

Module Interface Specification for Sun Catcher

Sharon (Yu-Shiuan) Wu

November 25, 2019

1 Revision History

Date	Version	Notes
2019/11/25	1.0	First Version
Date 2	1.1	Notes

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at <https://github.com/sharyuwu/optimum-tilt-of-solar-panels/blob/master/docs/SRS/SRS.pdf>

[Also add any additional symbols, abbreviations or acronyms —SS]

Contents

1	Revision History	i
2	Symbols, Abbreviations and Acronyms	ii
3	Introduction	1
4	Notation	1
5	Module Decomposition	1
6	MIS of Control Module	3
6.1	Module	3
6.2	Uses	3
6.3	Syntax	3
6.3.1	Exported Constants	3
6.3.2	Exported Access Programs	3
6.4	Semantics	3
6.4.1	State Variables	3
6.4.2	Environment Variables	3
6.4.3	Assumptions	3
6.4.4	Access Routine Semantics	3
7	Sun Catcher Type Module	4
7.1	Module	4
7.2	Uses	4
7.3	Exported Types	5
7.4	Syntax	5
7.4.1	Exported Constants	5
7.4.2	Exported Access Programs	5
7.5	Semantics	5
7.5.1	State Variables	5
7.5.2	Environment Variables	5
7.5.3	Assumptions	5
8	Day Duration ADT Module	5
8.1	Template Module	5
8.2	Uses	6
8.3	Exported Types	6
8.4	Syntax	6
8.4.1	Exported Constants	6
8.4.2	Exported Access Programs	6
8.5	Semantics	6

8.5.1	State Variables	6
8.5.2	Environment Variables	6
8.5.3	Assumptions	6
8.5.4	Access Routine Semantics	6
8.5.5	Local Functions	7
9	MIS of Input Parameters Module	7
9.1	Module	7
9.2	Uses	7
9.3	Syntax	8
9.3.1	Exported Constants	8
9.3.2	Exported Access Programs	8
9.4	Semantics	8
9.4.1	State Variables	8
9.4.2	Environment Variables	8
9.4.3	Assumptions	8
9.4.4	Access Routine Semantics	8
9.4.5	Local Functions	9
10	MIS of Input Verification Module	9
10.1	Module	9
10.2	Uses	9
10.3	Syntax	10
10.3.1	Exported Constants	10
10.3.2	Exported Access Programs	10
10.4	Semantics	10
10.4.1	State Variables	10
10.4.2	Environment Variables	10
10.4.3	Assumptions	10
10.4.4	Access Routine Semantics	10
10.4.5	Local Functions	10
11	MIS of Output Parameters Module	10
11.1	Module	10
11.2	Uses	10
11.3	Syntax	11
11.3.1	Exported Constants	11
11.3.2	Exported Access Programs	11
11.4	Semantics	11
11.4.1	State Variables	11
11.4.2	Environment Variables	11
11.4.3	Assumptions	11
11.4.4	Access Routine Semantics	11

11.4.5 Local Functions	12
12 MIS of Table-layout Module	12
12.1 Module	12
12.2 Uses	12
12.3 Syntax	12
12.3.1 Exported Constants	12
12.3.2 Exported Access Programs	12
12.4 Semantics	12
12.4.1 State Variables	12
12.4.2 Environment Variables	12
12.4.3 Assumptions	13
12.4.4 Access Routine Semantics	13
12.4.5 Local Functions	13
13 MIS of Optimum Tilt Angle Module	13
13.1 Module	13
13.2 Uses	13
13.3 Syntax	14
13.3.1 Exported Constants	14
13.3.2 Exported Access Programs	14
13.4 Semantics	14
13.4.1 State Variables	14
13.4.2 Environment Variables	14
13.4.3 Assumptions	14
13.4.4 Access Routine Semantics	14
13.4.5 Local Functions	15
14 MIS of Solar Energy Absorption Module	15
14.1 Module	15
14.2 Uses	15
14.3 Syntax	16
14.3.1 Exported Constants	16
14.3.2 Exported Access Programs	16
14.4 Semantics	16
14.4.1 State Variables	16
14.4.2 Environment Variables	16
14.4.3 Assumptions	16
14.4.4 Access Routine Semantics	16
14.4.5 Local Functions	17

