
| <b>T_Splitter</b>      | splitter for heat transfer fluid streams  | general  |
| <b>T_TCM_Htex</b>      | heat exchanger for transfer from thermofluid on hot side to TCM fluid on cold side  | specific |
| <b>T_W_Htex</b>        | heat exchanger for transfer from thermofluid on hot side to water on cold side      | general  |
| <b>T_wall_htex</b>     | heat exchanger with wall transferring heat from thermooil (T) to another side       | specific |
| <b>W_Compressor</b>    | compressor for steam  | general  |
| <b>W_Connector</b>     | connector for closed loops  | general  |
| <b>W_Heat_sink</b>     | heat sink for water streams   | general  |
| <b>W_Heat_source</b>   | heat source for water streams   | general  |
| <b>W_Htex</b>          | general purpose heat exchanger, water on both sides                                 | general  |
| <b>W_Mixer</b>         | mixer for water streams   | general  |
| <b>W_OR_Htex</b>       | heat exchanger for transfer from water on hot side to OR fluid on cold side         | adapted  |
| <b>W_Pipe</b>          | pipe for water  | general  |
| <b>W_Pump</b>          | pump for water  | general  |

|                          |   |          |
|--------------------------|---|----------|
| <b>W_Sink</b>            | sink for a water stream   | general  |
| <b>W_Source</b>          | source for a water stream   | general  |
| <b>W_Splitter</b>        | splitter for water streams  | general  |
| <b>W_T_Htex</b>          | heat exchanger for transfer from water on hot side to thermofluids on cold side | general  |
| <b>W_Valve</b>           | valve for water   | general  |
| <b>W_Xprescription</b>   | prescription/calculation of steam quality                                       | general  |
| <b>electricity_meter</b> | electrical power consumption or production meter                                | specific |
| <b>flow_meter</b>        | flow meter  | specific |
| <b>free_var</b>          | free variable   | general  |
| <b>gear</b>              | gears   | general  |
| <b>generator</b>         | generator   | general  |
| <b>heat_meter</b>        | heat consumption or production meter  | specific |
| <b>mech_loss</b>         | mechanical loss   | general  |
| <b>motor</b>             | motor   | general  |
| <b>optimization</b>      | optimization element  | general  |
| <b>time_counter</b>      | time counter used for measuring and displaying time intervals                   | specific |
| <b>wall_OR_htex</b>      | heat exchanger with wall transferring heat from wall to organic fluid (OR)      | specific |
| <b>wall_T_htex</b>       | heat exchanger with wall transferring heat from wall to thermooil (T)           | specific |

## 2.2. Available connections

Table 2: Available Connections

| Connection Name   | Description  | Type     |
|-------------------|--|----------|
| <b>G_Stream</b>   | stream using gas composition   | general  |
| <b>OR_Stream</b>  | stream using working fluids for ORC, refrigeration and heat pump processes | adapted  |
| <b>TCM_Stream</b> | thermochemical storage material (TCM) stream                               | specific |
| <b>T_Stream</b>   | stream representing a heat transfer fluid (thermofluid)                    | general  |
| <b>W_Stream</b>   | stream for water   | general  |

|                     |  |          |
|---------------------|--|----------|
| <b>q_cond_trans</b> | conductive heat transfer through a wall (solid boundary) | specific |
| <b>shaft</b>        | shaft  | general  |

## 2.3. Available global objects

*Table 3: Available Globals*

| <b>Global Name</b>    | <b>Description</b>   | <b>Type</b> |
|-----------------------|--|-------------|
| <b>G_Composition</b>  | chemical composition of a gaseous working fluid                          | general     |
| <b>OR_Composition</b> | working fluid (usually organic hydrocarbons and some other alternatives) | adapted     |
| <b>TCM</b>            | thermochemical material working pair                                     | specific    |
| <b>T_Composition</b>  | heat transfer fluid  | general     |
| <b>time_period</b>    | time period used for a certain operation                                 | specific    |

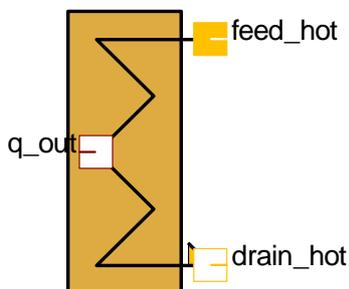
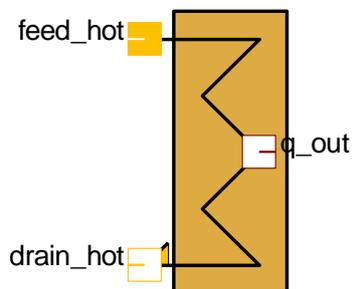
## 3. Component Model Details

This chapter provides details of the updated RESTORE specific component models contained in the last version of the RESTORE\_Lib model library. It presents the model description, lists the model equations and variables representing individual models, and displays the icons representing the individual units.

### 3.1. OR\_wall\_htex

#### Purpose

Heat exchanger with wall transferring heat from organic fluid (OR) to another side.



#### Connections

OR\_Stream: feed\_hot  
OR\_Stream: drain\_hot  
q\_cond\_trans: q\_out

#### 3.1.1. OR\_wall\_htex

#### Purpose

design model

#### Model equations

# heat transfer from fluid inside the tubes through the wall

# mass transfer

```
f_mass: feed_hot.mass = drain_hot.mass;

# pressure drop through the wall tubes
f_delta_p:      feed_hot.p - delta_p = drain_hot.p;

# hot fluid is cooled down
f_energy: feed_hot.mass*feed_hot.h - q_trans = drain_hot.mass*drain_hot.h;

# temperatue change
f_delta_t: feed_hot.t - delta_t = drain_hot.t;

# conductive heat transfer through the wall
f_q_q_trans:    q_trans = q_out.q_trans;

# transfer of prevailing temperatures
ifl Type == cocurrent then
  f_t_hot_in_co: q_out.t_hot_in = feed_hot.t;
  f_t_hot_out_co: q_out.t_hot_out = drain_hot.t;
endifl

ifl Type == counter_current then
  f_t_hot_in_counter:    q_out.t_hot_in = drain_hot.t;
  f_t_hot_out_counter:  q_out.t_hot_out = feed_hot.t;
endifl

# additional calculations and data for operating points of the working fluid with phase change
# (inside the tubes)
# heat is flowing out, working fluid is cooled and may condense

# pressure drop is assumed to be linear between feed and drain

# intermediate point is either inlet conditions (if already 2-phase) or where condensation starts
# (saturated vapour)
f_h_II_aux:      h_II_aux = feed_hot.Composition.o_h_px(feed_hot.p, 1.0);

f_h_II:  if (feed_hot.h > h_II_aux) then
          h_II = feed_hot.Composition.o_h_px(p_II, 1.0);
        else
          h_II = feed_hot.h;
f_p_II:  if (feed_hot.h > h_II_aux) then
          p_II = drain_hot.p + (h_II-drain_hot.h)/(feed_hot.h-
drain_hot.h)*(feed_hot.p - drain_hot.p);
        else
          p_II = feed_hot.p;

f_t_II:  if (feed_hot.h > h_II_aux) then
          t_II = feed_hot.Composition.o_tsat_p(p_II);
        else
          t_II = feed_hot.t;

f_s_II:  if (feed_hot.h > h_II_aux) then
          s_II = feed_hot.Composition.o_s_px(p_II, 1.0);
```

```

else
    s_II = feed_hot.s;

# condensation end point is saturated liquid
f_h_sl:  h_sl = feed_hot.Composition.o_h_px(p_sl, 0.0);
f_p_sl:  p_sl = drain_hot.p + (h_sl-drain_hot.h)/(feed_hot.h-drain_hot.h)*(feed_hot.p -
drain_hot.p);
f_t_sl:  t_sl = feed_hot.Composition.o_tsat_p(p_sl);
f_s_sl:  s_sl = feed_hot.Composition.o_s_px(p_sl, 0.0);

# test conditions
t_q_trans:      test (q_trans >= 0.0) warning "q_trans negative";
t_delta_p:      test (delta_p >= 0.0) warning "delta_p negative";

# feed and drain terminal must reference the same composition
ifl ref(feed_hot.Composition) != ref(drain_hot.Composition) then
    t_Comp:      test(1!=1) error "Composition must be the same at feed and drain";
endifl

```

#### Parameters

delta\_p                      absolute pressure drop

#### Variables

|          |   |
|----------|---|
| delta_t  | temperature difference between feed and drain mass flow |
| q_trans  | transferred heat  |
| p_II     | intermediate pressure (sat vap resp. inlet)             |
| t_II     | intermediate temperature (sat vap resp. inlet)          |
| h_II     | intermediate enthalpy (sat vap resp. inlet)             |
| s_II     | intermediate entropy (sat vap resp. inlet)              |
| h_II_aux | auxiliary vapour enthalpy                               |
| p_sl     | saturated condensate pressure                           |
| t_sl     | saturated condensate temperature                        |
| h_sl     | saturated condensate enthalpy                           |
| s_sl     | saturated condensate entropy                            |

#### Switches

Type (cocurrent, counter\_current)                      defines the type of heat exchanger

## 3.2. RESTORE\_EA

#### Purpose

Component model supplying economic analysis information.



### 3.2.1. RESTORE\_EA

#### Purpose

default model for economic analysis

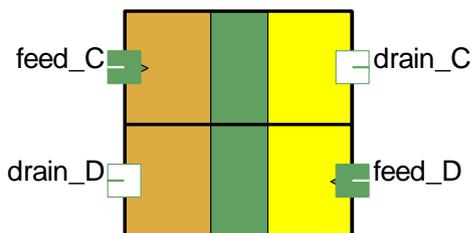
#### Model equations

- # Currently the model is used to switch the economic analysis on and off.
- # It does not directly contain any model variables and equations.

## 3.3. TCM\_CD\_Transformer

#### Purpose

Time shifting transformer for thermochemical materials to allow combining charging and discharging in one flowsheet.



#### Connections

- TCM\_Stream: feed\_C
- TCM\_Stream: drain\_D
- TCM\_Stream: drain\_C
- TCM\_Stream: feed\_D

### 3.3.1. TCM\_CD\_Transformer

#### Purpose

default model

#### Model equations

- # The model allows to transform the mass flows between charging and discharging.
- # It can be used to simulate successive charging and discharging operation in a single calculation.
- # Thereby it is possible to create closed charging and discharging cycles with steady-state models.

- # Note that the cycle will be closed for oil and the thermochemical material, but open for water,
- # as water is expelled from the charging reactor and added to the discharging reactor. It will be

```
a half-open cycle.
# In this way the model acts as a connector and consequently some of the mass balances for
either C or D are redundant!

# charging to discharging mass flow ratio
f_TCM_ratio:    feed_C.mass = TCM_ratio * drain_C.mass;

# mass balances

# charged material
# total and individual (one is redundant)
#f_mass_C:      feed_C.mass = drain_C.mass;

# For the individual material, the mass fraction is used here.
# Thereby one can easily transform the mass flow, as the fractions remain constant.
#f_w_Oil_C:     drain_C.w_Oil = feed_C.w_Oil;
f_w_Material1_C: feed_C.w_Material1 = drain_C.w_Material1;
f_w_Material2_C: feed_C.w_Material2 = drain_C.w_Material2;
f_w_Water_C:   feed_C.w_Water = drain_C.w_Water;

# discharged material
# total and individual (one is redundant)
#f_mass_D:      feed_D.mass * TCM_ratio = drain_D.mass;

# For the individual material, the mass fraction is used here.
# Thereby one can easily transform the mass flow, as the fractions remain constant.
#f_w_Oil_D:     drain_D.w_Oil = feed_D.w_Oil; (redundant, as oil is circulated in a
closed cycle)
#f_w_Material1_D: feed_D.w_Material1 = drain_D.w_Material1; (redundant, as TCM is
circulated in a closed cycle)

# Material 1 must not be converted to Material 2 (and vice versa).
# Hence the composition of of one of the two materials is passed through, then it is balanced.
_w_Material2_D: feed_D.w_Material2 = drain_D.w_Material2;

# Note here the balance for water
f_w_Water_D:   feed_D.w_Water = drain_D.w_Water;

# state properties remain constant
f_t_C:    feed_C.t = drain_C.t;
f_p_C:    feed_C.p = drain_C.p;
f_t_D:    feed_D.t = drain_D.t;
f_p_D:    feed_D.p = drain_D.p;

# test conditions

# tests for matching mass balances
t_mass: test (abs(feed_D.mass*TCM_ratio - drain_D.mass) < 1e-3)    error "overall mass
balance not correct!";
t_mass_Oil: test (abs(feed_D.mass_Oil*TCM_ratio - drain_D.mass_Oil) < 1e-3)    error
"oil mass balance not correct!";
```

```
t_mass_Material1:      test (abs(feed_D.mass_Material1*TCM_ratio -
drain_D.mass_Material1) < 1e-3) error "material 1 mass balance not correct!";
t_mass_Material2:      test (abs(feed_D.mass_Material2*TCM_ratio -
drain_D.mass_Material2) < 1e-3) error "material 2 mass balance not correct!";

# no change of substances
t_conn_material1_C: test (feed_C.TCM.ID1 == drain_C.TCM.ID1)
                        error "different material 1 at feed_C and drain_C!";
t_conn_material2_C: test (feed_C.TCM.ID2 == drain_C.TCM.ID2)
                        error "different material 2 at feed_C and drain_C!";
t_conn_oil_C: test (feed_C.T_Composition.FluidID == drain_C.T_Composition.FluidID)
                  error "different oil at feed_C and drain_C!";
t_conn_material1_D: test (feed_D.TCM.ID1 == drain_D.TCM.ID1)
                        error "different material 1 at feed_D and drain_D!";
t_conn_material2_D: test (feed_D.TCM.ID2 == drain_D.TCM.ID2)
                        error "different material 2 at feed_D and drain_D!";
t_conn_oil_D: test (feed_D.T_Composition.FluidID == drain_D.T_Composition.FluidID)
                  error "different oil at feed_D and drain_D!";
```

#### Variables

TCM\_ratio                      ratio of TCM charging to discharging mass flow

## 3.4. TCM\_Connector

#### Purpose

Connector element that removes the redundant mass balance equations from closed TCM loops.



#### Connections

TCM\_Stream: drain  
TCM\_Stream: feed

### 3.4.1. TCM\_Connector

#### Purpose

This model has to be used in closed loops. The mass balance is not defined, pressure and enthalpy of the connected streams are passed through.

#### Model equations

```
# mass balances are not defined!

# transfer of state variables
f_p:    feed.p = drain.p;
f_t:    feed.t = drain.t;
```

```
# tests for matching mass balances
t_mass: test (abs(feed.mass - drain.mass) < 1e-3)      error "mass balance not correct!";
t_mass_Material1:      test (abs(feed.mass_Material1 - drain.mass_Material1) < 1e-3) error
"material 1 mass balance not correct!";
t_mass_Material2:      test (abs(feed.mass_Material2 - drain.mass_Material2) < 1e-3) error
"material 2 mass balance not correct!";
t_mass_Water: test (abs(feed.mass_Water - drain.mass_Water) < 1e-3) error "water mass
balance not correct!";
t_mass_Oil:      test (abs (drain.mass_Oil - drain.mass_Oil) < 1e-3) error "oil mass balance not
correct!";

# no change of substances
t_conn_material1: test (feed.TCM.ID1 == drain.TCM.ID1)
                    error "different material 1 at feed and drain!";
t_conn_material2: test (feed.TCM.ID2 == drain.TCM.ID2)
                    error "different material 2 at feed and drain!";
t_conn_oil: test (feed.T_Composition.FluidID == drain.T_Composition.FluidID)
                error "different oil at feed and drain!";
```

### 3.4.2. TCM\_Connector\_Oil\_TCM

#### Purpose

This model has to be used in closed oil loops, which are open for water. In such a situation, water is balanced. The mass balances are not defined for oil and thermochemical materials, pressure and temperature of the connected TCM\_Streams are passed through.

#### Model equations

```
# This model must be used for cycles which are closed for oil and the thermochemical material,
but open for water.
# This is the case for a charging/discharging cycle where water is expelled from the charging
reactor and added to
# the discharging reactor. It will be a half-open cycle.

# Mass balances for the closed loop materials (oil and TCM) are not defined.

# The process where this model is used is open for water, which must be balanced.
# Hence there is a balance for water built in.
f_mass_Water: feed.mass_Water = drain.mass_Water;

# Material 1 must not be converted to Material 2 (and vice versa).
# Hence the composition of of one of the two materials is passed through, then it is balanced.
f_w_Material2: feed.w_Material2 = drain.w_Material2;

# transfer of state variables
f_p: feed.p = drain.p;
f_t: feed.t = drain.t;

# tests for matching mass balances
t_mass: test (abs(feed.mass - drain.mass) < 1e-3)      error "overall mass balance not
```

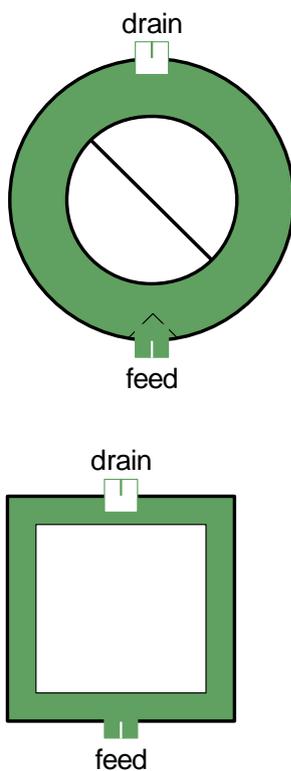
```
correct!";
t_mass_Oil:      test (abs(feed.mass_Oil - drain.mass_Oil) < 1e-3)      error "oil mass
balance not correct!";
t_mass_Material1:      test (abs(feed.mass_Material1 - drain.mass_Material1) < 1e-3) error
"material 1 mass balance not correct!";
t_mass_Material2:      test (abs(feed.mass_Material2 - drain.mass_Material2) < 1e-3) error
"material 2 mass balance not correct!";

# no change of substances
t_conn_material1: test (feed.TCM.ID1 == drain.TCM.ID1)
                    error "different material 1 at feed and drain!";
t_conn_material2: test (feed.TCM.ID2 == drain.TCM.ID2)
                    error "different material 2 at feed and drain!";
t_conn_oil: test (feed.T_Composition.FluidID == drain.T_Composition.FluidID)
               error "different oil at feed and drain!";
```

### 3.5. TCM\_Heat\_sink

#### Purpose

Heat sink or general heat consumer in a heat network for thermochemical materials



#### Connections

TCM\_Stream: feed  
TCM\_Stream: drain

### 3.5.1. TCM\_Heat\_sink

#### Purpose

default model

#### Model equations

```
# mass balance

f_mass:                feed.mass = drain.mass;
#f_massOil:           drain.mass_Oil = feed.mass_Oil;
f_massMaterial1: feed.mass_Material1 = drain.mass_Material1;
f_massMaterial2: feed.mass_Material2 = drain.mass_Material2;
f_massWater:         feed.mass_Water = drain.mass_Water;

# energy balance
# q_trans > 0

f_energy: feed.mass * feed.h - q_trans = drain.mass * drain.h;

f_delta_t: feed.t - delta_t = drain.t;
f_delta_p:  feed.p - delta_p = drain.p;

# test conditions

t_q_trans:  test (q_trans >= 0.0) warning "q_trans negative";
t_delta_p:  test (delta_p >= 0.0) warning "delta_p negative";
t_delta_t:  test (delta_t >= 0.0) warning "delta_t negative";

# no change of substances
t_conn_material1: test (feed.TCM.ID1 == drain.TCM.ID1)
                  error "different material 1 at feed and drain!";
t_conn_material2: test (feed.TCM.ID2 == drain.TCM.ID2)
                  error "different material 2 at feed and drain!";
t_conn_oil: test (feed.T_Composition.FluidID == drain.T_Composition.FluidID)
            error "different oil at feed and drain!";
```

#### Parameters

delta\_p absolute pressure drop

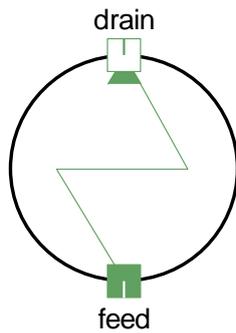
#### Variables

q\_trans transferred heat  
delta\_t temperature difference between feed and drain mass flow

### 3.6. TCM\_Heat\_source

#### Purpose

General heat source in a heat network for thermochemical materials.



### Connections

TCM\_Stream: feed  
TCM\_Stream: drain

### 3.6.1. TCM\_Heat\_source

#### Purpose

default model

#### Model equations

# mass balance

```
f_mass:          feed.mass = drain.mass;
#f_massOil:      drain.mass_Oil = feed.mass_Oil;
f_massMaterial1: feed.mass_Material1 = drain.mass_Material1;
f_massMaterial2: feed.mass_Material2 = drain.mass_Material2;
f_massWater:     feed.mass_Water = drain.mass_Water;
```

# energy balance

```
f_energy: q_trans = drain.mass * drain.h - feed.mass * feed.h;
```

```
f_delta_t: feed.t + delta_t = drain.t;
f_delta_p:  feed.p - delta_p = drain.p;
```

# test conditions

```
t_q_trans:      test (q_trans >= 0.0) warning "q_trans negative";
t_delta_p:      test (delta_p >= 0.0) warning "delta_p negative";
t_delta_t:      test (delta_t >= 0.0) warning "delta_t negative";
```

# no change of substances

```
t_conn_material1: test (feed.TCM.ID1 == drain.TCM.ID1)
                    error "different material 1 at feed and drain!";
t_conn_material2: test (feed.TCM.ID2 == drain.TCM.ID2)
                    error "different material 2 at feed and drain!";
t_conn_oil:        test (feed.T_Composition.FluidID == drain.T_Composition.FluidID)
                    error "different oil at feed and drain!";
```

#### Parameters

delta\_p absolute pressure drop

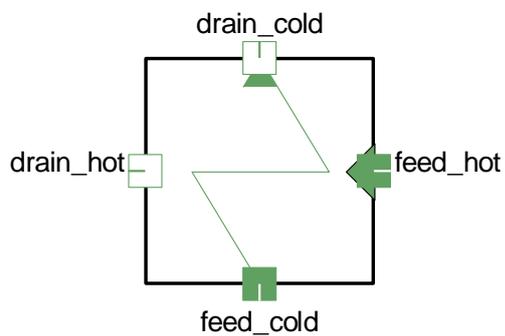
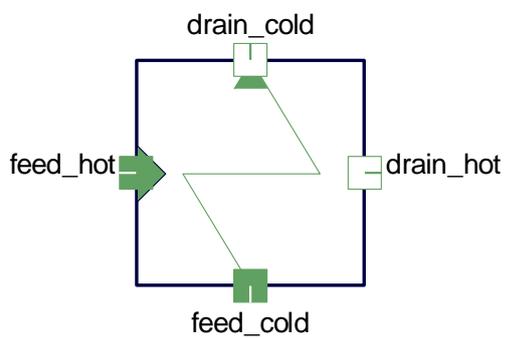
### Variables

|         |   |
|---------|---|
| q_trans | transferred heat                              |
| delta_t | temperature difference between drain and feed |

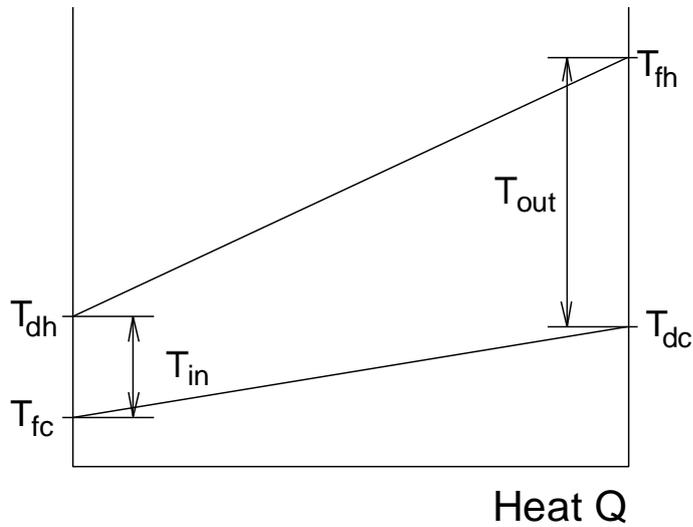
## 3.7. TCM\_Htex

### Purpose

Heat exchanger for transfer from TCM fluid on the hot side to TCM fluid on the cold side. Cocurrent or counter current flow.

