CS207 Project Document

0. Team Roles and Schedule

Team Division of Labor:

Contribution percentage: 1:1:1

- **Lan Yiming** (蓝一鸣 12312808)
  - Main FSM structure
  - Basic functionalities test
- **Xu Yi** (胥熠 12312710) :
  - System design & extra functions
  - On board test
- **Li Yuxuan** (李雨轩 12312728) :
  - Debugs
  - Documentation

Planned Schedule & Implementation Status :

We planned to finish the basic project functions at week 13 and finish all bonuses at week 14 so that we could have our project defence at week 15.

- **11.6:**
  - Team up
- **11.28:**
  - First formal team discussion:
    the raw structure of the system and work distribution
- **12.1:**
  - Implement basic project functionalities
- **12.6:**
  - Integration
  - On-board testing and debugging
  - Implement project bonus: speaker
- **12.12**
  - Implement project bonus: keyboard
- **12.20:**
  - Project Defense
- **12.26:**
  - Documentation writing

1. System Function List

Power_off: The machine is off.

All parameters are reset to default.

Can toggle to **standby mode** by pressing center button and gesture power-on operation.

The machine can be turned off by holding center button for 3 sec on any other state. While the center button is held, the speaker makes sound as.

Standby mode: The machine is on, but no tasks.

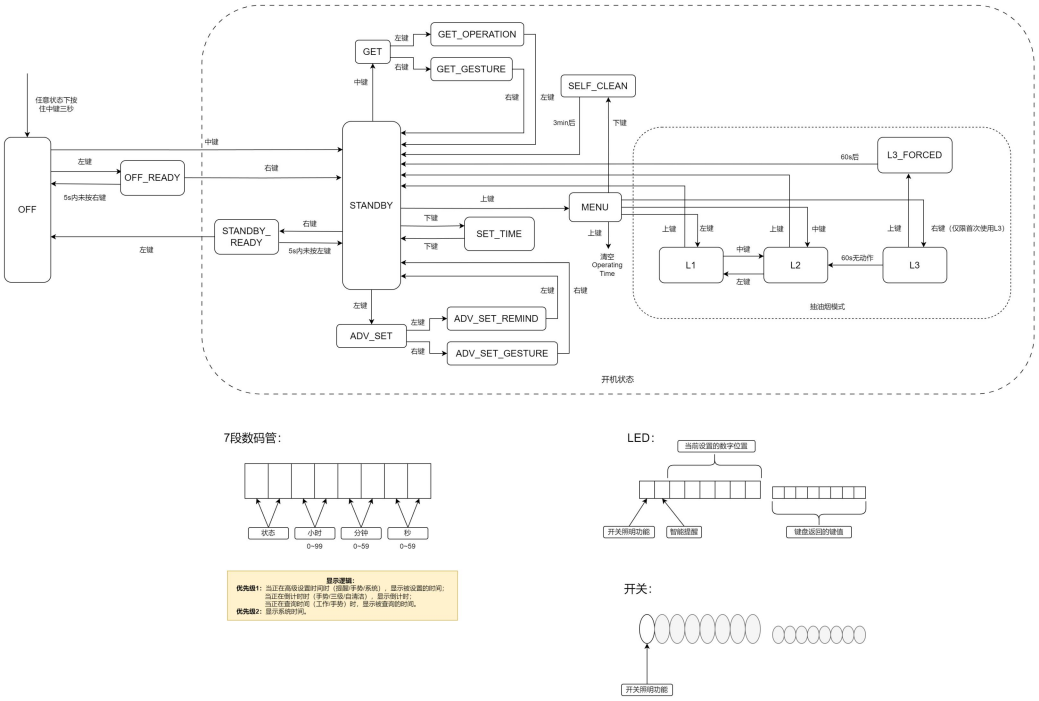
Can shutdown by gesture power-off operation.
Can toggle to getting current operation time/gesture switch time by pressing center+left/right, and toggle back by pressing the last pressed button.
Can toggle to setting operation time reminder/gesture switch time by pressing left+left/right, and toggle back by pressing the last pressed button.
Can toggle to setting system time by pressing down, and toggle back by pressing the last pressed button.
Can toggle to **menu** by pressing up.
Menu: The machine is to receive the input to determine what work to do.
Can toggle to **L1 working state** by pressing left button.
Can toggle to **L2 working state** by pressing center button.
Can toggle to **L3 working state** by pressing right button.
Can toggle to **self clean mode** state by pressing down button.
Working mode:
Include **L1, L2, L3 and L3 forced** states.
At any state in this mode, the speaker makes sound in a different frequency.