15 MIS of Sun Intensity Equation Module	17
15.1 Module	17
15.2 Uses	17
15.3 Syntax	17
15.3.1 Exported Constants	17
15.3.2 Exported Access Programs	17
15.4 Semantics	17
15.4.1 State Variables	17
15.4.2 Environment Variables	17
15.4.3 Assumptions	18
15.4.4 Access Routine Semantics	18
15.4.5 Local Functions	18
16 MIS of Zenith Angle Equation Module	18
16.1 Module	18
16.2 Uses	18
16.3 Syntax	18
16.3.1 Exported Constants	18
16.3.2 Exported Access Programs	18
16.4 Semantics	19
16.4.1 State Variables	19
16.4.2 Environment Variables	19
16.4.3 Assumptions	19
16.4.4 Access Routine Semantics	19
16.4.5 Local Functions	19
17 MIS of Read File Module	20
17.1 Module	20
17.2 Uses	20
17.3 Syntax	20
17.3.1 Exported Constants	20
17.3.2 Exported Access Programs	20
17.4 Semantics	20
17.4.1 State Variables	20
17.4.2 Environment Variables	20
17.4.3 Assumptions	20
17.4.4 Access Routine Semantics	20
17.4.5 Local Functions	21
18 Appendix	23

3 Introduction

The following document details the Module Interface Specifications for Sun Catcher[Fill in your project name and description —SS]

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at <https://github.com/sharyuwu/optimum-tilt-of-solar-panels>. [provide the url for your repo —SS]

4 Notation

[You should describe your notation. You can use what is below as a starting point. —SS]

The structure of the MIS for modules comes from Hoffman and Strooper (1995), with the addition that template modules have been adapted from Ghezzi et al. (2003). The mathematical notation comes from Chapter 3 of Hoffman and Strooper (1995). For instance, the symbol $:=$ is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | \dots | c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by Sun Catcher.

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

The specification of Sun Catcher 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, Sun Catcher 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.

5 Module Decomposition

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

Level 1	Level 2
Hardware-Hiding Module	
Behaviour-Hiding Module	Control Module Input Parameters Module Input Verification Module Output Parameters Module Read File Module Solar Energy Absorption Module Optimum Tilt Angle Module Sun Intensity Equation Module Zenith Angle Equation Module
Software Decision Module	Day Duration ADT Module Table-layout Module Sun Catcher Type Module

Table 1: Module Hierarchy

6 MIS of Control Module

6.1 Module

Control

6.2 Uses

InputPara⁹

InputVer¹⁰

OutputPara¹¹

TiltAng¹³

Energy¹⁴

SunInten¹⁵

ZenithAng¹⁶

ReadFile¹⁷

6.3 Syntax

6.3.1 Exported Constants

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
mian	-	-	-

6.4 Semantics

6.4.1 State Variables

dayS: DayT

dayE: DayT

6.4.2 Environment Variables

6.4.3 Assumptions

6.4.4 Access Routine Semantics

main ():

- transition: Implement the InputPara and the environment variables for Output by following steps.

Get (Φ_P : DegreeT, P_{A_h} : real, P_{A_w} : real, $year_{Start}$: natural number, $month_{Start}$: natural number, day_{Start} : natural number, $year_{End}$: natural number, $month_{End}$: natural

number, day_{End} : natural number) and (filename: string) from users' input.

$dayS := \langle day_{Start}, month_{Start}, year_{Start} \rangle$
 $dayE := \langle day_{End}, month_{End}, year_{End} \rangle$

load_anale_data (filename)
InputPara.init ()

Verify the input values using Input Verification Module
verified (Φ_P , P_{A_h} , P_{A_w} , dayS, dayE)

Get the zenith angle between dayS and dayE
 $\theta_{S_{date}} = \text{getzen} ()$

Get the optimum tilt angle using $\theta_{S_{date}}$
setilt ($\theta_{S_{date}}$)
 $\theta_T = \text{getilt} ()$

Get the estimated solar absorption using Optimum Tilt Angle Module
setenergy (P_{A_w} , P_{A_h} , getmaxInten (), $\theta_{S_{date}}$)
 $P_E = \text{getavenergy} ()$

Output P_E and θ_T to the Output paramaters Module
addresult (θ_T , P_E)

Display the output values on the screen using the Table-layout Module
setable (getresult ())
display ()

7 Sun Catcher Type Module

7.1 Module

SunCatTy

7.2 Uses

N/A

7.3 Exported Types

DegreeT = \mathbb{R}

DayT = a tuple of (d: \mathbb{N} , m: \mathbb{N} , y: \mathbb{N})

AnalemmaT = a tuple of (x: \mathbb{R} ,y: \mathbb{R} , z: \mathbb{R})

7.4 Syntax

7.4.1 Exported Constants

N/A

7.4.2 Exported Access Programs

N/A

7.5 Semantics

N/A

7.5.1 State Variables

N/A

7.5.2 Environment Variables

N/A

7.5.3 Assumptions

For DayT the day will always lie between 1 and 31, the month will always lie between 1 and 12, and year will always be greater than 1.