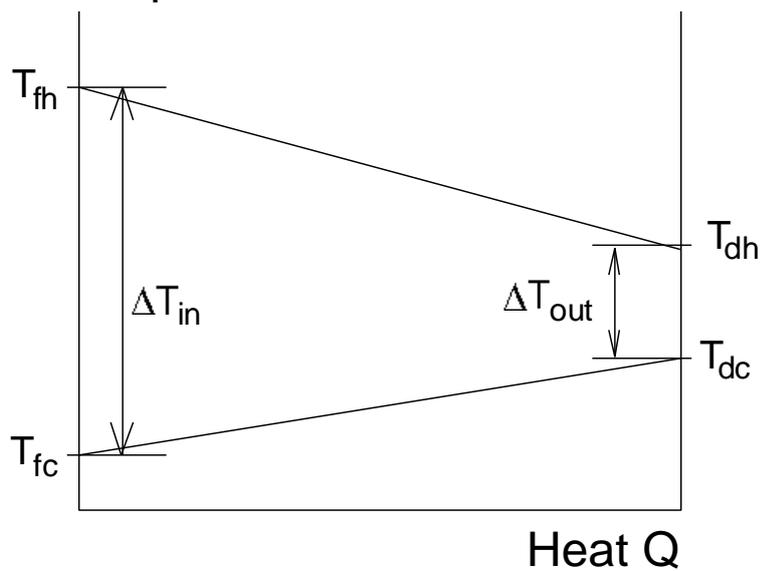


## Temperature T



QT-diagram, counter current

## Temperatur T



QT-diagram, cocurrent

### Connections

TCM\_Stream: drain\_hot  
TCM\_Stream: feed\_hot  
TCM\_Stream: drain\_cold  
TCM\_Stream: feed\_cold

### 3.7.1. TCM\_Htex

#### Purpose

design model

#### Model equations

# mass balances

```
# hot side total and individual mass balances (one is redundant)
f_mass_hot:          feed_hot.mass = drain_hot.mass;
#f_mass_Oil_hot:    feed_hot.mass_Oil = drain_hot.mass_Oil;
f_mass_Material1_hot:  feed_hot.mass_Material1 = drain_hot.mass_Material1;
f_mass_Material2_hot:  feed_hot.mass_Material2 = drain_hot.mass_Material2;
f_mass_Water_hot:     feed_hot.mass_Water = drain_hot.mass_Water;

# cold side total and individual mass balances (one is redundant)
f_mass_cold:         feed_cold.mass = drain_cold.mass;
#f_mass_Oil_cold:    feed_cold.mass_Oil = drain_cold.mass_Oil;
f_mass_Material1_cold: feed_cold.mass_Material1 = drain_cold.mass_Material1;
f_mass_Material2_cold: feed_cold.mass_Material2 = drain_cold.mass_Material2;
f_mass_Water_cold:   feed_cold.mass_Water = drain_cold.mass_Water;

# pressure drops

f_delta_p_hot:       feed_hot.p - delta_p_hot = drain_hot.p;
f_delta_p_cold:      feed_cold.p - delta_p_cold = drain_cold.p;

# energy balance

f_e_hot: feed_hot.mass * (feed_hot.h - drain_hot.h) * (1.0 - heat_loss/100) = q_trans;
f_e_cold: feed_cold.mass * (drain_cold.h - feed_cold.h) = q_trans;

# temperature differences
# They are differently defined for co- and counter current heat exchangers.

ifl Type == cocurrent then
  f_dt_in_co:   feed_hot.t - dt_in = feed_cold.t;
  f_dt_out_co:  drain_hot.t - dt_out = drain_cold.t;
endifl

ifl Type == counter_current then
  f_dt_in_counter:  drain_hot.t - dt_in = feed_cold.t;
  f_dt_out_counter: feed_hot.t - dt_out = drain_cold.t;
endifl

# Mean temperature difference is the logarithmic temperature difference.
# For nearly equal inlet and outlet temperature differences, it is approximated
# by the arithmetic mean (which is the asymptotic approximation).
f_MTD: if abs(dt_in/dt_out) >= 1.2 || abs(dt_out/dt_in) >= 1.2 then
  MTD * ln(dt_in/dt_out) = (dt_in - dt_out);
else
  MTD = (dt_in + dt_out) / 2.0;

f_htc_area:   q_trans = htc_area * MTD;

# tests
t_dt_in: test(dt_in > 0.0) error "dt_in <= 0.0";
t_dt_out: test(dt_out > 0.0) error "dt_out <= 0.0";
t_q_trans: test(q_trans > 0.0) error "q_trans <= 0.0";
```

```
# no change of substances
t_drain_hot_material1: test (feed_hot.TCM.ID1 == drain_hot.TCM.ID1)
                        error "Different material 1 at feed_hot and drain_hot!";
t_drain_hot_material2: test (feed_hot.TCM.ID2 == drain_hot.TCM.ID2)
                        error "Different material 2 at feed_hot and drain_hot!";
t_drain_hot_oil: test (feed_hot.T_Composition.FluidID == drain_hot.T_Composition.FluidID)
                    error "Different oil at feed_hot and drain_hot!";
t_drain_cold_material1: test (feed_cold.TCM.ID1 == drain_cold.TCM.ID1)
                        error "Different material 1 at feed_cold and drain_cold!";
t_drain_cold_material2: test (feed_cold.TCM.ID2 == drain_cold.TCM.ID2)
                        error "Different material 2 at feed_cold and drain_cold!";
t_drain_cold_oil: test (feed_cold.T_Composition.FluidID == drain_cold.T_Composition.FluidID)
                    error "Different oil at feed_cold and drain_cold!";
```

### Parameters

|              |   |
|--------------|---|
| delta_p_hot  | pressure drop of the high temperature side  |
| delta_p_cold | pressure drop of the low temperature side   |
| heat_loss    | percentage of heat lost to the surroundings |

### Variables

|          |  |
|----------|--|
| dt_in    | temperature difference at the inlet of the low temperature side  |
| dt_out   | temperature difference at the outlet of the low temperature side |
| htc_area | heat transfer coefficient x transfer area                        |
| MTD      | mean temperature difference                                      |
| q_trans  | transferred heat   |

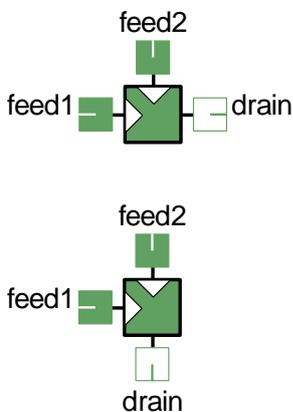
### Switches

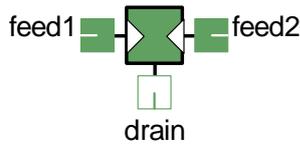
|                                   |                                    |
|-----------------------------------|------------------------------------|
| Type (cocurrent, counter_current) | defines the type of heat exchanger |
|-----------------------------------|------------------------------------|

## 3.8. TCM\_Mixer

### Purpose

Mixer for TCM streams.





### Connections

TCM\_Stream: feed1  
TCM\_Stream: feed2  
TCM\_Stream: drain

## 3.8.1. TCM\_Mixer

### Purpose

mixer model for two incoming streams

### Model equations

```
# mass balances

# overall
f_mass:          feed1.mass + feed2.mass = drain.mass;

# one of the individual mass balances is redundant and hence omitted
#f_mass_Oil:     feed1.mass_Oil + feed2.mass_Oil = drain.mass_Oil;
f_mass_Material1: feed1.mass_Material1 + feed2.mass_Material1 =
drain.mass_Material1;
f_mass_Material2: feed1.mass_Material2 + feed2.mass_Material2 =
drain.mass_Material2;
f_Mass_Water:    feed1.mass_Water + feed2.mass_Water = drain.mass_Water;

# pressure drops
f_p1: feed1.p - delta_p_1 = drain.p;
f_p2: feed2.p - delta_p_2 = drain.p;

# energy balance
f_energy: feed1.h * feed1.mass + feed2.h * feed2.mass = drain.h * drain.mass;

# test for positive pressure drops
t_delta_p_1: test (delta_p_1 >= 0.0)      warning "pressure drop delta_p_1 is negative";
t_delta_p_2: test (delta_p_2 >= 0.0)      warning "pressure drop delta_p_2 is negative";

# substances must be the same
t_feed1_material1: test (feed1.TCM.ID1 == drain.TCM.ID1)
                    error "different material 1 at feed1 and drain!";
t_feed1_material2: test (feed1.TCM.ID2 == drain.TCM.ID2)
                    error "different material 2 at feed1 and drain!";
t_feed1_oil: test (feed1.T_Composition.FluidID == drain.T_Composition.FluidID)
              error "different oil at feed1 and drain!";
t_feed2_material1: test (feed2.TCM.ID1 == drain.TCM.ID1)
                    error "different material 1 at feed2 and drain!";
t_feed2_material2: test (feed2.TCM.ID2 == drain.TCM.ID2)
```

```
error "different material 2 at feed2 and drain!";  
t_feed2_oil: test (feed2.T_Composition.FluidID == drain.T_Composition.FluidID)  
error "different oil at feed2 and drain!";
```

#### Variables

|           |   |
|-----------|---|
| delta_p_1 | absolute pressure drop of a stream attached to feed_1 |
| delta_p_2 | absolute pressure drop of stream attached to feed_2   |

## 3.9. TCM\_Pipe

### Purpose

Pipe for thermochemical material (TCM).



### Connections

TCM\_Stream: drain  
TCM\_Stream: feed

### 3.9.1. TCM\_Pipe

#### Purpose

design model

#### Model equations

```
# mass balances  
# total and individual (one is redundant)  
f_mass: feed.mass = drain.mass;  
f_mass_Material1: feed.mass_Material1 = drain.mass_Material1;  
f_mass_Material2: feed.mass_Material2 = drain.mass_Material2;  
f_mass_Water: feed.mass_Water = drain.mass_Water;  
# f_mass_Oil: drain.mass_Oil = drain.mass_Oil;  
  
# pressure drop equations  
f_delta_p_rel: (1.0-pressure_drop)*feed.p = drain.p;  
f_delta_p_abs: feed.p - delta_p = drain.p;  
  
# temperature loss  
f_delta_t: feed.t-delta_t = drain.t;  
  
# enthalpy loss  
f_delta_h: feed.h-delta_h = drain.h;  
  
# energy balance with heat loss to environment  
f_energy: feed.mass*(feed.h-drain.h) = q_loss;  
  
# test conditions
```

```
t_delta_p:      test(delta_p>=0.0) warning "pressure drop negative";

# no change of substances
t_conn_material1: test (feed.TCM.ID1 == drain.TCM.ID1)
                    error "different material 1 at feed and drain!";
t_conn_material2: test (feed.TCM.ID2 == drain.TCM.ID2)
                    error "different material 2 at feed and drain!";
t_conn_oil: test (feed.T_Composition.FluidID == drain.T_Composition.FluidID)
               error "different oil at feed and drain!";
```

### Variables

|               |                        |
|---------------|------------------------|
| delta_p       | absolute pressure drop |
| pressure_drop | relative pressure drop |
| q_loss        | heat loss              |
| delta_t       | temperature loss       |
| delta_h       | enthalpy loss          |

## 3.9.2. TCM\_Pipe\_pl

### Purpose

Off-design model. This model describes the dependence of the pressure drop upon the mass flow.

### Model equations

```
# mass balances
# total and individual (one is redundant)
f_mass: feed.mass = drain.mass;
f_mass_Material1: feed.mass_Material1 = drain.mass_Material1;
f_mass_Material2: feed.mass_Material2 = drain.mass_Material2;
f_mass_Water: feed.mass_Water = drain.mass_Water;
# f_mass_Oil: drain.mass_Oil = drain.mass_Oil;

# pressure drop equations
f_delta_p_rel: (1.0-pressure_drop)*feed.p = drain.p;
f_delta_p_abs: feed.p - delta_p = drain.p;

# temperature loss
f_delta_t: feed.t-delta_t = drain.t;

# enthalpy loss
f_delta_h: feed.h-delta_h = drain.h;

# energy balance with heat loss to environment
f_energy: feed.mass*(feed.h-drain.h) = q_loss;

# part-load relationships
# dependence pressure drop - mass flow taking
f_delta_p_pl: delta_p/delta_p0 = (feed.mass/massf0)^2;
# end of part-load relationships
```

```
# test conditions
t_delta_p:      test(delta_p>=0.0) warning "pressure drop negative";

# no change of substances
t_conn_material1: test (feed.TCM.ID1 == drain.TCM.ID1)
                  error "different material 1 at feed and drain!";
t_conn_material2: test (feed.TCM.ID2 == drain.TCM.ID2)
                  error "different material 2 at feed and drain!";
t_conn_oil: test (feed.T_Composition.FluidID == drain.T_Composition.FluidID)
               error "different oil at feed and drain!";
```

#### Parameters

|          |                                      |
|----------|--------------------------------------|
| delta_p0 | absolute pressure drop, design point |
| massf0   | mass flow, design point.             |

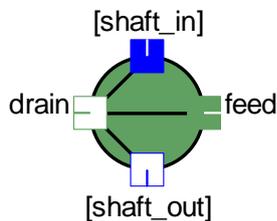
#### Variables

|               |                        |
|---------------|------------------------|
| delta_p       | absolute pressure drop |
| pressure_drop | relative pressure drop |
| q_loss        | heat loss              |
| delta_t       | temperature loss       |
| delta_h       | enthalpy loss          |

## 3.10. TCM\_Pump

### Purpose

Pump for TCM fluids.



### Connections

shaft: shaft\_in  
shaft: shaft\_out  
shaft: shaft\_in  
shaft: shaft\_out  
TCM\_Stream: feed  
TCM\_Stream: drain

### 3.10.1. TCM\_Pump

#### Purpose

design model

#### Model equations

```
# mass balances
# total and individual (one is redundant)
f_mass: feed.mass = drain.mass;
f_mass_Material1: feed.mass_Material1 = drain.mass_Material1;
f_mass_Material2: feed.mass_Material2 = drain.mass_Material2;
f_mass_Water: feed.mass_Water = drain.mass_Water;
# f_mass_Oil: drain.mass_Oil = drain.mass_Oil;

# pump efficiency definition
f_eta_p: (drain.p - feed.p)*feed.v*100 / eta_p = drain.h - feed.h;

# both sides connected
ifl ref(shaft_in) && ref(shaft_out) then
  f_powera: (feed.h - drain.h)* feed.mass / eta_m + shaft_in.power - shaft_out.power = 0.0;
endifl

# left side shaft only
ifl ref(shaft_in) && !ref(shaft_out) then
  f_powerb: (feed.h - drain.h)* feed.mass / eta_m + shaft_in.power = 0.0;
endifl

# right side shaft only
ifl !ref(shaft_in) && ref(shaft_out) then
```

```
f_powerc:      (feed.h - drain.h) * feed.mass / eta_m - shaft_out.power = 0.0;
endifl

# test conditions
t_delta_p:     test((drain.p - feed.p) >= 0.0)   warning "outlet pressure lower than inlet
pressure";
t_eta_p_low:   test ( eta_p >= 0.0)              error "pump efficiency < 0.0";
t_eta_p_high:  test ( eta_p <= 1.0)              error "pump efficiency > 1.0";
t_eta_m_low:   test ( eta_m >= 0.0)              error "mechanical efficiency < 0.0";
t_eta_m_high:  test ( eta_m <= 1.0)              error "mechanical efficiency > 1.0";

# no water vapour at inlet
t_p_cav: test (if(feed.w_Water < 1e-6) then feed.p_required else feed.p >= feed.p_required)
           warning "water vapour at inlet, cavitation will occur!";

# no change of substances
t_conn_material1: test (feed.TCM.ID1 == drain.TCM.ID1)
                   error "different material 1 at feed and drain!";
t_conn_material2: test (feed.TCM.ID2 == drain.TCM.ID2)
                   error "different material 2 at feed and drain!";
t_conn_oil: test (feed.T_Composition.FluidID == drain.T_Composition.FluidID)
              error "different oil at feed and drain!";
```

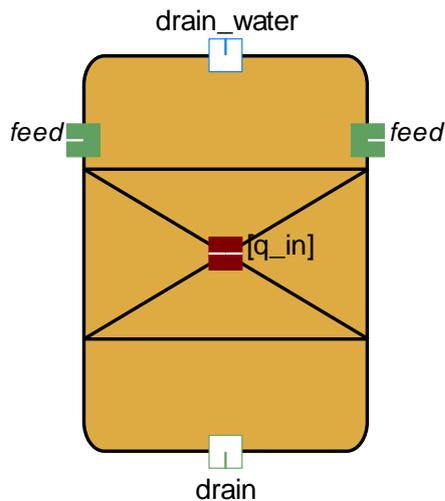
#### Parameters

|       |                       |
|-------|-----------------------|
| eta_p | pump efficiency       |
| eta_m | mechanical efficiency |

### 3.11. TCM\_Reactor\_Charging

#### Purpose

Reactor for charging step of thermochemical material (TCM). High temperature side transferring heat to the reactor is optional. Different heat delivering working fluids (OR\_ or heat transfer (T\_) fluids ) can be connected.



### Connections

TCM\_Stream: feed  
TCM\_Stream: drain  
W\_Stream: drain\_water  
q\_cond\_trans: q\_in

### 3.11.1. TCM\_Reactor\_Charging

#### Purpose

Design model for isothermal reactor charging the TCM.

