

ESDC - Project Specifications

Mastermind 2-Player Game

Contents

- 1. Introduction**
 - Game Instructions
 - Sequence Possibilities
 - Result Feedback
- 2. Detailed Specifications**
 - General System State Diagram
 - System Instructions
- 3. User Interface and Design Symbol**
 - User Interface
 - Design Symbol
- 4. Communications Protocol**
 - Connections
 - Frame Description
- 5. What If Situations**

1. Introduction

Mastermind is a code-breaking game created in the 70s. We have chosen this game as our ESDC project because it was invented by Mordecai Meirowitz, an Israeli postmaster and telecommunications expert, and as new students of the Telecommunication Engineering Master it is an honour to implement this game.

In order to decide the rules of the game we have read the general instructions of the Mastermind game and we have been playing the mobile app Mastermind (by *Rottz Games*) to get ideas related to difficulty levels and user interface items. The instructions given in the following lines are our specific design instructions.

Game Instructions

In this game, a secret sequence of four pegs is randomly proposed by the machine. The sequence contains colors that can be repeated. Also, the sequence can contain an empty peg. Two players are participating in our game and their objective is to guess the secret sequence in less attempts than their opponent, with a maximum of ten attempts.

Players are given one name (Alice or Bob) depending on when they connect to the game. The first player to connect becomes Alice and the second one becomes Bob. Alice is in charge of choosing the game characteristics and Bob has to agree with her to let the game start.

The secret sequence will be generated in Alice's FPGA, which will send the sequence to Bob's FPGA. Although players have the secret sequence in their FPGAs, they never have access to it. Otherwise, the game would be corrupted.

No player will have advantage with respect to the other player because they will start at the same time. In each turn each player tries to guess the sequence simultaneously to the other player. **We decided that both players have to guess the same code in order to increase competitiveness.**

Every time players try to guess the secret sequence, they get feedback from what we call the *small result pegs*. This is explained with more detail in the subsection *Result Feedback*. Players will also get the feedback of the opponent's attempt in order to know if they are doing better or worse than the opponent. However, players only see on the screen their attempt sequences, not their opponent's ones.

Sequence Possibilities

Alice will have to answer two questions. Depending on the answers, the generated secret sequence will be different. In **Table 1.1** we can see the sequence possibilities for each combination of answers. Color possibilities are shown in **Figure 1.1**.

Blue	32 #0000FF
Magenta	36 #FF00FF
Red	40 #FF0000
Yellow	44 #FFFF00
Green	48 #00FF00
Cyan	52 #00FFFF

Figure 1.1. Color possibilities for VGA screen.


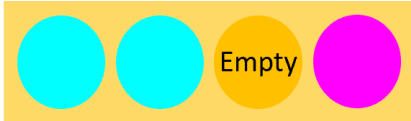
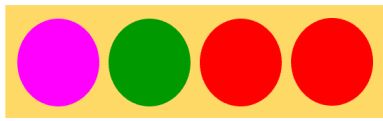
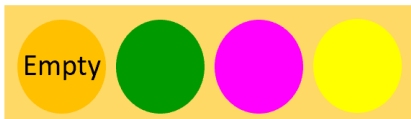
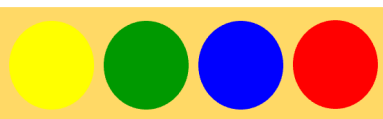
4 PEG sequences 		Do you want to have one empty peg in the secret sequence?	
		Yes	No
Do you want to have one repeated color in the secret sequence?	Yes	<p>Pegs can be Blue, Magenta, Red, Yellow, Green and Cyan. Also, there is one empty peg and one color is repeated once. Therefore, the amount of different colors in the secret sequence is 2.</p> <p>Example sequence:</p> 	<p>Pegs can be Blue, Magenta, Red, Yellow, Green and Cyan. One of the colors in the sequence is repeated once. Therefore, the amount of different colors in the secret sequence is 3.</p> <p>Example sequence:</p> 
	No	<p>Pegs can be Blue, Magenta, Red, Yellow, Green and Cyan but also there is one empty peg. Therefore, the amount of different colors in the secret sequence is 3.</p> <p>Example sequence:</p> 	<p>Regular sequence. Pegs can be Blue, Magenta, Red, Yellow, Green and Cyan. Each color can appear only one time.</p> <p>Example sequence:</p> 

Table 1.1. Description of the possible sequences depending on Alice's answers to questions.