8 Day Duration ADT Module

8.1 Template Module

DayDurTy

8.2 Uses

SunCatTy⁷

8.3 Exported Types

DayDurT = ?

8.4 Syntax

8.4.1 Exported Constants

8.4.2 Exported Access Programs

Name	In	Out	Exceptions
new DayDurT	a tuple of (DayT, DayT)	DayDurT	invalid_argument
getdurlist	-	a sequence of integer	-

8.5 Semantics

8.5.1 State Variables

dayS : DayT

dayE : DayT

daduraL: a sequence of integer

dateL: a sequence of DayT

8.5.2 Environment Variables

N/A

8.5.3 Assumptions

In getdiff (day1, day2), assume that day 2 is always greater than day 1.

8.5.4 Access Routine Semantics

new DayDurT (s,e):

- transition: dayS, dayE, periheD.d, periheD.m := s, e, 21, 12;

dateL := || (s = e \Rightarrow \langle e \rangle | otherwise \Rightarrow \langle s, setlist (getnext ()) \rangle)

- output: $\text{out} := \text{self}$
- exception: $\text{exc} := (\text{s} > \text{e} \Rightarrow \text{invalid_argument})$

$\text{getdurlist} ()$:

- transition: $\text{daduraL} := \parallel (d: \text{DayT} \mid d \in \text{dateL} \bullet [\text{Countdiff} (\text{perihelion} (d), d)]$
- output: $\text{out} := \text{daduraL}$
- exception:

8.5.5 Local Functions

$\text{setlist} : \text{DayT} \rightarrow \text{a sequence of DayT}$

$\text{setlist} (\text{day}) := \parallel (\text{day} = \text{dayE} \Rightarrow \langle \text{dayE} \rangle \mid \text{day} < \text{dayE} \Rightarrow [\text{day}, \text{setlist} (\text{getnext} (\text{day}, \text{dayE}))])$

$\text{getnext} : \text{DayT} \times \text{DayT} \rightarrow \text{DayT}$

$\text{getnext} (d1, d2) := (d1 < d2 \Rightarrow \text{The next day of } d1 \text{ according to the calendar} \mid \text{otherwise} \Rightarrow \text{dateL} [0])$

$\text{perihelion} : \text{DayT} \rightarrow \text{DayT}$

$\text{perihelion} (\text{day}) := (\text{day.m} = 12 \wedge \text{day.d} \geq 21 \Rightarrow \text{day.y} \mid \text{otherwise} \Rightarrow \text{day.y} - 1)$

$\text{Countdiff} : \text{DayT} \times \text{DayT} \rightarrow \text{integer}$

$\text{Countdiff} (\text{day1}, \text{day2}) := + (\text{day1} = \text{day2} \Rightarrow 0 \mid \text{otherwise} \Rightarrow \text{Countdiff} (\text{getnext} (\text{day1}, \text{day2}), \text{day2}) + 1)$

9 MIS of Input Parameters Module

9.1 Module

InputPara

9.2 Uses

HardH

9.3 Syntax

9.3.1 Exported Constants

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
init	-	-	Key_error
getla	-	DegreeT	-
getph	-	real	-
getpw	-	real	-
getds	-	DayT	-
getde	-	DayT	-

9.4 Semantics

9.4.1 State Variables

latitude: DegreeT
dayS: DayT
dayE: DayT
panH: real
panW real

9.4.2 Environment Variables

key: Input variables from keyboard.

9.4.3 Assumptions

When users chick the submit bottom from system interface, InputPara.init () implement.

