# Module Interface Specification for Program Name

Isobel Bicket

November 26, 2017

# 1 Revision History

| Date | Version | Notes |
| --- | --- | --- |
| November 26, 2017 | 1.0 | Initial draft |

# 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 **?**, 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 Program Name.

| 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 (-$\infty$, $\infty$) |

The specification of Program Name 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, Program Name 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 | |
| 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

# 6 MIS of Hardware Hiding Module

[Use labels for cross-referencing —SS]

## 6.1 Module

HardwareHiding

## 6.2 Uses

## 6.3 Syntax

### 6.3.1 Exported Access Programs

| Name | In | Out | Exceptions |
|------|-----|-----|------------|
| [accessProg —SS] | - | - | - |

## 6.4 Semantics

### 6.4.1 State Variables

### 6.4.2 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]

- output: [if appropriate —SS]

- exception: [if appropriate —SS]

# 7 MIS of Behaviour Hiding Module

[Use labels for cross-referencing —SS]

## 7.1 Module

BehaviourHiding

## 7.2 Uses

## 7.3 Syntax

### 7.3.1 Exported Access Programs

| Name | In | Out | Exceptions |
|------|----|----|------------|
| [accessProg —SS] | - | - | - |

## 7.4 Semantics

### 7.4.1 State Variables

### 7.4.2 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]

- output: [if appropriate —SS]

- exception: [if appropriate —SS]

# 8 MIS of Import csv Module

[Use labels for cross-referencing —SS]

## 8.1 Module

ImportCSV

## 8.2 Uses

## 8.3 Syntax

### 8.3.1 Exported Access Programs

| Name | In | Out | Exceptions |
|------|----|----|------------|
| [accessProg —SS] | - | - | - |

## 8.4 Semantics

### 8.4.1 State Variables

### 8.4.2 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]

- output: [if appropriate —SS]

- exception: [if appropriate —SS]

# 9 MIS of Import dm3 Module

## 9.1 Module

ImportDM3

## 9.2 Uses

- Array data structure

- Hardware hiding

- Data Spectrum Image

## 9.3 Syntax

### 9.3.1 Exported Access Programs

| Name | In | Out | Exceptions |
|------|-----|-----|------------|
| ReadDM3 | filename: string | SI: $\mathbb{R}^{X \times Y \times E}$, meta-data: dict | NO FILE, WRONG FILETYPE, NO DATA FOUND |

## 9.4 Semantics

### 9.4.1 State Variables

### 9.4.2 Environment Variables

- filedm3

### 9.4.3 Access Routine Semantics

ImportDM3():

- input:

- transition:

- output:

- exception:

# 10 MIS of Import h5 Module

## 10.1 Module

ImportH5

## 10.2 Uses

## 10.3 Syntax

### 10.3.1 Exported Access Programs

| Name | In | Out | Exceptions |
|------|-----|-----|------------|
| [accessProg —SS] | - | - | - |

## 10.4 Semantics

### 10.4.1 State Variables

### 10.4.2 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]

- output: [if appropriate —SS]

- exception: [if appropriate —SS]

# 11 MIS of Import rpl Module

[Use labels for cross-referencing —SS]

## 11.1   Module

ImportRPL

## 11.2   Uses

## 11.3   Syntax

### 11.3.1   Exported Access Programs

| Name | In | Out | Exceptions |
|---|---|---|---|
| [accessProg —SS] | - | - | - |

## 11.4   Semantics

### 11.4.1   State Variables

### 11.4.2   Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]

- output: [if appropriate —SS]

- exception: [if appropriate —SS]

# 12   MIS of Export csv Module

[Use labels for cross-referencing —SS]

## 12.1   Module

ExportCSV

## 12.2   Uses

## 12.3   Syntax

### 12.3.1   Exported Access Programs

| Name | In | Out | Exceptions |
|---|---|---|---|
| WriteCSV | - | - | - |
| FormatCSV | - | - | - |
| Verify1D | - | - | - |

## 12.4 Semantics

### 12.4.1 State Variables

### 12.4.2 Access Routine Semantics

WriteCSV():

- transition: Writes data to csv file

- output: csv file

- exception:

FormatCSV():

- transition: Formats data to prepare it to write to csv file

- output: formatted data

- exception:

Verify1D():

- transition: Verifies that the input data is of the correct format (a 1D spectrum) and has a spectral range and an intensity array of equal length

- output: formatted data

- exception:

# 13 MIS of Export h5 Module

[Use labels for cross-referencing —SS]

## 13.1 Module

ExportH5

## 13.2 Uses

## 13.3 Syntax

### 13.3.1 Exported Access Programs

| Name | In | Out | Exceptions |
|------|-----|-----|------------|
| [accessProg —SS] | - | - | - |

## 13.4   Semantics

### 13.4.1   State Variables

### 13.4.2   Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]

- output: [if appropriate —SS]

- exception: [if appropriate —SS]

# 14   MIS of Export png Module

[Use labels for cross-referencing —SS]