Result Feedback

Every time players try to guess the secret sequence, they get feedback from what we call the *small result pegs*. Players get a small black peg for each correctly guessed peg, a small white peg for each guess with the correct color but in the wrong position and an empty small peg for each wrong guess. This is explained in a clearer way in the diagram shown in **Figure 1.2**.

The order of the small pegs is randomized. Therefore, if we get a black small peg in the first position, it does not mean that the first guessed peg is correct. It can be any of the other pegs. Also, players will see the small result pegs of their opponent in order to keep track of the game evolution by knowing if they are doing it worse or better than their opponent.

2. Detailed Specifications

General System State Diagram

In **Figure 2.1**, we can see a general state diagram of the game. This is not meant to be the state diagram of any block of our design, it is just a guide that helps to understand the functioning of our system. However, in this state diagram, no frame types are named. In order to understand how frames will be used in our system, it is required to read the following subsection *System Instructions*.

System Instructions

- **[INITIAL STAGE]** The game starts with two boards connected. LED LG0 is turned on in both boards to inform that they still have to be identified. The player who first presses [#] will become Alice and therefore her LG0 LED will turn off and her LG1 LED will turn on. The frame I AM ALICE will be sent to the other board.
- The opponent's board will receive this frame and its player will become Bob by pressing [*]. The frame I AM BOB will be sent to Alice. Bob's LG0 LED will turn off and his LG2 LED will turn on.
- When Alice receives the I AM BOB frame, she is asked (on the VGA monitor) to answer two questions related to the generation of the secret sequence. First, she will decide if the code has one repeated peg by pressing [A] to accept or [B] to cancel. Second, she will decide if the code has an empty peg by also pressing [A] to accept or [B] to cancel.
- After she decides the characteristics, she will send them to Bob in the frame GAME CHARACTERISTICS. When Bob receives the characteristics, he will see them on his monitor screen. At this moment, he will have to decide if he agrees or not with them by pressing [A] to accept or [B] to cancel. He will send the frame ACCEPT CHARACTERISTICS with the LSB set to 0 (rejects) or 1 (accepts). Alice has to try options until Bob agrees with her.
- The secret sequence is generated in Alice's FPGA. Alice sends this sequence to Bob in the frame SECRET SEQUENCE. However, it is a secret sequence and none of the players will ever have access to it.
- When Bob correctly receives the secret sequence, he acknowledges Alice by sending the BOB IS READY TO PLAY frame. This frame makes sure that both players can start guessing the sequence. From this point, both FPGAs behave in the same way and the game starts. LG7 is turned on in both boards.
- **[GUESSING STAGE]** Each user can begin guessing the sequence by pressing the keys 1-7 of their keyboard as indicated in **Tab. 3.1**. When the system is waiting for the user to introduce a sequence, the LG6 LED is on. If a user wants to change the sequence, they can press [B] to reset it. If a user is sure about their proposed sequence, they can fix it by pressing [A].
- When a user fixes the sequence, the LED LG6 is turned off and the board computes the small result pegs and sends them to the other user with the frame type RESULT PEGS. Therefore, each user has their own small result pegs and the small result

pegs of their opponent. At this moment, each user sees on the screen their result pegs and the result pegs of their opponent.

- With this information (own result pegs and opponent's result pegs) each user can calculate if the game is finished or has to continue. If any of the players has correctly guessed the sequence or this last turn was the tenth turn, LG1 or LG2 (depending if user is Alice or Bob) and LG7 are turned off and the message *Game Over / You Win / It's a Tie* is shown on each screen, depending on the result. Back to **[INITIAL STAGE]**
- If the game continues, then both users go back to entering the code (and LED LG6 is turned on again). Back to **[GUESSING STAGE]**.

