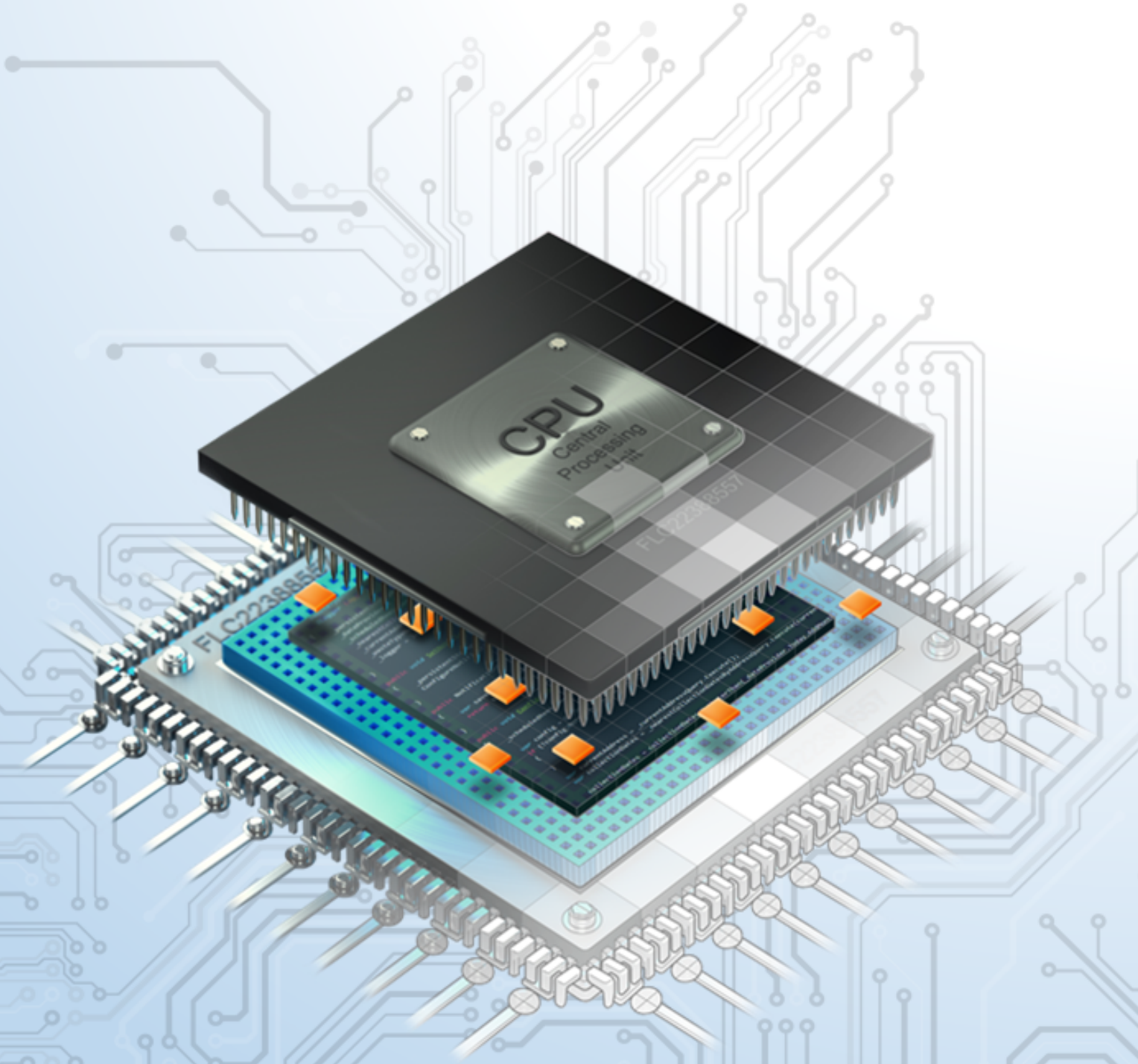
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
COMPUTER ENGINEERING

