

Module Interface Specification for Chess Connect

Team #4,
Alexander Van Kralingen
Arshdeep Aujla
Jonathan Cels
Joshua Chapman
Rupinder Nagra

January 18, 2023

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

1	Revision History	i
2	Symbols, Abbreviations and Acronyms	ii
3	Introduction	1
4	Notation	1
5	Module Decomposition	1
6	MIS of Web Application Input 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
6.4.5	Local Functions	4
7	MIS of Display 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	5
7.4.4	Access Routine Semantics	5
7.4.5	Local Functions	6
8	MIS of Web Application Output 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	7
8.4.4	Access Routine Semantics	7
8.4.5	Local Functions	7
9	MIS of User Mode Module	8
9.1	Module	8
9.2	Uses	8
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 Board Module	10
10.1	Module	10
10.2	Uses	10
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	11
10.4.5	Local Functions	11
11	MIS of Web Application Game State Module	12
11.1	Module	12
11.2	Uses	12
11.3	Syntax	12
11.3.1	Exported Constants	12
11.3.2	Exported Access Programs	12
11.4	Semantics	12
11.4.1	State Variables	12
11.4.2	Environment Variables	12
11.4.3	Assumptions	12

11.4.4	Access Routine Semantics	12
11.4.5	Local Functions	13
12	MIS of Engine Module	14
12.1	Module	14
12.2	Uses	14
12.3	Syntax	14
12.3.1	Exported Constants	14
12.3.2	Exported Access Programs	14
12.4	Semantics	14
12.4.1	State Variables	14
12.4.2	Environment Variables	14
12.4.3	Assumptions	14
12.4.4	Access Routine Semantics	14
12.4.5	Local Functions	15
13	Teensy Input from Mega Module	16
13.1	Module	16
13.2	Uses	16
13.3	Syntax	16
13.3.1	Exported Constants	16
13.3.2	Exported Access Programs	16
13.4	Semantics	16
13.4.1	State Variables	16
13.4.2	Environment Variables	16
13.4.3	Assumptions	16
13.4.4	Access Routine Semantics	16
13.4.5	Local Functions	17
14	Teensy Input Bluetooth from Web App Module	18
14.1	Module	18
14.2	Uses	18
14.3	Syntax	18
14.3.1	Exported Constants	18
14.3.2	Exported Access Programs	18
14.4	Semantics	18
14.4.1	State Variables	18
14.4.2	Environment Variables	18
14.4.3	Assumptions	18
14.4.4	Access Routine Semantics	18
14.4.5	Local Functions	19

15	Teensy Output Bluetooth to Web App Module	20
15.1	Module	20
15.2	Uses	20
15.3	Syntax	20
15.3.1	Exported Constants	20
15.3.2	Exported Access Programs	20
15.4	Semantics	20
15.4.1	State Variables	20
15.4.2	Environment Variables	20
15.4.3	Assumptions	20
15.4.4	Access Routine Semantics	20
15.4.5	Local Functions	21
16	Appendix	23

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 $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | \dots | c_n \Rightarrow r_n)$.

The following table summarizes the primitive data types used by Chess Connect.

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 Chess Connect 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, Chess Connect 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 Web Application Input Module

6.1 Module

Web Application Input

6.2 Uses

[Board Module](#)

[User Mode Module](#)

6.3 Syntax

6.3.1 Exported Constants

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
parseInput	string	seq of string	invalidInput

6.4 Semantics

6.4.1 State Variables

inputString: string #String containing FEN string, user mode, game termination state, and delimiting characters

6.4.2 Environment Variables

N/A

6.4.3 Assumptions

N/A

6.4.4 Access Routine Semantics

parseInput():

- output: sequence of strings. The first is the FEN string, the second is the user mode, the third is the game termination state.
- exception: invalidInput if any of validFen, validUserMode, or validGameTermination return false.

