



EE 306 - Microprocessors

Project Report of “Chess Clock”

Efe ERTEKİN - 041701006

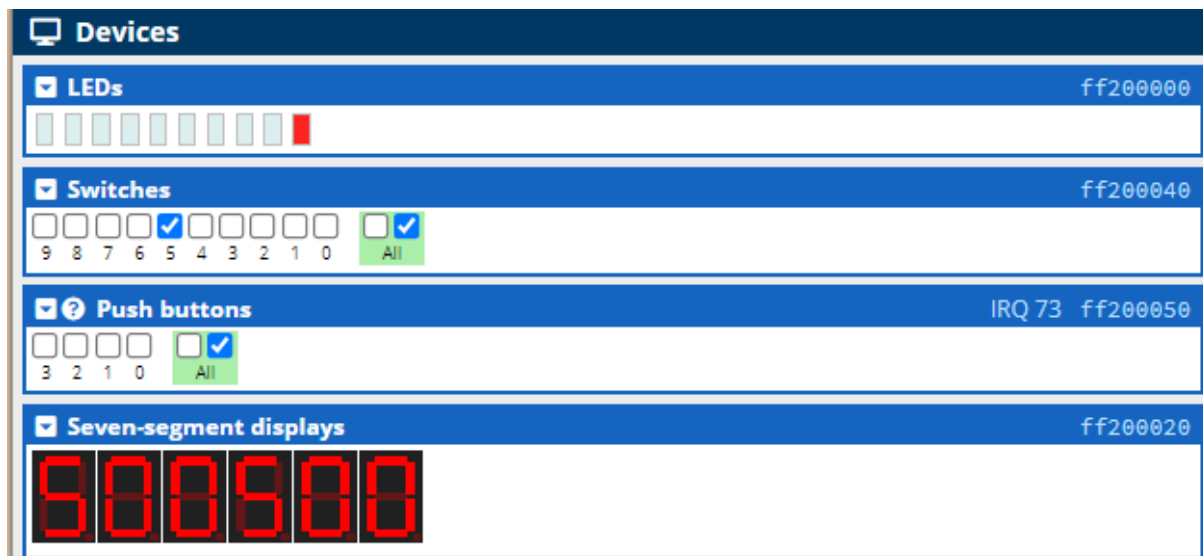
Submitted: 20.05.2021

Abstract:

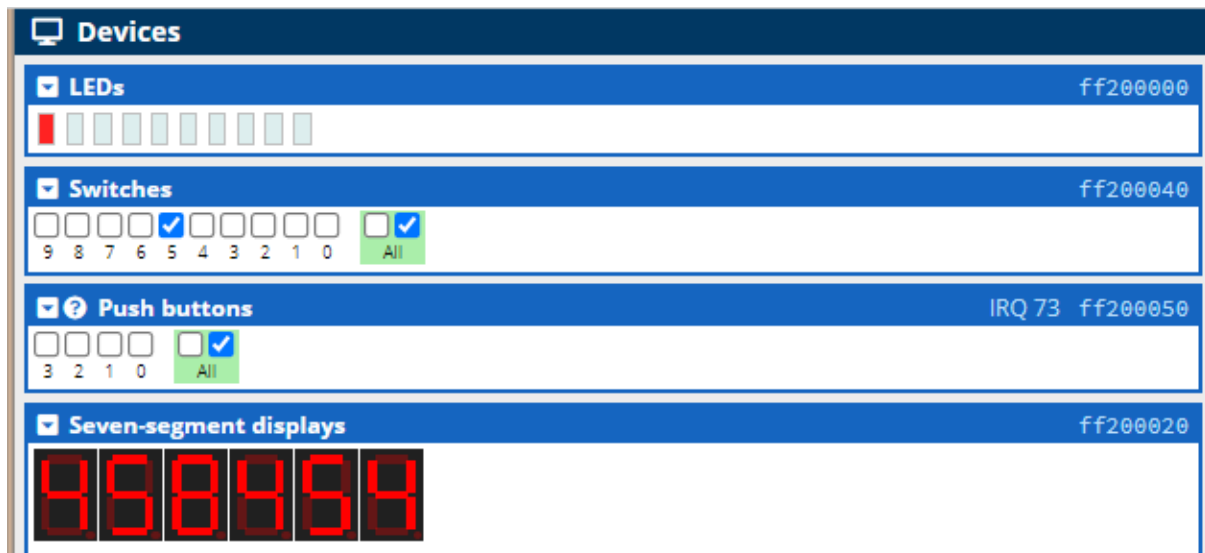
This is a Chess Clock. You can choose the time for the chess game, then start playing against your friends. You don't have to follow the remaining time of yours because this project is handling that for you. You can end your turn, reset the game, and start/pause the game with push buttons. Choose the game's mode (time) with switches. Easy to use and clearly shows every situation on the game. If you stopped the game, it would turn on the right-most LED; if game is on, left-most LED will be on; and lastly, if the game is finished which is one of the players run out of time, all LEDs will be on.