#### Model equations

# mass balances

# Due to the summary balance in the TCM\_stream, one of the mass balances is redundant.

# The overall mass balance is used just for testing correctness, see test section

#f\_mass: feed.mass = drain.mass + drain\_water.mass;

f\_mass\_oil: feed.mass\_Oil = drain.mass\_Oil;

f\_mass\_Material1: drain.mass\_Material1 = (1 - conv\_TCM) \* feed.mass\_Material1;

f\_mass\_Material2: drain.mass\_Material2 = feed.mass\_Material2 + conv\_TCM \*  
feed.mass\_Material1 / molarmass\_Material1 \* molarmass\_Material2;

# Feed may contain small amounts of water which will also be evaporated.

f\_mass\_Water: drain\_water.mass = ny\_H2O\_TCM \* conv\_TCM \*  
feed.mass\_Material1 / molarmass\_Material1 \* 18.0153 + feed.mass\_Water;

# All water will be evaporated and goes to drain\_water. No water in exiting TCM.

f\_no\_Water\_drain: drain.mass\_Water = 0.0;

#-----

f\_molarmass\_Material1: molarmass\_Material1 = feed.TCM.TCM\_M(feed.TCM.ID1);

f\_molarmass\_Material2: molarmass\_Material2 = feed.TCM.TCM\_M(feed.TCM.ID2);

```
# stoichiometric coefficient for the extracted water:
f_ny_H2O:
#Boric acid:
if feed.TCM.ID1 <= 1.1 then
  ny_H2O_TCM = 1.0;
else
#Coppersulphate pentahydrate:
if feed.TCM.ID1 <= 3.1 then
  ny_H2O_TCM = 4.0;
else
#Calciumchloride dihydrate:
if feed.TCM.ID1 <= 5.1 then
  ny_H2O_TCM = 2.0;
else
#Potassiumcarbonate 1.5 hydrate:
if feed.TCM.ID1 <= 7.1 then
  ny_H2O_TCM = 1.5;
else
  ny_H2O_TCM = -1.;
#-----
# pressure balance

f_p1:      drain.p = feed.p - delta_p;
f_p2:      drain.p = drain_water.p;

# energy balance

# temperature

f_t1:      drain_water.t = t_reac;
f_t2:      drain.t = t_reac;

# heat transfer

f_q_trans:  q_trans = q_reac + q_preheat;

ifl ref(q_in) then # the hot side is connected

  # conductive heat transfer through the wall
  f_q_q_trans:  q_trans = q_in.q_trans;

  # transfer of prevailing temperatures at the heat transfer surface
  # the reactor is isothermal conditions
  f_t_cold_in:  q_in.t_cold_in = t_reac;
  f_t_cold_out: q_in.t_cold_out = t_reac;
endifl

# preheating, considering there is no water in the feed

f_q_preheat:  q_preheat = drain.h_Material1 * feed.mass_Material1 + drain.h_Material2 *
feed.mass_Material2 + drain.h_Oil * feed.mass_Oil + drain.h_Water * feed.mass_Water - feed.h
* feed.mass;
```

# reaction of dehydration

```
f_q_reac:q_reac = (hWater - 285.83 / 18.0153 * 1000) * ((ny_H2O_TCM * conv_TCM *  
feed.mass_Material1 / molarmass_Material1 * 18.0153) ) +  
((drain.mass_Material2 - feed.mass_Material2) * (drain.h_Material2 +  
H_298_15_Material2 / molarmass_Material2 * 1000) -  
((feed.mass_Material1 - drain.mass_Material1) * (drain.h_Material1 +  
H_298_15_Material1 / molarmass_Material1 * 1000));
```

```
f_hWater: hWater = drain_water.fhpt(drain.p, drain.t, 1,0,0,0,0,0,0,0,0,0,0,0,0,0,0) -  
drain_water.fhpt(1, 25, 1,0,0,0,0,0,0,0,0,0,0,0,0,0);
```

# enthalpies of formation of the working pair materials

f\_H\_298\_15\_Material1:

#Boric acid:

if feed.TCM.ID1 <= 1.1 then

H\_298\_15\_Material1 = - 1093.99;

else

#Coppersulphate pentahydrate:

if feed.TCM.ID1 <= 3.1 then

H\_298\_15\_Material1 = -2276.512;

else

#Calciumchloride dihydrate:

if feed.TCM.ID1 <= 5.1 then

H\_298\_15\_Material1 = - 1403.9;

else

#Potassiumcarbonate 1.5H2O:

if feed.TCM.ID1 <= 7.1 then

H\_298\_15\_Material1 = -1612.930;

else

H\_298\_15\_Material1 = 0.0;

#-----

f\_H\_298\_15\_Material2:

#Metaboric acid:

if feed.TCM.ID2 <= 2.1 then

H\_298\_15\_Material2 = - 802.78;

else

#Coppersulphate monohydrate:

if feed.TCM.ID2 <= 4.1 then

H\_298\_15\_Material2 = -1082.818;

else

#Calciumchloride:

if feed.TCM.ID2 <= 6.1 then

H\_298\_15\_Material2 = - 795.8;

else

```
#Potassiumcarbonate:
if feed.TCM.ID2 <= 8.1 then
  H_298_15_Material2 = -1151.499;
else
  H_298_15_Material2 = 0.0;

# minimum and maximum temperatures of the reactions that can be used as range
temperatures for the charging reactor

f_t_min:
#Boric acid:
if feed.TCM.ID1 <= 1.1 then
  t_min = 145.0;
else

#Coppersulphate pentahydrate:
if feed.TCM.ID1 <= 3.1 then
  t_min = 80.0;
else

#Calciumchloride dihydrate:
if feed.TCM.ID1 <= 5.1 then
  t_min = 175.0;
else
#Potassiumcarbonate 1.5H2O:
if feed.TCM.ID1 <= 7.1 then
  t_min = 135.0;
else
  t_min = 10.0;
#-----

f_t_max:
#Metaboric acid:
if feed.TCM.ID2 <= 2.1 then
  t_max = 165.0;
else

#Coppersulphate monohydrate:
if feed.TCM.ID2 <= 4.1 then
  t_max = 130.0;
else

#Calciumchloride:
if feed.TCM.ID2 <= 6.1 then
  t_max = 210.0;
else

#Potassiumcarbonate:
if feed.TCM.ID2 <= 8.1 then
  t_max = 150.0;
```

```
else
  t_max = 0.0;

#-----
f_delta_H_reac: delta_H_reac = (drain.h_Material2 * molarmass_Material2 / 1000 +
H_298_15_Material2 + ny_H2O_TCM * (hWater * 18.0153 / 1000 - 285.83) -
      (feed.h_Material1 * molarmass_Material1 / 1000 + H_298_15_Material1)) /
molarmass_Material1 * 1000;

# equilibrium curves
# See Deliverable 2.4 eq. (2) and Table 2 for details. Data from experiments is used.

f_delta_H_reac_0:
#Boric acid:
if feed.TCM.ID1 <= 1.1 then
  delta_H_reac_0 = 125.2;
else

#Coppersulphate pentahydrate:
if feed.TCM.ID1 <= 3.1 then
  delta_H_reac_0 = 124.2;
else

#Calciumchloride dihydrate:
if feed.TCM.ID1 <= 5.1 then
  delta_H_reac_0 = 60.5;
else
#Potassiumcarbonate 1.5H2O:
if feed.TCM.ID1 <= 7.1 then
  delta_H_reac_0 = 158.6;
else
  delta_H_reac_0 = 0.0;

f_delta_S_reac_0:
#Boric acid:
if feed.TCM.ID1 <= 1.1 then
  delta_S_reac_0 = 298.1;
else

#Coppersulphate pentahydrate:
if feed.TCM.ID1 <= 3.1 then
  delta_S_reac_0 = 328.0;
else

#Calciumchloride dihydrate:
if feed.TCM.ID1 <= 5.1 then
  delta_S_reac_0 = 135.5;
else
#Potassiumcarbonate 1.5H2O:
if feed.TCM.ID1 <= 7.1 then
  delta_S_reac_0 = 375.6;
```

```
else
  delta_S_reac_0 = 0.0;

# stoichiometric coefficient for the equilibrium reactions (as they are different from
ny_H2O_TCM):
f_ny_equilib:
#Boric acid:
if feed.TCM.ID1 <= 1.1 then
  ny_equilib = 1.0;
else
#Coppersulphate pentahydrate:
if feed.TCM.ID1 <= 3.1 then
  ny_equilib = 2.0;      # reaction product is trihydrate
else
#Calciumchloride dihydrate:
if feed.TCM.ID1 <= 5.1 then
  ny_equilib = 1.0;      # reaction product is monohydrate
else
#Potassiumcarbonate 1.5 hydrate:
if feed.TCM.ID1 <= 7.1 then
  ny_equilib = 1.5;
else
  ny_equilib = -1.;

# Van't Hoff curve gives the equilibrium temperature-pressure relationship
# The reference pressure p+ is 1 bar and is cancelled out of the equation
# reactor pressure is drain TCM outlet pressure
f_t_equilib: ln(drain.p) = delta_S_reac_0/8.314/ny_equilib -
delta_H_reac_0*1000/8.314/ny_equilib/(t_equilib+273.15);

# test conditions
t_conv_TCM1:      test (conv_TCM >= 0.0) error "Negative conversion rate!";
t_conv_TCM2:      test (conv_TCM <= 1.0) error "Conversion rate higher than 1!";
t_q_reac:          test (q_reac >= 0.0)          error "Heat consumed by reaction is
negative!";

# temperature range for charging
t_t_min: test (t_reac >= t_min) warning "Temperature in reactor is too low!";
t_t_max: test (t_reac <= t_max) warning "Temperature in reactor is too high!";

# pressure and temperature must be on the right side from the equilibrium curve
t_t_equilib:      test (t_reac > t_equilib) warning "Charging conditions not met, decrease p or
increase t in reactor!";

# Using overall mass balance for testing correct balancing
t_mass: test (drain.mass + drain_water.mass - feed.mass < 1e-3) warning "Overall mass
balance incorrect!";

# no change of substances
t_drain_material1: test (feed.TCM.ID1 == drain.TCM.ID1)
                    error "Different material 1 at feed and drain!";
```

```
t_drain_material2: test (feed.TCM.ID2 == drain.TCM.ID2)
                    error "Different material 2 at feed and drain!";
t_drain_oil: test (feed.T_Composition.FluidID == drain.T_Composition.FluidID)
               error "Different oil at feed and drain!";
```

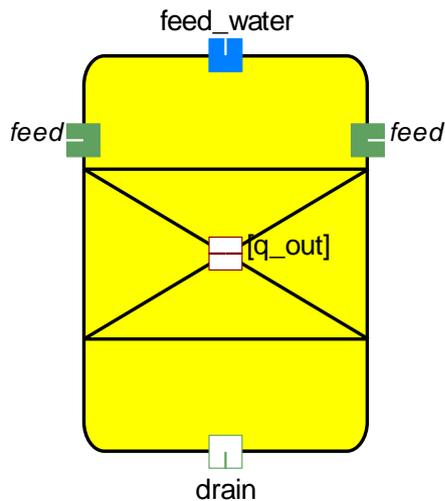
### Variables

|                      |   |
|----------------------|---|
| delta_p              | pressure drop in the reactor  |
| conv_TCM             | conversion fraction of TCM  |
|                      | This was variable X.  |
| t_reac               | temperature of the reaction   |
| q_trans              | transferred heat  |
| q_reac               | heat consumed by the reaction                                       |
| q_preheat            | heat required to bring all incoming streams to reaction temperature |
| delta_H_reac         | heat reaction of dehydration of TCM                                 |
| hWater               | water enthalpy at reaction temperature                              |
| molar mass_Material1 | molar mass of TCM 1   |
| molar mass_Material2 | molar mass of TCM 2   |
| H_298_15_Material1   | enthalpy of formation of TCM1                                       |
| H_298_15_Material2   | enthalpy of formation of TCM2                                       |
| ny_H2O_TCM           | stoichiometric coefficient of the extracted water                   |
| t_min                | minimum allowed temperature at the reactor inlet                    |
|                      | Tstart  |
| t_max                | maximum allowed temperature in the reactor                          |
|                      | Tmax  |
| t_equilib            | reaction equilibrium temperature                                    |
| delta_H_reac_0       | standard enthalpy of reaction                                       |
|                      | Spec from D2.5 chapter 3.2  |
| delta_S_reac_0       | standard entropy of reaction  |
|                      | Spec from D2.5 chapter 3.2  |
| ny_equilib           | stoichiometric coefficient of the equilibrium reactions             |

## 3.12. TCM\_Reactor\_Discharging

### Purpose

Reactor for discharging step of thermochemical material (TCM). Low temperature side receiving heat from the reactor is optional. Different heat receiving working fluids (OR\_ or heat transfer (T\_) fluids) can be connected.



### Connections

TCM\_Stream: feed  
TCM\_Stream: drain  
W\_Stream: feed\_water  
q\_cond\_trans: q\_out

### 3.12.1. TCM\_Reactor\_Discharging

#### Purpose

Design model for isothermal reactor to discharge the TCM.

#### Model equations

# mass balances

```
f_Oil:          feed.mass_Oil = drain.mass_Oil;
f_Material2:    if (lambda > 1) then
                 drain.mass_Material2 = (1 - conv_TCM) * feed.mass_Material2;
                 else
                 drain.mass_Material2 = (1 - conv_TCM * lambda) * feed.mass_Material2;
```

```
f_Material1:
if (lambda > 1) then
    drain.mass_Material1 = feed.mass_Material1 + conv_TCM * feed.mass_Material2 /
molar mass_Material2 * molar mass_Material1;
else
    drain.mass_Material1 = feed.mass_Material1 + lambda * conv_TCM *
feed.mass_Material2 / molar mass_Material2 * molar mass_Material1;
```

# stoichiometric coefficient for the extracted water:

```
f_ny_H2O:
#Boric acid:
if feed.TCM.ID1 <= 1.1 then
    ny_H2O_TCM = 1.0;
else
```

```
#Coppersulphate pentahydrate:
```

```

if feed.TCM.ID1 <= 3.1 then
  ny_H2O_TCM = 4.0;
else
  #Calciumchloride dihydrate:
  if feed.TCM.ID1 <= 5.1 then
    ny_H2O_TCM = 2.0;
  else
    #Potassiumcarbonate 1.5 hydrate:
    if feed.TCM.ID1 <= 7.1 then
      ny_H2O_TCM = 1.5;
    else
      ny_H2O_TCM = -1.0;

f_Water: if (lambda > 1) then
  if (feed_water.mass > ny_H2O_TCM * conv_TCM * feed.mass_Material2 /
molar mass_Material2 * 18.0153) then
    drain.mass_Water = feed_water.mass - ny_H2O_TCM * conv_TCM *
feed.mass_Material2 / molar mass_Material2 * 18.0153;
  else
    drain.mass_Water = 0;
  else
    drain.mass_Water = feed_water.mass - ny_H2O_TCM * conv_TCM * lambda
* feed.mass_Material2 / molar mass_Material2 * 18.0153;

f_lambda:      lambda = (feed_water.mass / 18.0153) / (ny_H2O_TCM *
feed.mass_Material2 / molar mass_Material2);

f_molar mass_Material1: molar mass_Material1 = feed.TCM.TCM_M(feed.TCM.ID1);
f_molar mass_Material2: molar mass_Material2 = feed.TCM.TCM_M(feed.TCM.ID2);

# pressure drops
f_delta_p_TCM: feed.p - delta_p_TCM = drain.p;
f_delta_p_water: feed_water.p - delta_p_water = drain.p;

# pressure in the reactor
f_p2:      p = drain.p;

# energy balance

# calculation of reaction temperature t_reac (mixing temperature)

f_mix:      0.0 = (h_Material2_treac - feed.h_Material2) * feed.mass_Material2 +
              (h_Material1_treac - feed.h_Material1) * feed.mass_Material1
+
              (h_Oil_treac - feed.h_Oil) * feed.mass_Oil +
              (feed_water.fhpt(p, t_reac, 1,0,0,0,0,0,0,0,0,0,0,0,0,0,0) -
feed_water.h) * feed_water.mass -
              q_preheat; # a positive q_preaheat means heat is needed to
preheat the feed

# enthalpies at t_reac

```

```
f_hOil_treac:    h_Oil_treac = feed.T_Composition.htf_h_t(t_reac) -
feed.T_Composition.htf_h_t(25);

f_hWater_treac: h_Water_treac = feed_water.fhpt(p, t_reac, 1,0,0,0,0,0,0,0,0,0,0) -
feed_water.fhpt(1.0, 25.0, 1,0,0,0,0,0,0,0,0,0,0);

f_h_Material1_t_reac: h_Material1_treac = feed.TCM.TCM_h_t(t_reac, feed.TCM.ID1) -
feed.TCM.TCM_h_t(25.0, feed.TCM.ID1);

f_h_Material2_t_reac: h_Material2_treac = feed.TCM.TCM_h_t(t_reac, feed.TCM.ID2) -
feed.TCM.TCM_h_t(25.0, feed.TCM.ID2);

f_H_298_15_Material1:
#Boric acid:
if feed.TCM.ID1 <= 1.1 then
    H_298_15_Material1 = - 1093.99; # kJ/mol
else

#Coppersulphate pentahydrate:
if feed.TCM.ID1 <= 3.1 then
    H_298_15_Material1 = -2276.512;
else

#Calciumchloride dihydrate:
if feed.TCM.ID1 <= 5.1 then
    H_298_15_Material1 = - 1403.9;
else

#Potassiumcarbonate 1.5H2O:
if feed.TCM.ID1 <= 7.1 then
    H_298_15_Material1 = -1612.930;
else
    H_298_15_Material1 = 0.0;
#-----

f_H_298_15_Material2:
#Metaboric acid:
if feed.TCM.ID2 <= 2.1 then
    H_298_15_Material2 = - 802.78;
else

#Coppersulphate monohydrate:
if feed.TCM.ID2 <= 4.1 then
    H_298_15_Material2 = -1082.818;
else

#Calciumchloride:
if feed.TCM.ID2 <= 6.1 then
    H_298_15_Material2 = - 795.8;
else
#Potassiumcarbonate:
if feed.TCM.ID2 <= 8.1 then
```

```
H_298_15_Material2 = -1151.499;
else
  H_298_15_Material2 = 0.0;

# minimum and maximum temperatures of the reactions that can be used as range
temperatures for the charging reactor
###
# These are preliminary values until more realistic data is made available!
###

f_t_min:
#Boric acid:
#if feed.TCM.ID1 <= 1.1 then
  #t_min = 0.0;
#else

#Coppersulphate pentahydrate:
#if feed.TCM.ID1 <= 3.1 then
  #t_min = 0.0;
#else

#Calciumchloride dihydrate:
#if feed.TCM.ID1 <= 5.1 then
  #t_min = 0.0;
#else
#Potassiumcarbonate 1.5H2O:
#if feed.TCM.ID1 <= 7.1 then
  #t_min = 0.0;
#else
  t_min = 10.0;
#-----

f_t_max:
#Metaboric acid:
if feed.TCM.ID2 <= 2.1 then
  t_max = 165.0;
else

#Coppersulphate monohydrate:
if feed.TCM.ID2 <= 4.1 then
  t_max = 130.0;
else

#Calciumchloride:
if feed.TCM.ID2 <= 6.1 then
  t_max = 210.0;
else

#Potassiumcarbonate:
if feed.TCM.ID2 <= 8.1 then
  t_max = 150.0;
```

```

else
  t_max = 0.0;

#-----

# reaction of rehydration

f_q_reac:      q_reac = -((drain.mass_Material1 - feed.mass_Material1) * (h_Material1_treac
+ H_298_15_Material1 / molarmass_Material1 * 1000) -
                (feed_water.mass - drain.mass_Water) * (h_Water_treac -
285.83 / 18.0153 * 1000) -
                (feed.mass_Material2 - drain.mass_Material2) *
(h_Material2_treac + H_298_15_Material2 / molarmass_Material2 * 1000));
#q_reac is positive if energy is released (exothermal reaction)

f_delta_H_reac: delta_H_reac =      (h_Material2_treac * molarmass_Material2 / 1000 +
H_298_15_Material2 + h_Water_treac * 18.0153 / 1000 - 285.83 -
                (h_Material1_treac * molarmass_Material1 / 1000 +
H_298_15_Material1)) / molarmass_Material1 * 1000;

# isothermal discharging
# outlet temperature of the TCM from the reactor:
f_drain_t:      drain.t = t_reac;

# for discharging, q_reac minus heat required to reach reactor temp minus any losses is
transferred to the heat receiving side
f_q_trans:      q_trans = q_reac - q_loss - q_preheat;

ifl ref(q_out) then # the cold side is connected

  # conductive heat transfer through the wall
  f_q_q_trans:  q_trans = q_out.q_trans;

  # transfer of prevailing temperatures at the heat transfer surface
  # the reactor is isothermal conditions
  f_t_hot_in:   q_out.t_hot_in = t_reac;
  f_t_hot_out:  q_out.t_hot_out = t_reac;
endifl

# equilibrium curves
# See Deliverable 2.4 eq. (2) and Table 2 for details. Data from experiments is used.

f_delta_H_reac_0:
#Boric acid:
if feed.TCM.ID1 <= 1.1 then
  delta_H_reac_0 = 125.2;
else

#Coppersulphate pentahydrate:
if feed.TCM.ID1 <= 3.1 then

```

```
    delta_H_reac_0 = 124.2;
else

#Calciumchloride dihydrate:
if feed.TCM.ID1 <= 5.1 then
    delta_H_reac_0 = 60.5;
else
#Potassiumcarbonate 1.5H2O:
if feed.TCM.ID1 <= 7.1 then
    delta_H_reac_0 = 158.6;
else
    delta_H_reac_0 = 0.0;

f_delta_S_reac_0:
#Boric acid:
if feed.TCM.ID1 <= 1.1 then
    delta_S_reac_0 = 298.1;
else

#Coppersulphate pentahydrate:
if feed.TCM.ID1 <= 3.1 then
    delta_S_reac_0 = 328.0;
else

#Calciumchloride dihydrate:
if feed.TCM.ID1 <= 5.1 then
    delta_S_reac_0 = 135.5;
else
#Potassiumcarbonate 1.5H2O:
if feed.TCM.ID1 <= 7.1 then
    delta_S_reac_0 = 375.6;
else
    delta_S_reac_0 = 0.0;

# stoichiometric coefficient for the equilibrium reactions (as they are different from
ny_H2O_TCM):
f_ny_equilib:
#Boric acid:
if feed.TCM.ID1 <= 1.1 then
    ny_equilib = 1.0;
else
#Coppersulphate pentahydrate:
if feed.TCM.ID1 <= 3.1 then
    ny_equilib = 2.0;      # reaction product is trihydrate
else
#Calciumchloride dihydrate:
if feed.TCM.ID1 <= 5.1 then
    ny_equilib = 1.0;      # reaction product is monohydrate
else
#Potassiumcarbonate 1.5 hydrate:
if feed.TCM.ID1 <= 7.1 then
```

```

ny_equilib = 1.5;
else
ny_equilib = -1.;

# Van't Hoff curve gives the equilibrium temperature-pressure relationship
# The reference pressure p+ is 1 bar and is cancelled out of the equation
# reactor pressure is drain TCM outlet pressure
f_t_equilib: ln(drain.p) = delta_S_reac_0/8.314/ny_equilib -
delta_H_reac_0*1000/8.314/ny_equilib/(t_equilib+273.15);

# test conditions

t_X1:          test (conv_TCM >= 0.0) error "Negative conversion rate!";
t_X2:          test (conv_TCM <= 1.0) error "Conversion rate higher than 1!";
t_q_reac: test (q_reac >= 0.0)   error "Heat of reaction is negative!";
t_q_trans:     test (q_trans >= 0.0)   warning "No heat can be extracted, preheat feed or
lower reactor temperature!";

# temperature range for discharging
t_t_min: test (t_reac >= t_min) warning "Temperature in reactor is too low!";
t_t_max: test (t_reac <= t_max) warning "Temperature in reactor is too high!";

# pressure and temperature must be on the left side from the equilibrium curve
t_t_equilib: test (t_reac < t_equilib) warning "Discharging conditions not met, increase p or
decrease t in reactor!";

# substances must be the same
t_feed_material1: test (feed.TCM.ID1 == drain.TCM.ID1)
                    error "Different material 1 at feed and drain!";
t_feed_material2: test (feed.TCM.ID2 == drain.TCM.ID2)
                    error "Different material 2 at feed and drain!";
t_feed_oil: test (feed.T_Composition.FluidID == drain.T_Composition.FluidID)
              error "Different oil at feed and drain!";

```

## Variables

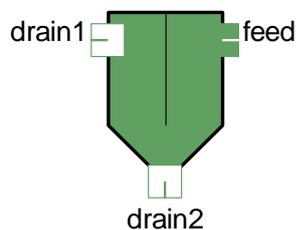
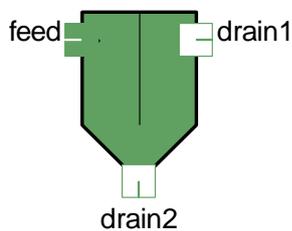
|               |   |
|---------------|---|
| p             | pressure  |
| delta_p_TCM   | pressure drop of TCM into the reactor                               |
| delta_p_water | pressure drop of water into the reactor                             |
| conv_TCM      | conversion fraction of TCM  |
|               | Was variable X  |
| t_reac        | temperature of rehydration reaction                                 |
| q_trans       | transferred heat  |
| q_reac        | heat produced by the reaction                                       |
| q_loss        | heat loss to environment  |
|               | lost heat (<0)  |
| q_preheat     | heat required to bring all incoming streams to reaction temperature |
| delta_H_reac  | heat reaction of rehydration of TCM                                 |
| lambda        | hydration factor  |
| h_Oil_treac   | thermal oil enthalpy after mixing with water (at t_reac)            |

|                     |   |
|---------------------|---|
| h_Water_treac       | water enthalpy at treac                                     |
| h_Material1_treac   | enthalpy at reaction temperature of material1               |
| h_Material2_treac   | enthalpy at reaction temperature of material2               |
| molarmass_Material1 | molar mass of TCM 1   |
| molarmass_Material2 | molar mass of TCM 2   |
| H_298_15_Material1  | enthalpy of formation of TCM1                               |
| H_298_15_Material2  | enthalpy of formation of TCM2                               |
| ny_H2O_TCM          | stoichiometric coefficient of the extracted water           |
| t_min               | minimum allowed temperature at the reactor inlet<br>Tstart  |
| t_max               | maximum allowed temperature in the reactor<br>Tmax          |
| t_equilib           | reaction equilibrium temperature                            |
| delta_H_reac_0      | standard enthalpy of reaction<br>Spec from D2.5 chapter 3.2 |
| delta_S_reac_0      | standard entropy of reaction<br>Spec from D2.5 chapter 3.2  |
| ny_equilib          | stoichiometric coefficient of the equilibrium reactions     |

### 3.13. TCM\_Separator

#### Purpose

Separator for an incoming thermochemical material stream.