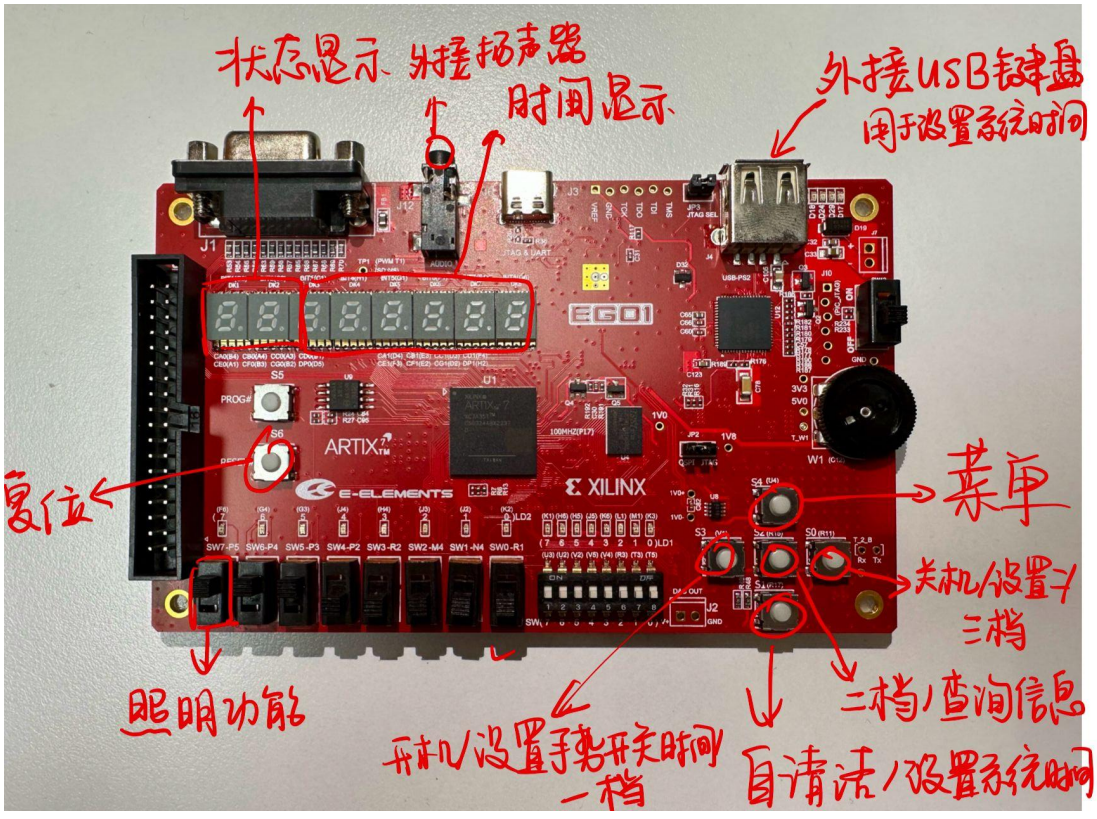
L1 state: The machine is working at minimum power gear.
Can toggle to **standby mode** by pressing up button.
Can toggle to **L2 state** by pressing center button.
L2 state: The machine is working at median power gear.
Can toggle to **standby mode** by pressing up button.
Can toggle to **L1 state** by pressing left button.
L3 state: The machine is working at maximum power gear.
Can toggle to **L3 forced state** by pressing up button.
Can toggle to **L2 state** after 1 min without operation.
Can only be entered once in one power on.
L3 forced state:
The machine is working at maximum power gear,Toggles to **standby mode** after 1 min.

Self clean mode: The machine self-clean itself, after the cleaning, the working time will be set into zero.
Toggles to standby mode after a certain period (default 3 min).
Lighting function: A LED will be lighted to give light.
Displayed at a EGO1 LED port. Can be turned on or off with a DIP switch in any state when power on.
Get Operation: The 7-seg tubes will display how long has machine worked.
Get Gesture: The 7-seg tubes will display how long the gesture time is.
Set Remind: The machine will receive the input from keyboard and set the remind time, when the working time is reached the remind time, a LED will be lighted to remind the user to clean the machine.
Set Gesture: The machine will receive the input from keyboard and set the gesture time. If the gesture time is 5s, then user will have 5 seconds to finish his gesture.

