Binario Game

Functional Specification

Ray Sun
15 June 2018

Description: The system allows the user to play the game Binario on an 8 by 8 red/green LED matrix grid. Initially, the grid is populated with a predetermined starting tableau consisting of various red, green, and unlit positions. Various preprogrammed starting tableaus may be selected with a pair of rotary encoders. Upon selection, the starting tableau is displayed, a starting melody is played, and the game begins.

The objective of Binario is to fill the unlit positions such that all positions are filled, every row and column has exactly four (4) green and red positions each, no row or column has more than two positions with the same color in a row, no two rows are identical, and no two columns are identical.

During gameplay, the user may change the color of the currently selected position, cycling through red, green, and unlit, and also move around the game grid using the encoders. The user is only permitted to change open positions in the starting tableau. If an illegal move (i.e. change of a predefined, fixed starting position) is attempted, a distinctive warning melody alerts the user while the grid does not change. If the game is filled in incorrectly, the same warning sound notifies the user. The user is permitted to rectify the incorrect solution. When the correct solution has been entered, a victory melody is played and changes to the grid are disabled to allow the user to examine the solution. The user may then reset the system to select another starting tableau.

Input: All user input is through a pair of rotary encoders with integrated pushbutton switches.

Switch Name	**Switch Type**	**Description**
Left encoder (Up/Down encoder)	Rotary encoder (with pushbutton)	Game selection: Rotation – no effect Press – no effect Gameplay: Rotation – moves the cursor position vertically Press –resets the game; prompts user to select a new starting tableau.
Right encoder (Left/Right Encoder)	Rotary encoder (with pushbutton)	Game selection: Rotation – no effect Press – Selects displayed starting tableau, starts the game Gameplay: Rotation – moves the cursor position horizontally Press – cycles the current position through red, green, and unlit (open)

The starting tableaus and their solutions are preprogrammed in the system in a 1 KB serial EEROM (93C46A serial EEPROM chip).

Output: An 8 by 8 red/green LED matrix displays the current state of the game, with the current position to be adjusted (“cursor”) indicated by a position that blinks between two colors. A standard speaker is used to provide audio feedback for game selection and start (a short, distinctive

User Interface: melody), changing the color of a grid position (another melody), illegal moves and incorrect solution (two short "denied" beep) and game completion (a "victory" melody).
Upon startup, the system displays an introductory animation consisting of a starburst animation with mixed colors, followed by the following scrolling message (with alternating coloring of each character)

◄ B i n a r i o ►

and then another starburst animation. Then the system displays a preprogrammed starting tableau. By rotating the left/right encoder the user may cycle through eight (8) available starting tableaus. The tableaus displayed wrap around should the user attempt to advance past the available tableaus. The currently selected tableau is shown on the display, which blinks to indicate game selection.

The user confirms the selection of starting tableau and starts the game by pressing the left/right encoder. The selected starting tableau remains on the display, display blinking is stopped, and the game starts, indicated by a distinctive melody (the "1-Up" sound from *Super Mario Bros*). A flashing grid cell indicating the current position, the cursor (which blinks in the manner described below) appears. The initial cursor position is the upper left-hand corner.

During gameplay, the current state of the game is displayed on the LED matrix. The cursor, whose position the user may change by rotating the encoders, is indicated by a position on the grid that flashes between two alternating colors according to the following patterns.

Cursor Position Status	**Position Color**	**Cursor Color 1**	**Cursor Color 2**
Free (not part of starting tableau)	Clear	Clear	Red
	Red	Red	Green
	Green	Green	Clear
Fixed (part of starting tableau)	Clear	Clear	Yellow
	Red	Red	Yellow
	Green	Green	Yellow

In particular, if the cursor is on a free position, the cursor blinks between the current position color and the color that would be updated if the user cycles the color by pressing the left/right encoder. If the cursor is on a fixed position (i.e. one that is part of the starting tableau), the cursor blinks between the position color and yellow (both red and green LEDs on). With this scheme, the user may always determine the current color of the cursor position, whether or not the position may be changed, and (if the position is free) the next color.

The user may change the cursor position by rotating the left/right encoder (for horizontal left/right movement) or the up/down encoder (for vertical up/down movement).

If the current cursor position may be changed, pressing the left/right encoder switch cycles the color of the current position through red, green, and open (clear). A short tune (the coin collection sound tune from *Super Mario Bros.)* is played on the speaker.

Pressing the up/down encoder switch resets the game. The user is prompted to select a starting tableau again, as in initialization. Once the starting tableau is selected, the game is initialized as on the initial game selection and start, and the user may commence play as usual.

Should an illegal move (i.e. changing a fixed position from the initial tableau) be attempted, the speaker would alert the user with a warning sound, comprising two short beeps in a "denied" fashion. In this case, the cursor position will not change color.

Should the user fills the grid incorrectly, the "denied" sound is also played. This warning sound will not be played again until the user clears at least one position on the grid, and fills the grid incorrectly again (unless the user attempts to change a fixed position, in which case the warning wil play regardless). Should the user attempts to move the cursor beyond the edge of the grid, the cursor position will not change.

Game completion is indicated by a distinctive tune (the "stage cleared" tune from *Super Mario Bros.)* played on the speaker. While this tune is playing, the display blinks solid green. After the tune plays, the game state is restored on the display and the cursor is turned off, allowing the user to view the solution. Furthermore, changes to the grid are disabled (requiring a system reset or a up/down switch press to start a new game).

Error Handling: If the user attempts an illegal move (i.e. changing one of the fixed starting tableau positions), the speaker will sound a distinctive "denied" sound comprising two short beeps, and the grid will not change. Attempted movement of the current position off the edges of the grid are handled by checking for such movement and not updating the current position.

If there is a power failure, the system will reset when power is restored, allowing the user to start a new game by selecting a starting tableau.

Algorithms: (game state update)

CanChange– check whether a desired change of a grid space is legal
IsDone – check whether a desired move (of the current position) will run off the grid
HasWon – determine whether or not the game is solved, given a stored solution

Data Structures (game state)

Data structures pertinent to the game state are described in the table below:

Data Structure	**Description**
`gameStateBuf` `dispBuf`	16-byte buffers of column data for each of the 16 columns (8 red, 8 green) of LEDs in the display. These buffers are organized as following Physical Col LEDs in col 0 R00, R01, ..., R07 1 R10, R11, ..., R17 : 8 R71, R72, ..., R77 0 G00, G01, ..., G07 1 G10, G01, ..., G07 : 8 G70, G71, ..., G77 where the indices of each LED indicate the physical column number, then the row number, with (0,0) as the upper left hand corner. `gameStateBuf` is read and written by the game state updating logic, handling user input and checking for game conditions. `dispBuf` is updated from `gameStateBuf` and used to display the game state on the LED matrix.
`gameSolution`	8-byte game solution data for the current game, organized column-wise where the first byte is the first physical column on the display. A `1` indicates a red pixel, while a `0` indicates a green pixel.
`gameFixedPos`	8-byte data for the current game indicating which positions in the game are fixed (part of the initial tableau) or user-changeable, organized column-wise where the first byte is the first physical column on the display. A `1` indicates a fixed position, while a `0` indicates a free position.

Global Variables: None.

Limitations: The user cannot view the introductory message once it has been played without resetting or power-cycling the microprocessor. Upon a reset, the player is always prompted to select a starting tableau; the player cannot reset to the initially-selected tableau.

The state of the game is volatile; gameplay cannot be saved mid-play.

Known Bugs: None.

Special Notes: None.