Module Interface Specification for RwaveDetection

Junwei Lin

April 8, 2025

1 Revision History

Date	Version	Notes
Mar 17, 2025	1.0	Creation

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at https://github.com/Lychee-acaca/CAS741/blob/main/docs/SRS/SRS.pdf.

Contents

1 Revision History **i**

2 Symbols, Abbreviations and Acronyms **ii**

3 Introduction **1**

4 Notation **1**

5 Module Decomposition **1**

6 MIS of Hardware-Hiding 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 . . . 4
6.4.4 Access Routine Semantics . . . 4
6.4.5 Local Functions . . . 4

7 MIS of General Digital Filter Module **5**
7.1 Module . . . 5
7.2 Uses . . . 5
7.3 Syntax . . . 5
7.3.1 Exported Constants . . . 5
7.3.2 Exported Access Programs . . . 5
7.4 Semantics . . . 5
7.4.1 State Variables . . . 5
7.4.2 Environment Variables . . . 5
7.4.3 Assumptions . . . 6
7.4.4 Access Routine Semantics . . . 6
7.4.5 Local Functions . . . 6

8 MIS of Specified IIR Filter Module **7**
8.1 Module . . . 7
8.2 Uses . . . 7
8.3 Syntax . . . 7
8.3.1 Exported Constants . . . 7
8.3.2 Exported Access Programs . . . 7

8.4 Semantics . . . 7
8.4.1 State Variables . . . 7
8.4.2 Environment Variables . . . 7
8.4.3 Assumptions . . . 8
8.4.4 Access Routine Semantics . . . 8
8.4.5 Local Functions . . . 8

9 MIS of Math Module **9**
9.1 Module . . . 9
9.2 Uses . . . 9
9.3 Syntax . . . 9
9.3.1 Exported Constants . . . 9
9.3.2 Exported Access Programs . . . 9
9.4 Semantics . . . 9
9.4.1 State Variables . . . 9
9.4.2 Environment Variables . . . 9
9.4.3 Assumptions . . . 10
9.4.4 Access Routine Semantics . . . 10
9.4.5 Local Functions . . . 10

10 MIS of Pan Tompkins Algorithm Module **11**
10.1 Module . . . 11
10.2 Uses . . . 11
10.3 Syntax . . . 11
10.3.1 Exported Constants . . . 11
10.3.2 Exported Access Programs . . . 11
10.4 Semantics . . . 11
10.4.1 State Variables . . . 11
10.4.2 Environment Variables . . . 11
10.4.3 Assumptions . . . 12
10.4.4 Access Routine Semantics . . . 12
10.4.5 Local Functions . . . 12

11 MIS of Annotated Data RMSE Module **13**
11.1 Module . . . 13
11.2 Uses . . . 13
11.3 Syntax . . . 13
11.3.1 Exported Constants . . . 13
11.3.2 Exported Access Programs . . . 13
11.4 Semantics . . . 13
11.4.1 State Variables . . . 13
11.4.2 Environment Variables . . . 13
11.4.3 Assumptions . . . 14

11.4.4 Access Routine Semantics . . . 14
11.4.5 Local Functions . . . 14

12 MIS of Rwave Detect Module **15**
12.1 Module . . . 15
12.2 Uses . . . 15
12.3 Syntax . . . 15
12.3.1 Exported Constants . . . 15
12.3.2 Exported Access Programs . . . 15
12.4 Semantics . . . 15
12.4.1 State Variables . . . 15
12.4.2 Environment Variables . . . 16
12.4.3 Assumptions . . . 16
12.4.4 Access Routine Semantics . . . 16
12.4.5 Local Functions . . . 16

13 MIS of Input Output Processing Module **17**
13.1 Module . . . 17
13.2 Uses . . . 17
13.3 Syntax . . . 17
13.3.1 Exported Constants . . . 17
13.3.2 Exported Access Programs . . . 17
13.4 Semantics . . . 17
13.4.1 State Variables . . . 17
13.4.2 Environment Variables . . . 17
13.4.3 Assumptions . . . 18
13.4.4 Access Routine Semantics . . . 18
13.4.5 Local Functions . . . 18

