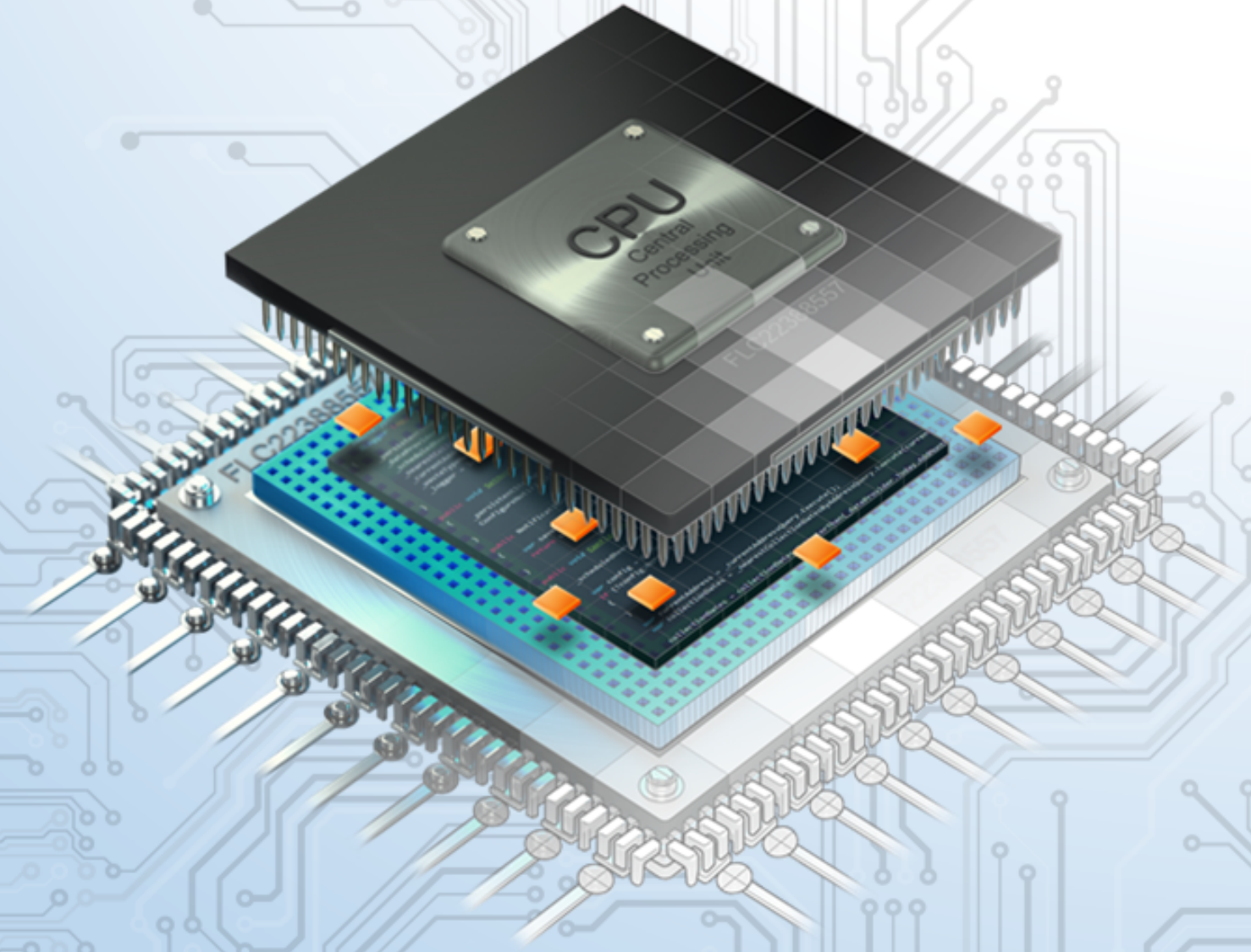
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
COMPUTER ENGINEERING

