Module Interface Specification for Solar Water Heating Systems Incorporating Phase Change Material

Brooks MacLachlan and Spencer Smith

May 4, 2018

1 Symbols, Abbreviations and Acronyms

See SRS Documentation at https://github.com/smiths/swhs

Contents

2 Introduction

The following document details the Module Interface Specifications for the implemented modules in a program simulating a Solar Water Heating System with Phase Change Material. It is intended to ease navigation through the program for design and maintenance purposes.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at https://github.com/smiths/swhs.

The specification is given in terms of functions, rather than sequences. For instance, the predicted temperature of the water is given as a function of time ($\mathbb{R} \to \mathbb{R}$), not as a sequence ($\mathbb{R}^n$). This approach is more straightforward for the specification, but in the implementation stage, it will likely be necessary to introduce a sequence, assuming that a numerical solver is used for the system of ODEs.

3 Notation

The structure of the MIS for modules comes from **?**, with the addition that template modules have been adapted from **?**. The mathematical notation comes from Chapter 3 of **?**. For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | ... | c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by SWHS.

Data Type	Notation	Description
character	char	a single symbol or digit
integer	$\mathbb{Z}$	a number without a fractional component in (-∞, ∞)
natural number	$\mathbb{N}$	a number without a fractional component in [1, ∞)
real	$\mathbb{R}$	any number in (-∞, ∞)

The specification of SWHS uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, SWHS uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

4 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	**Level 2**
Hardware-Hiding	
Behaviour-Hiding	Input Parameters Output Format Output Verification Temperature ODEs Energy Equations Control Module Specification Parameters Module
Software Decision	Sequence Data Structure ODE Solver Plotting

Table 1: Module Hierarchy

5 MIS of Control Module

5.1 Module

main

5.2 Uses

Param (Section **??**), Temperature (Section **??**), Solver (Section **??**), Energy (Section **??**), verify_output (Section **??**), plot (Section **??**), output (Section **??**)

5.3 Syntax

5.3.1 Exported Access Programs

Name	In	Out	Exceptions
main	-	-	-

5.4 Semantics

5.4.1 State Variables

None

5.4.2 Access Routine Semantics

main():

- transition: Modify the state of Param module and the environment variables for the Plot and Output modules by following these steps

Get (filenameIn: string) and (filenameOut: string) from user

load_params(filenameIn)

#Find temperature function $(T_W^{\text{Solid}}, T_W^{\text{Melting}}, T_W^{\text{Liquid}}, T_P^{\text{Solid}}, T_P^{\text{Melting}}, T_P^{\text{Liquid}})$*, and energy* (Q_P) *and times of transition between solid, melting and liquid phases* $(t_{\text{melt}}^{\text{init}}, t_{\text{melt}}^{\text{final}})$

$t_{\text{melt}}^{\text{init}}, [T_W^{\text{Solid}}, T_P^{\text{Solid}}]^T := \text{solve}(\text{ODE\_SolidPCM}, 0.0, [T_{\text{init}}, T_{\text{init}}]^T, \text{event\_StartMelt}, t_{\text{final}})$

$t_{\text{melt}}^{\text{final}}, [T_W^{\text{Melting}}, T_P^{\text{Melting}}, Q_p]^T := \text{solve}(\text{ODE\_MeltingPCM}, t_{\text{melt}}^{\text{init}}, [T_W^{\text{Solid}}(t_{\text{melt}}^{\text{init}}), T_P^{\text{Solid}}(t_{\text{melt}}^{\text{init}}), 0.0]^T,$ event_EndMelt, $t_{\text{final}})$

$[T_W^{\text{Liquid}}, T_P^{\text{Liquid}}]^T := \text{solveNoE}(\text{ODE\_LiquidPCM}, t_{\text{melt}}^{\text{final}}, [T_W^{\text{Melting}}(t_{\text{melt}}^{\text{final}}), T_P^{\text{Melting}}(t_{\text{melt}}^{\text{final}})]^T, t_{\text{final}})$