#### Connections

TCM\_Stream: feed  
TCM\_Stream: drain1  
TCM\_Stream: drain2

#### 3.13.1. TCM\_Separator

#### Purpose

allows solids-liquid separation of an incoming TCM stream

### Model equations

```
# mass balances

# overall
f_mass:          feed.mass = drain1.mass + drain2.mass;

# one of the individual mass balances is redundant and hence omitted
#f_mass_Oil:     feed.mass_Oil = drain1.mass_Oil + drain2.mass_Oil;
f_mass_Material1: feed.mass_Material1 = drain1.mass_Material1 +
drain2.mass_Material1;
f_mass_Material2: feed.mass_Material2 = drain2.mass_Material2 +
drain1.mass_Material2;
f_mass_Water:    feed.mass_Water = drain1.mass_Water + drain2.mass_Water;

# separation, individual split ratios for the different substances
f_phi_Oil:       drain1.mass_Oil = phi_Oil * feed.mass_Oil;
f_phi_Material1: drain1.mass_Material1 = phi_Material1 * feed.mass_Material1;
f_phi_Material2: drain1.mass_Material2 = phi_Material2 * feed.mass_Material2;
f_phi_Water:     drain1.mass_Water = phi_Water * feed.mass_Water;

# process conditions do not change, no pressure drops, no thermal losses

# pressures constant
f_p1:           feed.p = drain1.p;
f_p2:           feed.p = drain2.p;

# temperatures constant
f_t1:           feed.t = drain1.t;
f_t2:           feed.t = drain2.t;

# no change of substances
t_drain1_material1: test (feed.TCM.ID1 == drain1.TCM.ID1)
                    error "different material 1 at feed and drain1!";
t_drain1_material2: test (feed.TCM.ID2 == drain1.TCM.ID2)
                    error "different material 2 at feed and drain1!";
t_drain1_oil: test (feed.T_Composition.FluidID == drain1.T_Composition.FluidID)
                error "different oil at feed and drain1!";
t_drain2_material1: test (feed.TCM.ID1 == drain2.TCM.ID1)
                    error "different material 1 at feed and drain2!";
t_drain2_material2: test (feed.TCM.ID2 == drain2.TCM.ID2)
                    error "different material 2 at feed and drain2!";
t_drain2_oil: test (feed.T_Composition.FluidID == drain2.T_Composition.FluidID)
                error "different oil at feed and drain2!";
```

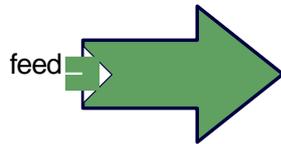
### Variables

|               |  |
|---------------|--|
| phi_Oil       | splitting ratio of extracted oil going to drain1       |
| phi_Material1 | splitting ratio of extracted Material1 going to drain1 |
| phi_Material2 | splitting ratio of extracted Material2 going to drain1 |
| phi_Water     | splitting ratio of water going to drain1               |

## 3.14. TCM\_Sink

### Purpose

Sink for a thermochemical material stream.



### Connections

TCM\_Stream: feed

### 3.14.1. TCM\_Sink

#### Purpose

sink modelling the release of a stream

#### Model equations

f\_mass: mass = feed.mass;

f\_p: p = feed.p;

f\_t: t = feed.t;

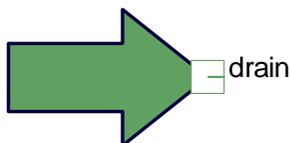
#### Variables

|      |                    |
|------|--------------------|
| mass | mass flow          |
| p    | outlet pressure    |
| t    | outlet temperature |

## 3.15. TCM\_Source

### Purpose

Source for a thermochemical material stream.



### Connections

TCM\_Stream: drain

### 3.15.1. TCM\_Source

#### Purpose

source modeling the inlet of a stream

#### Model equations

f\_mass: mass = drain.mass;

f\_p: p = drain.p;

f\_t: t = drain.t;

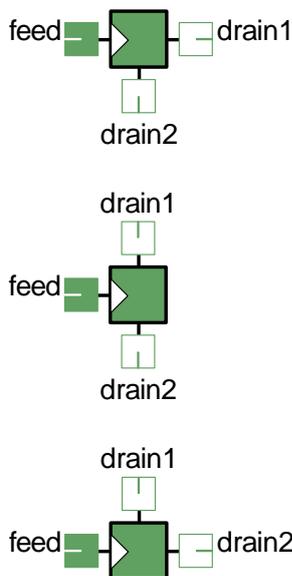
### Variables

|      |             |
|------|-------------|
| mass | mass flow   |
| p    | pressure    |
| t    | temperature |

## 3.16. TCM\_Splitter

### Purpose

Splitter for an incoming thermochemical material stream.



### Connections

TCM\_Stream: feed  
TCM\_Stream: drain1  
TCM\_Stream: drain2

### 3.16.1. TCM\_Splitter

### Purpose

splits an incoming TCM stream into two outgoing TCM streams with equal thermodynamic properties

### Model equations

# mass balances

# overall

f<sub>mass</sub>: feed.mass = drain1.mass + drain2.mass;

# one of the individual mass balances is redundant and hence omitted

#f<sub>mass\_Oil</sub>: feed.mass\_Oil = drain1.mass\_Oil + drain2.mass\_Oil;

f<sub>mass\_Material1</sub>: feed.mass\_Material1 = drain1.mass\_Material1 +  
drain2.mass\_Material1;

```
f_mass_Material2:      feed.mass_Material2 = drain2.mass_Material2 +
drain1.mass_Material2;
f_Mass_Water:         feed.mass_Water = drain1.mass_Water + drain2.mass_Water;

# composition (material fractions) and process conditions do not change

# one of the fractions is redundant and hence omitted
#f_frac_Oil:         feed.w_Material1 = drain1.w_Material1;
f_frac_Material1:    feed.w_Material1 = drain1.w_Material1;
f_frac_Material2:    feed.w_Material2 = drain1.w_Material2;
f_frac_Water:        feed.w_Water = drain1.w_Water;

# pressures identical
f_p1:                feed.p = drain1.p;
f_p2:                feed.p = drain2.p;

# temperatures identical
f_t1:                feed.t = drain1.t;
f_t2:                feed.t = drain2.t;

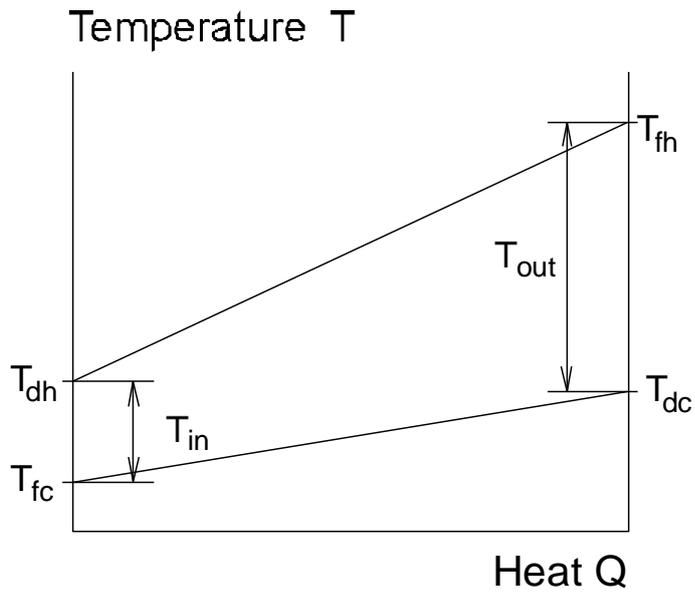
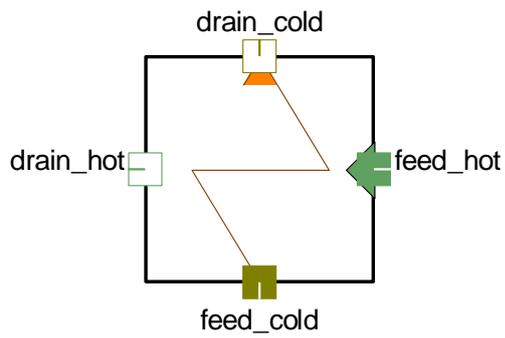
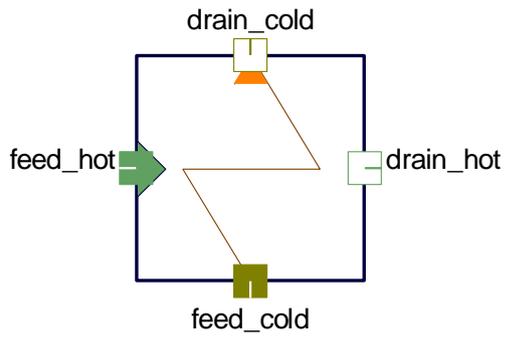
# tests

# no change of substances
t_drain1_material1: test (feed.TCM.ID1 == drain1.TCM.ID1)
                    error "different material 1 at feed and drain1!";
t_drain1_material2: test (feed.TCM.ID2 == drain1.TCM.ID2)
                    error "different material 2 at feed and drain1!";
t_drain1_oil: test (feed.T_Composition.FluidID == drain1.T_Composition.FluidID)
                    error "different oil at feed and drain1!";
t_drain2_material1: test (feed.TCM.ID1 == drain2.TCM.ID1)
                    error "different material 1 at feed and drain2!";
t_drain2_material2: test (feed.TCM.ID2 == drain2.TCM.ID2)
                    error "different material 2 at feed and drain2!";
t_drain2_oil: test (feed.T_Composition.FluidID == drain2.T_Composition.FluidID)
                    error "different oil at feed and drain2!";
```

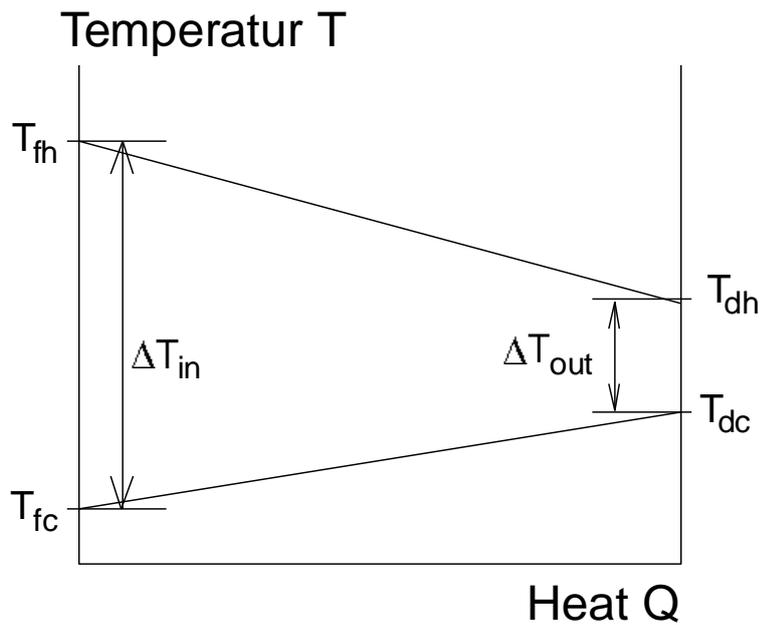
### 3.17. TCM\_T\_Htex

#### Purpose

Heat exchanger for transfer from TCM fluid on the hot side to thermofluid on the cold side. Cocurrent or counter current flow.



QT-diagram, counter current



QT-diagram, cocurrent

#### Connections

TCM\_Stream: drain\_hot  
TCM\_Stream: feed\_hot  
T\_Stream: feed\_cold  
T\_Stream: drain\_cold

### 3.17.1. TCM\_T\_Htex

#### Purpose

design model

#### Model equations

# mass balances

# hot side total and individual mass balances (one is redundant)

f\_mass\_hot: feed\_hot.mass = drain\_hot.mass;

#f\_mass\_Oil\_hot: feed\_hot.mass\_Oil = drain\_hot.mass\_Oil;

f\_mass\_Material1\_hot: feed\_hot.mass\_Material1 = drain\_hot.mass\_Material1;

f\_mass\_Material2\_hot: feed\_hot.mass\_Material2 = drain\_hot.mass\_Material2;

f\_mass\_Water\_hot: feed\_hot.mass\_Water = drain\_hot.mass\_Water;

# cold side mass balance

f\_mass\_cold: feed\_cold.mass = drain\_cold.mass;

# pressure drops

f\_delta\_p\_hot: feed\_hot.p - delta\_p\_hot = drain\_hot.p;

f\_delta\_p\_cold: feed\_cold.p - delta\_p\_cold = drain\_cold.p;

# energy balance

```
f_e_hot: feed_hot.mass * (feed_hot.h-drain_hot.h) * (1.0 - heat_loss/100) = q_trans;
f_e_cold: feed_cold.mass * (drain_cold.h - feed_cold.h) = q_trans;

# temperature differences
# They are differently defined for co- and counter current heat exchangers.

ifl Type == cocurrent then
  f_dt_in_co:    feed_hot.t - dt_in = feed_cold.t;
  f_dt_out_co:   drain_hot.t - dt_out = drain_cold.t;
endifl

ifl Type == counter_current then
  f_dt_in_counter:    drain_hot.t - dt_in = feed_cold.t;
  f_dt_out_counter:   feed_hot.t - dt_out = drain_cold.t;
endifl

# Mean temperature difference is the logarithmic temperature difference.
# For nearly equal inlet and outlet temperature differences, it is approximated
# by the arithmetic mean (which is the asymptotic approximation).
f_MTD: if abs(dt_in/dt_out) >=1.2 || abs(dt_out/dt_in) >=1.2 then
  MTD * ln(dt_in/dt_out) = (dt_in-dt_out);
else
  MTD = (dt_in+dt_out)/2.0;

f_htc_area:    q_trans = htc_area * MTD;

# tests
t_dt_in:  test(dt_in > 0.0)error "dt_in <= 0.0";
t_dt_out: test(dt_out > 0.0)  error "dt_out <= 0.0";
t_q_trans: test(q_trans > 0.0)  error "q_trans <= 0.0";

# no change of substances
t_drain_hot_material1: test (feed_hot.TCM.ID1 == drain_hot.TCM.ID1)
  error "Different material 1 at feed_hot and drain_hot!";
t_drain_hot_material2: test (feed_hot.TCM.ID2 == drain_hot.TCM.ID2)
  error "Different material 2 at feed_hot and drain_hot!";
t_drain_hot_oil: test (feed_hot.T_Composition.FluidID == drain_hot.T_Composition.FluidID)
  error "Different oil at feed_hot and drain_hot!";

# feed and drain thermooil must use the same fluid
t_conn_cold: test (feed_cold.T_Composition.FluidID == drain_cold.T_Composition.FluidID)
  error "different thermofluid at feed and drain cold temperature side!";
```

### Parameters

|              |   |
|--------------|---|
| delta_p_hot  | pressure drop of the high temperature side  |
| delta_p_cold | pressure drop of the low temperature side   |
| heat_loss    | percentage of heat lost to the surroundings |

### Variables

|        |  |
|--------|--|
| dt_in  | temperature difference at the inlet of the low temperature side  |
| dt_out | temperature difference at the outlet of the low temperature side |

|          |   |
|----------|---|
| htc_area | heat transfer coefficient x transfer area |
| MTD      | mean temperature difference               |
| q_trans  | transferred heat                          |

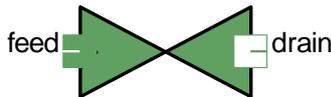
#### Switches

Type (cocurrent, counter\_current) defines the type of heat exchanger

## 3.18. TCM\_Valve

### Purpose

Valve for thermochemical material stream.



### Connections

TCM\_Stream: feed  
TCM\_Stream: drain

### 3.18.1. TCM\_Valve

#### Purpose

adiabatic expansion in a valve

#### Model equations

# mass balances

# overall

f\_mass: feed.mass = drain.mass;

f\_mass\_Material1: feed.mass\_Material1 = drain.mass\_Material1;

f\_mass\_Material2: feed.mass\_Material2 = drain.mass\_Material2;

f\_mass\_Water: feed.mass\_Water = drain.mass\_Water;

# f\_mass\_Oil: drain.mass\_Oil = drain.mass\_Oil;

# pressure drop

f\_delta\_p\_rel: (1.0-pressure\_drop) \* feed.p = drain.p;

f\_delta\_p\_abs: feed.p - delta\_p = drain.p;

# adiabatic expansion, enthalpy is constant

f\_h: feed.h = drain.h;

# test conditions

t\_delta\_p: test (delta\_p >= 0.0) warning "delta\_p < 0.0";

# no change of substances

t\_conn\_material1: test (feed.TCM.ID1 == drain.TCM.ID1)  
error "different material 1 at feed and drain!";

t\_conn\_material2: test (feed.TCM.ID2 == drain.TCM.ID2)  
error "different material 2 at feed and drain!";

t\_conn\_oil: test (feed.T\_Composition.FluidID == drain.T\_Composition.FluidID)

error "different oil at feed and drain!";

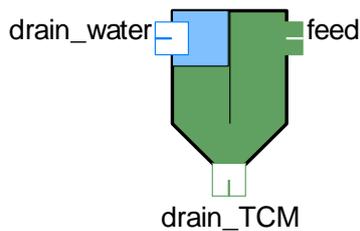
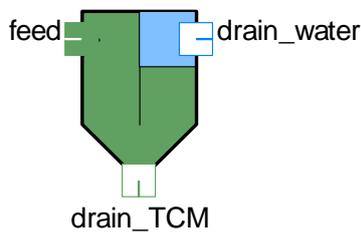
### Variables

|               |                        |
|---------------|------------------------|
| pressure_drop | relative pressure drop |
| delta_p       | absolute pressure drop |

## 3.19. TCM\_W\_Separator

### Purpose

Separator for pure water from an incoming thermochemical material stream.



### Connections

TCM\_Stream: feed  
TCM\_Stream: drain\_TCM  
W\_Stream: drain\_water

### 3.19.1. TCM\_W\_Separator

#### Purpose

allows separation of water from an incoming TCM stream

#### Model equations

# mass balances

# overall

f\_mass:  $feed.mass = drain\_water.mass + drain\_TCM.mass;$

# one of the individual mass balances is redundant and hence omitted

#f\_mass\_Oil:  $feed.mass\_Oil = drain\_TCM.mass\_Oil;$

f\_mass\_Material1:  $feed.mass\_Material1 = drain\_TCM.mass\_Material1;$

f\_mass\_Material2:  $feed.mass\_Material2 = drain\_TCM.mass\_Material2;$

```
f_mass_Water:          feed.mass_Water = drain_water.mass;

# water is 100% separated
f_sep_Water:          drain_TCM.mass_Water = 0.0;

# process conditions do not change, no pressure drops, no thermal losses

# pressures constant
f_p1:                feed.p = drain_water.p;
f_p2:                feed.p = drain_TCM.p;

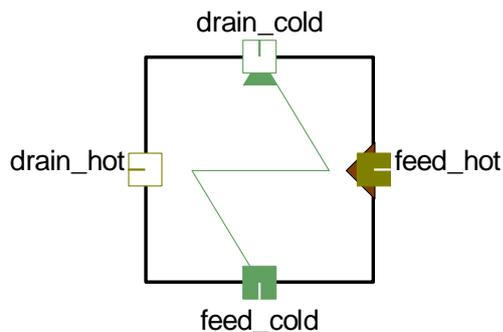
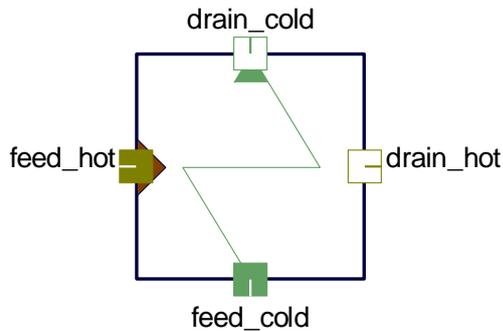
# temperatures constant
f_t1:                feed.t = drain_water.t;
f_t2:                feed.t = drain_TCM.t;

# no change of substances
t_drain_TCM_material1: test (feed.TCM.ID1 == drain_TCM.TCM.ID1)
                        error "different material 1 at feed and drain_TCM!";
t_drain_TCM_material2: test (feed.TCM.ID2 == drain_TCM.TCM.ID2)
                        error "different material 2 at feed and drain_TCM!";
t_drain_TCM_oil: test (feed.T_Composition.FluidID == drain_TCM.T_Composition.FluidID)
                    error "different oil at feed and drain_TCM!";
```

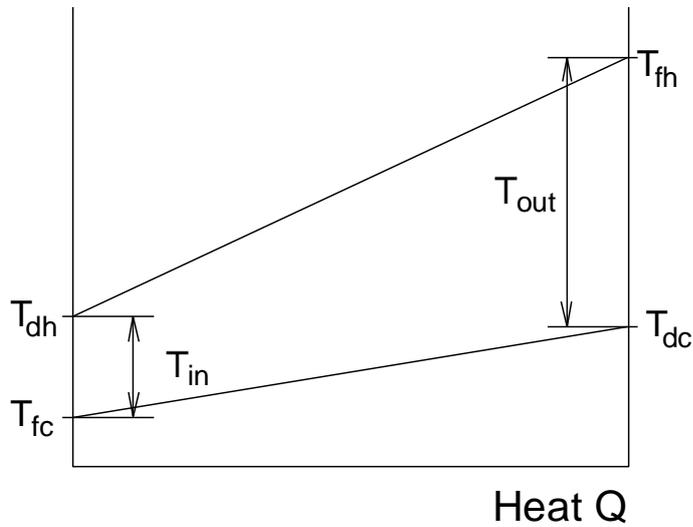
### 3.20. T\_TCM\_Htex

#### Purpose

Heat exchanger for transfer from thermofluid on the hot side to TCM fluid on the cold side. Cocurrent or counter current flow.