14 MIS of Error Handler Module **19**
14.1 Module . . . 19
14.2 Uses . . . 19
14.3 Syntax . . . 19
14.3.1 Exported Constants . . . 19
14.3.2 Exported Access Programs . . . 19
14.4 Semantics . . . 19
14.4.1 State Variables . . . 19
14.4.2 Environment Variables . . . 19
14.4.3 Assumptions . . . 19
14.4.4 Access Routine Semantics . . . 20
14.4.5 Local Functions . . . 20

15 MIS of Logger Module **21**
15.1 Module . . . 21
15.2 Uses . . . 21
15.3 Syntax . . . 21
15.3.1 Exported Constants . . . 21
15.3.2 Exported Access Programs . . . 21
15.4 Semantics . . . 21
15.4.1 State Variables . . . 21
15.4.2 Environment Variables . . . 21
15.4.3 Assumptions . . . 21
15.4.4 Access Routine Semantics . . . 22
15.4.5 Local Functions . . . 22

3 Introduction

The following document details the Module Interface Specifications for RwaveDetection.

This project focuses on reimplementing the Pan-Tompkins algorithm for R-wave detection in ECG signal processing.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at https://github.com/Lychee-acaca/CAS741.

4 Notation

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 RwaveDetection.

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 RwaveDetection uses some derived data types: sequences and strings. Sequences are lists filled with elements of the same data type, they are represented using a type with a superscript, for example, $\mathbb{R}^n$ represents a real number sequence of length n. Strings are sequences of characters. In addition, RwaveDetection 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	General Digital Filter Module Specified IIR Filter Module Math Module Pan Tompkins Algorithm Module Annotated Data RMSE Module
Software Decision Module	Rwave Detect Module Input Output Processing Module Error Handler Module Logger Module

Table 1: Module Hierarchy

6 MIS of Hardware-Hiding Module

This section outlines the formal specification of the Hardware-Hiding Module, which provides an abstraction layer for interacting with hardware devices.

6.1 Module

Hardware-Hiding

6.2 Uses

None

6.3 Syntax

This section describes the syntax used in the Hardware-Hiding Module, including the constants, programs, and routines that can be accessed.

6.3.1 Exported Constants

None

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
open	string, $\mathbb{N}$, $\mathbb{N}$	$\mathbb{Z}$	-
write	$\mathbb{Z}$, $\mathbb{N}^n$, $\mathbb{N}$	$\mathbb{N}$	-
read	$\mathbb{Z}$, $\mathbb{N}^n$, $\mathbb{N}$	$\mathbb{N}$	-

6.4 Semantics

This section provides the detailed functionality and behavior of the module's functions.

6.4.1 State Variables

None

6.4.2 Environment Variables

None

6.4.3 Assumptions

- The input parameters to open must be valid and correspond to existing hardware components.
- The write operation assumes that the hardware device is available and writable.
- The read operation assumes that the hardware device is readable.

6.4.4 Access Routine Semantics

open(pathname, flags, mode):
    See https://en.wikipedia.org/wiki/Open_(system_call).
write(fd, buf, count):
    See https://en.wikipedia.org/wiki/Write_(system_call).
read(fd, buf, count):
    See https://en.wikipedia.org/wiki/Read_(system_call).

6.4.5 Local Functions

None

7 MIS of General Digital Filter Module

This section outlines the formal specification of the General Digital Filter Module, which provides linear filter functions for filtering in the form of difference equations.

7.1 Module

General Digital Filter

7.2 Uses

- Error Handler Module (Section 14)
- Logger Module (Section 15)

7.3 Syntax

This section describes the syntax used in the General Digital Filter Module, including the constants, programs, and routines that can be accessed.

7.3.1 Exported Constants

None

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
lfilter	$\mathbb{R}^n$, $\mathbb{R}^m$, $\mathbb{R}^t$	$\mathbb{R}^t$	-

7.4 Semantics

This section provides the detailed functionality and behavior of the module's functions.