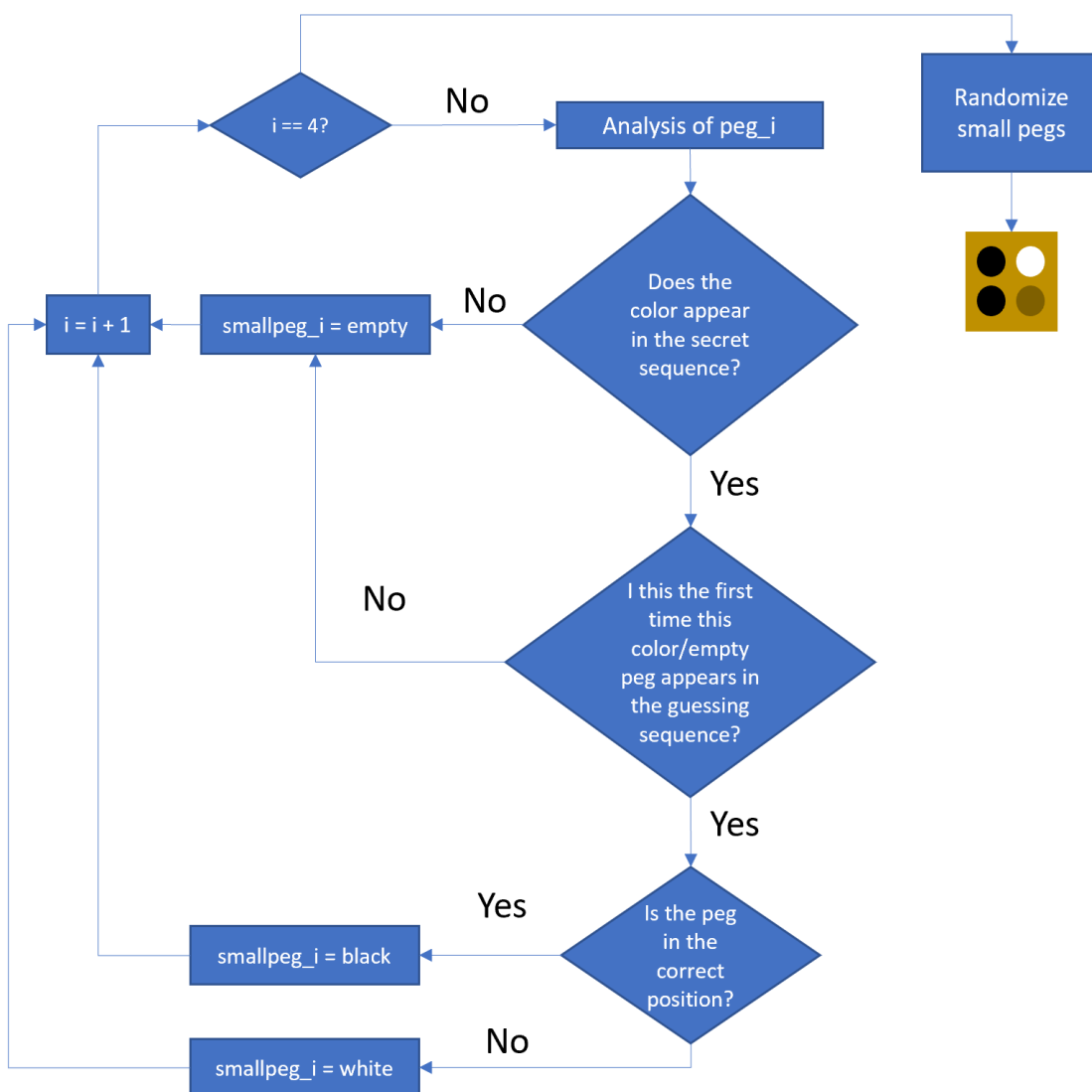


Figure 1.2. Diagram of the behaviour of the small result pegs.