## 14.1   Module

ExportPNG

## 14.2   Uses

## 14.3   Syntax

### 14.3.1   Exported Access Programs

| Name | In | Out | Exceptions |
|------|----|----|-----------|
| [accessProg —SS] | - | - | - |

## 14.4   Semantics

### 14.4.1   State Variables

### 14.4.2   Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]

- output: [if appropriate —SS]

- exception: [if appropriate —SS]

# 15    MIS of Export rpl Module

[Use labels for cross-referencing —SS]

## 15.1    Module

ExportRPL

## 15.2    Uses

## 15.3    Syntax

### 15.3.1    Exported Access Programs

| Name | In | Out | Exceptions |
|------|-----|-----|------------|
| [accessProg —SS] | - | - | - |

## 15.4    Semantics

### 15.4.1    State Variables

### 15.4.2    Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]

- output: [if appropriate —SS]

- exception: [if appropriate —SS]

# 16    MIS of Data Processing Richardson-Lucy Deconvolution Module

## 16.1    Module

RLDeconvolution

## 16.2    Uses

Array Data Structure

## 16.3 Syntax

### 16.3.1 Exported Access Programs

| Name | In | Out | Exceptions |
|------|-----|------|------------|
| RLDeconvolution | S, iterations, threads | S, deconvolved SI | - |
| SIDeconvolution | - | - | - |

## 16.4 Semantics

### 16.4.1 State Variables

N/A

### 16.4.2 Access Routine Semantics

RLDeconvolution():

- input: S, S, iterations, threads

- transition:

- output: deconvolved spectrum

- exception: Divide by zero!

SIDeconvolution():

- input: SI, iterations, S, threads

- transition:

- output: Deconvolved spectrum image

- exception: divide by zero

# 17 MIS of Data Processing Normalization Module

[Use labels for cross-referencing —SS]

## 17.1 Module

Normalization

## 17.2 Uses

## 17.3 Syntax

### 17.3.1 Exported Access Programs

| Name | In | Out | Exceptions |
|------|-----|-----|-----------|
| [accessProg —SS] | - | - | - |

## 17.4 Semantics

### 17.4.1 State Variables

### 17.4.2 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]

- output: [if appropriate —SS]

- exception: [if appropriate —SS]

# 18 MIS of Data Processing Gain Correction Module

[Use labels for cross-referencing —SS]

## 18.1 Module

GainCorr

## 18.2 Uses

## 18.3 Syntax

### 18.3.1 Exported Access Programs

| Name | In | Out | Exceptions |
|------|-----|-----|-----------|
| [accessProg —SS] | - | - | - |

## 18.4 Semantics

### 18.4.1 State Variables

### 18.4.2 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]

- output: [if appropriate —SS]

- exception: [if appropriate —SS]

# 19 MIS of Data Processing Background Correction Module

[Use labels for cross-referencing —SS]

## 19.1 Module

BackgroundCorr

## 19.2 Uses

## 19.3 Syntax

### 19.3.1 Exported Access Programs

| Name | In | Out | Exceptions |
|------|-----|-----|------------|
| [accessProg —SS] | - | - | - |

## 19.4 Semantics

### 19.4.1 State Variables

### 19.4.2 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]

- output: [if appropriate —SS]

- exception: [if appropriate —SS]

# 20 MIS of Data Extraction 1D Slice Module

[Use labels for cross-referencing —SS]

## 20.1 Module

Slice1D

## 20.2 Uses

## 20.3 Syntax

### 20.3.1 Exported Access Programs

| Name | In | Out | Exceptions |
|------|-----|-----|------------|
| CreateMask | - | - | - |
| ApplyMask | - | - | - |

## 20.4 Semantics

### 20.4.1 State Variables

- Mask (2D array of booleans)

### 20.4.2 Access Routine Semantics

CreateMask():

- transition: Creation of the mask for a 2d dataset - relies on user interaction

- output:

- exception:

[should this be here, or in display? —Author]
ApplyMask():

- transition: Applies 2d mask to dataset

- output:

- exception:

# 21 MIS of Data Extraction 2D Mask Module

[Use labels for cross-referencing —SS]

## 21.1   Module

Mask2D

## 21.2   Uses

## 21.3   Syntax

### 21.3.1   Exported Access Programs

| Name | In | Out | Exceptions |
|---|---|---|---|
| [accessProg —SS] | - | - | - |

## 21.4   Semantics

### 21.4.1   State Variables

### 21.4.2   Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]

- output: [if appropriate —SS]

- exception: [if appropriate —SS]

# 22   MIS of Data Extraction 3D Mask Module

[Use labels for cross-referencing —SS]

## 22.1   Module

Mask3D

## 22.2   Uses

## 22.3   Syntax

### 22.3.1   Exported Access Programs

| Name | In | Out | Exceptions |
|---|---|---|---|
| [accessProg —SS] | - | - | - |

## 22.4 Semantics

### 22.4.1 State Variables

mask3d

### 22.4.2 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]

- output: [if appropriate —SS]