Microcontroller



Dr. Le Trong Nhan

VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

Microcontroller - Microprocessor - CC02

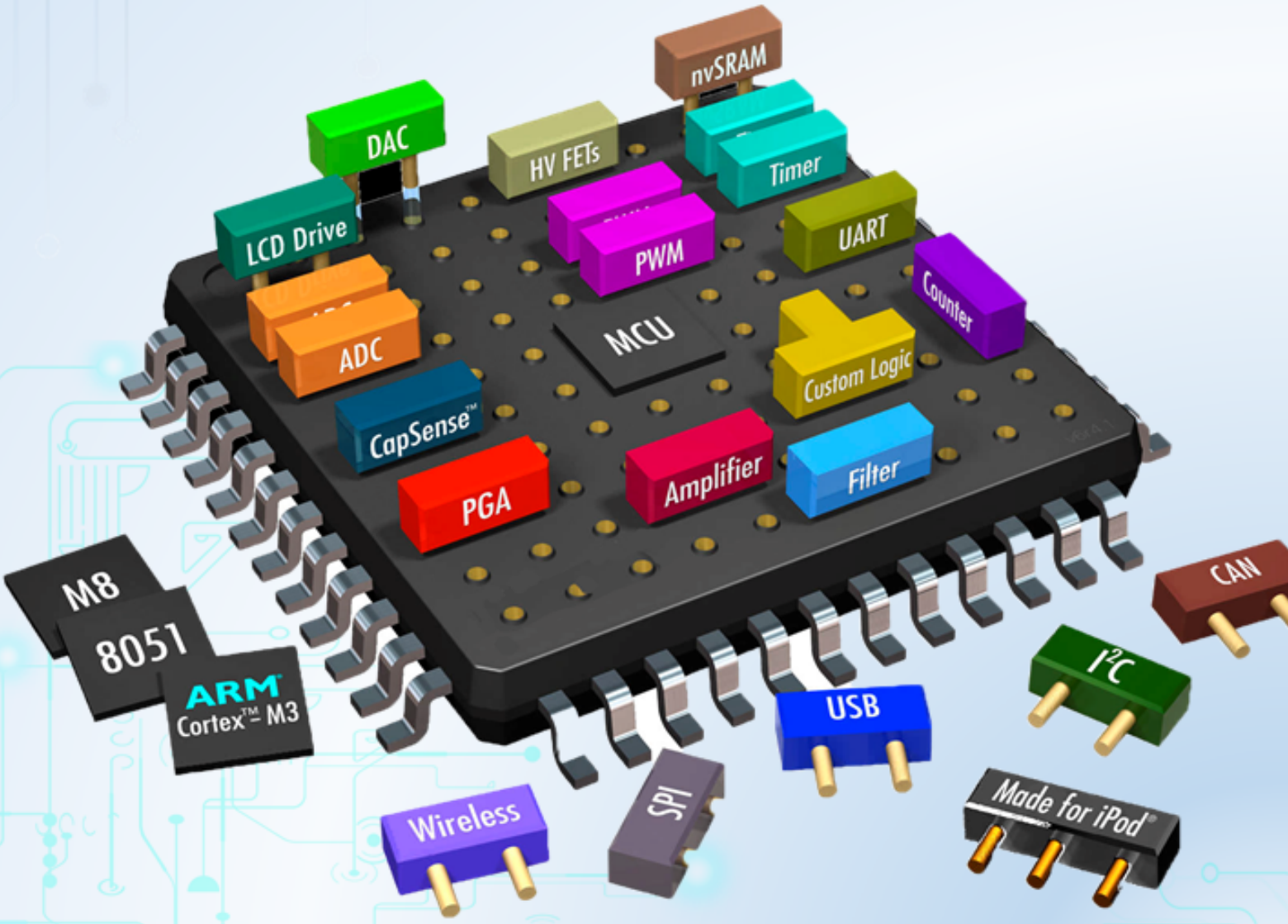
Report for Midterm Project

Le Tuan Hung - 2052508

Mục lục

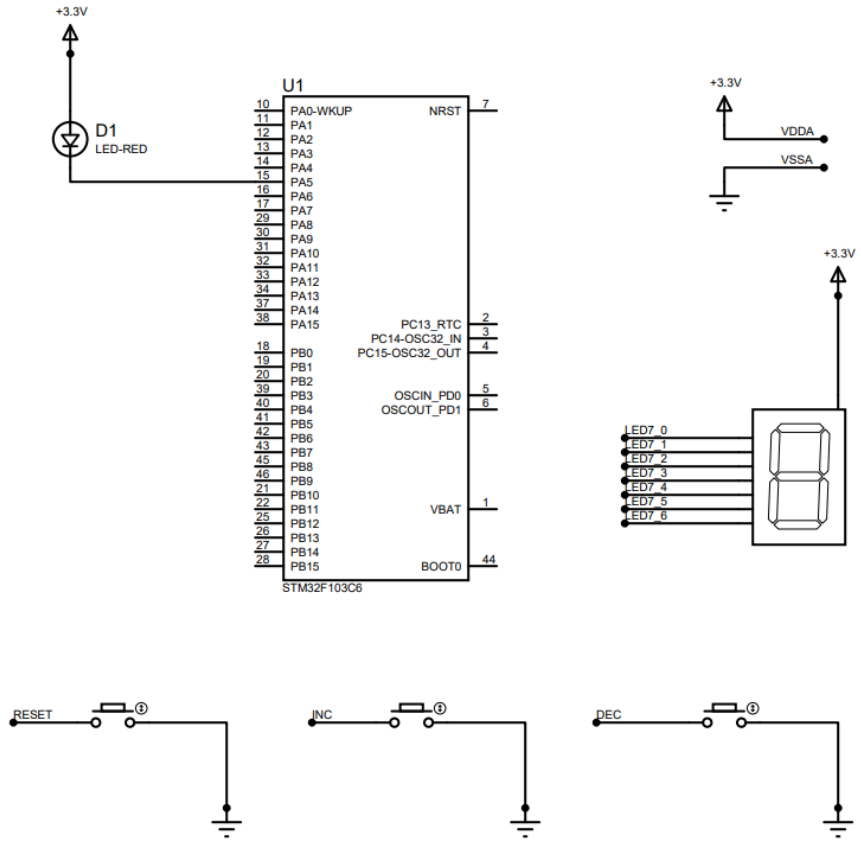
CHƯƠNG 1

MIDTERM 2022



1 Introduction

In this midterm project, a count-down system is designed and implemented in Proteus simulation. As it can be seen from Fig. 1.1, main components used in this project are the STM32F103C6, one LED, one LED7 segment and 3 different buttons.



Hình 1.1: *Proteus schematic for count-down system*

The main functions of the system are listed bellow:

- LED7 segment is used to display a counter ranging from 0 to 9.
- The **RESET** button is used to reset the counter value to 0. Meanwhile, the **INC** and **DEC** buttons are used to increase and decrease the counter value, respectively. There are two events need to handle for these buttons, including the normal-press and long-press.
- The D1 LED is blinking every second, which is normally used to monitor the execution of the system.

Students are supposed to following the section bellow, to finalize the project and fill in reports for their implementations. Some important notes for your midterm are listed bellow:

- The timer interrupt is 10ms. The value for counter is 9 (10 is also acceptable) when the pre-scaller is 7999.