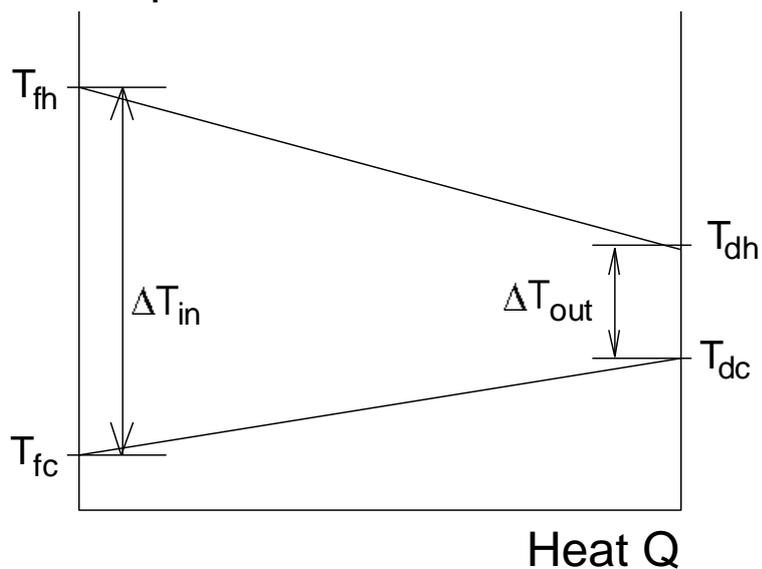


## Temperature T



QT-diagram, counter current

## Temperatur T



QT-diagram, cocurrent

### Connections

TCM\_Stream: drain\_cold  
TCM\_Stream: feed\_cold  
T\_Stream: feed\_hot  
T\_Stream: drain\_hot

### 3.20.1. T\_TCM\_Htex

#### Purpose

design model

#### Model equations

# mass balances

```
# hot side mass balance
f_mass_hot:          feed_hot.mass = drain_hot.mass;

# cold side total and individual mass balances (one is redundant)
f_mass_cold:          feed_cold.mass = drain_cold.mass;
#f_mass_Oil_cold:     feed_cold.mass_Oil = drain_cold.mass_Oil;
f_mass_Material1_cold: feed_cold.mass_Material1 = drain_cold.mass_Material1;
f_mass_Material2_cold: feed_cold.mass_Material2 = drain_cold.mass_Material2;
f_mass_Water_cold:    feed_cold.mass_Water = drain_cold.mass_Water;

# pressure drops

f_delta_p_hot:       feed_hot.p - delta_p_hot = drain_hot.p;
f_delta_p_cold:       feed_cold.p - delta_p_cold = drain_cold.p;

# energy balance

f_e_hot: feed_hot.mass * (feed_hot.h-drain_hot.h) * (1.0 - heat_loss/100) = q_trans;
f_e_cold: feed_cold.mass * (drain_cold.h - feed_cold.h) = q_trans;

# temperature differences
# They are differently defined for co and counter current heat exchangers.

ifl Type == cocurrent then
  f_dt_in_co:    feed_hot.t - dt_in = feed_cold.t;
  f_dt_out_co:   drain_hot.t - dt_out = drain_cold.t;
endifl

ifl Type == counter_current then
  f_dt_in_counter:    drain_hot.t - dt_in = feed_cold.t;
  f_dt_out_counter:   feed_hot.t - dt_out = drain_cold.t;
endifl

# Mean temperature difference is the logarithmic temperature difference.
# For nearly equal inlet and outlet temperature differences, it is approximated
# by the arithmetic mean (which is the asymptotic approximation).
f_MTD: if abs(dt_in/dt_out) >=1.2 || abs(dt_out/dt_in) >=1.2 then
  MTD * ln(dt_in/dt_out) = (dt_in-dt_out);
else
  MTD = (dt_in+dt_out)/2.0;

f_htc_area:    q_trans = htc_area * MTD;

# tests
t_dt_in:  test(dt_in > 0.0) error "dt_in <= 0.0";
t_dt_out: test(dt_out > 0.0) error "dt_out <= 0.0";
t_q_trans: test(q_trans > 0.0) error "q_trans <= 0.0";

# no change of substances
# feed and drain thermooil must use the same fluid
t_conn_hot: test (feed_hot.T_Composition.FluidID == drain_hot.T_Composition.FluidID)
```

error "different thermofluid at feed and drain hot temperature side!";

```
t_drain_cold_material1: test (feed_cold.TCM.ID1 == drain_cold.TCM.ID1)
    error "Different material 1 at feed_cold and drain_cold!";
t_drain_cold_material2: test (feed_cold.TCM.ID2 == drain_cold.TCM.ID2)
    error "Different material 2 at feed_cold and drain_cold!";
t_drain_cold_oil: test (feed_cold.T_Composition.FluidID == drain_cold.T_Composition.FluidID)
    error "Different oil at feed_cold and drain_cold!";
```

### Parameters

|              |   |
|--------------|---|
| delta_p_hot  | pressure drop of the high temperature side  |
| delta_p_cold | pressure drop of the low temperature side   |
| heat_loss    | percentage of heat lost to the surroundings |

### Variables

|          |  |
|----------|--|
| dt_in    | temperature difference at the inlet of the low temperature side  |
| dt_out   | temperature difference at the outlet of the low temperature side |
| htc_area | heat transfer coefficient x transfer area                        |
| MTD      | mean temperature difference                                      |
| q_trans  | transferred heat   |

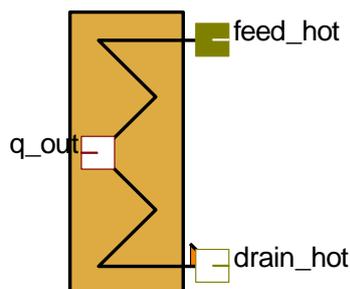
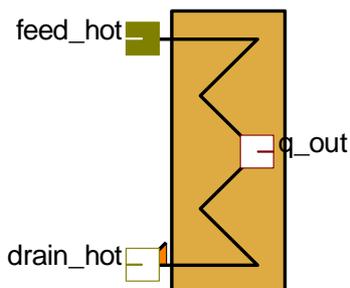
### Switches

|                                   |                                    |
|-----------------------------------|------------------------------------|
| Type (cocurrent, counter_current) | defines the type of heat exchanger |
|-----------------------------------|------------------------------------|

## 3.21. T\_wall\_htex

### Purpose

Heat exchanger with wall transferring heat from thermooil (T) to another side.



## Connections

q\_cond\_trans: q\_out  
T\_Stream: feed\_hot  
T\_Stream: drain\_hot

### 3.21.1. T\_wall\_htex

#### Purpose

design model

#### Model equations

```
# heat transfer from fluid inside the tubes through the wall

# mass transfer
f_mass: feed_hot.mass = drain_hot.mass;

# pressure drop through the wall tubes
f_delta_p:      feed_hot.p - delta_p = drain_hot.p;

# hot fluid is cooled down
f_energy: feed_hot.mass*feed_hot.h - q_trans = drain_hot.mass*drain_hot.h;

# temperatue change
f_delta_t: feed_hot.t - delta_t = drain_hot.t;

# conductive heat transfer through the wall
f_q_q_trans:    q_trans = q_out.q_trans;

# transfer of prevailing temperatures
ifl Type == cocurrent then
  f_t_hot_in_co: q_out.t_hot_in = feed_hot.t;
  f_t_hot_out_co: q_out.t_hot_out = drain_hot.t;
endifl

ifl Type == counter_current then
  f_t_hot_in_counter:    q_out.t_hot_in = drain_hot.t;
  f_t_hot_out_counter:  q_out.t_hot_out = feed_hot.t;
endifl

# test conditions
t_q_trans:      test (q_trans >= 0.0) warning "q_trans negative";
t_delta_p:     test (delta_p >= 0.0) warning "delta_p negative";

# feed and drain thermooil must use the same fluid
t_conn_hot: test (feed_hot.T_Composition.FluidID == drain_hot.T_Composition.FluidID)
              error "different thermofluid at feed and drain!";
```

#### Parameters

delta\_p absolute pressure drop

#### Variables

delta\_t temperature difference between feed and drain mass flow

q\_trans                      transferred heat

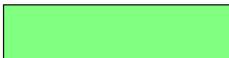
#### Switches

Type (cocurrent, counter\_current)                      defines the type of heat exchanger

## 3.22. electricity\_meter

### Purpose

electrical power consumption or production meter



### Global objects

time\_period: time\_period operating time period to be counted

### 3.22.1. electricity\_meter

#### Purpose

steady-state model giving instantaneous results

#### Model equations

# The model acts as a configurable electricity (electrical power) meter for an arbitrary flowsheet.

# The electrical power is assigned in a free equation in the flowsheet.

# For steady-state calculations, the variable is just made available.

#### Variables

power                      active power

### 3.22.2. electricity\_meter\_dyn

#### Purpose

dynamic model doing integration over time

#### Model equations

# The model acts as a configurable electricity (electrical power) meter for an arbitrary flowsheet.

# The electrical power is assigned in a free equation in the flowsheet and summarized here.

# integration over time

f\_energy\_sum: energy' = power;

#### Variables

power                      active power

energy                      electrical energy

### 3.22.3. electricity\_meter\_period

#### Purpose

steady-state model giving instantaneous results and displaying data for a certain operating period

#### Model equations

# The model acts as a configurable electricity (electrical power) meter for an arbitrary flowsheet.  
# The electrical power is assigned in a free equation in the flowsheet.

# For steady-state calculations, the power times a certain operating time gives  
# the energy consumption or production for that period.

ifl ref(time\_period) then

    f\_energy:            energy = power\*time\_period.operating\_time;

endifl

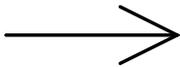
#### Variables

power                      active power  
energy                      electrical energy

## 3.23. flow\_meter

### Purpose

flow meter



### Global objects

time\_period: time\_period operating time period to be counted

### 3.23.1. flow\_meter

#### Purpose

steady-state model giving instantaneous results

#### Model equations

# The model acts as a configurable mass flow meter for an arbitrary flowsheet.

# The mass flow is assigned in free equations in the flowsheet.

# For steady-state calculations, the variable is just made available.

#### Variables

mass\_flow                      mass flow

### 3.23.2. flow\_meter\_dyn

#### Purpose

dynamic model doing integration over time

#### Model equations

# The model acts as a configurable mass flow meter for an arbitrary flowsheet.

# The mass flow is assigned in a free equation in the flowsheet and summarized here.

# integration over time

f\_Mass\_sum:    Mass' = mass\_flow;

#### Variables

mass\_flow                      mass flow

Mass                              mass

### 3.23.3. flow\_meter\_period

#### Purpose

steady-state model giving instantaneous results and displaying data for a certain operating period

#### Model equations

- # The model acts as a configurable mass flow meter for an arbitrary flowsheet.
- # The mass flow is assigned in free equations in the flowsheet.
  
- # For steady-state calculations, the variable is just made available.
  
- # For steady-state calculations, the mass flow times a certain operating time gives
- # the total Mass flowing through the flow meter for that period.

ifl ref(time\_period) then

```
f_mass:Mass = mass_flow*time_period.operating_time;
```

endifl

#### Variables

|           |           |
|-----------|-----------|
| mass_flow | mass flow |
| Mass      | mass      |

### 3.24. heat\_meter

#### Purpose

heat consumption or production meter



#### Global objects

time\_period: time\_period operating time period to be counted

### 3.24.1. heat\_meter

#### Purpose

steady-state model giving instantaneous results

#### Model equations

- # The model acts as a configurable heat (thermal energy) meter for an arbitrary flowsheet.
- # The thermal power is assigned in a free equation in the flowsheet.

# For steady-state calculations, the variable is just made available.

#### Variables

q\_th                      thermal energy

### 3.24.2. heat\_meter\_dyn

#### Purpose

dynamic model doing integration over time

#### Model equations

# The model acts as a configurable heat (thermal energy) meter for an arbitrary flowsheet.

# The thermal power is assigned in a free equation in the flowsheet and summarized here.

# integration over time

f\_Q\_th\_sum:      Q\_th\_sum' = q\_th;

#### Variables

q\_th                      thermal energy from producers

Q\_th\_sum                sum of thermal energy production or consumption

### 3.24.3. heat\_meter\_period

#### Purpose

steady-state model giving instantaneous results and displaying data for a certain operating period

#### Model equations

# The model acts as a configurable heat (thermal energy) meter for an arbitrary flowsheet.

# The thermal power is assigned in a free equation in the flowsheet.

# For steady-state calculations, the variable is just made available.

# For steady-state calculations, the thermal power times a certain operating time gives

# the thermal energy consumption or production for that period.

ifl ref(time\_period) then

f\_Q\_th\_sum:      Q\_th\_sum = q\_th\*time\_period.operating\_time;

endifl

#### Variables

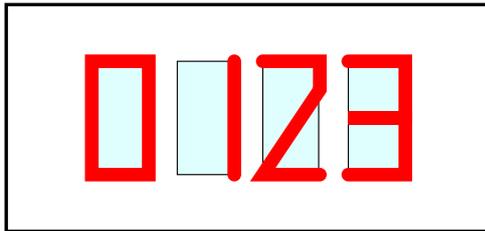
q\_th                      thermal energy

Q\_th\_sum                sum of thermal energy production or consumption

### 3.25. time\_counter

#### Purpose

Time counter used for measuring and displaying time intervals.



### 3.25.1. time\_counter\_dyn

#### Purpose

time counter model for dynamic calculations

#### Model equations

```
# The model displays the concept of time in the calculations.

# Timers can be categorized into two main types.
# The word "timer" is usually reserved for devices that counts down from a specified time
interval,
# while devices that do the opposite, measuring elapsed time by counting upwards from zero,
are called
# time counters or stopwatches.
# A simple example of the first type of timer is an hourglass.

# See also time-to-digital converter (TDC), time digitizer and time counter (TC).

# For steady-state calculations, current time represents a certain point in time.

# Using the built-in currenttime() function only works in dynamic calculations.
f_current_time: current_time = currenttime();
```

#### Variables

current\_time                    variable that keeps count of the current time

### 3.25.2. time\_counter

#### Purpose

time counter model for steady-state calculations

#### Model equations

```
# The model displays the concept of time in the calculations.

# Timers can be categorized into two main types.
# The word "timer" is usually reserved for devices that counts down from a specified time
interval,
# while devices that do the opposite, measuring elapsed time by counting upwards from zero,
are called
# time counters or stopwatches.
# A simple example of the first type of timer is an hourglass.
```

# See also time-to-digital converter (TDC), time digitizer and time counter (TC).

# For steady-state calculations, current time represents a certain point in time.

# Using the built-in currenttime() function only works in dynamic calculations.

# f\_current\_time: current\_time = currenttime();

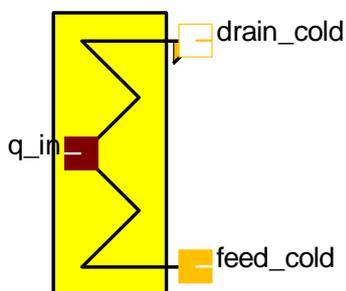
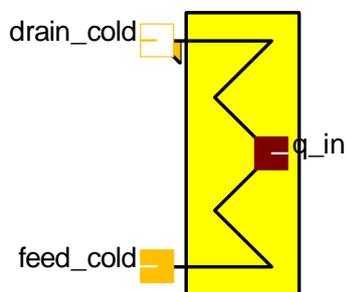
#### Parameters

current\_time            variable that keeps count of the current time

## 3.26. wall\_OR\_htex

### Purpose

Heat exchanger with wall transferring heat from wall to organic fluid (OR).



### Connections

OR\_Stream: feed\_cold  
OR\_Stream: drain\_cold  
q\_cond\_trans: q\_in

### 3.26.1. wall\_OR\_htex

#### Purpose

design model

#### Model equations

# heat transfer from wall to fluid inside the tubes

# mass transfer

```
f_mass: feed_cold.mass = drain_cold.mass;

# pressure drop through the wall tubes
f_delta_p:      feed_cold.p - delta_p = drain_cold.p;

# cold fluid is heat up
f_energy: feed_cold.mass*feed_cold.h + q_trans = drain_cold.mass*drain_cold.h;

# temperatue change
f_delta_t: feed_cold.t + delta_t = drain_cold.t;

# conductive heat transfer through the wall
f_q_q_trans:    q_trans = q_in.q_trans;

# transfer of prevailing temperatures
# for the cold side cocurrent/counter current does not make a difference
f_t_cold_in:    q_in.t_cold_in = feed_cold.t;
f_t_cold_out:   q_in.t_cold_out = drain_cold.t;

# additional calculations and data for operating points of the working fluid with phase change
# (inside the tubes)
# heat is flowing in, working fluid is heated and may evaporate

p_crit:  p_crit = feed_cold.Composition.o_pcrit();
s_crit:  s_crit = feed_cold.Composition.o_scrit();

# mean evaporation pressure
p_evap:  p_evap = feed_cold.p - delta_p;

# discriminating subcritical and supercritical boiler conditions
f_x_feed: if (p_evap <= p_crit) then
            feed_cold.h - feed_cold.Composition.o_h_px( p_evap, 0) - x_feed*
(feed_cold.Composition.o_h_px( p_evap, 1.0) - feed_cold.Composition.o_h_px( p_evap, 0)) =
0.0;
        else
            x_feed = 999.0;

f_x_drain:  if (p_evap <= p_crit) then
            drain_cold.h - drain_cold.Composition.o_h_px( p_evap, 0) - x_drain*
(drain_cold.Composition.o_h_px( p_evap, 1.0) - drain_cold.Composition.o_h_px( p_evap, 0)) =
0.0;
        else
            x_drain = -999;

# start of evaporation resp.. intermediate point I
f_s_l_aux: s_l_aux = feed_cold.s + (drain_cold.s - feed_cold.s) * 4.0/9.0;

f_s_l:  if (x_feed <= 0.0) then
            s_l = feed_cold.Composition.o_s_px(p_evap, 0.0);
        else
            if(s_l_aux <= s_crit) then
```

```
        s_l = s_crit;
    else
        s_l = feed_cold.s + (drain_cold.s - feed_cold.s) * 4.0/9.0;

f_p_l:   if (p_evap <= p_crit && x_feed <= 1.0) then
        p_l = p_evap;
    else
        p_l = feed_cold.p - delta_p*8/9;

f_t_l:   t_l = feed_cold.Composition.o_t_ps(p_l, s_l);

f_h_l:   s_l = drain_cold.Composition.o_s_ph(p_l, h_l);

# end of evaporation resp. intermediate point II
f_s_II_aux: s_II_aux = feed_cold.s + (drain_cold.s - feed_cold.s) * 5.0/9.0;

f_s_II:  if (x_drain >= 1) then
        if (x_feed <= 1.0) then
            s_II = drain_cold.Composition.o_s_px(p_evap, 1.0);
        else
            if(s_II_aux < s_crit) then
                s_II = s_crit;
            else
                s_II = feed_cold.s + (drain_cold.s - feed_cold.s) * 5/9;
        else
            if (x_feed <= 1.0) then
                s_II = feed_cold.Composition.o_s_px(p_II, 0.0);
            else
                if(s_II_aux < s_crit) then
                    s_II = s_crit;
                else
                    s_II = feed_cold.s + (drain_cold.s - feed_cold.s) * 5/9;

f_p_II:  if (p_evap <= p_crit && x_feed <= 1.0) then
        p_II = p_evap;
    else
        p_II = drain_cold.p + delta_p*8.0/9.0;

f_t_II:  t_II = drain_cold.Composition.o_t_ps(p_II, s_II);

f_h_II:  s_II = drain_cold.Composition.o_s_ph(p_II, h_II);

# calculating intermediate data points to obtain a temperature profile (6 in total for each
property)

# subcritical design gives additional points in liquid and vapour region (3 each)
# supercritical design evenly spreads entropy s between inlet and outlet

f_s_1:   s_1 = feed_cold.s + (s_l-feed_cold.s)*1/4;
f_p_1:   p_1 = feed_cold.p - (feed_cold.p-p_l)*1/4;
```

```

f_t_1:    t_1 = feed_cold.Composition.o_t_ps(p_1, s_1);

f_s_2:    s_2 = feed_cold.s + (s_l-feed_cold.s)*2/4;
f_p_2:    p_2 = feed_cold.p - (feed_cold.p-p_l)*2/4;
f_t_2:    t_2 = feed_cold.Composition.o_t_ps(p_2, s_2);

f_s_3:    s_3 = feed_cold.s + (s_l-feed_cold.s)*3/4;
f_p_3:    p_3 = feed_cold.p - (feed_cold.p-p_l)*3/4;
f_t_3:    t_3 = feed_cold.Composition.o_t_ps(p_3, s_3);

f_s_4:    s_4 = s_ll + (drain_cold.s-s_ll)*1/4;
f_p_4:    p_4 = p_ll - (p_ll-drain_cold.p)*1/4;
f_t_4:    t_4 = feed_cold.Composition.o_t_ps(p_4, s_4);

f_s_5:    s_5 = s_ll + (drain_cold.s-s_ll)*2/4;
f_p_5:    p_5 = p_ll - (p_ll-drain_cold.p)*2/4;
f_t_5:    t_5 = feed_cold.Composition.o_t_ps(p_5, s_5);

f_s_6:    s_6 = s_ll + (drain_cold.s-s_ll)*3/4;
f_p_6:    p_6 = p_ll - (p_ll-drain_cold.p)*3/4;
f_t_6:    t_6 = feed_cold.Composition.o_t_ps(p_6, s_6);

# test conditions
t_q_trans:    test (q_trans >= 0.0) warning "q_trans negative";
t_delta_p:    test (delta_p >= 0.0) warning "delta_p negative";

# feed and drain terminal must reference the same composition
ifl ref(feed_cold.Composition) != ref(drain_cold.Composition) then
  t_Comp:    test(1!=1) error "Composition must be the same at feed and drain";
endifl

```

## Parameters

delta\_p                      absolute pressure drop

## Variables

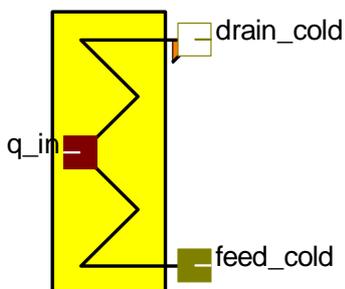
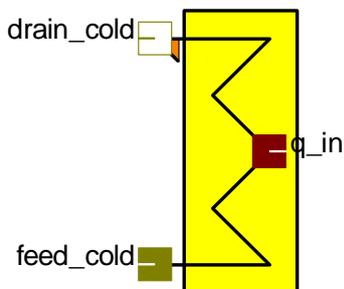
|          |  |
|----------|--|
| delta_t  | temperature difference between feed and drain mass flow                  |
| q_trans  | transferred heat   |
| p_evap   | --- profile calculation: mean evaporation pressure                       |
| p_l      | --- profile calculation: intermediate pressure                           |
| t_l      | --- profile calculation: intermediate temperature (sat liq resp. inlet)  |
| h_l      | intermediate enthalpy (sat liq resp. inlet)                              |
| s_l      | --- profile calculation: intermediate entropy (sat liq resp. inlet)      |
| s_l_aux  | --- profile calculation: auxiliary variable (entropy)                    |
| p_ll     | --- profile calculation: intermediate pressure (sat vap resp. outlet)    |
| t_ll     | --- profile calculation: intermediate temperature (sat vap resp. outlet) |
| h_ll     | intermediate enthalpy (sat vap resp. outlet)                             |
| s_ll     | --- profile calculation: intermediate entropy (sat vap resp. outlet)     |
| s_ll_aux | --- profile calculation: auxiliary variable (entropy)                    |
| x_feed   | --- profile calculation: steam quality at feed                           |
| x_drain  | --- profile calculation: steam quality at drain                          |

|        |   |
|--------|---|
| p_1    | --- profile calculation: intermediate pressure #1               |
| t_1    | --- profile calculation: intermediate temperatures #1           |
| s_1    | --- profile calculation: intermediate entropy #1                |
| p_2    | --- profile calculation: intermediate pressure #2               |
| t_2    | --- profile calculation: intermediate temperature #2            |
| s_2    | --- profile calculation: intermediate entropy #2                |
| p_3    | --- profile calculation: intermediate pressure #3               |
| t_3    | --- profile calculation: intermediate temperature #3            |
| s_3    | --- profile calculation: intermediate entropy #3                |
| p_4    | --- profile calculation: intermediate pressure #4               |
| t_4    | --- profile calculation: intermediate temperature #4            |
| s_4    | --- profile calculation: intermediate entropy #4                |
| p_5    | --- profile calculation: intermediate pressure #5               |
| t_5    | --- profile calculation: intermediate temperature #5            |
| s_5    | --- profile calculation: intermediate entropy #5                |
| p_6    | --- profile calculation: intermediate pressure #6               |
| t_6    | --- profile calculation: intermediate temperature #6            |
| s_6    | --- profile calculation: intermediate entropy #6                |
| p_crit | --- profile calculation: critical pressure of the working fluid |
| s_crit | --- profile calculation: entropy at critical point              |

### 3.27. wall\_T\_htex

#### Purpose

Heat exchanger with wall transferring heat from wall to thermooil (T).