9.4.4 Access Routine Semantics

init ():

- transition: Get the values from users' input.
latitude, panH, panW, dayS, dayE := HardH.la, HardH.pn, HardH.pw, (HardH.ds, HardH.ms, HardH.ys), (HardH.de, HardH.me, HardH.ye)
- output:
- exception: if the data type of captured values do not match with the parameters' data type \Rightarrow Key_error

getla ():

- transition:

- output: latitude
- exception:

getph ():

- transition:
- output: panH
- exception:

getpw ():

- transition:
- output: panW
- exception:

getds ():

- transition:
- output: dayS
- exception:

getde ():

- transition:
- output: dayE
- exception:

9.4.5 Local Functions

10 MIS of Input Verification Module

10.1 Module

InputVer

10.2 Uses

InputPara9

10.3 Syntax

10.3.1 Exported Constants

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
verified	DegreeT, real, real, DayT, DayT,	-	invaild_lan, invaild_pan, invaild_day

10.4 Semantics

10.4.1 State Variables

10.4.2 Environment Variables

10.4.3 Assumptions

The input values is input from InputPara

10.4.4 Access Routine Semantics

verified (lan, ph, pw, ds, de):

- transition:
- output:
- exception: $\text{exc} := (\text{lan} > 90 \wedge \text{lan} < -90 \Rightarrow \text{invaild_lan} \mid \text{ph} < 0 \wedge \text{pw} < 0 \Rightarrow \text{invaild_pan} \mid \text{if ds and de is a invalid date in Gregorian calendar} \Rightarrow \text{invaild_day})$

10.4.5 Local Functions

11 MIS of Output Parameters Module

11.1 Module

OutputPara

11.2 Uses

Table¹²

11.3 Syntax

11.3.1 Exported Constants

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
init	-	-	-
addresult	DegreeT, real	-	-
getresult	-	a sequence of a tuple of (DegreeT, real, integer)	-

11.4 Semantics

11.4.1 State Variables

resultL: a sequence of a tuple of (t: DegreeT, e: real, diff: integer)

11.4.2 Environment Variables

11.4.3 Assumptions

OutputPara.init () always implement before any export access programs in output parameters module. OutputPara.init () will only be called once.

11.4.4 Access Routine Semantics

init ():

- transition: resultL := { }
- output:
- exception:

addresult (angle, energy):

- transition: resultL := || (|resultL| = 0 \Rightarrow [< angle, energy, 100 >] | angle \notin resultL.t \Rightarrow [< angle, energy, getdiff (resultL.e, energy) >])
- output:
- exception:

getresult ():

- transition:
- output: out := resultL
- exception:

11.4.5 Local Functions

getdiff : $\text{real} \times \text{real} \rightarrow \text{integer}$
getdiff (e1, e2) = $\frac{e_2}{e_1} \times 100$

12 MIS of Table-layout Module

12.1 Module

Table

12.2 Uses

OutputPara11
HardH

12.3 Syntax

12.3.1 Exported Constants

hRow1: "Tilt Angle" hRow2: "Energy Absorption" hRow3: "Energy Competition"

12.3.2 Exported Access Programs

Name	In	Out	Exceptions
setable	a sequece of a tuple of (DegreeT, real, integer)	-	-
display	-	-	-

12.4 Semantics

12.4.1 State Variables

resultL: a sequence of a tulple of (t: DegreeT, e: real, diff: integer)

12.4.2 Environment Variables

Output variables will display on the screen.

12.4.3 Assumptions

12.4.4 Access Routine Semantics

setable (result):

- transition: resultL := result
- output:
- exception:

display ():

- transition:
- output: out := display a table that shows the result on the screen.

The text file has the following format, where the first row following heading: hRow1, denotes the result of optimal tilt angle; the second row following heading, hRow2, denotes the result of estimating energy absorption; and the third row following heading, hRow3, denotes the result of different energy absorption between resultL.e [0] and resultL.e [1.. | resultL|] followed by a symbol %

Tilt Angle	resultL.t [0]	...	resultL.t [resultL]
Energy Absorption	resultL.e [0]	...	resultL.e [resultL]
Energy Competition	resultL.diff [0] %	...	resultL.diff [resultL] %

- exception:

12.4.5 Local Functions

13 MIS of Optimum Tilt Angle Module

[Use labels for cross-referencing —SS]

13.1 Module

TitleAng [Short name for the module —SS]

13.2 Uses

SunInten¹⁵

ZenithAng¹⁶

13.3 Syntax

13.3.1 Exported Constants

$I_S := 1.35$

13.3.2 Exported Access Programs

Name	In	Out	Exceptions
setilt	a sequence of DegreeT	-	-
getilt	-	DegreeT	-
getmaxInten	-	real	out_empty
[accessProg —SS]	-	-	-