7.4.1 State Variables

None

7.4.2 Environment Variables

None

7.4.3 Assumptions

- The filter coefficients b and a are valid real-valued sequences.
- The input sequence x is a real-valued time series of appropriate length.
- The first coefficient of a is nonzero to ensure system stability.

7.4.4 Access Routine Semantics

lfilter(b, a, x):

- transition: Calculate the filtered result of the sequence x based on the given filter parameters b and a.
- output: The result of the sequence x after being filtered.

7.4.5 Local Functions

None

8 MIS of Specified IIR Filter Module

This section outlines the formal specification of the Specified IIR Filter Module, which provides parameter derivation functions for Butterworth and Chebyshev filters.

8.1 Module

Specified IIR Filter

8.2 Uses

- General Digital Filter Module (Section 7)
- Error Handler Module (Section 14)
- Logger Module (Section 15)

8.3 Syntax

This section describes the syntax used in the Specified IIR Filter Module, including the constants, programs, and routines that can be accessed.

8.3.1 Exported Constants

None

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
butter	$\mathbb{N}$, $\mathbb{R}$, string	$\mathbb{R}^n$, $\mathbb{R}^m$	ResultExceededExpectation
cheby1	$\mathbb{N}$, $\mathbb{R}$, $\mathbb{R}$, string	$\mathbb{R}^n$, $\mathbb{R}^m$	ResultExceededExpectation
cheby2	$\mathbb{N}$, $\mathbb{R}$, $\mathbb{R}$, string	$\mathbb{R}^n$, $\mathbb{R}^m$	ResultExceededExpectation

8.4 Semantics

This section provides the detailed functionality and behavior of the module's functions.

8.4.1 State Variables

None

8.4.2 Environment Variables

None

8.4.3 Assumptions

- The system performing the calculations can handle floating-point precision properly.

8.4.4 Access Routine Semantics

butter(N, Wn, btype):

- transition: Given the order N, cutoff angular frequency W_n, and type of the Butterworth filter (high/low), calculate the parameters.
- output: The parameter sequences b and a of the filter.
- exception: When the output result clearly does not meet expectations, throw the ResultExceededExpectationError.

cheby1(N, rp, Wn, btype):

- transition: Given the order N, the maximum ripple rp, the cutoff angular frequency W_n, and type of the Chebyshev I filter (high/low), calculate the parameters.
- output: The parameter sequences b and a of the filter.
- exception: When the output result clearly does not meet expectations, throw the ResultExceededExpectationError.

cheby2(N, rs, Wn, btype):

- transition: Given the order N, the minimum attenuation rs, the cutoff angular frequency W_n, and type of the Chebyshev II filter (high/low), calculate the parameters.
- output: The parameter sequences b and a of the filter.
- exception: When the output result clearly does not meet expectations, throw the ResultExceededExpectationError.

8.4.5 Local Functions

None

9 MIS of Math Module

This section outlines the formal specification of the Math Module, which provides three mathematical functions: squaring, thresholding, and RMSE calculation.

9.1 Module

Math

9.2 Uses

- Error Handler Module (Section 14)
- Logger Module (Section 15)

9.3 Syntax

This section describes the syntax used in the Math Module, including the constants, programs, and routines that can be accessed.

9.3.1 Exported Constants

None

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
calSquare	$\mathbb{R}^n$	$\mathbb{R}^n$	-
calThreshold	$\mathbb{R}^n$, $\mathbb{R}^n$	$\mathbb{R}^n$	InvalidInput
calRMSE	$\mathbb{R}^n$, $\mathbb{R}^n$	$\mathbb{R}$	InvalidInput

9.4 Semantics

This section provides the detailed functionality and behavior of the module's functions.

9.4.1 State Variables

None

9.4.2 Environment Variables

None

9.4.3 Assumptions

- All input sequences are non-empty.

