Reaction Game with ERIKA OS

Learning focus:
OSEK, tasks, events, alarms

Your task is to develop a simple reaction game similar to the one developed during the Presemester class using Erika OS and to extend this functionality with some Arcadian light effects.

Marking

Correct Design and Implementation Reaction Game	6 points
Correct Design and Implementation “Arcadian Style”	4 points

You need 4 points in order to pass the lab!

Print the exercise and bring a handwritten solution together with your code to the lab.

1 Reaction Game

1.1 Requirements

Your task is to develop an embedded application with the following requirements

- The application runs as an Autosar Application
- Upon startup, the program will show a welcome message via the serial port.
- After pressing one of the two white buttons, the program will wait for a random time. After waiting for 1s to 3s a random value (1 or 2) will be displayed on both 7segment displays
- The user has to press the right button in case a ‘1’ is displayed and the left button in case a ‘2’ is displayed
- In case the correct button is being pressed, the measured reaction time in [ms] will be shown and the game can be started again by pressing one of the two buttons.
- In case a wrong button is pressed, an error message will be displayed and the game can be started again by pressing one of the two buttons.
- In case the user does not press a button within 1 s, the message “Too slow” will appear and the game can be started again by pressing one of the two buttons.
- One game consists out of 10 rounds
- At the end of a game, print the score (i.e. correct number of button pressed), the total time and the average time

Example output via the serial port:

```
Reaction test program round 1
press one of the two buttons to start...

Great - correct button pressed
Reaction time in ms: 249

==================================================
```

1.2 Analysis

Events are input signals which can be fired by the user or by the system to initiate a state change.

Example:

- After pressing button1 or button2, the program will wait for a random time.

Which events are mentioned in this requirement?

Event 1: ev_buttonpress

Event 2: ___________________________

Which state transition is mentioned in this requirement?

When the button is pressed for the first time, it triggers an event to start the game by waiting for a random time (done via alarm) and shows the number in seven segment display. Now the state is transitioned from Game_ISIDLE to Game_ISWAITINGBUTTON state. On pressing the button once again, the event is triggered again to show if the button pressed is correct/wrong or Too Slow.
The transition is as follows:
Game_ISIDLE ----------> Game_ISWAITINGBUTTON.

In addition to the two user input events, two additional system events can be identified from the requirements.

Event 3: ___________________________

Event 4: ___________________________

1.3 Erika elements

Alarms can be used to implement time signals. They can be compared with timer interrupts but provide more flexibility. Furthermore, several alarms can share one physical timer resource.

Provide the code snippets which will create the 1ms tick for the base counter.

```
ISR(systick_handler)
{
 CounterTick(cnt_systick);
}


int main()
{
EE_systick_set_period(MILLISECONDS_TO_TICKS(1, BCLK__BUS_CLK__HZ));
EE_systick_enable_int();

}
```

In our application, we will use the alarm `alrm_Tick1ms` which will be fired every 1ms. Every time the alarm is fired, the task `tsk_Timer` will be called. Alternatively, you may use a callback handler for this.

Check the API call `SetRelAlarm` and provide the code to configure the alarm to be fired every 1ms.

`SetRelAlarm(`__alarm_count__, __1__, __1__)

Alternatively, an alarm can also be configured to be fired only once. Where might we need this feature and how do you configure this?

We might need an alarm to be fired only once after a button is pressed at the start of the game. We need this feature when there arrives a circumstances, in which a function must be called only once.

For example:

// This alarm is called once to wait for a random time (approx for 1,2 or 3 seconds) after a button is pressed at the start of the game.
SetRelAlarm(alarm_seven_seg_display_wait, 1000*randomtime,0);

This can be configured by setting the TickType cycle column in a SetRelAlarm function to 0.

Add ONE ISR function for the button pressed. Will this function be an ISR1 or ISR2 category ISR? Explain.

It will be ISR2 category. If we want to use OS calls in the interrupt handler e.g., firing events, we have to make it category 2 interrupt.