#Combine temperatures for $0 \leq t \leq t_{final}$

$T_W(t) = (0 \leq t < t_{\text{melt}}^{\text{init}} \Rightarrow T_W^{\text{Solid}} | t_{\text{melt}}^{\text{init}} \leq t < t_{\text{melt}}^{\text{final}} \Rightarrow T_W^{\text{Melting}} | t_{\text{melt}}^{\text{final}} \leq t \leq t_{\text{final}} \Rightarrow T_W^{\text{Liquid}})$

$T_P(t) = (0 \leq t < t_{\text{melt}}^{\text{init}} \Rightarrow T_P^{\text{Solid}} | t_{\text{melt}}^{\text{init}} \leq t < t_{\text{melt}}^{\text{final}} \Rightarrow T_P^{\text{Melting}} | t_{\text{melt}}^{\text{final}} \leq t \leq t_{\text{final}} \Rightarrow T_P^{\text{Liquid}})$

#Energy values $(E_W(t), E_P(t))$ *for* $0 \leq t \leq t_{\text{final}}$

$E_W(t) = (0 \leq t < t_{\text{melt}}^{\text{init}} \Rightarrow \text{energyWater}(T_W^{\text{Solid}}) | t_{\text{melt}}^{\text{init}} \leq t < t_{\text{melt}}^{\text{final}} \Rightarrow \text{energyWater}(T_W^{\text{Melting}}) | t_{\text{melt}}^{\text{final}} \leq t \leq t_{\text{final}} \Rightarrow \text{energyWater}(T_W^{\text{Liquid}}))$

$E_P(t) = (0 \leq t < t_{\text{melt}}^{\text{init}} \Rightarrow \text{energySolidPCM}(T_P^{\text{Solid}}) | t_{\text{melt}}^{\text{init}} \leq t < t_{\text{melt}}^{\text{final}} \Rightarrow \text{energyMeltingPCM}(Q_P) | t_{\text{melt}}^{\text{final}} \leq t \leq t_{\text{final}} \Rightarrow \text{energyLiquidPCM}(T_P^{\text{Liquid}}))$

#Output calculated values to a file and to a plot. Verify calculated values obey conservation of energy.

verify_output(T_w, T_p, E_w, E_p, t_{final})

plot(T_w, T_p, E_w, E_p, t_{final})

output(filenameOut, T_w, T_p, E_w, E_p, t_{final})

6 MIS of Input Parameters Module

The secrets of this module are the data structure for input parameters, how the values are input and how the values are verified. The load and verify secrets are isolated to their own access programs.

6.1 Module

Param

6.2 Uses

SpecParam (Section **??**)

6.3 Syntax

Name	**In**	**Out**	**Exceptions**
load_params	string	-	FileError
verify_params	-	-	badLength, badDiam, badPCMVolume, badPCMAndTankVol, badPCMArea, badPCMDensity, badMeltTemp, badCoilAndInitTemp, badCoilTemp, badPCMHeatCapSolid, badPCMHeatCapLiquid, badHeatFusion, badCoilArea, badWaterDensity, badWaterHeatCap, badCoilCoeff, badPCMCoeff, badInitTemp, badFinalTime, badInitAndMeltTemp
L	-	$\mathbb{R}$	
D	-	$\mathbb{R}$	
V_P	-	$\mathbb{R}$	
A_P	-	$\mathbb{R}$	
...	...	...	
m_W^{noPCM}	-	$\mathbb{R}$	
τ_W^{noPCM}	-	$\mathbb{R}$	

Should verify_params have a boolean output?

