Module Interface Specification for Sun Catcher

Author Name

November 20, 2019

1 Revision History

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at [give url —SS]

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

Contents

3 Introduction

The following document details the Module Interface Specifications for [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 . . . . [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 $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | ... | c_n \Rightarrow r_n)$.

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 (-∞, ∞)
natural number	$\mathbb{N}$	a number without a fractional component in [1, ∞)
real	$\mathbb{R}$	any number in (-∞, ∞)
degree	$\mathbb{R}$	any number in (-∞, ∞)

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 Verify Module Output Parameters Module Solar Energy Absorption Module Optimum Tilt Angle Module Sun Intensity Equation Module Zenith Angle Equation Module Sun Declination Module Days Module
Software Decision Module	Count days Module Table-layout Module Sequence Data Structure Module

Table 1: Module Hierarchy

6 Day ADT Module

6.1 Template Module

Day

6.2 Uses

N/A

6.3 Exported Types

DayT = ?

6.4 Syntax

6.4.1 Exported Constants

6.4.2 Exported Access Programs

Name	In	Out	Exceptions
initday	$\mathbb{N}$,$\mathbb{N}$,$\mathbb{N}$	DayT	-
day	-	natural number	-
month	-	natural number	-
year	-	natural number	-

6.5 Semantics

6.5.1 State Variables

d1 :real
m1 :real
y1 :real

6.5.2 Environment Variables

N/A

6.5.3 Assumptions

initday(d,m,y):

- transition: d1, m1, y1 := d,m,y
- output: out := self

day( ):

- transition:
- output: out := d1

month( ):

- transition:
- output: out := m1
- exception:

year( ):

- transition:
- output: out := y1
- exception:

7 Analemma ADT Module

7.1 Template Module

Analemma

7.2 Uses

N/A

7.3 Exported Types

AnalemmaT = ?

7.4 Syntax

7.4.1 Exported Constants

7.4.2 Exported Access Programs

Name	In	Out	Exceptions
initanale	real,real,real	AnalemmaT	-
x	-	real	-
y	-	real	-
z	-	real	-

7.5 Semantics

7.5.1 State Variables

x1 :real
y1 :real
z1 :real

7.5.2 Environment Variables

N/A

7.5.3 Assumptions

initanale(x,y,z):

- transition: x1, y1, z1 := x,y,z
- output: out := self

x( ):

- transition:
- output: out := x1

y( ):

- transition:
- output: out := y1
- exception:

z( ):

- transition:
- output: out := z1
- exception:

8 MIS of Optimum Tilt Angle Module

[Use labels for cross-referencing —SS]

8.1 Module

tile angle [Short name for the module —SS]

8.2 Uses

Sun Intensity,zenith angle

8.3 Syntax

8.3.1 Exported Constants

I_S := 1.35
$I_{S_{\text{total}}}$: real
$I_{S_{\text{total}}}$:= intenSum(I_S)
$\theta_{S_{\text{date}}}$: a sequence of degree
$\theta_{S_{\text{date}}}$:= zenith( )

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
angle	-	degree	-
[accessProg —SS]	-	-	-

8.4 Semantics

8.4.1 State Variables

max.inten: real
max.inten := intenSingle($I_{S_{\text{total}}}$, $\theta_{S_{\text{date}}}[0]$),
max.deg : degree

max.deg = $\theta_{S_{\text{date}}}[0]$

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

8.4.2 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]

8.4.3 Assumptions

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

8.4.4 Access Routine Semantics

[accessProg —SS] angle( ):

- transition:
  ($\forall$z: degree | z $\in \theta_{S_{\text{date}}}$ • max.inten = ifMax (max.inten, intenSingle($I_{S_{\text{total}}}$,z))$\Rightarrow$ nothing change | otherwise $\Rightarrow$ max.inten = intenSingle($I_{S_{\text{total}}}$,z), max.deg = z)
- output: max.deg
- exception: [if appropriate —SS]

[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]

8.4.5 Local Functions

ifMax: real $\times$ real $\rightarrow$ real
ifMax(x, y) = (x $\geq$ y $\Rightarrow$ x | y > x $\Rightarrow$ 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]

9 MIS of Solar Energy Absorption Module

9.1 Module

Energy Absorption

9.2 Uses

input parameter, solar intensity, tilt angle,zenith angle

9.3 Syntax

9.3.1 Exported Constants

$I_S := 1.35$
$I_{S_{\text{total}}}$: real
$I_{S_{\text{total}}}$:= intenSum(I_S)
$I_{S_{\text{max}}}$: real
$I_{S_{\text{max}}}$:= intenSingle(I_S, angle( ))
$\theta_{S_{\text{date}}}$: a sequence of degree
$\theta_{S_{\text{date}}}$:= zenith( )

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
energy	-	real[ ]	-

9.4 Semantics

9.4.1 State Variables

$I_{S_{\text{daily}}}$: real[ ]

9.4.2 Environment Variables

N/A

9.4.3 Assumptions

energy( ):

- transition: $I_{S_{\text{daily}}}$:= ($\forall$ x : degree | x $\in \theta_{S_{\text{date}}}$ $\bullet$ intenSingle($I_{S_{\text{max}}}$, x))
- output: out := ($\forall$ x: real | x $\in I_{S_{\text{daily}}}$ $\bullet$ $P_{A_{\text{w}}} \times P_{A_{\text{h}}} \times 18.7 \times 0.75 \times$ x)