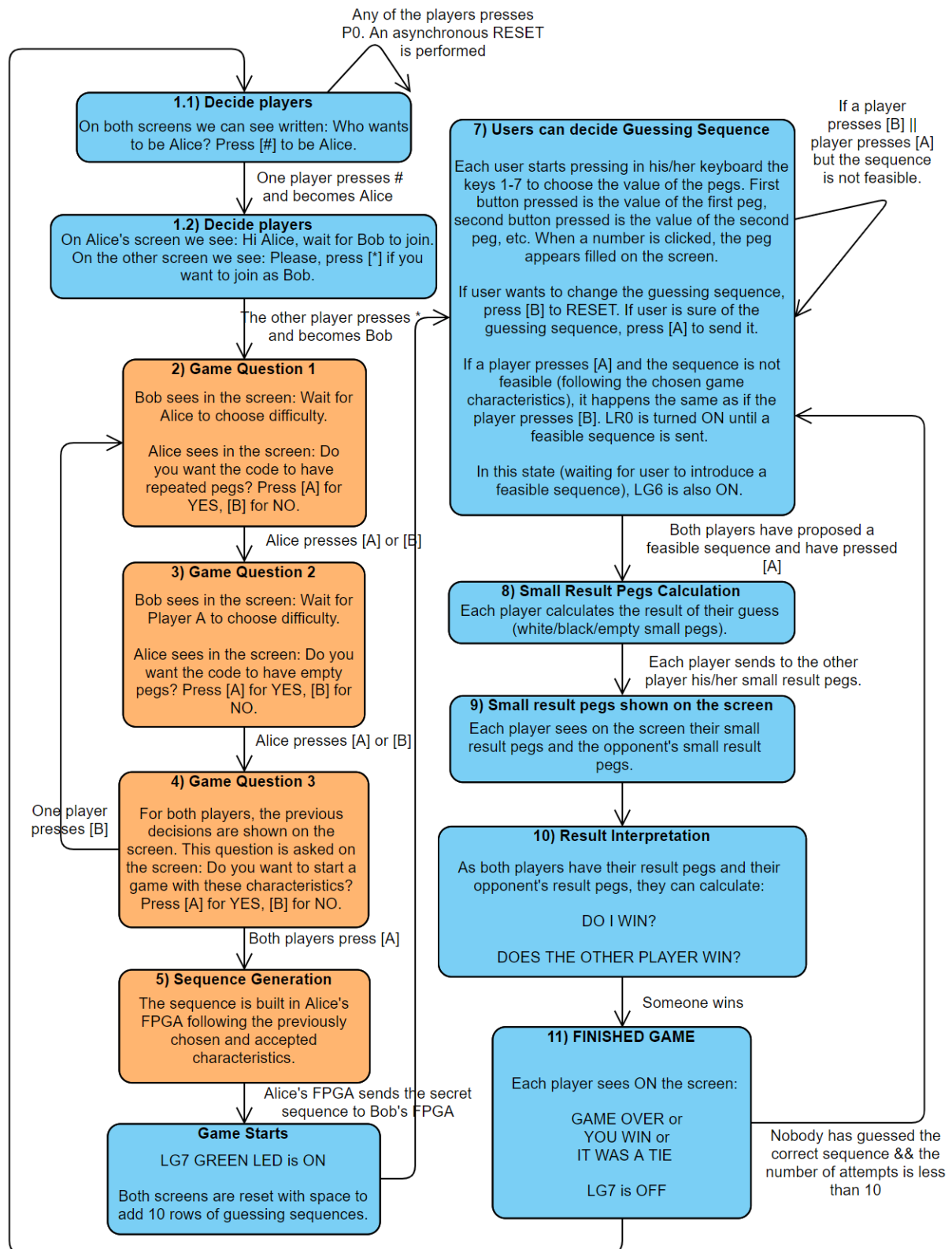


Figure 2.1. General State Diagram of our system. Orange states are states where Alice and Bob function in different ways.

3. User Interface and Design Symbol

User Interface

- Keyboard. We can see the use of each key in **Table 3.1**.
- ALTERA DE2 CYCLONE II:
 - Push button 0 (P0) to perform a global asynchronous reset.
 - RED LEDS
 - **LR0**: on if the user has entered a wrong guessing sequence (state 7 from **Figure 2.1**).
 - **LR1**: on if there has been an error when identifying both players (state 1.1 and 1.2 from **Figure 2.1**).
 - GREEN LEDS
 - **LG7**: on if the game is in progress (state 6-11 from **Figure 2.1**).
 - **LG6**: on if the user can enter a sequence (state 7 from **Figure 2.1**).
 - **LG0**: on if the player can still choose character (from **Figure 2.1**, state 1.1 in case of Alice but state 1.1 and 1.2 in case of Bob).
 - **LG1**: on if the user is Alice (until returning to state 1.1 from **Figure 2.1**).
 - **LG2**: on if the user is Bob (until returning to state 1.1 from **Figure 2.1**).
- Monitor screen connected by VGA to the FPGA. One monitor screen per FPGA (two int total, as there are two players in the game).
- RS-232 for the communication between the two FPGAs.