6.4.5 Local Functions

Name	In	Out	Exceptions
validFen	string	boolean	
validUserMode	string	boolean	
validGameTermination	string	boolean	

7 MIS of Display Module

7.1 Module

Display

7.2 Uses

[Board Module](#)

7.3 Syntax

7.3.1 Exported Constants

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
drawSquare	string		
drawBoard	seq of (seq of int)		
displayGameTermination	int		
setBackground	string		

7.4 Semantics

7.4.1 State Variables

N/A

7.4.2 Environment Variables

N/A

7.4.3 Assumptions

N/A

7.4.4 Access Routine Semantics

drawSquare():

- output: Draw board square
- exception: none

drawBoard():

- transition: Uses drawSquare to display the game board

- exception: none

displayGameTermination():

- transition: Displays game termination state (checkmate, stalemate, etc.)
- exception: none

setBackground():

- transition: Sets the background colors of the display.
- exception: none

7.4.5 Local Functions

N/A

8 MIS of Web Application Output Module

8.1 Module

Web Application Output

8.2 Uses

[Engine Module](#)

[Game State Module](#)

8.3 Syntax

8.3.1 Exported Constants

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
sendData	string	string	

8.4 Semantics

8.4.1 State Variables

N/A

8.4.2 Environment Variables

N/A

8.4.3 Assumptions

N/A

8.4.4 Access Routine Semantics

sendData(string):

- output: string #Encodes game state (none, check, checkmate, stalemate), and 3 engine-generated moves
- exception: none

8.4.5 Local Functions

N/A

9 MIS of User Mode Module

9.1 Module

User Mode

9.2 Uses

[Engine Module](#)

9.3 Syntax

9.3.1 Exported Constants

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
getUserMode		string	
setUserMode	string		

9.4 Semantics

9.4.1 State Variables

userMode: string #Represents the current user mode (Normal, Beginner, Engine)

9.4.2 Environment Variables

N/A

9.4.3 Assumptions

N/A

9.4.4 Access Routine Semantics

getMode():

- output: string

$output := userMode$

- exception: none

setMode(string):

- transition: Sets userMode to the input user mode

$userMode := input$

- exception: none

9.4.5 Local Functions

N/A

10 MIS of Board Module

10.1 Module

Board

10.2 Uses

[Engine Module](#)

[Game State Module](#)

10.3 Syntax

10.3.1 Exported Constants

```
#define letters ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h']  
#define startFEN = 'rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq -  
0 1'  
#define boardDimension = 8
```

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
initialize			
getXYPosition	int	tuple of int	invalidIndex
getPosition	int	tuple of int	
getFenString		string	
setFenString	string		

10.4 Semantics

10.4.1 State Variables

fenString: string #Stores FEN string of current game position

10.4.2 Environment Variables

N/A

10.4.3 Assumptions

initialize is called before any other access routine.

10.4.4 Access Routine Semantics

initialize():

- transition: #Initializes fenString to the starting chess board position

$$fenString := startFEN$$

- exception: none

getXYPosition(int: squareInd):

- output: #X and Y number coordinate for an input square number. Eg. getXYPosition(14) returns (0, 6).

$$out := (squareInd // boardDimension, squareInd \% boardDimension)$$

- exception: none

getPosition(int: squareInd):

- output: #letter and number coordinate for an input square number. Eg. getPosition(14) returns 'g7'.

$$out := 'letters[squareInd \% boardDimension]' \\ + 'boardDimension - (squareInd // boardDimension)'$$

- exception: none

getFenString():

- output:

$$out := fenString$$

- exception: none

setFenString(string: fen):

- transition:

$$fenString := fen$$

- exception: none

10.4.5 Local Functions

N/A

11 MIS of Web Application Game State Module

11.1 Module

Web Application Game State

11.2 Uses

N/A

11.3 Syntax

11.3.1 Exported Constants

11.3.2 Exported Access Programs

Name	In	Out	Exceptions
isCheck	string	boolean	
isCheckmate	string	boolean	
isStalemate	string	boolean	

11.4 Semantics

11.4.1 State Variables

N/A

11.4.2 Environment Variables

N/A

11.4.3 Assumptions

N/A

11.4.4 Access Routine Semantics

isCheck():