6.4 Semantics

6.4.1 Environment Variables

inputFile: sequence of string *#f[i] is the ith string in the text file f*

6.4.2 State Variables

From R1
L: $\mathbb{R}$
D: $\mathbb{R}$
V_P: $\mathbb{R}$
A_P: $\mathbb{R}$
ρ_P : $\mathbb{R}$
T^P_{melt}: $\mathbb{R}$
C^S_P: $\mathbb{R}$
C^L_P: $\mathbb{R}$
H_f: $\mathbb{R}$
A_C: $\mathbb{R}$
T_C: $\mathbb{R}$
ρ_W: $\mathbb{R}$
C_W: $\mathbb{R}$
h_C: $\mathbb{R}$
h_P: $\mathbb{R}$
T_{init}: $\mathbb{R}$
t_{step}: $\mathbb{R}$
t_{final}: $\mathbb{R}$
$AbsTol$: $\mathbb{R}$
$RelTol$: $\mathbb{R}$
$ConsTol$: $\mathbb{R}$

From R2
V_{tank}: $\mathbb{R}$
m_W: $\mathbb{R}$
m_P: $\mathbb{R}$

From R3
τ_W: $\mathbb{R}$
η: $\mathbb{R}$
τ^S_P: $\mathbb{R}$
τ^L_P: $\mathbb{R}$

To Support IM4
$E_{P\text{melt}}^{\text{init}}$: $\mathbb{R}$
$E_{P\text{melt}}^{\text{all}}$: $\mathbb{R}$

To Support Testing
m_W^{noPCM}: $\mathbb{R}$
τ_W^{noPCM}: $\mathbb{R}$

6.4.3 Assumptions

- load_params will be called before the values of any state variables will be accessed.
- The file contains the string equivalents of the numeric values for each input parameter in order, each on a new line. The order is the same as in the table in R1 of the SRS. Any comments in the input file should be denoted with a '#' symbol.

6.4.4 Access Routine Semantics

Param.L:

- output: $out := L$
- exception: none

Param.D:

- output: $out := D$
- exception: none

...

Param.m_W^{noPCM}:

- output: $out := m_W^{\text{noPCM}}$
- exception: none

Param.τ_W^{noPCM}:

- output: $out := \tau_W^{\text{noPCM}}$
- exception: none

load_params(s):

- transition: The filename s is first associated with the file f. inputFile is used to modify the state variables using the following procedural specification:

  1. Read data sequentially from inputFile to populate the state variables from R1 (L to $ConsTol$).
  2. Calculate the derived quantities (all other state variables) as follows:
     - $V_{\text{tank}} := \pi \times L \times (\frac{D}{2})^2$
     - $m_W := \rho_w(V_t - V_p)$
     - $m_P := \rho_p V_p$
     - $\tau_W := \frac{m_w C_w}{A_c h_c}$
     - $\eta := \frac{h_p A_p}{h_c A_c}$
     - $\tau_P^S := \frac{m_p C_{ps}}{h_p A_p}$
     - $\tau_P^L := \frac{m_p C_{pl}}{h_p A_p}$
     - $E_{P\text{melt}}^{\text{init}} := C_{ps} m_p (T_{\text{melt}} - T_{\text{init}})$
     - $E_{P\text{melt}}^{\text{all}} := H_f m_p$
     - $m_W^{\text{noPCM}} := \rho_w V_t$
     - $\tau_W^{\text{noPCM}} := \frac{m_W^{\text{noPCM}} C_w}{h_c A_c}$
  3. verify_params()

- exception: exc := a file name s cannot be found OR the format of inputFile is incorrect $\Rightarrow$ FileError

verify_params():