Keyboard Key	Use of Key
1	Red
2	Green
3	Blue
4	Yellow
5	Magenta
6	Cyan
7	Empty
[A]	Accept button
[B]	Cancel button
*	Key to become Bob
#	Key to become Alice

Table 3.1. Use of each game Keyboard Key.

Design Symbol

The design symbol of our general system is the following (see **Figure 3.1**).



Figure 3.1. System Design Symbol.

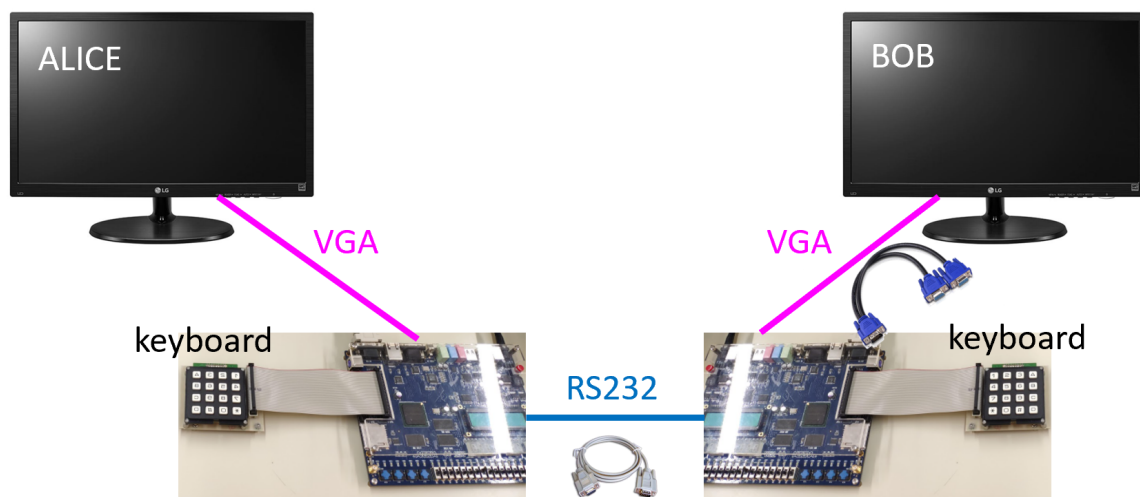


Figure 3.2. User Interface Scheme.

4. Communications Protocol

Connections

The boards will communicate through a serial cable RS-232, using the three wires (Tx, Rx and GND) and plugged to the 9 PINS RS-232 board connector. In this project we will assume that there are no communication errors. However, we want to discuss this with the teachers.

Regarding the system connections (please see **Figure 3.2**), it is also important to highlight the VGA connection between the two FPGAs and the two monitors available in the Laboratory. It is important to remember that baud rates must be the same for both FPGAs when establishing the connection.

Frame Description

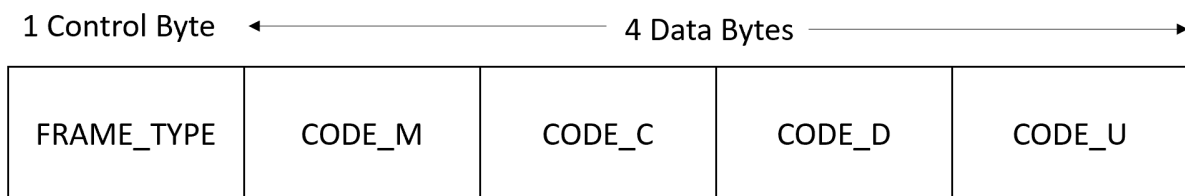


Figure 4.1. Frame structure (one box per byte).