1.4 State Maschine

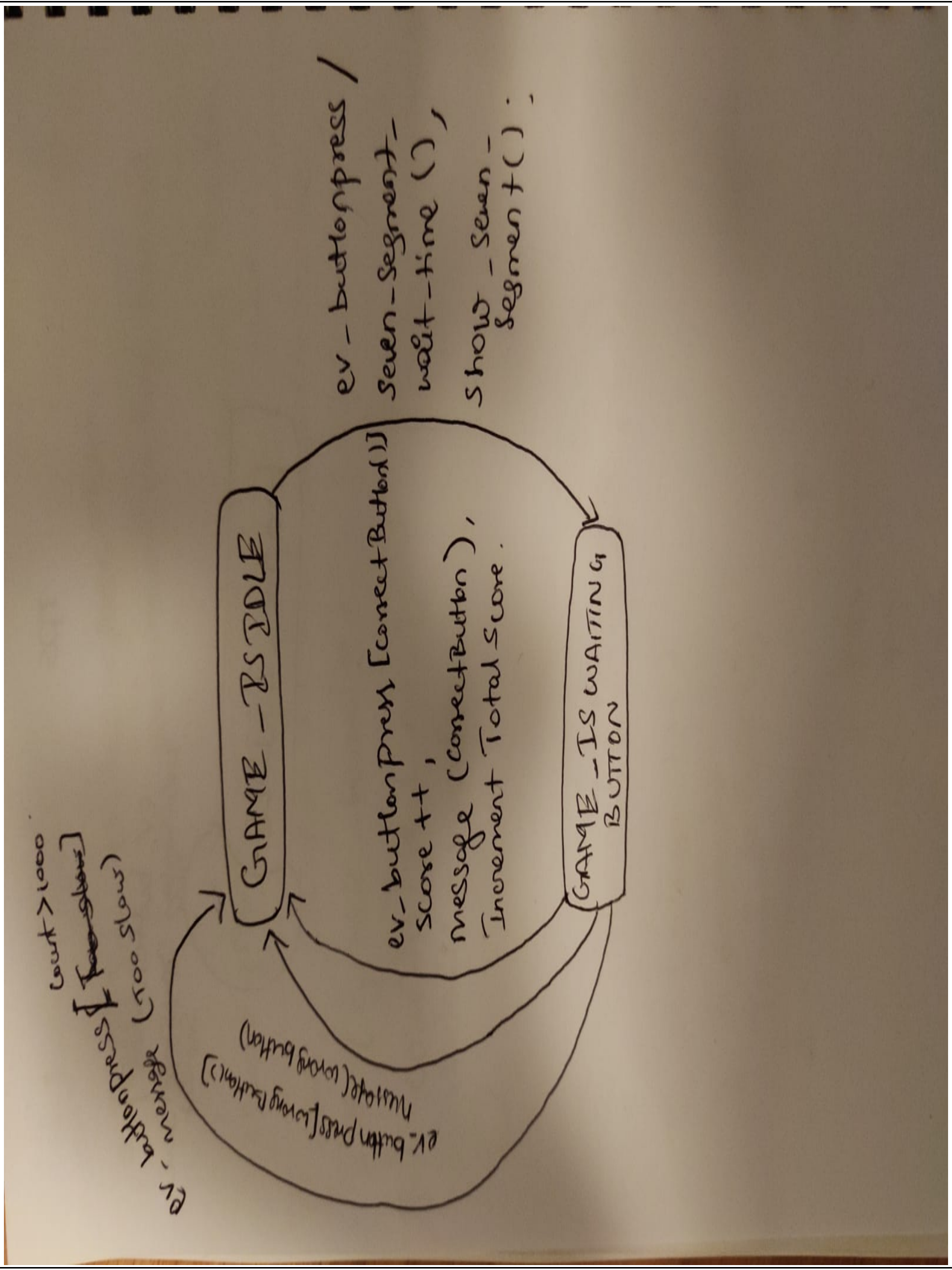
Draw a state machine of the system.

Describe the states using the **Is<WaitingforSomething>** mini sentence convention.

Use the following convention to describe a transition between two states:

Event [optional guard] / Action

A guard is an additional condition which is checked when an event occurs. In our game, this could be for example a check, if the correct button has been pressed. Guards are very often functions returning a Boolean value.


ev_buttonpress / seven_segment_wait_time(), show_seven_segment();
GAME_IS IDLE
GAME_IS WAITING BUTTON
ev_buttonPress [correctButton()] score++, message (correctButton), (increment Total score).
ev_buttonPress [wrong button()] message (wrong button)
ev_buttonpress [count>1000] message (too slow)

Implement the code for the program based on the given requirements and design.

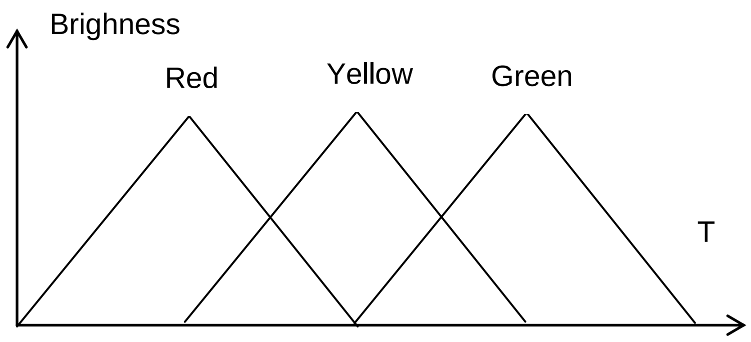
2 Arcadian Style

2.1 Requirements

In order to make the game a bit fancier, we want to add some Arcadian style light effect. The light effects may not interfere with rest of the functionality.

Fader:

In a first step, we want to create a fading traveling light. The three LED’s are glowing using the following pattern:



You might want to check some 80ies Knightrider movies to get into the grove, e.g. https://www.youtube.com/watch?v=Mo8Qls0HnWo

Glower:

Using the RGB LED, we want to implement an easily configurable glowing function. Using a const - table like the following (pseudo code)

```
const RG__Glow_t RG_glowtable_1[] = {
//Red   Green Blue  TimeInMS
{255,   0,    0,    500},
{0,     255,  0,    500},
{0,     0,    255,  500},
{0,     0,    0,    100},
{255,   255,  255,  100},
{0,     0,    0,    100},
{255,   255,  255,  100},
{0,     0,    0,    100},
{255,   255,  255,  100}
};
```

will create the sequence:

- 500ms red
- 500ms green
- 500ms blue
- 100ms off
- 100ms white
- 100ms off

- 100ms white
- 100ms off
- 100ms white

This sequence will be repeated permanently.

Provide a good declaration for `RG__Glow_t`

```
typedef struct
{
        uint8_t Red;
        uint8_t Green;
        uint8_t Blue;
        uint8_t delay;
}RG__Glow_t
```

How many tasks do you need for the new Arcadian functions? Explain.

We can take more than one Task for the Arcardian features, but ideally, we can impliment it in one task by calling both fadar and glower functions simultaneously.