Module Interface Specification for Audio360

Team #6, Six Sense
Omar Alam
Sathurshan Arulmohan
Nirmal Chaudhari
Kalp Shah
Jay Sharma

November 12, 2025

1 Revision History

Date	Version	Notes
2025-11-13	1.0	Initial write-up

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 [1], with the addition that template modules have been adapted from [2]. The mathematical notation comes from Chapter 3 of [1]. 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 Audio360.

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 Audio360 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, Audio360 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 Data Types

6.1 Generics

6.1.1 int32

32 bits signed integer ($\mathbb{Z}$).

6.1.2 int64

64 bits signed integer ($\mathbb{Z}$).

6.1.3 uint32

32 bits unsigned integer ($\mathbb{N}$).

6.1.4 uint64

64 bits unsigned integer ($\mathbb{N}$).

6.1.5 float32

32 bits floating point ($\mathbb{R}$).

6.1.6 float64

64 bits floating point ($\mathbb{R}$).

6.2 Enums

6.2.1 Audio360State

1. AudioClassificationProcess: State when audio classification is running.
2. DirectionalAnalysisProcess: State when directional analysis is running.
3. OutputProcess: State when output processing is running.

6.2.2 Audio360Status

1. Uninitialized: Audio360 Engine is not initialized.
2. Initialized: Audio360 Engine is initialized, but not ready.
3. Ready: Audio360 Engine ready for requests.
4. Running: Audio360 Engine is running. Can not accept new requests.
5. Error: Audio360 Engine is stuck at an unhandled error.

6.3 Data Structures

7 MIS of [Module Name —SS]

[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]

7.1 Module

[Short name for the module —SS]

7.2 Uses

7.3 Syntax

7.3.1 Exported Constants

7.3.2 Exported Access Programs

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

7.4 Semantics

7.4.1 State Variables

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

7.4.2 Environment Variables

[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]

7.4.3 Assumptions

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

7.4.4 Access Routine Semantics

[accessProg —SS]():

- transition: [if appropriate —SS]
- output: [if appropriate —SS]
- 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]

7.4.5 Local Functions

[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]

8 MIS of Audio360 Engine - M??

8.1 Module

Orchestrates the overall audio processing by receiving raw input data and managing module communication.

8.2 Uses

1. Audio classification
2. Directional Analysis
3. Output
4. Audio Sampling Module

8.3 Syntax

8.3.1 Exported Constants

None

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
runProgram	-	-	ProgramStartFailure, ProgramRunTimeFailure

8.4 Semantics

8.4.1 State Variables

- state [Audio360State]: Determines the current state of the the Audio360 engine. Each state will run a specific module in used modules.

8.4.2 Environment Variables

- status [Audio360Status]: Status of the module. This encapsulates initialization, running, ready, or errored.

8.4.3 Assumptions

None

8.4.4 Access Routine Semantics

run():

- transition: The state machine in figure 1 outlines the transition of this module.

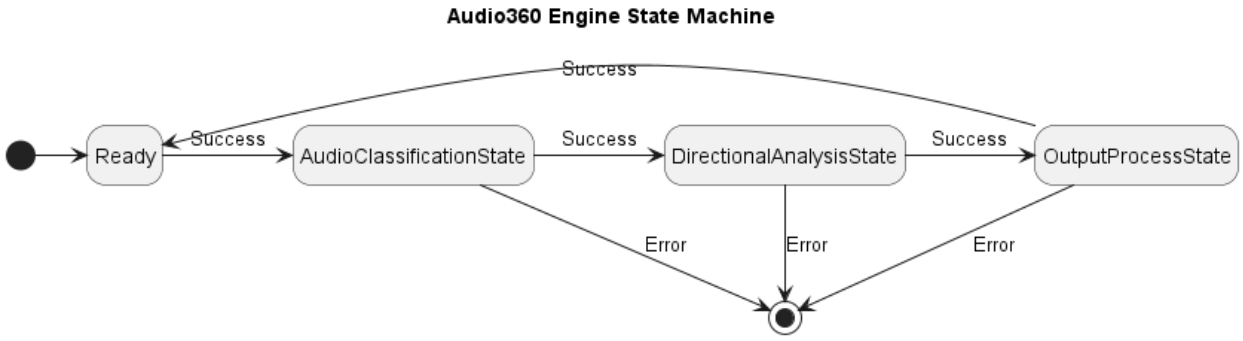

Figure 1: Internal state machine of Audio360 Engine module.

- output: None
- exception: ProgramStartFailure, ProgramRunTimeFailure

8.4.5 Local Functions

None

References

[1] D. M. Hoffman and P. A. Strooper, *Software Design, Automated Testing, and Maintenance: A Practical Approach*. New York, NY, USA: International Thomson Computer Press, 1995. [Online]. Available: http://citeseer.ist.psu.edu/428727.html

[2] C. Ghezzi, M. Jazayeri, and D. Mandrioli, *Fundamentals of Software Engineering*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall, 2003.

9 Appendix

[Extra information if required —SS]

Appendix — Reflection

1. What went well while writing this deliverable?

   Sathurshan: The system was decomposed into modules that was small enough. This allowed the team to easily split up the work without having much dependencies on each other. Furthermore, writing the MIS has helped the team further align with design decisions and formally document.

2. What pain points did you experience during this deliverable, and how did you resolve them?

   Sathurshan: The main pain point was the deliverable deadline as the team was asked to finish the first revision in a week while preparing for the proof of concept of demo. The team expressed the concerns with the professor, and fornately received an extension allowing us to put more effort into this document.

3. Which of your design decisions stemmed from speaking to your client(s) or a proxy (e.g. your peers, stakeholders, potential users)? For those that were not, why, and where did they come from?

   Sathurshan: The supervisor mentionned the difficulty of getting access to hardware that will meet our software specification. As a result, we designed the system such that the software is portable and can take input from different sources. This ensures that the capstone project can still be successful in the case we are not able to find the right hardware within our budget.

4. While creating the design doc, what parts of your other documents (e.g. requirements, hazard analysis, etc), it any, needed to be changed, and why?

   Sathurshan: SRS needed to be updated. It was updated because part of writing this document required reviewing the SRS. From this, I have found parts of the requirements that can be improved based on recent better understanding of the system.

5. What are the limitations of your solution? Put another way, given unlimited resources, what could you do to make the project better? (LO_ProbSolutions)

   Sathurshan: The limitation of the solution is compute and memory power. Given further time, the team would be able to design a system that is more low level and is optimized in accomplishing the tasks that are required.

6. Give a brief overview of other design solutions you considered. What are the benefits and tradeoffs of those other designs compared with the chosen design? From all the potential options, why did you select the documented design? (LO_Explores)

   Sathurshan: Personally, there was not a lot of time to think about other designs. The team has been implementing the software before this document was created since the proof of concept is one week after this document is due. Thus, the team already

considered and analyzed high level designs months ago and is too far back to document them.