9.4.4 Access Routine Semantics

calSquare(x):

- transition: Input a sequence x, compute a new sequence where each element is the square of the corresponding element in x.
- output: The sequence after squaring.

calThreshold(x, thres):

- transition: Input a sequence x and a threshold sequence thres, then compute the result of applying this threshold to x.
- output: The result of applying the threshold to the sequence x.
- exception: When the lengths of x and thres are not equal, throw an InvalidInput exception.

calRMSE(x, y):

- transition: Input two sequences x and y of the same length, then compute the RMSE.
- output: The RMSE of sequence x and sequence y.
- exception: When the lengths of x and y are not equal, throw an InvalidInput exception.

9.4.5 Local Functions

None

10 MIS of Pan Tompkins Algorithm Module

This section outlines the formal specification of the Pan Tompkins Algorithm Module, which provides the implementation of the Pan-Tompkins algorithm.

10.1 Module

Pan Tompkins Algorithm

10.2 Uses

- Specified IIR Filter Module (Section 8)
- General Digital Filter Module (Section 7)
- Math Module (Section 9)
- Error Handler Module (Section 14)
- Logger Module (Section 15)

10.3 Syntax

This section describes the syntax used in the Pan Tompkins Algorithm Module, including the constants, programs, and routines that can be accessed.

10.3.1 Exported Constants

None

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
findPeak	$\mathbb{R}^n$, $\mathbb{N}$	$\mathbb{Z}^m$	InvalidInput

10.4 Semantics

This section provides the detailed functionality and behavior of the module's functions.

10.4.1 State Variables

None

10.4.2 Environment Variables

None

10.4.3 Assumptions

- The input sequence rawSignal is a non-empty array of real numbers.
- The input sequence rawSignal is filterable.

10.4.4 Access Routine Semantics

findPeak(rawSignal, fs):

- transition: Input the raw signal rawSignal and the sampling frequency fs, and use the Pan-Tompkins algorithm to compute the R-wave peak indices.
- output: An integer sequence representing the R-wave peak indices.
- exception: When the input signal is shorter than the specified minimum length, throw an InvalidInput exception.

10.4.5 Local Functions

None

11 MIS of Annotated Data RMSE Module

This section outlines the formal specification of the Annotated Data RMSE Module, which provides functionality to compare algorithm results with annotated data to determine the magnitude of algorithmic errors.

11.1 Module

Annotated Data RMSE

11.2 Uses

- Math Module (Section 9)
- Error Handler Module (Section 14)
- Logger Module (Section 15)

11.3 Syntax

This section describes the syntax used in the Annotated Data RMSE Module, including the constants, programs, and routines that can be accessed.

11.3.1 Exported Constants

None

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
signalRMSE	$\mathbb{Z}^n$, $\mathbb{Z}^m$	$\mathbb{R}$	NoMatchingPoint

11.4 Semantics

This section provides the detailed functionality and behavior of the module's functions.

11.4.1 State Variables

None

11.4.2 Environment Variables

None

11.4.3 Assumptions

- The implementation of the Pan-Tompkins algorithm is correct.

11.4.4 Access Routine Semantics

signalRMSE(rawIndices, annIndices):

- transition: Input two index sequences rawIndices and annIndices (possibly of unequal lengths), match the nearby peak points, and compute the RMSE for the matched points.
- output: A real number representing the RMSE.
- exception: When no pairs of points match or the number of matching points is too few, throw a NoMatchingPoint exception.

11.4.5 Local Functions

None

12 MIS of Rwave Detect Module

This section outlines the formal specification of the Rwave Detect Module, which provides initialization and invocation functionalities for other modules, serving as the entry point for the entire program.

12.1 Module

Rwave Detect

12.2 Uses

- Input Output Processing Module (Section 13)
- Pan Tompkins Algorithm Module (Section 10)
- Annotated Data RMSE Module (Section 11)
- Error Handler Module (Section 14)
- Logger Module (Section 15)

12.3 Syntax

This section describes the syntax used in the Rwave Detect Module, including the constants, programs, and routines that can be accessed.

12.3.1 Exported Constants

None

12.3.2 Exported Access Programs

Name	In	Out	Exceptions
mainEntry	-	-	-

12.4 Semantics

This section provides the detailed functionality and behavior of the module's functions.