- exception:

9.4.4 Local Functions

10 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 LaTeXfor hypperlinks to external documents. —SS]

10.1 Module

Sun Intensity

10.2 Uses

zenith angle

10.3 Syntax

10.3.1 Exported Constants

$I_S := 1.35$
$\theta_{S_{\text{date}}}$: a squence of degree
$\theta_{S_{\text{date}}}$:= zenith( )

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
intenSum	real	real	-
intenSingle	real, degree	real	-

10.4 Semantics

10.4.1 State Variables

10.4.2 Environment Variables

N/A

10.4.3 Assumptions

intenSum(i):

- output: out := $+(\forall$ d: degree | d $\in \theta_{S_{\text{date}}} \bullet I_S \cdot \frac{1.00}{i}^{sec(d)})$

- exception:

intenSingle(i, d):

- output: out $:= 1.35 \cdot \frac{1.00}{i}^{sec(d)}$
- exception:

10.4.4 Local Functions

11 MIS of Zenith Angle Equation Module

11.1 Module

Zenith angle

11.2 Uses

Date Duration Module14

11.3 Syntax

11.3.1 Exported Constants

Φ_P: degree
δ_{date}: a squence of degree
δ_{date}:= declination( )

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
zenith	-	degree[ ]	-

11.4 Semantics

11.4.1 State Variables

zenithS: a squence of degree

11.4.2 Environment Variables

N/A

11.4.3 Assumptions

[accessProg —SS]zenith( ):

- transition: zenithS := (∀ d: degree | d ∈ δ_{date} • $\Phi_P \times$ d $\geq 0 \Rightarrow \Phi_P$ - d | otherwise $\Rightarrow$ Φ_P + d)
- output: out := self
- exception:

11.4.4 Local Functions

12 MIS of Sun Declination Module

12.1 Module

Sun Declination

12.2 Uses

date duration

12.3 Syntax

12.3.1 Exported Constants

date: a squence of integer
date := dateduration( )

12.3.2 Exported Access Programs

Name	In	Out	Exceptions
intidec	-	AnalemmaT[ ]	-
declination	-	degree[ ]	-

12.4 Semantics

12.4.1 State Variables

declinationS: a squence of degree
initdec: a squence of AnalemmaT

12.4.2 Environment Variables

file: Input a file that contains a sequence of (x: real, y: real, z: real) of 366 days

12.4.3 Assumptions

intidec( ):

- transition: initdec := (∀ d: integer,∀ i: integer | d ∈ date, i ∈ [0...] • dec.z[i] := file[d].z, dec.x[i] := file[d].x,dec.y[i] := file[d].y)
- output: self
- exception:

declination( ):

- transition: declinationS := (∀: i:DayT | i ∈ intidec( ) • arcsin $\frac{i.z}{\sqrt{i.x^2+i.y^2+i.z^2}}$)
- output: self
- exception:

12.4.4 Local Functions

13 MIS of Days Module

13.1 Module

Days

13.2 Uses

Input parameter

13.3 Syntax

13.3.1 Exported Constants

end: DayT
end.d = day_{End}
end.m = $month_{\text{End}}$
end.y = $year_{\text{End}}$

13.3.2 Exported Access Programs

Name	In	Out	Exceptions
perihelion	DayT	DayT	-
dateduration	-	integer[ ]	-

(day.m = 12 ∧ day.d ≥ 21 ⇒ fixday.y = day.y | otherwise ⇒ fixday.y := day.y - 1)

13.4 Semantics

13.4.1 State Variables

days : a sequence of integer
fixday : DayT
start: DayT
start.d := day_{Start}
start.m := $month_{\text{Start}}$

start.y := $year_{\text{Start}}$

13.4.2 Environment Variables

None

13.4.3 Assumptions

perihelion(day):

- transition: fixday := (day.m = 12 $\wedge$ day.d $\geq$ 21 $\Rightarrow$ fixday.d = 21, fixday.m = 12, fixday.y = day.y | otherwise $\Rightarrow$ fixday.d = 21, fixday.m = 12, fixday.y := day.y - 1)
- output: out := fixday
- exception:

dateduration(day):

- transition: days := (negetiveD(start, end) = 1 $\Rightarrow$ countdays(perihelion(start), start)), start := adday(start)
- output: days
- exception:

13.4.4 Local Functions

negetiveD: DayT $\times$ DayT $\rightarrow$ integer
negetiveD(day1, day2) := (countdays(day1, day2) $\geq$ 0 $\Rightarrow$ 1 | otherwise $\Rightarrow$ -1)

14 MIS of Count Days Module

14.1 Module

Count days

14.2 Uses

N/A

14.3 Syntax

14.3.1 Exported Constants

14.3.2 Exported Access Programs

Name	In	Out	Exceptions
countdays	DayT, DayT	integer	-
addays	DayT	DayT	-

14.4 Semantics

14.4.1 State Variables

14.4.2 Environment Variables

None

14.4.3 Assumptions

countdays(day1, day2):

- transition:
- output: out := Count the days from the start date, day, to the end date, day, but not including the end date.
- exception:

adday(day):

- transition:
- output: out := the next day of the date, day.
- exception:

14.4.4 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.

15 Appendix

[Extra information if required —SS]