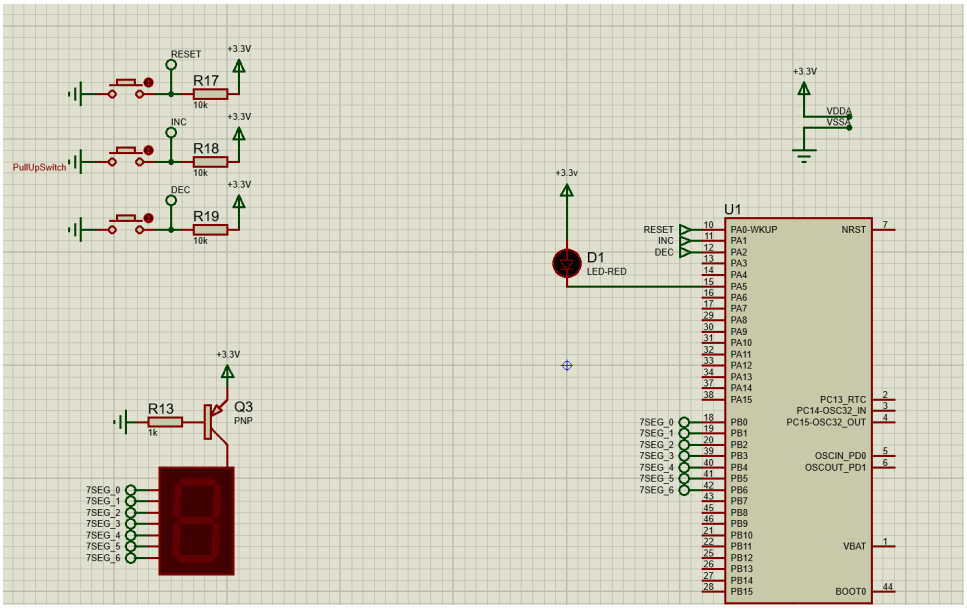
- All the buttons must be DEBOUNCING by using a timer interrupt service routing. A timeout for long press event is 3 seconds.
- There is no HAL_Delay() function in your source code. All the delay behavior must be based on a software timer.
- This report must be submitted with your answer.
- GitHub link for the source code and demo video link must be public access.

2 Implement and Report

2.1 Proteus schematic - 1 point

In this part, students propose the connection of the LED7 segment and 3 buttons to the STM32F103C6.

Your report: The schematic of your system is presented here. The screen can be captured and present in this part.



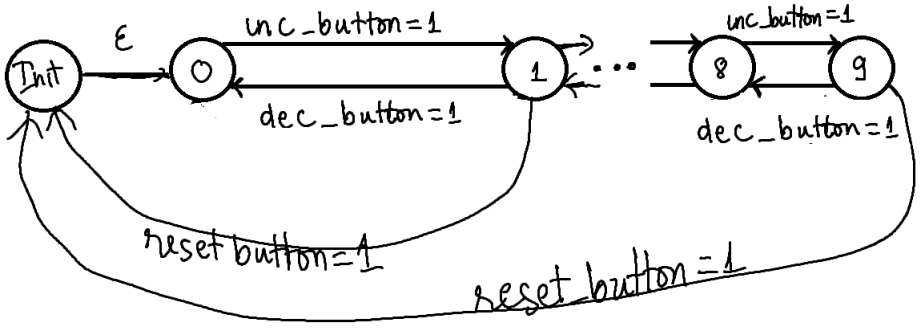
Hình 1.2: Long press behavior for INC button

2.2 State machine Step 1 - 2 points

A state machine is required in this step to perform just only the normal-press (or a button push) behavior of three buttons:

- Whenever the RESET is pressed, the counter value is 0.
- When INC is pressed, the counter is increased by 1. When counter is 9, it comes back to 0.
- When DEC is pressed, the counter is decreased by 1. When counter is 0, it rolls back to 9.

The value of the counter is displayed on the LED7 Segment.



Hình 1.3: Long press behavior for INC button

My report:

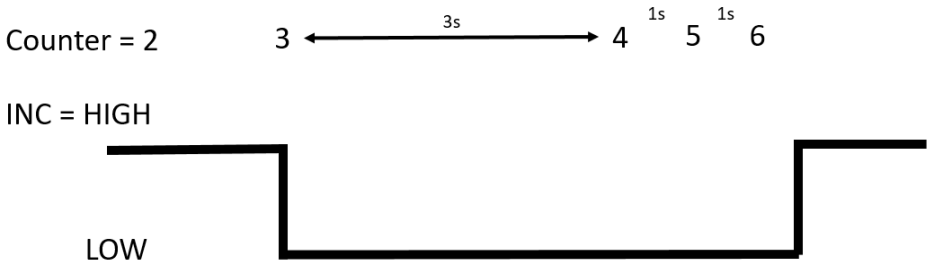
```
void fsm_simple_buttons_run(){
      switch (number_state){
    case INIT:
      number_state = NUM0;
      break;
    case NUM0:
      display7SEG(SegmentNumber[0]);
      if (isButtonPressed(RESET_BUTTON)){
        number_state = INIT;
      }
      if (isButtonPressed(INC_BUTTON)){
        number_state =  **NEXT_STATE**;
      }
      if (isButtonPressed(DEC_BUTTON))){
        number_state =  **PREV_STATE**;
      }
      break;
}
```