12.4.1 State Variables

- runningState: Record the stages of the program's execution, such as initialization, running, or unexpected termination.

12.4.2 Environment Variables

None

12.4.3 Assumptions

- The program will first initialize and enter this module during execution.

12.4.4 Access Routine Semantics

mainEntry:

- transition: As the entry point of the program, it will initialize other modules and make the corresponding function calls, coordinating the various modules.

12.4.5 Local Functions

None

13 MIS of Input Output Processing Module

This section outlines the formal specification of the Input Output Processing Module, which provides basic data input and output functionality by calling the Hardware-Hiding Module.

13.1 Module

Input Output Processing

13.2 Uses

- Hardware-Hiding Module (Section 6)
- Error Handler Module (Section 14)
- Logger Module (Section 15)

13.3 Syntax

This section describes the syntax used in the Input Output Processing Module, including the constants, programs, and routines that can be accessed.

13.3.1 Exported Constants

None

13.3.2 Exported Access Programs

Name	In	Out	Exceptions
readFromFile	string	-	FileError
writeToFile	string, x	-	FileError

13.4 Semantics

This section provides the detailed functionality and behavior of the module's functions.

13.4.1 State Variables

None

13.4.2 Environment Variables

None

13.4.3 Assumptions

- The operating system can perform file read and write operations correctly.

13.4.4 Access Routine Semantics

readFromFile(filename):

- transition: Read the ECG signal from a file at the specified path and convert it into a format that can be used by other modules.
- output: The ECG signal after format conversion.
- exception: When the operating system cannot find the corresponding file or the file's internal format is incorrect, throw a FileError.

writeToFile(filename, index):

- transition: Save the index sequence of the R-wave peaks to a file.
- exception: If the file is not writable, throw a FileError.

13.4.5 Local Functions

None

14 MIS of Error Handler Module

This section outlines the formal specification of the Error Handler Module, which provides handling functionality for exceptions that occur during the execution of the program.

14.1 Module

Error Handler

14.2 Uses

- Hardware-Hiding Module (Section 6)

14.3 Syntax

This section describes the syntax used in the Error Handler Module, including the constants, programs, and routines that can be accessed.

14.3.1 Exported Constants

None

14.3.2 Exported Access Programs

Name	In	Out	Exceptions
throwException	string	-	-

14.4 Semantics

This section provides the detailed functionality and behavior of the module's functions.

14.4.1 State Variables

None

14.4.2 Environment Variables

None

14.4.3 Assumptions

- The exception can be correctly logged and saved.

14.4.4 Access Routine Semantics

throwException(ex):

- transition: Throw an exception named ex, terminate the program execution, and output the relevant logs.

14.4.5 Local Functions

None

15 MIS of Logger Module

This section outlines the formal specification of the Logger Module, which provides functionality for logging output.

15.1 Module

Logger

15.2 Uses

- Hardware-Hiding Module (Section 6)

15.3 Syntax

This section describes the syntax used in the Logger Module, including the constants, programs, and routines that can be accessed.

15.3.1 Exported Constants

None

15.3.2 Exported Access Programs

Name	In	Out	Exceptions
log	string, $\mathbb{N}$	-	-
setLogLevel	$\mathbb{N}$	-	-
setLogPath	string	-	FileError

15.4 Semantics

This section provides the detailed functionality and behavior of the module's functions.

15.4.1 State Variables

- runningTime: The runtime of the program.

15.4.2 Environment Variables

- The default log level in the configuration file.

15.4.3 Assumptions

- Logs can be correctly output and stored.

15.4.4 Access Routine Semantics

log(msg, level):

- transition: If the level meets the output condition, log the message with the specified format and timestamp.

setLogLevel(level):

- transition: Suppress all log outputs below the specified level, and retain log outputs with a level greater than or equal to the specified level.

setLogPath(filename):

- transition: Set the log file save path.
- exception: If the file is not writable, throw a FileError.

15.4.5 Local Functions

None

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.