13.4 Semantics

13.4.1 State Variables

sunIn: real

zenithL: a sequence of DegreeT

maxsunIn: a tuple of (i: real, d: DegreeT)

[Not all modules will have state variables. State variables give the module a memory. —SS]

13.4.2 Environment Variables

N/A [This section is not necessary for all modules. Its purpose is to capture when the module has external interaction with the environment, such as for a device driver, screen interface, keyboard, file, etc. —SS]

13.4.3 Assumptions

[Try to minimize assumptions and anticipate programmer errors via exceptions, but for practical purposes assumptions are sometimes appropriate. —SS]

13.4.4 Access Routine Semantics

[accessProg —SS] setilt(zen):

- transition: $\text{sunIn}, \text{zenithL}, \text{maxsunIn} := \text{SunInten.sum}(I_S, \text{zen}), \text{zen}, \langle 0, 0 \rangle$
- output:
- exception:

[accessProg —SS] getilt ():

- transition: $\text{maxsunIn} := (j: \text{integer} \mid j \in [0.. |\text{zenithL}|]) \bullet j = 0 \Rightarrow \langle \text{SunInten.single}(\text{sunIn}, \text{zenithL} [0]), \text{zenithL} [0] \rangle \mid \text{ifMax}(\text{maxsunIn.i}, \text{SunInten.single}(\text{sunIn}, \text{zenithL} [j])) \neq \text{maxsunIn.i} \Rightarrow \langle \text{SunInten.single}(\text{sunIn}, \text{zenithL} [j]), \text{zenithL} [j] \rangle$
- output: $\text{out} := \text{maxsunIn.d}$
- exception:

getmaxInten ():

- transition:
- output: $\text{SunInten.single}(\text{sunIn}, \text{maxsunIn.d})$
- exception: $\text{exp} := \text{maxsunIn.i} = 0 \wedge \text{maxsunIn.d} \Rightarrow \text{out_empty}$

[A module without environment variables or state variables is unlikely to have a state transition. In this case a state transition can only occur if the module is changing the state of another module. —SS]

[Modules rarely have both a transition and an output. In most cases you will have one or the other. —SS]

13.4.5 Local Functions

ifMax: $\text{real} \times \text{real} \rightarrow \text{real}$

$\text{ifMax}(x, y) = (x \geq y \Rightarrow x \mid \text{otherwise} \Rightarrow y)$

[As appropriate —SS] [These functions are for the purpose of specification. They are not necessarily something that is going to be implemented explicitly. Even if they are implemented, they are not exported; they only have local scope. —SS]

14 MIS of Solar Energy Absorption Module

14.1 Module

Energy

14.2 Uses

InputPara⁹

ZenithAng¹⁶

TiltAng¹³

14.3 Syntax

14.3.1 Exported Constants

14.3.2 Exported Access Programs

Name	In	Out	Exceptions
setenergy	real, real, DegreeT, a sequence of DegreeT	-	-
getavenergy	-	real	-

14.4 Semantics

14.4.1 State Variables

energyL: a sequence of real

14.4.2 Environment Variables

N/A

14.4.3 Assumptions

ZenithAngle has been fully implement before implement Energy.init ().
Energy.init () is called first before any other exported access programs.

14.4.4 Access Routine Semantics

setenergy (pw, ph, maxinten, zen):

- transition: $\text{energyL} := \text{getenergy} (\text{getsunIn} (\text{zen}, \text{maxinten}), \text{pw}, \text{ph})$
- output:
- exception:

getavenergy ():

- transition:
- output: $\text{out} := + (i: \text{integer} \mid i \in [0.. |\text{energyL}|] \bullet \text{energyL}[i]) / |\text{energyL}|$
- exception:

14.4.5 Local Functions

getsunIn: a sequence of DegreeT \rightarrow a sequence of real

getsunIn (z, maxinten) = \parallel (i: integer | i \in [0.. |z|] • [SunInten.single(maxinten, z[i])])

getenergy: a sequence of real \times real \times real \rightarrow a sequence of real

getenergy (e, pw, ph) = \parallel (i: integer | i \in [0.. |e|] • [pw \times ph \times 18.7 \times 0.75 \times e [i]])

15 MIS of Sun Intensity Equation Module

[Use labels for cross-referencing —SS]

[You can reference SRS labels, such as R??. —SS]

[It is also possible to use L^AT_EX for hyperlinks to external documents. —SS]