Program 1.1: Implementation of the state machine

2.3 State machine Step 2 - 2 points

In this part, long-press events for INC and DEC buttons are added to the project. For a button, this event is raised after 3 seconds keep pressing the button.

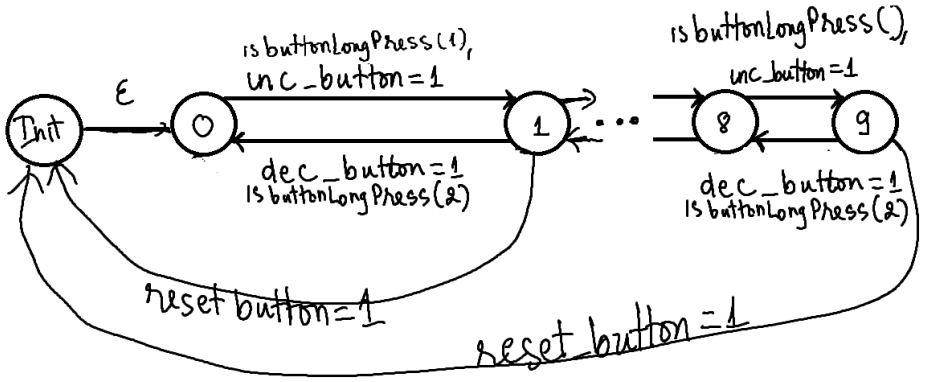
When a long-press event is detected, the value of the counter keeps changing every 1 second until the button is released. For example, the current value of the counter is 2 and the INC button is pressed. The value of the counter immediately increased by 1, or counter = 3. The INC button keeps pressing for 3 seconds, then the value of the counter is 4. As long as the INC button is pressed, the value continues increasing **every 1 second**. This behavior is illustrated in the Figure below:



Hình 1.4: *Long press behavior for INC button*

The behaviors of the DEC button are reversed to the INC button. The value of the counter is also rolled back if it reaches 0 or 9.

Your report: Present your whole state machine when the long press events are added.



Hình 1.5: *Long press behavior for INC button*

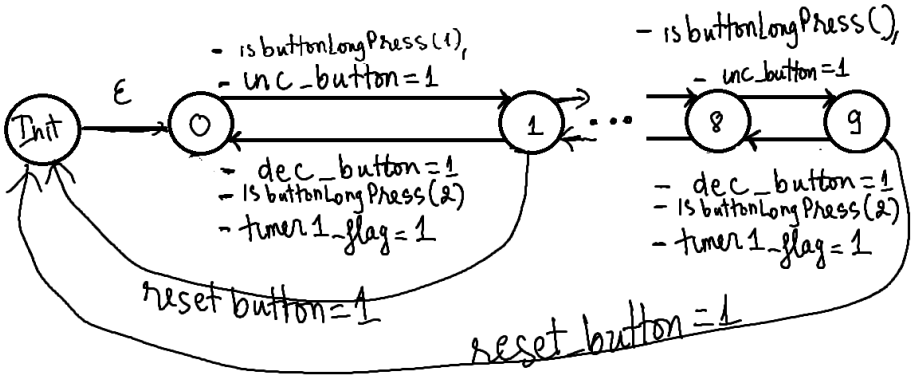
Your report: Present the main function, which is used to implement additional states. Minor changes in the previous source code are note required to present here.

```
if (isButtonPressed(INC_BUTTON)||isButtonLongPressed(
   INC_BUTTON)){
  number_state =  **NEXT_STATE**;
  setTimer1(10000);
  button_flag[1] = 0;
  button_flagLongPress[1] = 0;
}
if (isButtonPressed(DEC_BUTTON)||isButtonLongPressed(
   DEC_BUTTON)){
  number_state =  **PREV_STATE**;
  setTimer1(10000);
  button_flag[2] = 0;
}
```

2.4 State machine Step 3 - 2 points

Finally, where there is no button event after 10 seconds, the value of counter is counted down and stopped at 0. If the INC or DEC is pressed again, the status of the system comes back to the previous state, which is designed in Subsection 2 or 3.