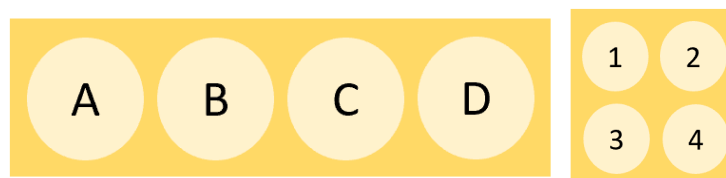


Figure 4.2. Sequence pegs (left) and small result pegs (right).

FRAME TYPE:

- I AM ALICE (A1)_{HEX}
- I AM BOB (A3)_{HEX}
- GAME CHARACTERISTICS (B1)_{HEX}
- ACCEPT CHARACTERISTICS (B3)_{HEX}
- BOB IS READY TO PLAY (B5)_{HEX}
- SECRET SEQUENCE (C1)_{HEX}
- RESULT PEGS (C3)_{HEX}

In the case of the frame type GAME CHARACTERISTICS, the values we want to know are whether we can have a repeated peg and whether there can be a repeated color in the secret sequence. This frame is sent at the beginning of the game, as many times as it is needed so that Bob accepts the game characteristics. In this case, we use the byte CODE_U as follows.

CODE_U: 0000 00U₁U₀, where U₁U₀ is giving the game characteristics following **Table 4.1**.

The four bytes of data are only needed if the frame type is SECRET SEQUENCE or RESULT PEGS. In both cases, we want to know the value of four pegs. In the case of the frame type SECRET SEQUENCE, the values we want to know are the colors of the secret sequence. This frame is sent at the beginning of the game, only once. We use the four data bytes as follows. In this case, only three bits for peg are required because colors will go from 1 to 7.

CODE_M: 0000 0M₂M₁M₀, where M₂M₁M₀ is the number associated with peg A

CODE_C: 0000 0C₂C₁C₀, where C₂C₁C₀ is the number associated with peg B

CODE_D: 0000 0D₂D₁D₀, where D₂D₁D₀ is the number associated with peg C

CODE_U: 0000 0U₂U₁U₀, where U₂U₁U₀ is the number associated with peg D

In the case of the frame type RESULT PEGS, the values we want to know are the colors of the result pegs of our opponent. This frame is sent at the end of each turn. In this case, only two bits for peg are required because colors will go from 0 to 2.

CODE_M: 0000 00M₁M₀, where M₁M₀ is the number associated with result peg 1

CODE_C: 0000 00C₁C₀, where C₁C₀ is the number associated with result peg 2

CODE_D: 0000 00D₁D₀, where D₁D₀ is the number associated with result peg 3

CODE_U: 0000 00U₁U₀, where U₁U₀ is the number associated with result peg 4

Finally, in the frame ACCEPT CHARACTERISTICS, there will be a bit in the byte CODE_U indicating whether Bob accepts or not the game characteristics proposed by Alice.

CODE_U: 0000 000U₀, where U₀ is 0 if Bob does not accept the characteristics or 1 if he does.

Repeated Peg (U ₁)	Repeated Color (U ₀)	Game characteristics
0	0	Regular sequence
0	1	Color is repeated once
1	0	One empty peg
1	1	One empty peg and a color is repeated once

Table 4.1. Bits in the frame type GAME CHARACTERISTICS.

5. What If Situations

The player proposes a not feasible guessing sequence. For example:

- **User presses [7] in case Alice has chosen not to have an empty peg in the secret sequence**
- **User repeats a color in case Alice has chosen not to have repeated colors**
- **User selects more than one empty peg (this is not a possible situation, see Table 1.1).**
- **User selects more than one repeated color (this is not a possible situation, see Table 1.1).**

In these cases, the system should tell the player that this action was wrong. We propose that after the player presses [A], the LR0 LED is turned on and the system asks again for a guessing sequence to the player. When the player finally proposes a feasible sequence, LR0 is turned off.

The two players press [#] to become Alice at the same time (or very fast)

If the two players receive the I AM ALICE frame, we will consider that the character selection went wrong and it will be restarted. Also, the LR1 LED will be turned on until Alice is correctly identified.