- out: out := none
- exception: exc :=

$\neg(L > 0)$	$\Rightarrow$ badLength
$\neg(L_{\min} \leq L \leq L_{\max})$	$\Rightarrow$ warnLength
$\neg(D > 0)$	$\Rightarrow$ badDiam
$\neg(\frac{D}{L}_{\min} \leq \frac{D}{L} \leq \frac{D}{L}_{\max})$	$\Rightarrow$ warnDiam
$\neg(V_P > 0)$	$\Rightarrow$ badPCMVolume
$\neg(V_P \geq \text{minfract} \cdot V_{\text{tank}}(D, L))$	$\Rightarrow$ warnPCMVol
$\neg(V_P < V_{\text{tank}}(D, L))$	$\Rightarrow$ badPCMAndTankVol
$\neg(A_P > 0)$	$\Rightarrow$ badPCMArea
$\neg(V_P \leq A_P \leq \frac{2}{h_{\min}} V_P)$	$\Rightarrow$ warnVolArea
$\neg(\rho_P > 0)$	$\Rightarrow$ badPCMDensity
$\neg(\rho_P^{\min} < \rho_P < \rho_P^{\max})$	$\Rightarrow$ warnPCMDensity

etc. See Appendix (Section **??**) for the complete list of exceptions and associated error messages.

6.5 Considerations

The value of each state variable can be accessed through its name (getter). An access program is available for each state variable. There are no setters for the state variables, since the values will be set and checked by load params and not changed for the life of the program.

7 MIS of Temperature ODEs Module

7.1 Module

Temperature

7.2 Uses

Param (Section ??)

7.3 Syntax

7.3.1 Exported Access Programs

Name	In	Out	Exceptions
ODE_SolidPCM	–	$(\mathbb{R}^3 \to \mathbb{R})^2$	-
ODE_MeltingPCM	–	$(\mathbb{R}^4 \to \mathbb{R})^3$	-
ODE_LiquidPCM	–	$(\mathbb{R}^3 \to \mathbb{R})^2$	-
event_StartMelt	–	$\mathbb{R}^2 \to \mathbb{R}$	-
event_EndMelt	–	$\mathbb{R}^3 \to \mathbb{R}$	-

7.4 Semantics

7.4.1 State Variables

none

7.4.2 Assumptions

none

7.4.3 Access Routine Semantics

ODE_SolidPCM():

- output: $out := \frac{d}{dt}\begin{bmatrix} T_W \\ T_P \end{bmatrix} = \begin{bmatrix} \frac{1}{\tau_W}[(T_C - T_W(t)) + \eta(T_P(t) - T_W(t))] \\ \frac{1}{\tau_P^S}(T_W(t) - T_P(t)) \end{bmatrix}$
- exception: none

ODE_MeltingPCM():

- output: $out := \dfrac{d}{dt}\begin{bmatrix} T_W \\ T_P \\ Q_P \end{bmatrix} = \begin{bmatrix} \frac{1}{\tau_W}[(T_C - T_W(t)) + \eta(T_P(t) - T_W(t))] \\ 0 \\ h_P A_P (T_W(t) - T^P_{\text{melt}}) \end{bmatrix}$
- exception: none

ODE_LiquidPCM():

- output: $out := \dfrac{d}{dt}\begin{bmatrix} T_W \\ T_P \end{bmatrix} = \begin{bmatrix} \frac{1}{\tau_W}[(T_C - T_W(t)) + \eta(T_P(t) - T_W(t))] \\ \frac{1}{\tau^L_P}(T_W(t) - T_P(t)) \end{bmatrix}$
- exception: none

event_StartMelt():

- output: $out := g([T_W, T_P]^T) = T^P_{\text{melt}} - T_P$
- exception: none

event_EndMelt():

- output: $out := g([T_W, T_P, Q_P]^T) = 1 - \phi$, where $\phi = \frac{Q_P}{E^{\text{all}}_{P\text{melt}}}$
- exception: none