- exception: [if appropriate —SS]

# 23 MIS of Display 1D Spectrum Module

[Use labels for cross-referencing —SS]

## 23.1 Module

Disp1D

## 23.2 Uses

Data 1D Spectrum Plotting library

## 23.3 Syntax

### 23.3.1 Exported Access Programs

| Name | In | Out | Exceptions |
|------|-----|-----|-----------|
| plot | - | - | - |

## 23.4 Semantics

### 23.4.1 State Variables

### 23.4.2 Environment Variables

fig

### 23.4.3   Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]

- output: [if appropriate —SS]

- exception: [if appropriate —SS]

# 24   MIS of Display 2D Image Module

[Use labels for cross-referencing —SS]

## 24.1   Module

Disp2D

## 24.2   Uses

## 24.3   Syntax

### 24.3.1   Exported Access Programs

| Name | In | Out | Exceptions |
| --- | --- | --- | --- |
| [accessProg —SS] | - | - | - |

## 24.4   Semantics

### 24.4.1   State Variables

### 24.4.2   Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]

- output: [if appropriate —SS]

- exception: [if appropriate —SS]

# 25   MIS of Display 3D Spectrum Image Module

[Use labels for cross-referencing —SS]

## 25.1 Module

Disp3D

## 25.2 Uses

- Data

- Plotting library

- 2D image plot

- 1D spectrum plot

## 25.3 Syntax

### 25.3.1 Exported Access Programs

| Name | In | Out | Exceptions |
|------|-----|-----|------------|
| [accessProg —SS] | - | - | - |

## 25.4 Semantics

### 25.4.1 State Variables

- axis2D image

- axis1D spectrum

- axis2D mask

- axis1D contrast

- axis colourbar

- polygons

- slicer

[do polygons and slicer belong here, or in the mask2d and slice1d modules? —Author]

### 25.4.2 Environment Variables

- Plotting window displayed on screen

- Keyboard keys and mouse buttons

### 25.4.3 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]

- output: [if appropriate —SS]

- exception: [if appropriate —SS]

# 26 MIS of Data 1D Spectrum Module

[Use labels for cross-referencing —SS]

## 26.1 Module

Spectrum

## 26.2 Uses

## 26.3 Syntax

### 26.3.1 Exported Access Programs

| Name | In | Out | Exceptions |
|------|-----|-----|-----------|
| [accessProg —SS] | - | - | - |

## 26.4 Semantics

### 26.4.1 State Variables

### 26.4.2 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]

- output: [if appropriate —SS]

- exception: [if appropriate —SS]

# 27 MIS of Data 2D Image Module

[Use labels for cross-referencing —SS]

## 27.1 Module

Image

## 27.2 Uses

## 27.3 Syntax

### 27.3.1 Exported Access Programs

| Name | In | Out | Exceptions |
|---|---|---|---|
| [accessProg —SS] | - | - | - |

## 27.4 Semantics

### 27.4.1 State Variables

### 27.4.2 Access Routine Semantics

[accessProg —SS]():

- transition:

- output:

- exception:

# 28 MIS of Data 3D Spectrum Image Module

## 28.1 Template Module

SI

## 28.2 Uses

- Array Data Structure

## 28.3 Syntax

### 28.3.1 Exported Access Programs

| Routine Name | In | Out | Exceptions |
|---|---|---|---|
| init | data | SI | - |

## 28.4 Semantics

### 28.4.1 State Variables

- data: $\mathbb{R}^{X \times Y \times K}$

- Imcal: $\mathbb{R}$

- dispersion: $\mathbb{R}$

- Srange: $\mathbb{R}^K$

- ZLP: $\mathbb{Z}$

- size: $\mathbb{N}^3$

- Slabel: string

- Sunit: string

- metadata: dict

### 28.4.2 State Invariant

### 28.4.3 Assumptions

### 28.4.4 Access Routine Semantics

init

- input:

- transition: Initialize all state variables

- output:

- exception:

# 29 MIS of Array Data Structure Module

## 29.1 Module

Array

## 29.2 Uses

## 29.3 Syntax

### 29.3.1 Exported Access Programs

| Name | In | Out | Exceptions |
|------|-----|-----|------------|
| [accessProg —SS] | - | - | - |

## 29.4 Semantics

### 29.4.1 State Variables

### 29.4.2 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]

- output: [if appropriate —SS]

- exception: [if appropriate —SS]

# 30 MIS of Plotting Library Module

[Use labels for cross-referencing —SS]

## 30.1 Module

Plotting

## 30.2 Uses

## 30.3 Syntax

### 30.3.1 Exported Access Programs

| Name | In | Out | Exceptions |
|------|-----|-----|------------|
| [accessProg —SS] | - | - | - |

## 30.4   Semantics

### 30.4.1   State Variables

### 30.4.2   Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]

- output: [if appropriate —SS]

- exception: [if appropriate —SS]

# 31  Appendix

[Extra information if required —SS]