Your report: Present your whole state machine for the 10s time-out event.



Hình 1.6: Long press behavior for INC button

Your report: Present a main function, which is used to implement additional states. Minor changes in the previous source code are note required to present here.

```
if (timer1_flag == 1){
    number_state =  **PREV_STATE** or NUM0 if NUM0**;
    setTimer1(1000);
    button_flag[2] = 0;
}
```

Final code: I will present the combination of those codes above, in only **1 case**, all remains behave the same and can be reached in the link https://github.com/thoriumrabbit/STM_Midterm

```
switch (number_state){
    case INIT:
      number_state = NUM0;
      setTimer1(10000);
      break;
    case NUM0:
      display7SEG(SegmentNumber[0]);
      if (isButtonPressed(RESET_BUTTON)){
        number_state = INIT;
        button_flag[0] = 0;
      }
      if (isButtonPressed(INC_BUTTON)||isButtonLongPressed(
   INC_BUTTON)){
        number_state =  NUM1; //Next state
        setTimer1(10000);
        button_flag[1] = 0;
        button_flagLongPress[1] = 0;
      }
      if (isButtonPressed(DEC_BUTTON)||isButtonLongPressed(
   DEC_BUTTON)){
        number_state =  NUM9; //Previous state
        setTimer1(10000);
        button_flag[2] = 0;
      }
      if (timer1_flag == 1){
        number_state =  NUM0; //Countdown stop at 0
        setTimer1(1000);
      }
      break;
```

in **main.c**, we also need to initialize and invoke the function in while loop and timer_callBack_function()

```
number_state = INIT;
setTimer0(1000);
while (1)
{
    debugLED_blinky();
    fsm_simple_buttons_run();
}
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
  timerRun();
  getKeyInput(0);
  getKeyInput(1);
  getKeyInput(2);
}
```

in **MX_TIM2_Init(void)**, i also define a variable called TIMER_CYCLE by using varible of htim2

```
      TIMER_CYCLE = 1000*(1 << htim2.Init.ClockDivision)
      *(htim2.Init.Period + 1)
          *( htim2.Init.Prescaler + 1)/8000000;
```

2.5 Led Blinky for Debugging - 1 point

Finally, for many projects based on microcontrollers, there is an LED that keeps blinking every second. In this project, the LED connected to PA5 is used to perform this feature.
my report: in "led_processing.c"

```
uint8_t SegmentNumber[10] = {
//Anode Common, A is MSB
  0x81,  //0
  0xCF,  //1
  0x92,  //2
  0x86,  //3
  0xCC,  //4
  0xA4,  //5
  0xA0,  //6
  0x8F,  //7
  0x00,  //8
  0x84   //9
};
void display7SEG(uint8_t num) {
  HAL_GPIO_WritePin(LED7_0_GPIO_Port, LED7_0_Pin, (num >>
   6) & 0x01);
  HAL_GPIO_WritePin(LED7_1_GPIO_Port, LED7_1_Pin, (num >>
   5) & 0x01);
  HAL_GPIO_WritePin(LED7_2_GPIO_Port, LED7_2_Pin, (num >>
   4) & 0x01);
  HAL_GPIO_WritePin(LED7_3_GPIO_Port, LED7_3_Pin, (num >>
   3) & 0x01);
  HAL_GPIO_WritePin(LED7_4_GPIO_Port, LED7_4_Pin, (num >>
   2) & 0x01);
  HAL_GPIO_WritePin(LED7_5_GPIO_Port, LED7_5_Pin, (num >>
   1) & 0x01);
  HAL_GPIO_WritePin(LED7_6_GPIO_Port, LED7_6_Pin, (num >>
   0) & 0x01);
}
void debugLED_blinky(){
  if(timer0_flag == 1){
    HAL_GPIO_TogglePin(LED_RED_GPIO_Port, LED_RED_Pin);
    setTimer0(1000);
  }
}
```