- output: True if the position is 'check', false otherwise
- exception: none

isCheckmate():

- output: True if the position is 'checkmate', false otherwise
- exception: none

isStalemate():

- output: True if the position is ‘stalemate’, false otherwise
- exception: none

11.4.5 Local Functions

N/A

12 MIS of Engine Module

12.1 Module

Engine

12.2 Uses

N/A

12.3 Syntax

12.3.1 Exported Constants

`#define depth` #How many layers of depth the chess engine should use to evaluate the position
`#define maxSearchTime` #The maximum time the chess engine should take to evaluate the position

12.3.2 Exported Access Programs

Name	In	Out	Exceptions
evaluatePosition	string	string	

12.4 Semantics

12.4.1 State Variables

N/A

12.4.2 Environment Variables

N/A

12.4.3 Assumptions

The depth and maxSearchTime values will be determined experimentally after the system is built. There is a trade-off between move quality and speed/depth of the search.

12.4.4 Access Routine Semantics

evaluatePosition(string):

- output: String containing 3 possible moves, calculated by a chess engine from the FEN input string
- exception: none

12.4.5 Local Functions

N/A

13 Teensy Input from Mega Module

13.1 Module

Teensy Input from Mega Module

13.2 Uses

Receives game state information from the Arduino Mega in the form of a FEN string.

13.3 Syntax

13.3.1 Exported Constants

`#define baud rate` #The baud rate of the serial communication system
`#define stringFormat` #The format of the string remains constant to perform proper communication between the two Arduinos.

13.3.2 Exported Access Programs

Name	In	Out	Exceptions
copyFen	string	string	
collectMode	string	binary	

13.4 Semantics

13.4.1 State Variables

local game state
local game mode

13.4.2 Environment Variables

N/A

13.4.3 Assumptions

The size of the string passed from the Arduino Mega aligns with the designed format.

13.4.4 Access Routine Semantics

receiveGameState(pin):

- Input: A serial line of data from an Rx pin

- Output: A string containing the FEN of the game state
- exception: none

13.4.5 Local Functions

N/A

14 Teensy Input Bluetooth from Web App Module

14.1 Module

Teensy Input via bluetooth from the Web Application

14.2 Uses

Receives best moves from the game engine contained in the Web Application.

14.3 Syntax

14.3.1 Exported Constants

`#define baud rate` #The baud rate of the serial communication system
`#define stringFormat` #The format of the string remains constant to perform proper communication between the two Arduinos.

14.3.2 Exported Access Programs

Name	In	Out	Exceptions
copyBestMove	string	string	

14.4 Semantics

14.4.1 State Variables

local bestMove1
local bestMove2
local bestMove3

14.4.2 Environment Variables

N/A

14.4.3 Assumptions

The size of the string passed from the Web Application aligns with the designed format.
The Web Application will always be able to return best moves with the given data.

14.4.4 Access Routine Semantics

receiveBestMove(pin):

- output: String containing 3 possible moves, calculated by a chess engine from the FEN input string

- exception: none

14.4.5 Local Functions

N/A

15 Teensy Output Bluetooth to Web App Module

15.1 Module

Teensy Output via bluetooth to the Web Application

15.2 Uses

Sends current game state and game mode to the Web Application via Bluetooth

15.3 Syntax

15.3.1 Exported Constants

`#define baud rate` #The baud rate of the serial communication system
`#define stringFormat` #The format of the string remains constant to perform proper communication between the two Arduinos.

15.3.2 Exported Access Programs

Name	In	Out	Exceptions
copyFEN	string	string	
copyGameMode	string	string	

15.4 Semantics

15.4.1 State Variables

local lastGameState
local currGameState

15.4.2 Environment Variables

N/A

15.4.3 Assumptions

The size of the string passed to the Web Application aligns with the designed format that the web application is expecting

15.4.4 Access Routine Semantics

sendCurrState(pin):

- output: String containing the current game state

- exception: none

sendCurrMode(pin):

- output: String containing the current game mode
- exception: none

15.4.5 Local Functions

N/A

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>.

16 Appendix

[Extra information if required —SS]