Microcontroller



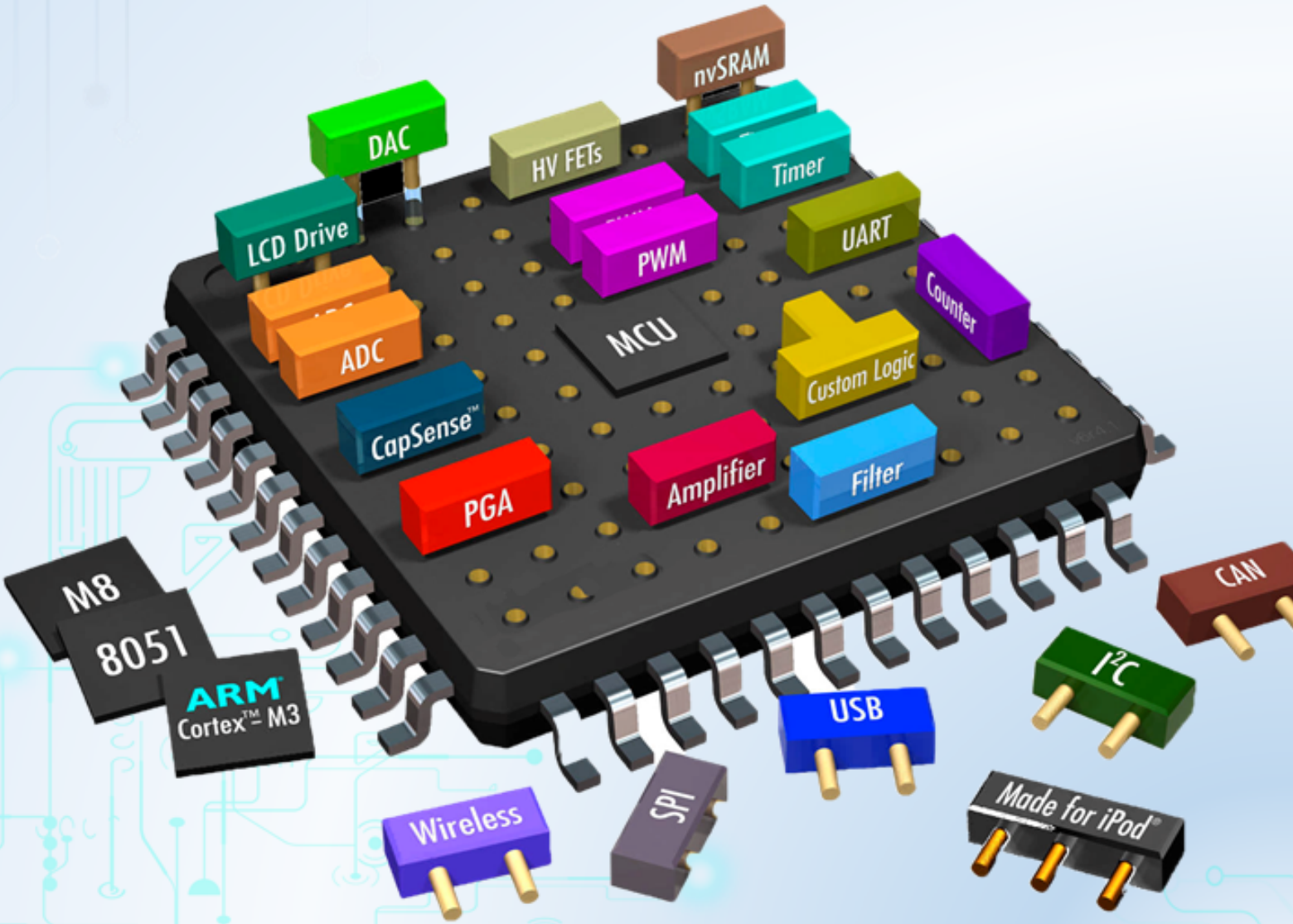
BÁO CÁO MIDTERM TEST MÔN HỌC MCU

Họ và tên: Nguyễn Ngọc Trí
MSSV: 2012286
Lớp lí thuyết: L02

Mục lục

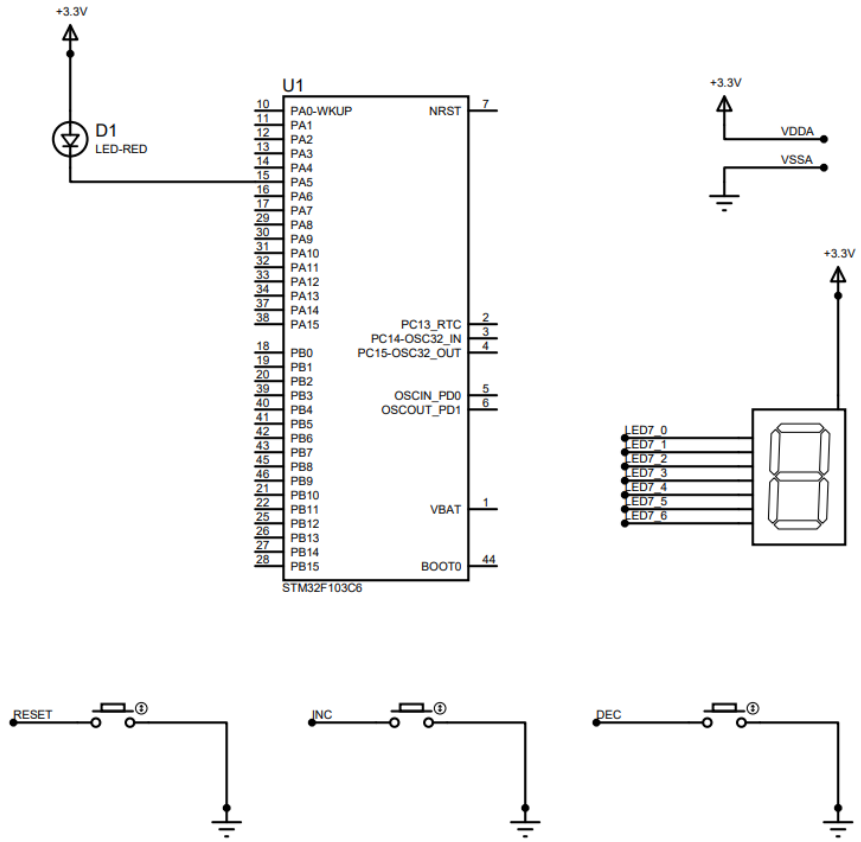
CHƯƠNG 1

GIỮA KÌ 2022



1 Giới thiệu

Trong project giữa kì này, một hệ thống đếm lùi sẽ được hiện thực trên phần mềm mô phỏng Proteus. Như được trình bày ở Hình 1.7, các thành phần chính của hệ thống bao gồm vi điều khiển STM32F103C6, một đèn LED, một LED 7 đoạn và 3 nút nhấn đơn.



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

Một số tính năng chính của hệ thống được trình bày như sau:

- LED 7 đoạn dùng để hiển thị giá trị của counter, có giá trị từ 0 đến 9.
- Nút **RESET** được dùng để reset giá trị của counter về 0. Trong khi đó, nút nhấn **INC** và **DEC** được dùng để tăng hoặc giảm giá trị của counter. Có 2 sự kiện cần phải xử lý cho các nút nhấn, là nhấn thường và nhấn giữa. Trong dự án này, một nút nhấn được coi là nhấn giữ, nếu nó giữ nguyên trạng thái đó trong 3 giây liên tiếp.
- Đèn LED D1 được dùng để theo dõi hoạt động của hệ thống, nó sẽ luân phiên chớp tắt mỗi giây.

Sinh viên sẽ hiện thực dự án của mình theo từng bước yêu cầu ở phần bên dưới. Trong mỗi phần, sinh viên cần trình bày các yêu cầu về report. Một số lưu ý quan trọng cho phần hiện thực như sau:

- Chu kì của ngắt timer là 10ms. Giá trị của counter khi cài đặt timer có thể là 9 hoặc 10 đều được chấp nhận (trong trường hợp này, prescaller là 7999).
- Tất cả các nút nhấn đều được xử lý chống rung bằng ngắt timer 10ms. Một nút nhấn sẽ được xem là nhấn đè nếu nó được giữ liên tục trong 3 giây.
- Không sử dụng HAL_Delay() trong việc hiện thực. Tất cả các hiệu ứng thời gian đều phải được hiện thực dựa trên software timer.
- Report này cần được nộp lại kèm theo các câu trả lời của sinh viên.
- Link github và video demo trong report này được chỉnh quyền truy cập public.