8 MIS of ODE Solver Module

#Bold font is used to indicate variables that are a sequence type

8.1 Module

Solver($n : \mathbb{N}$) *#n is the length of the sequences*

8.2 Uses

None

8.3 Syntax

8.3.1 Exported Access Programs

Name	In	Out	Except.
solve	$\mathbf{f} : (\mathbb{R}^{n+1} \to \mathbb{R})^n, t_0 : \mathbb{R}, \mathbf{y}_0 : \mathbb{R}^n, g : \mathbb{R}^n \to \mathbb{R}, t_{\text{fin}} : \mathbb{R}$	$t_1 : \mathbb{R}, \mathbf{y} : (\mathbb{R} \to \mathbb{R})^n$	ODE_ERR, NO_EVENT
solveNoE	$\mathbf{f} : (\mathbb{R}^{n+1} \to \mathbb{R})^n, t_0 : \mathbb{R}, \mathbf{y}_0 : \mathbb{R}^n, t_{\text{fin}} : \mathbb{R}$	$\mathbf{y} : (\mathbb{R} \to \mathbb{R})^n$	ODE_ERR

8.4 Semantics

8.4.1 State Variables

None

8.4.2 Access Routine Semantics

#Solving $\frac{d}{dt}\mathbf{y} = \mathbf{f}(t, \mathbf{y}(t))$

solve($\mathbf{f}, t_0, \mathbf{y}_0, g, t_{\text{fin}}$):

- output: $out := t_1, \mathbf{y}(t)$ where

$$\mathbf{y}(t) = \mathbf{y}_0 + \int_{t_0}^{t} \mathbf{f}(s, \mathbf{y}(s))ds$$

  with t_1 determined by the first time where $g(\mathbf{y}(t_1)) = 0$. $\mathbf{y}(t)$ is calculated from $t = t_0$ to $t = t_1$.

- exception: $exc :=$ ($\neg(\exists t : \mathbb{R} | t_0 \leq t \leq t_{\text{fin}} : g(\mathbf{y}(t)) = 0) \Rightarrow$ NO_EVENT | ODE Solver Fails $\Rightarrow$ ODE_ERR)

solveNoE($\mathbf{f}$, t_0, $\mathbf{y}_0$, t_{fin}):

- output: $out := \mathbf{y}(t)$ where

$$\mathbf{y}(t) = \mathbf{y}_0 + \int_{t_0}^{t_{\text{fin}}} \mathbf{f}(s, \mathbf{y}(s))ds$$

  $y(t)$ is calculated from $t = t_0$ to $t = t_{\text{fin}}$.

- exception: $exc :=$ (ODE Solver Fails $\Rightarrow$ ODE_ERR)

9 MIS of Energy Module

9.1 Module

Energy

9.2 Uses

Param (Section **??**)

9.3 Syntax

9.3.1 External Access Programs

Name	In	Out	Exceptions
energyWater	$\mathbb{R} \to \mathbb{R}$	$\mathbb{R} \to \mathbb{R}$	-
energySolidPCM	$\mathbb{R} \to \mathbb{R}$	$\mathbb{R} \to \mathbb{R}$	-
energyMeltingPCM	$\mathbb{R} \to \mathbb{R}$	$\mathbb{R} \to \mathbb{R}$	-
energyLiquidPCM	$\mathbb{R} \to \mathbb{R}$	$\mathbb{R} \to \mathbb{R}$	-

9.4 Semantics

9.4.1 State Variables

None

9.4.2 Assumptions

None

9.4.3 Access Routine Semantics

energyWater(T_W):

- output: $out := C_W m_W (T_W - T_{\text{init}})$
- exception: none

energySolidPCM(T_P):

- output: $out := C_P^S m_P (T_P - T_{\text{init}})$
- exception: none

energyMeltingPCM(Q_P):

- output: $out := E^{\text{init}}_{P\text{melt}} + Q_P$
- exception: none

energyLiquidPCM(T_P):

- output: $out := E^{\text{init}}_{P\text{melt}} + H_f m_p + C^L_P m_P (T_P(t) - T^P_{\text{melt}})$
- exception: none

10 MIS of Output Verification Module

10.1 Module

verify_output

10.2 Uses

Param (Section **??**)

10.3 Syntax

10.3.1 Exported Constant

ADMIS_ER = 1×10^{-6}

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
verify_output	$T_W(t) : \mathbb{R} \to \mathbb{R}, T_P(t) : \mathbb{R} \to \mathbb{R}, E_W(t) : \mathbb{R} \to \mathbb{R}, E_P(t) : \mathbb{R} \to \mathbb{R}, t_{\text{final}} : \mathbb{R}$	-	EWAT_NOT_CONSERVE, EPCM_NOT_CONSERVE

10.4 Semantics

10.4.1 State Variables

None

10.4.2 Assumptions

All of the fields of the input parameters structure have been assigned a value.