2. System Usage Instructions

The whole usage is given as follow:



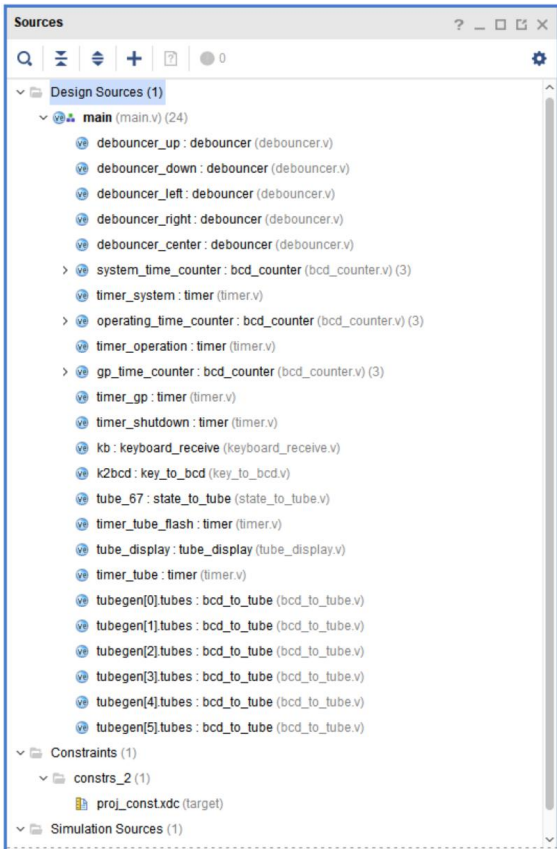


The left 2 tubes of 7-seg tube will display the state, the states representation are given as follows:

OFF	00	ADV_SET	09

OFF_READY	01	ADV_SET_REMIND	10
STANDBY	02	ADV_SET_GESTURE	11
STANDBY_READY	03	GET	12
MENU	04	GET_OPERATION	13
L1	05	GET_GESTURE	14
L2	06	SELF_CLEAN	15
L3	07	SET_TIME	16
L3_FORCED	08	ERROR	EE

3. System Structure & Sub Module Function Description



main.v: FSM structure, IO control integration, timers control, lighting function
debouncer.v: debouncer for EGO1 buttons
timer.v: one second timer
bcd_counter.v: time counter, including 4 modules:

bcd_counter: main module for the file, managing a time counter that count BCD_coded time in both regular and countdown form, or pause as needed.

twelve: managing hour counter, returning time count and whether the time count exceeds limit 11.

sixty: similar to twelve, managing minute and second counter, returning time count and whether the time count exceeds limit 59.

bcd_counter_one: bottom module for a designated counting feature and limit.

bcd_to_tube.v: decoder for BCD_coded time count to 7_seg tube control signal

state_to_tube.v: decoder for main FSM state to 7_seg tube control signal

keyboard_receive.v: includes a debouncer for ps2 keyboard signal clock and an FSM counter to manage and pass ps2 code

key_to_bcd: converter for ps2 key code to BCD number (used for time input)

tube_display.v: 7-seg tube display controller

main:

FSM structure that has overall 17 states (the diagram above);

Managing reset logics;

Managing 5 EGO1 button input and apply a **debouncer** for each of them

Managing a ps2 keyboard input using **keyboard_receive**, and use the input code for time setting with **key_to_bcd.**

Managing a system timer,

an operating timer for current operation time,

a general purposed timer for countdown for L3 state, L3_FORCED state, SELF_CLEAN state and gesture switch count, each of which uses a **bcd_counter** and a **timer** module.

Managing a shutdown timer for shutting down by holding center botton for 3 sec, using only **timer.**

Managing 7-seg tube display signal by processing output signal using **state_to_tube** for current state display and **bcd_to_tube** for system time/countdown/time setting display, and controlling 7-seg tube output by using **tube_display**.

Managing a time_setting block using the above keyboard input block and 7-segment display block for showing input content.

Managing a speaker controller module that make sound in the L1, L2, L3 and L3_FORCED working states and when the center button is pressed.

bcd_counter:

time counter module, receiving enable signal and reset time content from **main** and manage BCD_coded time for timers.

timer:

one second timer. Used for creating enable (toggle count) signal for **bcd_counter** and direct count for shutdown timer in **main.**

bcd_to_tube:

decoder for BCD_coded time count to 7_seg tube control signal, receive a 4-bit wide BCD-coded number and output a 8-bit tube control signal for **tube_display**.

state_to_tube:

decoder for main FSM state to 7_seg tube control signal. receive a 5-bit wide FSM state code signal and output a 16-bit tube control signal for **tube_display**.

tube_display:

7-seg tube display controller. manage the content on 7-segment tube display, receive

a 64-bit wide signal that holds the data to be displayed on the tubes. The module controls segments of the tubes that illuminate in a synchronic manner.
Some sub-modules are bonus and will be introduced in the following part.