15.1 Module

SunInten

15.2 Uses

ZenithAngle¹⁶

15.3 Syntax

15.3.1 Exported Constants

$I_S := 1.35$

15.3.2 Exported Access Programs

Name	In	Out	Exceptions
sum	real, a sequence of DegreeT	real	sequence_empty
single	real, DegreeT	real	-

15.4 Semantics

15.4.1 State Variables

15.4.2 Environment Variables

N/A

15.4.3 Assumptions

15.4.4 Access Routine Semantics

sum (e, z):

- output: out := + (i: integer | i ∈ [0.. |z|] •

$$I_S \cdot \frac{1.00}{e} \sec(z[i])$$

- exception: exc := |z| = 0 ⇒ sequence_empty

single (e, z):

- output: out :=

$$I_S \cdot \frac{1.00}{e} \sec(z)$$

- exception:

15.4.5 Local Functions

16 MIS of Zenith Angle Equation Module

16.1 Module

ZenithAng

16.2 Uses

DayDurTy⁸

SunCatTy⁷

16.3 Syntax

16.3.1 Exported Constants

16.3.2 Exported Access Programs

Name	In	Out	Exceptions
setzen	AnaIammaT, real, DayT, DayT	-	-
getzen	-	a sequence of DegreeT	sequence_empty

16.4 Semantics

16.4.1 State Variables

decS: a sequence of DegreeT

day: DayDurTy

latitude: real

analeL: a sequence of AnalemmaT

16.4.2 Environment Variables

16.4.3 Assumptions

Read File Module will always be implemented before Zenith angle Module.

InputPara Module will always be implemented before Zenith angle Module.

ZenithAng.setzen is called before any other access program.

ZenithAng.setzen is called by the Read File Module.

16.4.4 Access Routine Semantics

setzen (a, lan, ds, de):

- transition: $\text{analeL} := \parallel (i: \text{integer} \mid i \in [0..|a|] \bullet [\langle a.x[i], a.y[i], a.z[i] \rangle])$
day := new DayDurT (ds, de)
latitude := lan
decS := getdec (day.getdurlist)
- output:
- exception:

getzen ():

- transition:
- output: $\text{out} := \parallel (i: \text{integer} \mid i \in [0..|\text{decS}|] \bullet \text{latitude} \times \text{decS}[i] \geq 0 \Rightarrow [\text{latitude} - \text{decS}[i]] \mid \text{otherwise} \Rightarrow [\text{latitude} + \text{decS}[i]])$
- exception: $|\text{decS}| = 0 \Rightarrow \text{sequence_empty}$

16.4.5 Local Functions

getdec : a sequence of integer \rightarrow a sequence of DegreeT

getdec (d) := $\parallel (i: \text{integer} \mid i \in [0..|d|] \bullet$

$$[\arcsin \frac{\text{analeL}.z[d[i]]}{\sqrt{\text{analeL}.x^2[d[i]] + \text{analeL}.y^2[\text{dayL}[i]] + \text{analeL}.z^2[d[i]]}})]$$

17 MIS of Read File Module

17.1 Module

ReadFile

17.2 Uses

ZenithAng¹⁶

17.3 Syntax

17.3.1 Exported Constants

None

17.3.2 Exported Access Programs

Name	In	Out	Exceptions
load_anale_data	string	-	-

17.4 Semantics

17.4.1 State Variables

17.4.2 Environment Variables

anale_data: File listing Analemma data

17.4.3 Assumptions

The input file will match the given specification. ReadFile.load_anale_data(s) is called by the Input Parameters Module

17.4.4 Access Routine Semantics

load_anale_data (s):

- transition: read data from the file anale_data associated with the string s. Use this data to update the state of the ZenithAng module. load_anale_data (s) will first initialize ZenithAng (ZenithAng.setzen (anale, lan, ds, de)) before populating ZenithAng with analemma data that follows the types in SunCatTy.

The text file has the following format, where x_i, y_i, z_i denotes the vector that locates Earth relative to the Sun on the day i since perihelion.

All data values in a row are separated by commas. Rows are separated by a new line. The data shown below is for a total of 366 days.

x_0, y_0, z_0
x_1, y_1, z_1
x_2, y_2, z_2
.
.
.
x_365, y_365, z_365

- output:
- exception:

17.4.5 Local Functions

References

- Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.
- Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL <http://citeseer.ist.psu.edu/428727.html>.

18 Appendix

[Extra information if required —SS]