#### Connections

```
q_cond_trans: q_in  
T_Stream: feed_cold  
T_Stream: drain_cold
```

### 3.27.1. wall\_T\_htex

#### Purpose

design model

#### Model equations

```
# heat transfer from wall to fluid inside the tubes  
  
# mass transfer  
f_mass: feed_cold.mass = drain_cold.mass;  
  
# pressure drop through the wall tubes  
f_delta_p:      feed_cold.p - delta_p = drain_cold.p;  
  
# cold fluid is heat up  
f_energy: feed_cold.mass*feed_cold.h + q_trans = drain_cold.mass*drain_cold.h;  
  
# temperatue change  
f_delta_t: feed_cold.t + delta_t = drain_cold.t;  
  
# conductive heat transfer through the wall  
f_q_q_trans:    q_trans = q_in.q_trans;  
  
# transfer of prevailing temperatures  
# for the cold side cocurrent/counter current does not make a difference  
f_t_cold_in:    q_in.t_cold_in = feed_cold.t;  
f_t_cold_out:   q_in.t_cold_out = drain_cold.t;  
  
# test conditions  
t_q_trans:      test (q_trans >= 0.0) warning "q_trans negative";  
t_delta_p:      test (delta_p >= 0.0) warning "delta_p negative";  
  
# feed and drain thermooil must use the same fluid  
t_conn_cold:    test (feed_cold.T_Composition.FluidID == drain_cold.T_Composition.FluidID)  
                error "different thermofluid at feed and drain!";
```

#### Parameters

delta\_p absolute pressure drop

#### Variables

delta\_t temperature difference between feed and drain mass flow  
q\_trans transferred heat

## 3.28. OR\_Stream

#### Purpose

Transfer of mass. The transferred fluid is represented by an OR\_Composition object.

### Global objects

OR\_Composition: Composition defines the ORC working fluid in use

## 3.28.1. OR\_Stream

### Purpose

Transfer of mass. The transferred fluid is represented by an OR\_Composition object.

### Model equations

ft: if blocksize() == 1.0 && isconverged(p) && isconverged(t) then

h = Composition.o\_h\_pt(p, t);

else

t = Composition.o\_t\_ph(p, h);

fs: s = Composition.o\_s\_ph(p, h);

fv: v = Composition.o\_v\_ph(p, h);

t1: test (mass >=0.0) warning "mass flow negative";

### Variables

|      |                 |
|------|-----------------|
| p    | pressure        |
| t    | temperature     |
| h    | enthalpy        |
| s    | entropy         |
| v    | specific volume |
| mass | mass flow       |

## 3.29. TCM\_Stream

### Purpose

thermochemical storage material (TCM) stream

### Global objects

TCM: TCM Thermochemical material working pair

T\_Composition: T\_Composition Heat transfer fluid in use

## 3.29.1. TCM\_Stream

### Purpose

thermochemical storage material (TCM) stream

### Model equations

# thermochemical storage material stream

# It is a mixture of the working pair material suspended in oil plus water

# mass fraction balances

f\_mass\_frac: w\_Material1 + w\_Material2 + w\_Oil + w\_Water = 1.0;

```

f_mass_Material1:    mass_Material1 = mass * w_Material1;
f_mass_Material2:    mass_Material2 = mass * w_Material2;
f_mass_Oil:          mass_Oil = mass * w_Oil;
f_mass_Water:        mass_Water = mass * w_Water;

# various fractions for additional information

#fSolid_fraction:    Solid_fraction = Material1 + Material2 ;
f_w_Solids:          w_Solids = w_Material1 + w_Material2;

#fX_Material1_Oil:    X_Material1_Oil = Material1 / Oil;
#fX_Material2_Oil:    X_Material2_Oil = Material2 / Oil;
f_X_Material1_Oil:    X_Material1_Oil * w_Oil = w_Material1;
f_X_Material2_Oil:    X_Material2_Oil * w_Oil = w_Material2;

# Global thermodynamic properties

#fh:                 h = h_Oil * Oil + h_Material1 * Material1 + h_Material2 * Material2 + water *
h_Water;
#fv:                 v = v_Oil * Oil + v_Material2 * Material2 + v_Material1 * Material1 + water *
v_Water;

f_h:                 h = h_Oil * w_Oil + h_Material1 * w_Material1 + h_Material2 * w_Material2 +
h_Water * w_Water;
f_v:                 v = v_Oil * w_Oil + v_Material1 * w_Material1 + v_Material2 * w_Material2 +
v_Water * w_Water;

# individual specific volumes
f_v_Oil:             v_Oil = T_Composition.htf_v_t(t);
f_v_Water:           v_Water = fv(p, h_Water_int, 1, 0,0,0,0,0,0,0,0,0,0,0,0);
f_v_Material1:       v_Material1 = TCM.TCM_v_t(t, TCM.ID1);
f_v_Material2:       v_Material2 = TCM.TCM_v_t(t, TCM.ID2);

# density
f_rho:               rho * v = 1.0;

# enthalpies of the TCM stream

# Using enthalpies with different zero point definitions and making them consistent
# All enthalpies are defined here to be zero at 25°C

f_h_Oil:             h_Oil = T_Composition.htf_h_t(t) - T_Composition.htf_h_t(25);

f_h_Material1:       h_Material1 = TCM.TCM_h_t(t, TCM.ID1) - TCM.TCM_h_t(25, TCM.ID1);
f_h_Material2:       h_Material2 = TCM.TCM_h_t(t, TCM.ID2) - TCM.TCM_h_t(25, TCM.ID2);
#f_h_Material1:      h_Material1 = TCM.TCM_h_t(t, TCM.ID1);
#f_h_Material2:      h_Material2 = TCM.TCM_h_t(t, TCM.ID2);

f_h_Water:           h_Water = h_Water_int - fhpt(1, 25, 1, 0,0,0,0,0,0,0,0,0,0,0,0);

# for internal use (to allow using fv(p,h,...) above

```

```
f_h_Water_int:  h_Water_int = fhpt(p, t, 1, 0,0,0,0,0,0,0,0,0,0,0);

# Water phase and water vapour pressure

#f_Water_Phase:p_H2Os = (exp(73.649 - 7258.2/(t+273.15) - 7.3037*ln(t+273.15) +
(4.1653*10^(-6))*(t+273.15)^2))/100000;
f_p_H2Os:      p_H2Os*1e5 = exp(73.649 - 7258.2/(t+273.15) - 7.3037*ln(t+273.15) +
(4.1653*10^(-6))*(t+273.15)^2);

# -1 -> liquid /// 1 -> gas
f_Water_Phase: if (p > p_H2Os) then
                Water_Phase = -1;
                else
                Water_Phase = 1;

# Pressure for liquid water
f_p_required:  p_required = p_H2Os + 0.1;
```

### Variables

|                 |  |
|-----------------|--|
| p               | pressure   |
| t               | temperature                                      |
| h               | enthalpy   |
| v               | specific volume                                  |
| rho             | density  |
| mass            | total mass flow                                  |
| mass_Material1  | mass flow material 1                             |
| mass_Material2  | mass flow material 2                             |
| mass_Oil        | mass flow oil                                    |
| mass_Water      | mass flow water                                  |
| w_Material1     | mass fraction material 1                         |
| w_Material2     | mass fraction material 2                         |
| w_Oil           | mass fraction Oil                                |
| w_Water         | mass fraction Water                              |
| w_Solids        | solid fraction                                   |
| X_Material1_Oil | mass ratio of material1 to oil                   |
| X_Material2_Oil | mass ratio of material2 to oil                   |
| h_Material1     | enthalpy of material 1                           |
| h_Material2     | enthalpy of material 2                           |
| h_Oil           | enthalpy of oil                                  |
| h_Water         | enthalpy of water                                |
|                 | with the same zero point as the other substances |
| h_Water_int     | enthalpy of water (for internal processing)      |
| v_Material1     | specific volume material 1                       |
| v_Material2     | specific volume material 2                       |
| v_Oil           | specific volume oil                              |
| v_Water         | specific volume water                            |
| Water_Phase     | -1 liquid // +1 gas                              |
| p_H2Os          | saturation pressure of water                     |
| p_required      | minimal required pressure for liquid water       |

## 3.30. q\_cond\_trans

### Purpose

conductive heat transfer through a wall (solid boundary)

### 3.30.1. q\_cond\_trans

### Purpose

conductive heat transfer through a wall (solid boundary)

### Model equations

```
# temperature differences, MTD and heat transfer area*heat transfer coefficient

# definition of prevailing temperature differences
f_dt_in: dt_in = t_hot_in - t_cold_in;
f_dt_out: dt_out = t_hot_out - t_cold_out;

# Mean temperature difference is the logarithmic temperature difference.
# For nearly equal inlet and outlet temperature differences, it is approximated
# by the arithmetic mean (which is the asymptotic approximation).
f_MTD: if abs(dt_in/dt_out) >=1.2 || abs(dt_out/dt_in) >=1.2 then
    MTD * ln(dt_in/dt_out) = (dt_in-dt_out);
else
    MTD = (dt_in+dt_out)/2.0;

# heat transfer equation
f_q_trans: q_trans = htc_area * MTD;

# For now, there is no explicit variable which represents the heat transfer area.
# If there is an explicit area, also a heat transfer coefficient is required.
```

### Variables

|            |  |
|------------|--|
| q_trans    | transferred heat   |
| dt_in      | temperature difference at the inlet of the cooling stream  |
| dt_out     | temperature difference at the outlet of the cooling stream |
| MTD        | mean temperature difference                                |
| htc_area   | heat transfer coefficient x heat transfer area             |
| t_hot_in   | temperature of the hot side at the inlet                   |
| t_hot_out  | temperature of the hot side at the outlet                  |
| t_cold_in  | temperature of the cold side at the inlet                  |
| t_cold_out | temperature of the cold side at the outlet                 |

## 3.31. OR\_Composition

### Purpose

Represents an organic carbon-based working fluid and some other alternatives used in organic rankine cycles, refrigeration and heat pump processes.

### 3.31.1. OR\_Composition

#### Purpose

Represents an organic carbon-based working fluid and some other alternatives used in organic rankine cycles, refrigeration and heat pump processes.

#### Model equations

```
#####  
# The following are all coming from the REFPROP database  
  
# pure fluids  
  
# Ammonia  
ifl FluidName == Ammonia then  
  f_Ammonia: FluidID = 11;  
endifl  
  
ifl FluidName == Butane then  
  f_Butane: FluidID = 12;  
endifl  
  
ifl FluidName == Butene then  
  f_Butene: FluidID = 13;  
endifl  
  
# Carbon Dioxide  
ifl FluidName == CO2 then  
  f_CO2: FluidID = 14;  
endifl  
  
# Ethanol  
# Other names: ethyl alcohol  
# molecular formula C2H5OH  
ifl FluidName == Ethanol then  
  f_Ethanol: FluidID = 15;  
endifl  
  
ifl FluidName == Isobutane then  
  f_Isobutane: FluidID = 16;  
endifl  
  
ifl FluidName == Isobutene then  
  f_Isobutene: FluidID = 17;  
endifl  
  
# Three isomers exist for pentane:  
# n-pentane (linear molecule), isopentane and neopentane  
# All three have the molecular formula C5H12  
  
# Isopentane  
# Other name: methylbutane, 2-methylbutane
```

```
ifl FluidName == Isopentane then
  f_Isopentane: FluidID = 18;
endifl

# Neopentane
# Other name: dimethylpropane, 2,2-dimethylpropane
ifl FluidName == Neopentane then
  f_Neopentane: FluidID = 19;
endifl

# Pentane
# Other name: n-Pentane
ifl FluidName == Pentane then
  f_Pentane: FluidID = 20;
endifl

ifl FluidName == Propane then
  f_Propane: FluidID = 21;
endifl

# Toluene
# Other names: toluol, methylbenzene, phenylmethane
# molecular formula C7H8
ifl FluidName == Toluene then
  f_Toluene: FluidID = 22;
endifl

# Water
ifl FluidName == Water then
  f_Water: FluidID = 23;
endifl

# Cyclopropane
# molecular formula C3H6
ifl FluidName == Cyclopropane then
  f_Cyclopropane: FluidID = 24;
endifl

# Cyclopentane
# molecular formula C5H10
ifl FluidName == Cyclopentane then
  f_Cyclopentane: FluidID = 25;
endifl

# Cyclohexane
# molecular formula C6H12
ifl FluidName == Cyclohexane then
  f_Cyclohexane: FluidID = 26;
endifl

# Siloxanes
```

```
ifl FluidName == D4 then
  f_D4: FluidID = 31;
endifl
```

```
ifl FluidName == D5 then
  f_D5: FluidID = 32;
endifl
```

```
ifl FluidName == D6 then
  f_D6: FluidID = 33;
endifl
```

```
ifl FluidName == MDM then
  f_MDM: FluidID = 34;
endifl
```

```
ifl FluidName == MD2M then
  f_MD2M: FluidID = 35;
endifl
```

```
ifl FluidName == MD3M then
  f_MD3M: FluidID = 36;
endifl
```

```
ifl FluidName == MD4M then
  f_MD4M: FluidID = 37;
endifl
```

```
ifl FluidName == MM then
  f_MM: FluidID = 38;
endifl
```

```
# refrigerants
```

```
ifl FluidName == R123 then
  f_R123: FluidID = 51;
endifl
```

```
ifl FluidName == R1234yf then
  f_R1234yf: FluidID = 52;
endifl
```

```
ifl FluidName == R1234ze then
  f_R1234ze: FluidID = 53;
endifl
```

```
ifl FluidName == R124 then
  f_R124: FluidID = 54;
endifl
```

```
ifl FluidName == R125 then
  f_R125: FluidID = 55;
```

```
endifl

ifl FluidName == R134a then
  f_R134a: FluidID = 56;
endifl

ifl FluidName == R142b then
  f_R142b: FluidID = 57;
endifl

ifl FluidName == R143a then
  f_R143a: FluidID = 58;
endifl

ifl FluidName == R152a then
  f_R152a: FluidID = 59;
endifl

ifl FluidName == R227ea then
  f_R227ea: FluidID = 60;
endifl

ifl FluidName == R236ea then
  f_R236ea: FluidID = 61;
endifl

ifl FluidName == R236fa then
  f_R236fa: FluidID = 62;
endifl

ifl FluidName == R245ca then
  f_R245ca: FluidID = 63;
endifl

# R245fa
# Other name: 1,1,1,3,3-pentafluoropropane
# molecular formula CF3CH2CHF2
ifl FluidName == R245fa then
  f_R245fa: FluidID = 64;
endifl

# R365mfc
# Other name: 1,1,1,3,3-pentafluorobutane
# molecular formula CF3CH2CF2CH3
ifl FluidName == R365mfc then
  f_R365mfc: FluidID = 160;
endifl

# Novec649 (3M brand name)
# Other name: 1,1,1,2,2,4,5,5,5-nonafluoro-4-(trifluoromethyl)-3-pentanone
# Other name: FK-5-1-12
```

```
# molecular formula C6F12O
ifl FluidName == Novec649 then
  f_Novec649: FluidID = 153;
endifl

# mixtures (treated as pseudo-pure fluids)

ifl FluidName == R407C then
  f_R407C: FluidID = 83;
endifl

ifl FluidName == R410A then
  f_R410A: FluidID = 86;
endifl

ifl FluidName == R507A then
  f_R507A: FluidID = 91;
endifl

ifl FluidName == Air then
  f_Air: FluidID = 95;
endifl

# end of REFPROP section
#####

# user defined fluid uses the REFPROP calculation methods and file format
#ifl FluidName == _userdefined then # *.FLD is copied to userdefined1.fld file
# f_userdefined1: FluidID = -1;
#endifl
```

### Variables

FluidID                    fluid identifier

### Switches

FluidName (Air, Ammonia, Butane, Butene, CO2, Cyclohexane, Cyclopentane, Cyclopropane, D4, D5, D6, Ethanol, Isobutane, Isobutene, Isopentane, MD2M, MD3M, MD4M, MDM, MM, Neopentane, Novec649, Pentane, Propane, R123, R1234yf, R1234ze, R124, R125, R134a, R142b, R143a, R152a, R227ea, R236ea, R236fa, R245ca, R245fa, R365mfc, R407C, R410A, R507A, Toluene, Water)    working fluid to be used

## 3.32. TCM

### Purpose

thermochemical material working pair

### 3.32.1. TCM

### Purpose

thermochemical material working pair

### Model equations

# Selection of the thermochemical material system.

# Note that only certain selected working pairs of the available materials make sense.

```
# H3BO3 and HBO2
ifl Material_System == Boric_Acid then
  # boric acid H3BO3
  f_BA_1:      ID1 = 1;
  # metaboric acid HBO2
  f_BA_2:      ID2 = 2;
endifl

# CuSO4.5H2O and CuSO4.H2O
ifl Material_System == Copper_Sulfate then
  # copper sulfate pentahydrate CuSO4.5H2O
  f_CS_1:      ID1 = 3;
  # copper sulfate monohydrate CuSO4.H2O
  f_CSA_2:     ID2 = 4;
endifl

# CaCl2.2H2O and CaCl2
ifl Material_System == Calcium_Chloride then
  # calcium chloride dihydrate CaCl2.2H2O
  f_CC_1:      ID1 = 5;
  # anhydrous calcium chloride CaCl2
  f_CC_2:      ID2 = 6;
endifl

# K2CO3.3/2_H2O and K2CO3
ifl Material_System == Potassium_Carbonate then
  # potassium carbonate sesquihydrate K2CO3.3/2_H2O
  f_PC_1:      ID1 = 7;
  # anhydrous potassium carbonate K2CO3
  f_PC_2:      ID2 = 8;
endifl
```

#### Variables

|     |                       |
|-----|-----------------------|
| ID1 | identifier material 1 |
| ID2 | identifier material 2 |

#### Switches

Material\_System (Boric\_Acid, Calcium\_Chloride, Copper\_Sulfate, Potassium\_Carbonate)  
Selection of the thermochemical material system

### 3.33. time\_period

#### Purpose

time period used for a certain operation

### 3.33.1. time\_period

#### Purpose

time period used for a certain operation

#### Model equations

# This is a test for conversion and usability of the concept

f\_time:  $\text{operating\_hours} * 3600 = \text{operating\_time};$

#### Variables

operating\_hours      operating hours

operating\_time      operating time

## 4. Economic modelling

This chapter provides details of the final economic model adopted in the RESTORE project. The economic analysis is essential in the development of innovative storage systems, as RESTORE system, as it provides crucial information for the correct evaluation of the project concept. By evaluating financial risks and potential returns, it supports informed decision-making and strategic investment.

The RESTORE economic model aims to give estimations for carried out the economic analysis of the RESTORE concept supporting relevant project activities as the Task 1.7 – Socio-Economic Assessment for the RESTORE solution.

For this purpose, CENER has developed and implemented a specific financial model adapted to the particularities of a very innovative energy storage system as RESTORE concept represents in order to ease its economic evaluation.

In this context, a relatively complex model has been developed in order to improve the representation and results, considering a wide variety of financial features and parameters, while at the same time, attention has been paid to build a model user-friendly.

The model presents a high flexibility, making possible the users to modify a high number of parameters in order to customize their analysis and fulfil their requirements. The main financial parameters are shown in Table 4, with some default values that could be modified by the users.