4. Implementation Instructions for bonus

We've done 3 bonus: Gesture Operation; Extra output: Speaker; Extra Input: Keyboard.

Gesture Operation:

In main.v, take the ***power on*** gesture for example. When the state is on **OFF** and **left button is pressed**. Then the state will switch into **OFF_READY**, at the same time, a timer with gesture time will be set. The **OFF_READY** will end at these two circumstances: 1. When the timer reaches zero, which means that this gesture operation has failed, then the state will switch to **OFF**. 2. When the **right button is pressed.** This means that the gesture operation successes, then the state will switch to **STANDBY**, that is the machine is on.

The ***power off*** gesture shares the same principle.

Speaker Implementation:

In main.v, we manage a speaker controller module that make sound in the **L1, L2, L3** and **L3_FORCED** working states and when the **center button is pressed**.

The speaker itself is enabled making sounds by vibration. We find that if we switch the output signal at some frequencies, then the speaker will make sound of different tone. We set a register, which plus one at the positive edge of P17, then when the reg reaches a certain number like 120000, the output signal will be inverse, thus causing a vibration on the speaker. By controlling the certain number, we can successfully make different tones on L1, L2, and L3 mode, then it can tell the user what mode is the machine working at.

Keyboard Implementation:

keyboard_receive:

Interface with a ps2 keyboard that is used for time input. receives the ps2 clock signal (ps2_clk), and the ps2 data signal (ps2_data), outputs a signal indicating the key code, completion signal of a key receipt (to inform translation of **key_to_bcd** module), and a signal indicating if a key has been released (to inform a confirmed input digit for time input).

key_to_bcd:

convert key code input from **keyboard_receive** into BCD_coded digits.

All sub modules above except the two decoders work on a clock and a reset signal received from **main.**

In main.v, we manage a ps2 keyboard input using **keyboard_receive**, and use the input code for time setting with **key_to_bcd.** When the state is in **ADV_SET_GESTURE** or **ADV_SET_REMIND** or **SET_TIME**, the variable will be assigned to **the key_to_bcd** signal bit by bit.

5. Project Summary

We've learned a lot from this project experiences, and it could be summarized in the

following points:

- **Teamwork:**

  The teamwork is fluent more than expected.

  - Start the works as soon as possible. Our team started the formal discussion at week 12 and finished most of the project at week 13. This gives us enough time to finish all the bonuses and debugging.
  - Make sure everyone's part is independent. The whole project is large and we definitely don't want to waste time on integrating. E.g, one variable might be triggered by multiple conditions, we should assign all this variable code to one specified person to avoid overriding in Verilog.

- **Coding:**
  - The essence of modularization of the codes. We meet several problems in displaying the contents on the 7-seg tube. We made a mistake that we put too many things in a single variable so it has brought trouble to figure out which part ain't working properly. After the adjustments of modularization, we fix the bug easily. Also, this advantage also shows in writing the bonus part, for we can conveniently use the former code.
  - Avoid long regular expression. We meet several weird bugs and find causes in the long regular expression since the order of expression might not work as expected.
  - Use GitHub wisely. There are several existing projects on GitHub, we definitely will not to copy their code but we can imitate their project structure and instructions.
- **Testing:**
  - Make sure every possible path between states are tested. We made a serious mistake in final defense: When switching from L3 to L3_forced, we forget to refresh the timer and it hasn't been noticed during the on-board test.
  - It's highly appreciated that every members participate in the testing. We find that conducting testing by single person will inevitably miss some circumstances.

Overall, the project workflow is more fluent as expected, this might due to this semester's project don't seem to be as difficult as the previous semester.

6. Ideas for Project Questions

Simple calculator: Inspired by real calculator.

Implement a calculator that can do small integer (no more than 99999999, which is the display limit of 7-seg tube on EGO1) calculation, including add, subtract, multiply, divide (exact division) and mod operation.

Input can be in pre-order or post-order format (based on state of the machine). User decide the next input instance [number, operation, memory (last answer as a number), answer].

The calculator should correctly display the answer of the input when the user input to get the answer, or display error when one occur (answer out of range, 0 as divisor, stack not containing only the answer when getting answer, stack containing less than 2 number when input an operation in post-order input state, or any other operation errors).

The calculator should be able to store the last calculated answer and use it as an input number.

Bonus Points:

1. Use Input Devices (keyboard etc.) other than DIP switches and button switches.
2. Input in in-order format (for it need brackets and priority management).

No Reference is needed.

END.