2.6 Github and Demo

A link to your github presented the last commit of your project is provided in this section. This link contains all files in your STMCube project (configurations, header and source files)

https://github.com/thoriumrabbit/STM_Midterm

And a link for just one demo video is also needed to present here.

https://www.youtube.com/watch?v=OCirHSe1Rjk

3 Extra exercise - Engineer mindset -1 point

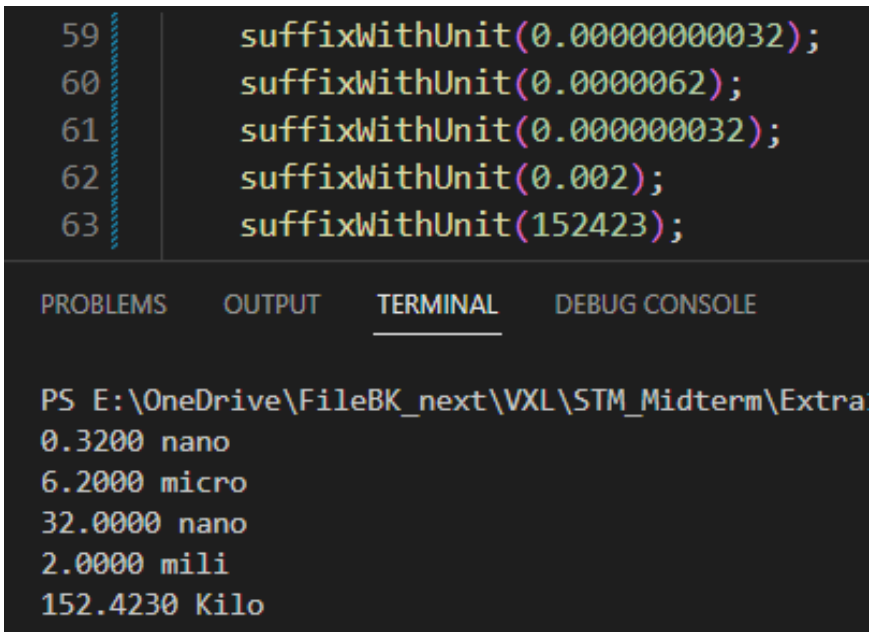
In this course, we encourage you to obtain an innovative mindset to solve daily problem. In this question, we would expect you to write a C program to solve the following problem.

Suffix with Unit

```
char* unitof10pow3(int i){
    switch (i){
    case -3:
        return " nano";
        break;
    case -2:
        return " micro";
        break;
    case -1:
        return " mili";
        break;
    case 0:
        return " ";
        break;
    case 1:
        return " Kilo";
        break;
    case 2:
        return " Mega";
        break;
    case 3:
        return " Giga";
        break;
    }
}
```

```
void suffixWithUnit(double number){
    double natural_part = number;
    int i = 0;
    if (number == 0);
    else if (number > 1000) {
        while(natural_part > 1 && i < 3){
            natural_part = natural_part / 1000;
            if (natural_part < 1) {
                natural_part *= 1000;
                break;
            }
            i++;
        }
    }
    else if (number < 1) {
        while (natural_part < 1 && i > -3){
            natural_part = natural_part * 1000;
            if (natural_part > 100){
                natural_part /= 1000;
                break;
            }
            i--;
        }
    }
    printf("%.4f%s\n",natural_part, unitof10pow3(i));
}
```

Answer: About the idea, when I input the number, I will multiply the number with 10^3 or 10^{-3}, if the natural part reaches the normal form (0.1 < x <= 1000) the algorithm will stop.



Hình 1.7: Long press behavior for INC button