Table 4 – Economic parameters and default values.

| Inputs                |   | Default value |
|-----------------------|---|---------------|
| <b>General</b>        | Useful life (years)                           | 28            |
|                       | Yearly inflation rate (%)                     | 2,50%         |
| <b>Heat charge</b>    | Overall heat feed-in tariff (€/kWh) Purchase  | 0.1           |
| <b>Heat discharge</b> | Overall heat feed-in tariff (€/kWh) Sales     | 0.2           |
|                       | Residual Value                                | 0             |
| <b>El. Charge</b>     | El. overall feed-in tariff (€/kWh) Purchase   | 0.3           |
| <b>El. Discharge</b>  | El. overall feed-in tariff (€/kWh) Sales      | 0.7           |
|                       | Residual Value                                | 0             |
| <b>Debt</b>           | Debt (% total investment)                     | 75%           |
|                       | Debt repayment period (years)                 | 15            |
|                       | Yearly debt interest (%)                      | 5,00%         |
|                       | Grace period - Interest-only period (years)   | 3             |
| <b>Equity</b>         | Equity (% total investment)                   | 25%           |
|                       | Yearly rate of return on equity capital (%)   | 12%           |
| <b>Other Expenses</b> | O&M costs per unit of net electricity (€/kWh) | 0,0250        |
|                       | Insurance cost (%)                            | 0,50%         |
|                       | Land occupation (m <sup>2</sup> )             | 0             |
|                       | Land cost per unit area (€/m <sup>2</sup> )   | 0,000         |
|                       | Decommissioning Cost                          | 0%            |
| <b>Taxes</b>          | Tax rate (%)                                  | 35,00%        |
|                       | MAT - Minimum Alternative Tax (%)             | 0,00%         |
|                       | Tax Credit Expiration Period (years)          | 10            |
|                       | Depreciation Schedule (yearly %)              | ----          |

On the other hand, the model also fixed in other to represent a realistic scenario for its calculations. In this context, the following aspects have been considered for the financial model:

- The inflation rate is considered to be constant over the investment life, and it is directly applied to the energy prices, O&M costs, land costs, etc.
- The Yearly Net Electricity Production and Consumption can be varied for each year by assigning a percentage of the reference production.
- The Yearly Net Heat Production and Consumption can be varied for each year by assigning a percentage of the reference production.
- The Plant Capital Cost investing schedule can be described by assigning a percentage of the total Plant Capital Cost for a number of years.
- The debt payment amortization system considered for the analysis is the French System (constant instalment).
- In addition to the corporate tax, a Minimum Alternate Tax, MAT (See description later in this document) and tax credit can be applied (set 0 years in the Tax Credit Expiration Period for not considering tax credit).
- The depreciation schedule considered for the capital assets can be provided as a percentage of the total investment for each year of the plant life. The appropriate calculation of the depreciation schedule has to be done by the user.
- Costs related to grid connection and energy deliver to the energy networks have not been considered.

The financial model calculates the cash flows for the project and, from these, the most common financial indicators, such as the Net Present Value (NPV), the Internal Rate of Return (IRR), and the Debt Service Coverage Ratio (DSCR) for each year. Besides, due to the particularities of the RESTORE concept, able to store and deliver energy in both forms: electricity and heat, the cost of the energy is analyzed from two different perspectives:

- Electricity approach, estimating the Levelized Cost Of Storage for Electricity (LCOS<sub>el</sub>) and considering the heat as a side product.
- Heat approach, estimating the Levelized Cost Of Heat (LCOS<sub>heat</sub>) and considering the electricity as a side product.

In summary, the main outputs of the economic model are shown in Table 5 below

*Table 5 – Main Outputs of the Economic input parameters from other models of the system.*

| Output                                    | Units |
|---|-------|
| Project Net Present Value (NPV)           | €     |
| Project Internal Return Rate (IRR)        | %     |
| Levelized Cost of Storage for Electricity | €/kWh |
| Levelized Cost of Storage for Heat        | €/kWh |
| Plant capital cost                        | (€)   |

In addition, the economic model includes a submodule responsible for estimating the investment cost of the facility, which is a key output required for the subsequent calculation of the final figures of merit — the main objective of the economic model.

Finally, the interaction of this model with the simulation of the virtual use cases, through their corresponding models implemented on the IPSE-GO platform, is essential. Those simulations provide relevant technical information, such as annual energy balances, which include the energy consumed and delivered by the system.

#### 4.1.1. The investment cost module

The investment cost module estimates the total investment cost of the system. The resulting value is then integrated into the overall economic model.

The total investment cost is calculated based on the estimated cost of the main components that constitute the RESTORE energy storage system. This includes both the key components of the organic cycles and those involved in the thermochemical energy storage subsystem.

The cost of each component is estimated using a specific correlation, which takes as input one or two characteristic parameters that reflect the nominal operating conditions and the size of the component. Each correlation is unique and defines how the cost scales with the component's size.

The following sections detail the correlation used for each main component.

##### ORC Turbine

The turbine used in the Organic Rankine Cycles (ORC) is a crucial element that should efficiently convert the thermal energy of the relatively low-temperature, low-pressure organic fluids into mechanical power. Specifically, an axial turbine where the working fluid flows parallel to the axis of rotation is chosen. The correlation defined for the cost estimation of these components is shown below:

$$C = F1 * 10^{(K1+K2*(\log_{10} x)+K3*(\log_{10} x)^2)}$$

Where x represents the turbine shaft power in kilowatt and parameters values are shown below:

|           |        |
|-----------|--------|
| <i>K1</i> | 2.266  |
| <i>K2</i> | 1.44   |
| <i>K3</i> | -0.178 |
| <i>F1</i> | 3.15   |

This correlation provides a result in USD, so it is required to apply the conversion factor change to euros.

##### Organic fluid compressor

The compressor used in the heat pump increases the pressure and temperature of the organic fluid, enabling heat transfer from the colder area (supply) to a warmer one (thermochemical reactor) while circulating the organic fluid through the system. The centrifugal compressor was chosen. The correlation defined for the cost estimation of these components is shown below:

$$C = F1 * 10^{(K1+K2*(\log_{10} x)+K3*(\log_{10} x)^2)}$$

Where x represents the compressor shift power in kilowatts and the parameters values are shown below:

|           |        |
|-----------|--------|
| <i>K1</i> | 2.29   |
| <i>K2</i> | 1.360  |
| <i>K3</i> | -0.103 |
| <i>F1</i> | 2,2    |

This correlation provides a result in USD, so it is required to apply the conversion factor change to euros.

### Reversible scroll compressor

A reversible scroll machine suitable for smaller-scale solutions. This is suitable for small applications and it is able to work as a generator during the discharge or as a compressor during the charge. The correlation defined for the cost estimation of these components is shown below:

$$C = K1 * x^{K2}$$

Where x represents the volumetric flow through the compressor in m<sup>3</sup>/s and the parameters values are shown below:

|           |        |
|-----------|--------|
| <i>K1</i> | 166000 |
| <i>K2</i> | 0.824  |

This correlation provided a result in GBP, so it is required to apply the conversion factor change to euros.

### Heat Exchangers (Shell and tube)

The heat exchangers are Shell and tube type and are connected to the organic cycle (organic fluid through the tubes). They are used inside the thermodynamic cycle for the evaporator of the heat pump, condenser of the ORC and regenerator of both cycles. The shell side considered material is SA-516, while the tube side material is SA-214. The Heat Exchangers are also used as a preheater of the Thermochemical reactor. The correlation defined for the cost estimation of these components is shown below:

$$C = k1 * \left(\frac{x - k2}{k3}\right)^{k4}$$

Where x represents the product of the global heat transfer coefficient and the heat exchange area in W/K, while the parameters values are shown below:

|           |      |
|-----------|------|
| <i>K1</i> | 175  |
| <i>K2</i> | 50   |
| <i>K3</i> | 28.5 |
| <i>F1</i> | 0.69 |

### Throttle valve

The throttle valve in a heat pump is a crucial component that reduces the pressure of the organic fluid before it enters the evaporator. This drop in pressure causes the fluid to cool significantly, allowing it to absorb heat efficiently at lower temperature. It helps regulate the flow and ensures proper phase change of the working fluid. A steel motor operated valve is chosen. The correlation defined for the cost estimation of these components is shown below:

$$C = K1 * x *$$

Where x represents the mass flow through the valve in kg/s while the parameter value for K1 is shown below:

|           |        |
|-----------|--------|
| <i>K1</i> | 105.46 |
|-----------|--------|

This correlation provides a result in USD, so it is required to apply the conversion factor change to euros

### Organic Fluid Pump

The organic fluid pump in an Organic Rankine Cycle (ORC) is responsible for pressurizing the working fluid before it enters the evaporator. It increases the fluid's pressure without significantly raising its temperature, enabling efficient heat absorption. The pump plays a key role in maintaining the organic fluid flow. The correlation defined for the cost estimation of these components is shown below:

$$C = F1 * 10^{(K1+K2*(\log_{10} x)+K3*(\log_{10} x)^2)}$$

Where x is the pump shift power in kW while the parameters values are shown below:

|           |       |
|-----------|-------|
| <i>K1</i> | 3.389 |
| <i>K2</i> | 0.054 |
| <i>K3</i> | 0.154 |
| <i>F1</i> | 2.268 |

This correlation provides a result in USD, so it is required to apply the conversion factor change to euros

### Thermochemical Reactor.

A stirrer continues the reactor where the thermal oil with the suspended thermochemical material is heated up during charge or cooled down during the discharge process. The reactor is made of steel 1.4301 and includes the coil connected to the organic cycle. The correlation defined for the cost estimation of these components is shown below:

$$C = k1 * x^{(1 - k2)}$$

Where x represents the heat transferred in nominal conditions during the charge mode in kW while the parameters values are shown below:

|           |       |
|-----------|-------|
| <i>K1</i> | 27334 |
| <i>K2</i> | 0.6   |

## Thermochemical Material Storage

The thermochemical material is stored at low temperature in high-density polyethylene (HDPE) tanks. These tanks must be able to store the solution of thermal oil plus TCM (with a high concentration of TCM) and present a cheaper way to store the thermochemical material. The tanks are reinforced in order to increase their structural resistance, allowing a maximum size of 5 m<sup>3</sup>. The estimated cost of each tank is around 1200 €, which means an economically competitive system for storing the TCM. In addition, the estimated cost of the TCM is 1000 € per ton.

Then, after the calculation, the cost must be updated considering the CEPCI factor using the following equation:

$$C_{\text{present}} = \frac{\text{CEPCI}_{\text{present}}}{\text{CEPCI}_{\text{correlation}}} \cdot C_{\text{correlation}}$$

Where  $C_{\text{present}}$  is the updated cost,  $\text{CEPCI}_{\text{present}}$  represents the index in the present year, and  $C_{\text{correlation}}$  and  $\text{CEPCI}_{\text{correlation}}$ , cost resulting from the correlation and the CEPCI in the year of the correlation. The CEPCI coefficient for each correlation is shown below:

|                              |      |
|------------------------------|------|
| Turbine                      | 2.02 |
| Compressor                   | 2.02 |
| Reversible scroll compressor | 1.05 |
| Heat Exchangers              | 1.0  |
| Throttle valve               | 2.02 |
| Organic Fluid Pump           | 2.02 |
| Thermochemical reactor       | 1.0  |
| Storage tanks                | 1.0  |

Finally, the module includes a contingency factor in order to consider additional costs related to auxiliary systems (piping, control, etc) and construction, adding an extra value for covering uncertainties. As the default value, the contingency cost is set as 15%.

### 4.1.2. The Levelized Cost of Storage for Electricity

The Levelized Cost Of Storage for Electricity (LCOS<sub>el</sub>) is an economic indicator of great usefulness when comparing technological options, concerning energy storage systems, from an economic point of view.

The LCOS<sub>el</sub> can be defined, in actual euros (or dollars), as the value that would have to be assigned to every unit of dispatchable electricity delivered by the energy storage system throughout a determined period in order to equal the total costs incurred during this period, also expressed in actual euros (or dollars).

In this approach, the electricity is considered the main product purpose of the study meanwhile the heat delivered is considered as a side product, thus representing an income in the calculations.

That means that, if  $Q_n$  is the total electric energy produced by the power plant in the year  $n$ ,  $d$  is the discount rate and  $C_n$  is the total cost incurred at the power plant in the year  $n$ , according to the definition of  $LCOS_{el}$ , the following equality:

$$\sum_{n=1}^N \frac{Q_n \times LCOS_{el}}{(1+d)^n} = \sum_{n=1}^N \frac{C_n}{(1+d)^n}$$

From this equality, the following mathematical expression can be obtained for the calculation of the  $LCOS_{el}$ :

$$LCOS_{el} = \frac{\sum_{n=1}^N \frac{C_n}{(1+d)^n}}{\sum_{n=1}^N \frac{Q_n}{(1+d)^n}}$$

Where:

- $LCOS_{el}$  is the Levelized Cost Of Storage for Electricity
- $n$  is the number of years (period) for carrying out this calculation, which usually coincides with the expected useful life of the power plant.
- $C_n$  is the net cost (cost minus benefits from side products) incurred during the year  $n$ , which includes, among others, the investment cost in the first years of construction and commissioning of the plant, O&M, cost associated to energy, as well as financial costs, including main depreciation and interest payment.
- $Q_n$  is the yearly electricity delivered by the storage system in the year  $n$ .
- $d$  is the discount rate, which indicates the present value of the future cash flows that depend on, among others, the debt interest, inflation rate and expected investment profitability.

The next step is to calculate the different factors required by the  $LCOS_{el}$  expression.

As a discount rate, the Weighted Average Cost of Capital (WACC) needed to finance the construction of the energy storage system has been used. The WACC is the minimum return that a company must earn on an existing asset base to satisfy its creditors, owners, and other providers of capital, so that they will not invest elsewhere. It can be calculated by using the following expression:

$$d = WACC = \frac{E_I \times R_{EI} + D_I \times R_{DI}}{E_I + D_I}$$

Where:

- $E_I$  is the self-financing (equity capital) of the investment
- $R_{EI}$  is the yearly rate of return on equity capital (profitability expected from self-financing)
- $D_I$  is the bank debt of the investment
- $R_{DI}$  is the yearly debt interest

In addition, the annual net costs include the initial investment, all type of costs derived from the achievement of a proper operation of the plant (operation and maintenance costs), energy cost, as well as financial costs, etc.

In addition, the value of these annual costs, as well as the relative weight of specific costs composing them, varies along the years. Next expression shows the way these costs can be calculated:

$$\sum_{n=1}^N \frac{C_n}{(1+d)^n} = \sum_{n=1}^N \frac{I_n + OM_n + FC_n + EC_n - TS_n}{(1+d)^n}$$

Where:

- $I_n$  is the investment of equity capital in the year  $n$ . As the LCOS value is calculated from a promoter point of view of the project regarding construction, commissioning and operation of system, only the equity capital is considered as investment; the rest of the capital is considered as a debt. This magnitude is typically higher than zero during the first years of the analysis, in which the system is under project and construction, and zero for the rest of the period considered for the calculation of the LCOS. Even if the simplest way to consider the investment is to consider that the total equity capital is invested in the initial instant, this simplification has not been considered as it underestimates the value of the money flow during the construction period. Instead, for each case, more realistic criteria can be rather considered according to the schedule of equity capital contributions during the whole useful lifetime of the power plant. This value is estimated by the CAPEX model mentioned in previous section.
- $OM_n$  is the annual operation and maintenance (O&M) cost in the year  $n$ . These costs include both O&M costs of the central (operation labour, maintenance, etc.) and land renting costs. This annual cost can be estimated as a percentage of the investment cost for larger plants, while in the case of smaller applications, an approach considering the number of workers and salary, and a small percentage for covering piece replacement and repairment, is more accurate.

- $EC_n$  is the annual energy cost in the year  $n$ . These costs include, expenses such as the cost of the electricity and heat used for charging the system, as well as incomes, associated to the side products (in this case the heat delivered).
- $FC_n$  is the annual financial cost in the year  $n$ . This cost includes both depreciation of the principal payment and the debt interest payment. As it has been already explained, the calculation of the LCOS has been carried out from the stockholder point of view, so the bank financing (debt) is considered as an expense throughout the years and not as an initial investment. Logically, in order to obtain the financial costs and their distribution during the years, a repayment period and/or a grace period has to be considered.
- $TS_n$  is the tax saving in the year  $n$ , which is calculated for each year as the tax rate considered multiplied by the sum of the deductible expenses: O&M expenses, depreciation of investments and the corresponding part of the financial costs interest payment. This term is sometimes not considered depending on the type of analysis performed (that means considering a tax rate value = 0%).

### 4.1.3. The Levelized Cost of Heat

Similarly, an estimation of the Levelized Cost of Storage for Heat can be conducted following a similar procedure. In this scenario, the calculation process is analogous to the aforementioned approach. However, the underlying assumption that differentiates both approaches is the consideration of the Heat as the primary product; meanwhile, the delivered electricity represents the side product which represents an income when calculating the energy costs.

### 4.1.4. Example of application of the economic model to a RESTORE virtual use-case.

In this example, the previously described economic model is applied to Virtual Use Case 2 (UC2) [13]. This use case is based on the integration of the RESTORE energy storage system with the Rohrdorfer cement plant and the local district heating network in Gmunden.

UC2 is characterised by the availability of high-temperature waste heat from the cement plant, which enables the integration of an auxiliary organic loop coupled to the heat pump. This configuration generates additional electricity during the charging phase, potentially covering the system's own electricity consumption during this process.

Further technical details and an extended description of UC2 are provided in Deliverable *D5.5 – Implementation and Validation of RESTORE Use Case II: Integration of Different Heat Sources in DH of Cement Industry* [13].

The economic model uses as input the results from the technical model of UC2, developed on the IPSE-GO platform using the RESTORE model library [13]. The main inputs related to the energy balances are summarised in Table 6.

*Table 6 – Main Inputs related to Energy Balances.*

| Energy Balances          |       |          |
|--------------------------|-------|----------|
| Annual electricity inlet | 5544  | MWh/year |
| Annual heat in           | 47398 | MWh/year |
| Annual heat out          | 33909 | MWh/year |

In addition, the economic model also receives, as input from the technical model in IPSE-GO, the key parameters that define the size of each main component of the RESTORE system.

These parameters are applied within the investment cost module, where the cost of each main component is estimated using the correlations described in Section 4.1.1.

Table 7 below summarises the results of these correlations, presenting the estimated cost for each component.

*Table 7 – Estimation of Components Costs.*

| Estimated Components Cost |                                     |                     |
|---------------------------|-------------------------------------|---------------------|
| Component                 | Correlation                         | Cost                |
| Thermochemical reactor    | Thermochemical reactor              | 328.316 €           |
| TCM preheat-cooling HEX   | Heat Exchange Shell and tubes       | 10.270 €            |
| DH Condenser              | Heat Exchange Shell and tubes       | 1.047.456 €         |
| Charge turbine            | Non-reversible turbine              | 567.613 €           |
| Charge pump               | Organic Fluid Pump                  | 34.052 €            |
| WH HEX                    | Heat Exchange Shell and tubes       | 2.487 €             |
| Throttle valve            | Throttle valve of the Organic Cycle | 2260 €              |
| Charge compressor         | Non-reversible compressor           | 196.152 €           |
| DH Evap-Cond              | Heat Exchange Shell and tubes       | 14.971 €            |
| Reheat HEX                | Heat Exchange Shell and tubes       | 27.714 €            |
| Discharge pump            | Organic Fluid Pump                  | 10.115 €            |
| Discharge turbine         | Non-reversible turbine              | 57.426 €            |
| Storage tanks             |                                     | 1.658.400 €         |
| TCM material              |                                     | 12.922.765 €        |
| Contingency               | 20%                                 | 3.375.592 €         |
| <b>TOTAL</b>              |                                     | <b>20.253.553 €</b> |

Additionally, as this case represents a large plant, the O&M annual cost is represented as 1.5% of the investment cost. Finally, the calculation and analysis of the cash flows is carried out following the methodology previously described, to conclude the figures of merit of the economic modelling. The figures of merit of this scenario are shown in Table 8 below.

*Table 8 – Energy Balance Inputs.*

| Inputs from the Energy Balance              |             |
|---|-------------|
| Levelized cost of Storage (for electricity) | 0.14 €/kWh  |
| Levelized cost of Storage (for heat)        | 0.05 €/kWh  |
| Internal Return Rate (IRR)                  | 7,45 %      |
| Net Present Value (NPV)                     | 1.276.403 € |

These conclusions provide a summary of the application of the economic model developed within RESTORE. A more detailed economic analysis, including the application of the model to the remaining RESTORE virtual use cases, as well as the evaluation of different scenarios and sensitivity analyses, will be presented in *Deliverable 1.9 – Report on the Assessment of Socio-Economic Sustainability (V2 Final)*. This report will include the calculation of the various figures of merit and a comprehensive discussion of the results.

## 5. Conclusion

This document (D5.2) reported about the final version of the component models developed for RESTORE, embedded in the dedicated model library RESTORE\_Lib, and described the process level of the overall RESTORE model, based on the TCES and on the rORC technologies, presenting also the economic model used for the RESTORE system.

The work reported was carried out in task T5.2 (RESTORE techno-economic modelling of the integrated system, M9-M47), led by CENER, initiated in task T5.1 (Modelling of individual components of the overall RESTORE system, M7-M30), led by SIMTECH. Task T5.2 developed the project required models for system level simulation (yearly performance simulations, covering off-design performance of cycles and constraints due to transient behaviour) and for techno-economic analysis. The RESTORE sub-systems, with TCES and rORC technologies, were modelled using the customized RESTORE\_Lib model library created in T5.1, enhanced with additional economic parameters, that are cost-correlations of the investment costs for the components.

The component models of the final version of the RESTORE\_Lib, specifically developed for the RESTORE project, were described in detail. The model equations and variables representing individual models were presented, with the icons representing the individual units.

An extensive and detailed economic model has been implemented. The model allows to the user to modify a broad number of details and considerations for the economic analysis. In addition, the model receives inputs from the technical simulations and the investment cost from the CAPEX calculation submodule. As result, the module shows to the user results such as the Net Present Value, the Internal Return Rate, the Investment Cost and the Levelized Cost of Storage for Heat and Electricity.

An example of the economic modelling used in the RESTORE Use-Case was presented in this report, and a more detailed economic analysis, including the application of the model to the remaining RESTORE virtual use cases, as well as the evaluation of different scenarios and sensitivity analyses, the calculation of the various figures of merit, and a comprehensive discussion of the results, will be presented in the RESTORE deliverable D1.9 (*Report on the Assessment of Socio-Economic Sustainability (V2 Final)*).