10.4.3 Access Routine Semantics

verify_output($T_W, T_P, E_W, E_P, t_{\text{final}}$):

- exception: exc := (

$(\forall t | 0 \leq t \leq t_{\text{final}} : \text{relErr}(E_W, \int_0^t h_C A_C (T_C - T_W(t))dt - \int_0^t h_P A_P (T_W(t) - T_P(t))dt) < \text{ADMIS\_ER}) \Rightarrow \text{EWAT\_NOT\_CONSERVE}$

|

$(\forall t | 0 \leq t \leq t_{\text{final}} : \text{relErr}(E_P, \int_0^t h_P A_P (T_W(t) - T_P(t))dt) < \text{ADMIS\_ER}) \Rightarrow \text{EPCM\_NOT\_CONSERVE}$

)

10.4.4 Local Functions

relErr: $\mathbb{R} \times \mathbb{R} \to \mathbb{R}$
relErr$(t, e) \equiv \frac{|t-e|}{|t|}$

11 MIS of Plotting Module

11.1 Module

plot

11.2 Uses

N/A

11.3 Syntax

11.3.1 Exported Access Programs

Name	In	Out	Exceptions
plot	$T_W(t) : \mathbb{R} \to \mathbb{R}, T_P(t) : \mathbb{R} \to \mathbb{R}, E_W(t) : \mathbb{R} \to \mathbb{R}, E_P(t) : \mathbb{R} \to \mathbb{R}$, $t_{\text{final}} : \mathbb{R}$	-	-

11.4 Semantics

11.4.1 State Variables

None

11.4.2 Environment Variables

win: 2D sequence of pixels displayed on the screen

11.4.3 Assumptions

None

11.4.4 Access Routine Semantics

plot(T_w, T_p, E_w, E_p, t_{final}):

- transition: Modify win to display a plot where the vertical axis is time and one horizontal axis is temperature and the other horizontal axis is energy. The time should run from 0 to t_{final}
- exception: none

12 MIS of Output Module

12.1 Module

output

12.2 Uses

Param (Section **??**)

12.3 Syntax

12.3.1 Exported Constants

max_width: integer

12.3.2 Exported Access Program

Name	In	Out	Exceptions
output	fname: string, $T_W(t) : \mathbb{R} \to \mathbb{R}, T_P(t) : \mathbb{R} \to \mathbb{R}, E_W(t) : \mathbb{R} \to \mathbb{R}, E_P(t) : \mathbb{R} \to \mathbb{R}$, $t_{\text{final}} : \mathbb{R}$	-	-

12.4 Semantics

12.4.1 State Variables

None

12.4.2 Environment Variables

file: A text file

12.4.3 Access Routine Semantics

output(fname, T_w, T_p, E_w, E_p, t_{final}):

- transition: Write to environment variable named fname the following: the input parameters from Param, and the calculated values T_w, T_p, E_w, E_p from times 0 to t_{final}. The functions will be output as sequences in this file. The spacing between points in the sequence should be selected so that the heating behaviour is captured in the data.
- exception: none

13 MIS of Specification Parameters

The secrets of this module is the value of the specification parameters.