- Push button 0 = Right-side of the timer's player ending turn button.
- Push button 1 = Left-side of the timer's player ending turn button.
- Push button 2 = Restart the game (You can choose different mode using switch and restart the game with that mode as well.)
- Push button 3 = Start/Stop the game. (At the beginning, you should use this button to start the game.)
- Switches are changing the mode of the game. Feel free to check out figures to find which modes are assigned to which switches. If you don't choose any switch or choose inappropriate switches, game automatically starts 3+0 game for you.

Figures:



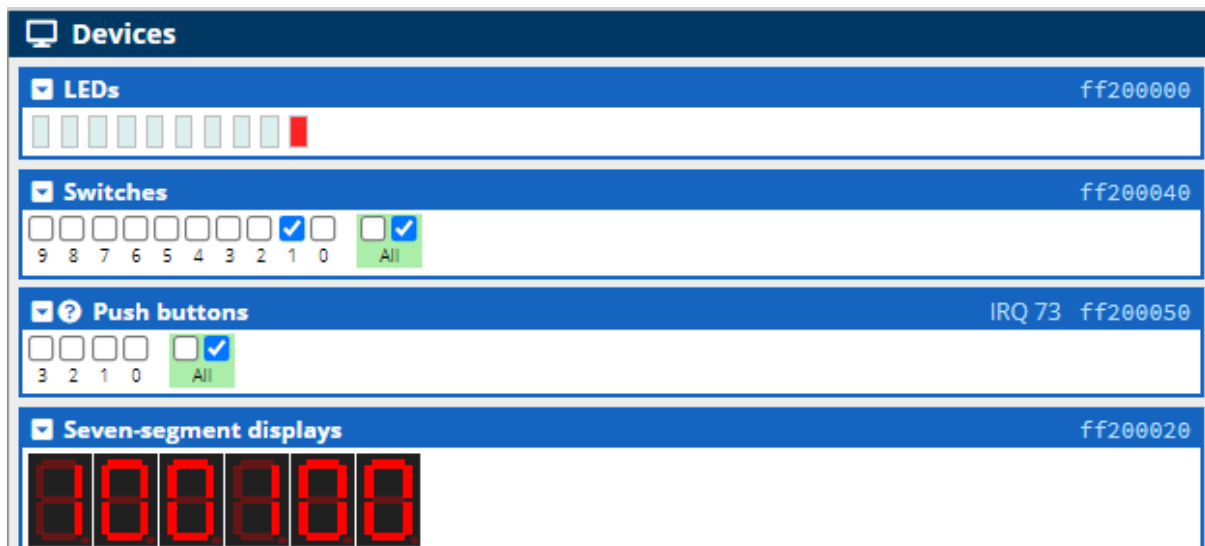
Figure#1: Game is stopped




Figure#2: Game is on




Figure#3: Game is done



Figure#4: 1+0 game's settings


Devices

☒ **LEDs** ff200000




☒ **Switches** ff200040

☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☒

9 8 7 6 5 4 3 2 1 0

☐ ☒

All

☒  **Push buttons** IRQ 73 ff200050


☐ ☐ ☐ ☐

3 2 1 0

☐ ☒

All

☒ **Seven-segment displays** ff200020



Figure#5: 1+3 game's setting

☒ LEDs
 ff200000

☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☒

☒ Switches
 ff200040

☐ ☐ ☐ ☐ ☐ ☒ ☐ ☐ ☐ ☐ ☒

9 8 7 6 5 4 3 2 1 0

☐ ☒ All

☒ ? Push buttons
 IRQ 73 ff200050

☐ ☐ ☐ ☐ ☒ ☒

3 2 1 0

☐ ☒ All

☒ Seven-segment displays
 ff200020



Figure#6: 3+0 game's setting

Devices

☒ LEDs ff200000

☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐ ☐

☒ Switches ff200040

☐ ☐ ☐ ☐ ☐ ☐ ☐ ☒ ☐ ☐
☐ ☒ All

9 8 7 6 5 4 3 2 1 0

☒ ? Push buttons IRQ 73 ff200050

☐ ☐ ☐ ☐
☐ ☒ All

3 2 1 0

☒ Seven-segment displays ff200020

8 0 0 3 0 0

Figure#7: 3+3 game's setting



Figure#8: 5+0 game's setting



Figure#9: 5+3 game's setting

Explanation of the project:

In this project, the creator's goal is the creation of a chess clock. With this, chess players will be able to see their remaining times. Times are shown in 7-segment displays. The format of the times is M:SS which is the first digit represents how many minutes left, and the last two digits represent how many seconds left. LEDs are showing game's status. If clock is stopped = right-most LED is on, if clock is active = left-most LED is on, if game is over = all LEDs are on.

Player1's time is on the right side of the 7-segment displays and player2's time is on the left time of the 7-segment displays. Players can interact with this clock using push buttons that are assigned for them. KEY3 of the push button is the run/pause button, KEY2 is a reset button that restarts the game, KEY1 is a button that assigned for player2 to indicate its turn is over, and lastly, KEY0 is a button that assigned for player1 to indicate its turn is over.

There are mods that can be chosen by players. For instance, players can start the timer with 5 minutes, 3 minutes, and 1 minute. In addition to this, also these games can be additive games which means when one of the players makes his move, the player will get an extra +3 seconds. Mods are choosable with switches. If you don't open any switches, the game will automatically start the 3+0 game. If you open **only** switch 1 = 1+0, switch 0 = 1+3, switch 3 = 3+0, switch 2 = 3+3, switch 5 = 5+0, and switch 4 = 5+3.

All pieces of information that are on the above are the main activities of this project. But there are many operations that occur to achieve these activities. Further explanation will be given below.

In this project, interrupts are used. So, there are many interrupts operations are handled. Firstly, the configuration of ARM GIC handled. Inside of the CONFIG_GIC, interrupts that will be used in the project are also handled by using the CONFIG_INTERRUPT subroutine. After these operations, Cortex-A9 Private Timer and Push buttons' interrupts can be handled by the processor.

Code goes into IDLE to start the main operations of the Chess Clock. Inside IDLE, firstly there is a RESET_ALL subroutine that is run. This subroutine is the insurance of the project. When KEY2 is used for the reset, this subroutine is handling that. Inside of the IDLE, there are mods for the game. You can choose one of the mods using switches. It is mentioned before.

When the code moves on, it reaches the Display0 subroutine that displays the first status of the game which is chosen mod's times. After that, there is a game_stopped subroutine. This checks whether the game is stopped or not. In the beginning, players always should use KEY3 to start the game because of this subroutine. Because there is a word (STOP) inside memory that keeps this status. And STOP starts with the value "0". After the game is started, the configuration of the private timer is handled with CONFIG_PRIV_TIME subroutine.

Now code is in the LOOP subroutine. Inside of this, there is always checking if the game is reset or not. Also, most of the interrupts come when the code is inside of the LOOP.

When the timer's interrupt arrives, the processor goes to 7. vector which is SERVICE_IRQ. Then, inside of it, there is a detection process that identifies the interrupt. After that, it goes to PRIV_TIME_ISR. Inside of that, firstly interrupt status is cleared. Then, finds which player's turn and keeps that player's time inside one of the registers so that it can check if time is "0" or not. If it is "0", then the game will be finished. Otherwise, it goes to the shifting subroutine. Inside the shifting subroutine, times are kept as decimal numbers in registers. They are stored digit by digit in registers. With this, display operation can be handled.

After keeping values inside registers, it goes to Display subroutine. Inside the Display, there are many operations. Firstly, it handles adding problems which are mentioned in the code part of the project but in short, if seconds exceed 60 seconds while adding 3 seconds, it handles this situation. After that, printing times. After printing operations, time decrease happens. It checks which player's turn is current, then with that information decrease the proper time (TIME1 or TIME2).

After all these operations, it goes to EXIT_IRQ. Inside of that, the code comes back to the inside of LOOP.

When push buttons' interrupt arrives, the same operations are happening. But the code goes to KEY_ISR instead of PRIV_TIME_ISR. Inside of that, firstly the processor clears the interrupt. After that, the processor identifies which key is pressed. KEY0 and KEY1 are assigned to players to dictate their turn is over. KEY2 is restarting the game. It is basically a

For HD version: <https://i.hizliresim.com/h4wostr.png>