The results presented in this deliverable provided important inputs to the development of the WP5 tasks T5.3 (Web-Platform Adaptation for RESTORE Dynamic and Techno-economic Modelling to represent the Use-Cases" (see [18]), T5.4 (Implementation, Optimization, Management & Validation of RESTORE Use-Cases using the Simulation Web Platform) and its sub-tasks T5.4.1-T5.4.6 (see [11], [12], [13], [14], [15], [16], [17]); as well as task T5.5 "Replication Strategy via Stakeholders additional Cases" (see [10]), having influence on its further development, for the exploitation phase of the project.

## 6. References

- [1] European Commission, Horizon 2020 Project RESTORE “Renewable Energy based seasonal Storage Technology in Order to Raise Environmental sustainability of DHC”, <https://cordis.europa.eu/project/id/101036766>, Grant Agreement GA Nr.101036766, August 20<sup>th</sup>, 2021.
- [2] SIMTECH GmbH, “The Process Simulation Environment IPSEpro”, <https://www.simtechnology.com/cms/ipsepro/process-simulation-and-heat-balance-software>, © 2023 SimTech GmbH.
- [3] SIMTECH GmbH, “IPSE GO: The Future of Simulation”, <https://about.ipsego.app/>, © 2023 SimTech GmbH.
- [4] European Commission, RESTORE Horizon 2020 Project GA Nr.101036766. Deliverable D1.1 - Report on Requirements and Specifications of the Overall Concept, ed.: F. Cabello (CENER), March 2023.
- [5] European Commission, RESTORE Horizon 2020 Project GA Nr.101036766. Deliverable D1.4 - Specifications of RESTORE Use-Cases and Models, ed.: F. Dargam (SIMTECH), E. Perz (SIMTECH), September 30<sup>th</sup>, 2023.
- [6] European Commission, RESTORE Horizon 2020 Project GA Nr.101036766. Deliverable D2.3 - Report on TCES Task D2.3 - Small-scale (1-2kW) TCES reactor tested and optimized, ed.: S. Denner (TU Wien), January 2024.
- [7] European Commission, RESTORE Horizon 2020 Project GA Nr.101036766. Deliverable D2.4 - Design report of the 30 kW/150 kWh TCES reactor, ed.: G. Wedl (TU WIEN), L. Schmieder (TU WIEN), F. Winter (TU WIEN), January 2024.
- [8] European Commission, RESTORE Horizon 2020 Project GA Nr.101036766. Deliverable D2.5 – Dedicated models for the reactor simulation, ed.: A. Werner (TU Wien), L. Schmieder (TU Wien), January 2024.
- [9] European Commission, RESTORE Horizon 2020 Project GA Nr.101036766. Deliverable D3.1 - Numerical model for HPORC systems optimization and application to different Test Cases, ed.: M. Astolfi (POLIMI), D. Alfani (POLIMI), October 2023.
- [10] European Commission, RESTORE Horizon 2020 Project GA Nr.101036766. Deliverable D5.10 - RESTORE Replication Strategy V1, ed.: F. Dargam (SIMTECH), E. Perz (SIMTECH), November 2023.
- [11] European Commission, RESTORE Horizon 2020 Project GA Nr.101036766. WP5-Task T5.4 RESTORE Use-Cases Modelling Summary, ed.: F. Dargam (SIMTECH), E. Perz (SIMTECH), S. Bergmann (SIMTECH), July 31<sup>st</sup>, 2025. DOI [10.5281/zenodo.16949460](https://doi.org/10.5281/zenodo.16949460).
- [12] European Commission, RESTORE Horizon 2020 Project GA Nr.101036766. Deliverable D5.4 - Implementation and validation of RESTORE Use-Case I: Residential / Industrial DH with Biomass and Solar Collectors, ed.: R. Pérez Santana (AALBORG CSP), M. Herrador Moreno (AALBORG CSP), July 31<sup>st</sup>, 2025. DOI [10.5281/zenodo.16962322](https://doi.org/10.5281/zenodo.16962322).
- [13] European Commission, RESTORE Horizon 2020 Project GA Nr.101036766. Deliverable D5.5 - Implementation and validation of RESTORE Use-Case II: Integration of different heat sources in DH of

Cement Industry, ed.: Rubén Garayoa Falcón (CENER), July 31<sup>st</sup>, 2025. DOI [10.5281/zenodo.16962505](https://doi.org/10.5281/zenodo.16962505).

[14] European Commission, RESTORE Horizon 2020 Project GA Nr.101036766. Deliverable D5.6 - Implementation and validation of RESTORE Use-Case III: Integration of different heat sources in DH of Paper Mills Industry, ed.: A. Werner (TU-WIEN), C. Gruber (ANDRITZ), July 31<sup>st</sup>, 2025. DOI [10.5281/zenodo.16962631](https://doi.org/10.5281/zenodo.16962631).

[15] European Commission, RESTORE Horizon 2020 Project GA Nr.101036766. Deliverable D5.7 - Implementation and validation of RESTORE Use-Case IV: Integration of different heat sources in DH of Steel-working industry, ed.: Mattia Piazza (TUR), S. Bergmann (SIMTECH), July 31<sup>st</sup>, 2025. DOI [10.5281/zenodo.16962969](https://doi.org/10.5281/zenodo.16962969).

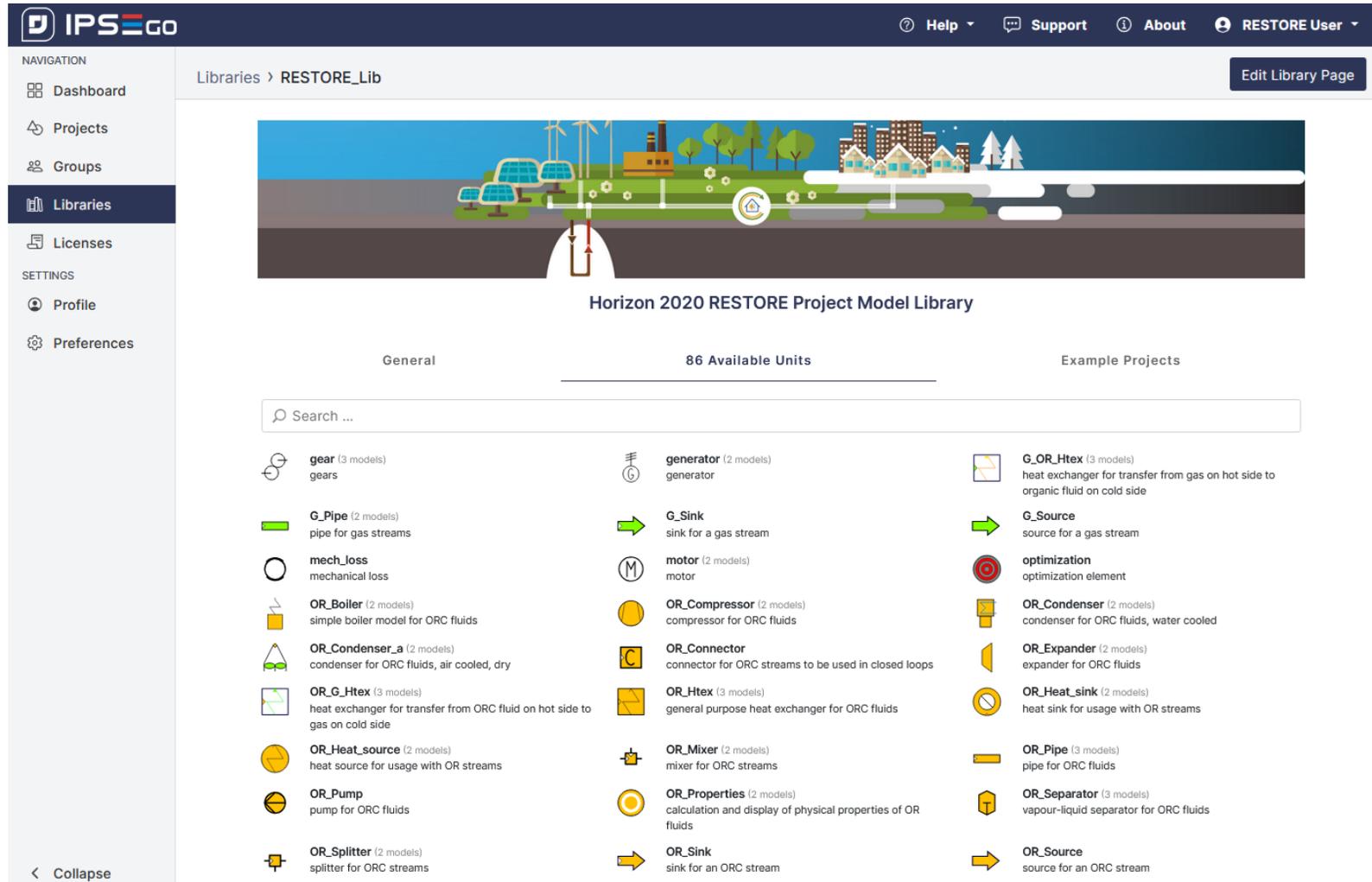
[16] European Commission, RESTORE Horizon 2020 Project GA Nr.101036766. Deliverable D5.8 - Implementation and validation of RESTORE Use-Case V: District heating with Geothermal Technology, ed.: Mattia Piazza (TUR), S. Bergmann (SIMTECH), July 31<sup>st</sup>, 2025. DOI [10.5281/zenodo.16963091](https://doi.org/10.5281/zenodo.16963091).

[17] European Commission, RESTORE Horizon 2020 Project GA Nr.101036766. Deliverable D5.9 - Implementation and validation of RESTORE Use-Case VI: Small-scale DHC network of Politecnico di Milano campus, ed.: Marco Astolfi (POLIMI), Dario Alfani (POLIMI), July 31<sup>st</sup>, 2025. DOI [10.5281/zenodo.16963299](https://doi.org/10.5281/zenodo.16963299).

[18] European Commission, RESTORE Horizon 2020 Project GA Nr.101036766. D5.3 Web-Platform adaptation for RESTORE dynamic and techno-economic modelling to represent different Use-Cases, ed.: E. Perz (SIMTECH), S. Kysela (SIMTECH), M. Perz (SIMTECH), January 31<sup>st</sup>, 2025. DOI [10.5281/zenodo.15341166](https://doi.org/10.5281/zenodo.15341166).

# Annex I. RESTORE Virtual Tool powered by IPSE GO

Landing Page of the RESTORE Virtual Tool powered by IPSE GO, showing the RESTORE\_Lib Available Units.



The screenshot displays the IPSE GO RESTORE\_Lib interface. The top navigation bar includes the IPSE GO logo, a user profile for 'RESTORE User', and links for Help, Support, and About. The left sidebar contains navigation options: Dashboard, Projects, Groups, Libraries (selected), Licenses, Profile, and Preferences. The main content area is titled 'Libraries > RESTORE\_Lib' and features a banner for the 'Horizon 2020 RESTORE Project Model Library'. Below the banner, there are three tabs: 'General', '86 Available Units' (selected), and 'Example Projects'. A search bar is located above a grid of 24 unit icons, each with a name, model count, and description. The units are organized into three columns.

| General   | 86 Available Units  | Example Projects  |
|---|---|---|
|  <b>gear</b> (3 models)<br>gears   |  <b>generator</b> (2 models)<br>generator   |  <b>G_OR_Htex</b> (3 models)<br>heat exchanger for transfer from gas on hot side to organic fluid on cold side |
|  <b>G_Pipe</b> (2 models)<br>pipe for gas streams   |  <b>G_Sink</b><br>sink for a gas stream  |  <b>G_Source</b><br>source for a gas stream   |
|  <b>mech_loss</b><br>mechanical loss   |  <b>motor</b> (2 models)<br>motor   |  <b>optimization</b><br>optimization element   |
|  <b>OR_Boiler</b> (2 models)<br>simple boiler model for ORC fluids   |  <b>OR_Compressor</b> (2 models)<br>compressor for ORC fluids                                   |  <b>OR_Condenser</b> (2 models)<br>condenser for ORC fluids, water cooled                                    |
|  <b>OR_Condenser_a</b> (2 models)<br>condenser for ORC fluids, air cooled, dry                             |  <b>OR_Connector</b><br>connector for ORC streams to be used in closed loops                    |  <b>OR_Expander</b> (2 models)<br>expander for ORC fluids  |
|  <b>OR_G_Htex</b> (3 models)<br>heat exchanger for transfer from ORC fluid on hot side to gas on cold side |  <b>OR_Htex</b> (3 models)<br>general purpose heat exchanger for ORC fluids                     |  <b>OR_Heat_sink</b> (2 models)<br>heat sink for usage with OR streams                                       |
|  <b>OR_Heat_source</b> (2 models)<br>heat source for usage with OR streams                                 |  <b>OR_Mixer</b> (2 models)<br>mixer for ORC streams  |  <b>OR_Pipe</b> (3 models)<br>pipe for ORC fluids  |
|  <b>OR_Pump</b><br>pump for ORC fluids   |  <b>OR_Properties</b> (2 models)<br>calculation and display of physical properties of OR fluids |  <b>OR_Separator</b> (3 models)<br>vapour-liquid separator for ORC fluids                                    |
|  <b>OR_Splitter</b> (2 models)<br>splitter for ORC streams   |  <b>OR_Sink</b><br>sink for an ORC stream   |  <b>OR_Source</b><br>source for an ORC stream  |

# Annex II. RESTORE Reference Design Model

RESTORE Reference Design Model used as a starting point for the setting up of the Use-cases.

The screenshot displays the IPSECO software interface for a RESTORE rORC with TCES simulation. The main workspace shows a detailed schematic diagram of the system, including components like generators, motors, compressors, condensers, expanders, pumps, valves, and storage units. The diagram is annotated with various parameters and flow directions. Below the schematic, there are several data tables, including one for 'IP Charging' and another for 'IP Discharging', which list various system parameters and their values.

The Object Manager on the right side of the interface lists the following objects:

| Class            | Object              |
|------------------|---------------------|
| generator        | generator001        |
| motor            | motor001            |
| motor            | motor002            |
| OR_Compressor    | OR_Compressor001    |
| OR_Condenser     | OR_Condenser001     |
| OR_Connector     | OR_Connector001     |
| OR_Connector     | OR_Connector002     |
| OR_Expander      | OR_Expander001      |
| OR_Htex          | OR_Htex001_C        |
| OR_Htex          | OR_Htex001_D        |
| OR_Pump          | OR_Pump001          |
| OR_Valve         | OR_Valve001         |
| OR_wall_htex     | OR_wall_htex001     |
| OR_Xprescription | OR_Xprescription001 |
| OR_Xprescription | OR_Xprescription002 |
| OR_Xprescription | OR_Xprescription003 |
| OR_Xprescription | OR_Xprescription006 |
| OR_Xprescription | OR_Xprescription007 |
| OR_Xprescription | OR_Xprescription008 |
| OR_Xprescription | OR_Xprescription101 |
| OR_Xprescription | OR_Xprescription102 |

At the bottom of the interface, a status bar indicates "No issues detected" and "IPSE GO v1.11.3 - RESTORE\_Lib".

# Annex III. RESTORE\_Lib Components

RESTORE\_Lib Component Models (as of July 2025). The model library was produced using IPSEpro-MDK and is used in IPSE GO.

|   |  |   |  |   |  |
|---|--|---|--|---|--|
|  <b>gear</b> (3 models)<br>gears   |  <b>generator</b> (2 models)<br>generator   |  <b>G_OR_Htex</b> (3 models)<br>heat exchanger for transfer from gas on hot side to organic fluid on cold side |  <b>TCM_Sink</b><br>sink for a TCM stream   |  <b>TCM_Source</b><br>source for a TCM stream  |  <b>TCM_Splitter</b><br>splitter for TCM streams  |
|  <b>G_Pipe</b> (2 models)<br>pipe for gas streams  |  <b>G_Sink</b><br>sink for a gas stream   |  <b>G_Source</b><br>source for a gas stream  |  <b>TCM_T_Htex</b><br>heat exchanger for transfer from TCM fluid on hot side to thermofluid on cold side      |  <b>TCM_Valve</b><br>valve for TCM stream  |  <b>TCM_W_Separator</b><br>separator for water from TCM stream  |
|  <b>mech_loss</b><br>mechanical loss   |  <b>motor</b> (2 models)<br>motor   |  <b>optimization</b><br>optimization element   |  <b>T_Connector</b><br>connector for heat transfer fluids to be used in closed loops                          |  <b>T_Heat_sink</b><br>heat sink for heat transfer fluids  |  <b>T_Heat_source</b><br>heat source for heat transfer fluids   |
|  <b>OR_Boiler</b> (2 models)<br>simple boiler model for ORC fluids   |  <b>OR_Compressor</b> (2 models)<br>compressor for ORC fluids   |  <b>OR_Condenser</b> (2 models)<br>condenser for ORC fluids, water cooled                                      |  <b>T_Htex</b> (3 models)<br>general purpose heat exchanger for heat transfer fluids                          |  <b>T_Mixer</b> (2 models)<br>mixer for heat transfer fluid streams  |  <b>T_OR_Htex</b> (3 models)<br>heat exchanger for transfer from thermofluid on hot side to ORC fluids on cold side |
|  <b>OR_Condenser_a</b> (2 models)<br>condenser for ORC fluids, air cooled, dry   |  <b>OR_Connector</b><br>connector for ORC streams to be used in closed loops  |  <b>OR_Expander</b> (2 models)<br>expander for ORC fluids  |  <b>T_Pipe</b> (3 models)<br>pipe for heat transfer fluids  |  <b>T_Pump</b><br>pump for heat transfer fluids  |  <b>T_Splitter</b> (2 models)<br>splitter for heat transfer fluid streams   |
|  <b>OR_G_Htex</b> (3 models)<br>heat exchanger for transfer from ORC fluid on hot side to gas on cold side   |  <b>OR_Htex</b> (3 models)<br>general purpose heat exchanger for ORC fluids   |  <b>OR_Heat_sink</b> (2 models)<br>heat sink for usage with OR streams   |  <b>T_Sink</b><br>sink for a heat transfer fluid stream   |  <b>T_Source</b><br>source for a heat transfer fluids  |  <b>T_TCM_Htex</b><br>heat exchanger for transfer from thermofluid on hot side to TCM fluid on cold side            |
|  <b>OR_Heat_source</b> (2 models)<br>heat source for usage with OR streams   |  <b>OR_Mixer</b> (2 models)<br>mixer for ORC streams  |  <b>OR_Pipe</b> (3 models)<br>pipe for ORC fluids  |  <b>T_W_Htex</b> (3 models)<br>heat exchanger for transfer from thermofluid on hot side to water on cold side |  <b>T_wall_htex</b><br>heat exchanger with wall transferring heat from thermool (T) to another side              |  <b>W_Compressor</b><br>compressor for steam  |
|  <b>OR_Pump</b><br>pump for ORC fluids   |  <b>OR_Properties</b> (2 models)<br>calculation and display of physical properties of OR fluids   |  <b>OR_Separator</b> (3 models)<br>vapour-liquid separator for ORC fluids                                      |  <b>W_Connector</b><br>connector for closed loops   |  <b>W_Heat_sink</b><br>heat sink for water streams   |  <b>W_Heat_source</b><br>heat source for water streams  |
|  <b>OR_Splitter</b> (2 models)<br>splitter for ORC streams   |  <b>OR_Sink</b><br>sink for an ORC stream   |  <b>OR_Source</b><br>source for an ORC stream  |  <b>W_Mixer</b> (2 models)<br>mixer for water streams   |  <b>W_OR_Htex</b> (3 models)<br>heat exchanger for transfer from water on hot side to OR fluid on cold side      |  <b>W_Pipe</b> (3 models)<br>pipe for water   |
|  <b>OR_Turbine</b> (4 models)<br>turbine for ORC fluids  |  <b>OR_T_Htex</b> (3 models)<br>heat exchanger for transfer from ORC fluid on hot side to thermofluid on cold side  |  <b>OR_Valve</b><br>valve for ORC fluid  |  <b>W_Pump</b><br>pump for water   |  <b>W_Sink</b><br>sink for a water stream   |  <b>W_Source</b><br>source for a water stream  |
|  <b>OR_W_Htex</b> (3 models)<br>heat exchanger for transfer from ORC fluid on hot side to water on cold side   |  <b>OR_Xprescription</b> (3 models)<br>prescription/calculation of vapor quality of an ORC fluid  |  <b>OR_wall_htex</b><br>heat exchanger with wall transferring heat from organic fluid (OR) to another side     |  <b>W_Splitter</b> (2 models)<br>splitter for water streams   |  <b>W_T_Htex</b> (3 models)<br>heat exchanger for transfer from water on hot side to thermofluids on cold side |  <b>W_Valve</b><br>valve for water  |
|  <b>RESTORE_EA</b><br>component model supplying economic analysis information  |  <b>TCM_CD_Transformer</b><br>transformer to combine charging and discharging operations of TCM   |  <b>TCM_Connector</b> (2 models)<br>connector for TCM to be used in closed loops                             |  <b>W_Xprescription</b><br>prescription/calculation of steam quality  |  <b>electricity_meter</b> (3 models)<br>electrical power consumption or production meter                       |  <b>flow_meter</b> (3 models)<br>flow meter   |
|  <b>TCM_Heat_sink</b><br>heat sink for TCM   |  <b>TCM_Heat_source</b><br>heat source for TCM  |  <b>TCM_Htex</b><br>heat exchanger for transfer from TCM fluid on hot side to TCM fluid on cold side         |  <b>free_var</b><br>free variable   |  <b>heat_meter</b> (3 models)<br>heat consumption or production meter  |  <b>time_counter</b> (2 models)<br>time counter used for measuring and displaying time intervals                  |
|  <b>TCM_Mixer</b><br>mixer for TCM streams   |  <b>TCM_Pipe</b> (2 models)<br>pipe for thermochemical material (TCM)   |  <b>TCM_Pump</b><br>pump for TCM fluids  |  <b>wall_OR_htex</b><br>heat exchanger with wall transferring heat from wall to organic fluid (OR)          |  <b>wall_T_htex</b><br>heat exchanger with wall transferring heat from wall to thermool (T)                    |  |
|  <b>TCM_Reactor_Charging</b><br>Reactor for charging step of TCM. High temperature side transferring heat to the reactor is optional. Different heat delivering working fluids (OR_ or T_) can be connected. |  <b>TCM_Reactor_Discharging</b><br>Reactor for discharging step of TCM. Low temperature side receiving heat from the reactor is optional. Different heat receiving working fluids (OR_ or T_) can be connected. |  <b>TCM_Separator</b><br>separator for TCM stream  |  |   |  |