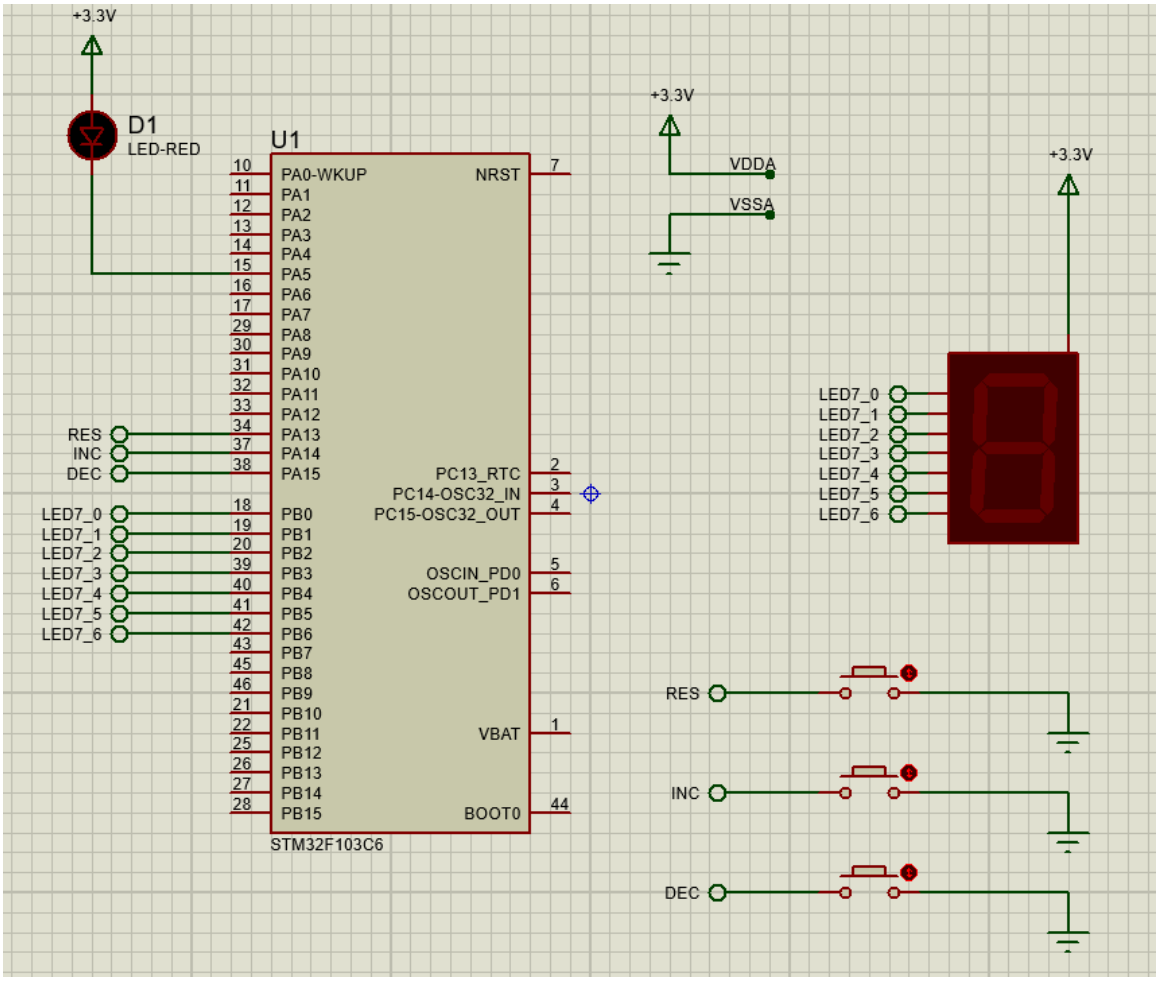
2 Hiện thực và Report

2.1 Sơ đồ nguyên lý trên Proteus - 1 điểm

Trong phần này, sinh viên đề xuất các kết nối của LED 7 đoạn và 3 nút nhấn với STM32F103C6.

Report: Trình bày sơ đồ nguyên lý tại đây. Sinh viên có thể chụp màn hình Proteus cho phần sơ đồ nguyên lý.

Sơ đồ nguyên lý trên Proteus em đề xuất như sau:



Hình 1.2: Report Proteus Schematic

Ở sơ đồ nguyên lý trên em đề xuất các kết nối như sau:
- Đèn LED_RED gắn vào chân PA5.
- Các button RES, INC, DEC ở lần lượt các chân PA13, PA14, PA15.
- 7 chân của LED 7 đoạn được em đặt tên lần lượt là LED7_0, LED7_1, LED7_2, LED7_3, LED7_4, LED7_5, LED7_6 sẽ được gắn lần lượt vào các chân từ PB0 đến PB6.