13.1 Module

SpecParam

13.2 Uses

N/A

13.3 Syntax

13.3.1 Exported Constants

From Table 2 in SRS
$L_{\text{min}} := 0.1$
$L_{\text{max}} := 50$
$\frac{D}{L}_{\text{min}} := 0.002$
$\frac{D}{L}_{\text{max}} := 200$
minfrac := 10^{-6}
$h_{\text{min}} := 0.001$
$\rho_P^{\text{min}} := 500$
$\rho_P^{\text{max}} := 20000$
$C_{P\text{min}}^S := 100$
$C_{P\text{max}}^S := 4000$
$C_{P\text{min}}^L := 100$
$C_{P\text{max}}^L := 5000$
$A_C^{\text{max}} := \pi(\frac{D}{2})^2$
$\rho_W^{\text{min}} := 950$
$\rho_W^{\text{max}} := 1000$
$C_W^{\text{min}} := 4170$
$C_W^{\text{max}} := 4210$
$h_C^{\text{min}} := 10$
$h_C^{\text{max}} := 10000$
$h_P^{\text{min}} := 10$
$h_P^{\text{max}} := 10000$
$t_{\text{final}}^{\text{max}} := 86400$

[$A_C^{\max}$ shouldn't be in this table of constants —SS]

13.4 Semantics

N/A

14 Appendix

Table 2: Possible Exceptions

Message ID	**Error Message**
badLength	Error: Tank length must be > 0
badDiam	Error: Tank diameter must be > 0
badPCMVolume	Error: PCM volume must be > 0
badPCMAndTankVol	Error: PCM volume must be < tank volume
badPCMArea	Error: PCM area must be > 0
badPCMDensity	Error: rho_p must be > 0
badMeltTemp	Error: Tmelt must be > 0 and < Tc
badCoilAndInitTemp	Error: Tc must be > Tinit
badCoilTemp	Error: Tc must be > 0 and < 100
badPCMHeatCapSolid	Error: C_ps must be > 0
badPCMHeatCapLiquid	Error: C_pl must be > 0
badHeatFusion	Error: Hf must be > 0
badCoilArea	Error: Ac must be > 0
badWaterDensity	Error: rho_w must be > 0
badWaterHeatCap	Error: C_w must be > 0
badCoilCoeff	Error: hc must be > 0
badPCMCoeff	Error: hp must be > 0
badInitTemp	Error: Tinit must be > 0 and < 100
badFinalTime	Error: tfinal must be > 0
badInitAndMeltTemp	Error: Tinit must be < Tmelt
ODE_ACCURACY	*reltol* and *abstol* were not satisfied by the ODE solver for a given solution step.
ODE_BAD_INPUT	Invalid input to ODE solver
ODE_MAXSTEP	ODE solver took $MaxStep$ steps and did not find solution
warnLength	Warning: It is recommended that 0.1 <= L <= 50
warnDiam	Warning: It is recommended that 0.002 <= D/L <= 200
warnPCMVol	Warning: It is recommended that Vp be >= 0.0001% of Vt
warnVolArea	Warning: It is recommended that Vp <= Ap <= (2/0.001) * Vp
warnPCMDensity	Warning: It is recommended that 500 < rho_p < 20000
warnPCMHeatCapSolid	Warning: It is recommended that 100 < C_ps < 4000

warnPCMHeatCapLiquid	Warning: It is recommended that 100 < C_pl < 5000
warnCoilArea	Warning: It is recommended that Ac <= pi * (D/2) ∧ 2
warnWaterDensity	Warning: It is recommended that 950 < rho_w <= 1000
warnWaterHeatCap	Warning: It is recommended that 4170 < C_w < 4210
warnCoilCoeff	Warning: It is recommended that 10 < hc < 10000
warnPCMCoeff	Warning: It is recommended that 10 < hp < 10000
warnFinalTime	Warning: It is recommended that 0 < tfinal < 86400
warnWaterError	Warning: There is greater than x% relative error between the energy in the water output and the expected output based on the law of conservation of energy. (Where x is the value of $ConsTol$)
warnPCMError	Warning: There is greater than x% relative error between the energy in the PCM output and the expected output based on the law of conservation of energy. (Where x is the value of $ConsTol$)