2.2 State machine Step 1 - 2 điểm

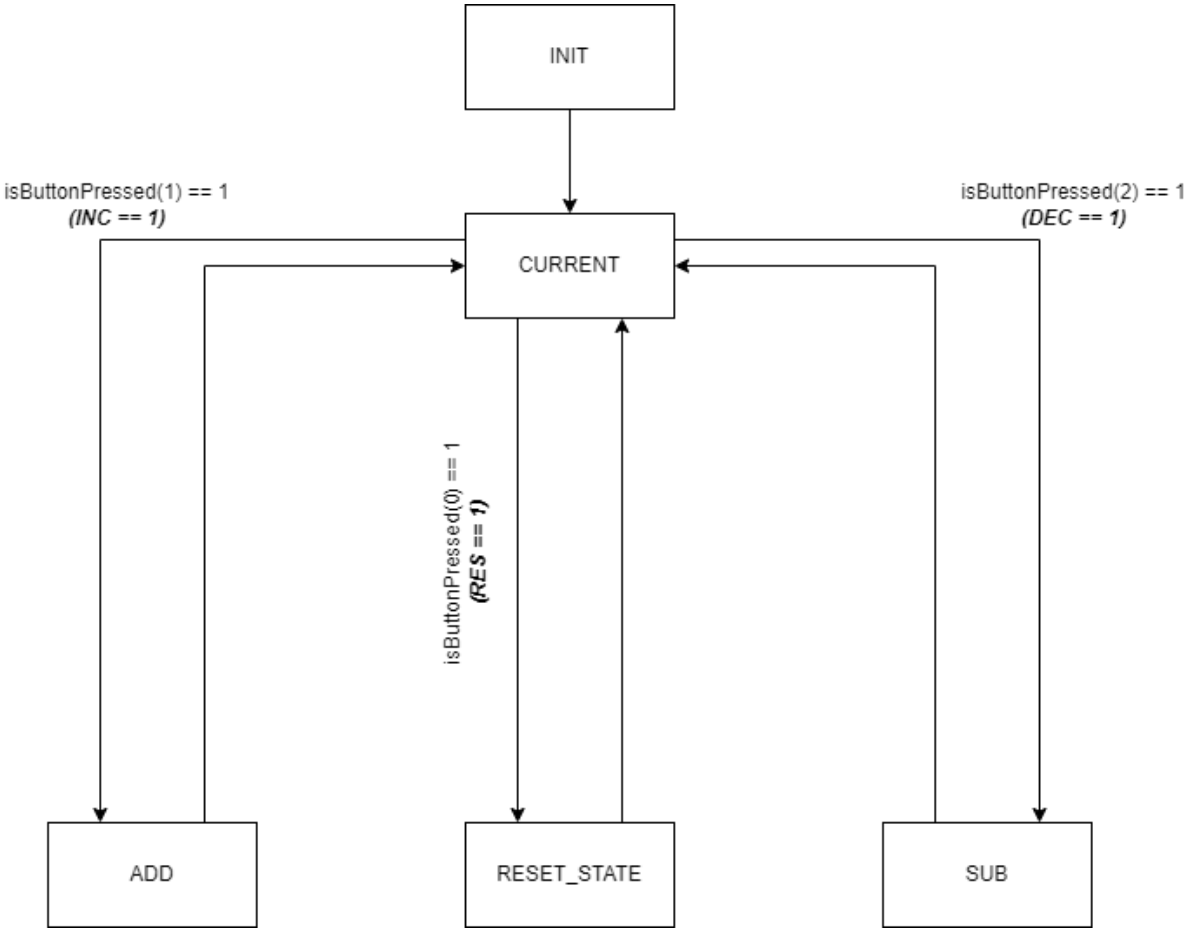
Một máy trạng thái sẽ được thiết kế để thực hiện các chức năng với trạng thái nhấn bình thường (normal-press) cho 3 nút nhấn, như sau:

- Khi nút RESET được nhấn, giá trị counter sẽ là 0.
- Khi INC được nhấn, giá trị của counter tăng 1 đơn vị. Khi nó đến 9, giá trị tiếp theo là 0.
- Khi DEC được nhấn, giá trị của counter giảm 1 đơn vị. Khi giá trị của nó là 0, giá trị tiếp theo là 9.

Giá trị của biến counter được hiển thị lên LED 7 đoạn.

Report: Trình bày máy trạng thái của hệ thống.

Hệ thống em thực hiện ở step 1 sẽ có máy trạng thái như hình dưới:



Hình 1.3: Report Step 1 - Finite State Machine

Giải thích sơ lược về máy trạng thái trên:

- Em đặt một biến có tên là "counter" dùng để hiển thị giá trị ra LED 7 đoạn (giá trị ban đầu được đặt là 0).

- Sơ đồ của em sẽ có trạng thái INIT ban đầu sẽ trỏ về trạng thái CURRENT, đây là trạng thái giúp giá trị hiển thị ra LED 7 đoạn được cố định và cũng từ trạng thái này em sẽ xét các điều kiện nhấn nút để chuyển đến các trạng thái khác.

- Ở trạng thái CURRENT sẽ có thể xảy ra những sự kiện như sau:

+ Nếu ta nhấn nút RES, thì ta sẽ chuyển đến trạng thái RESET_STATE, khi đó giá trị "counter" sẽ đưa về 0, và sau đó sẽ chuyển trạng thái trở về CURRENT.
+ Nếu ta nhấn nút INC, thì ta sẽ chuyển đến trạng thái ADD, khi đó giá trị "counter" sẽ được tăng 1 đơn vị nếu "counter" < 9. Còn nếu "counter" == 9, giá trị tiếp theo là 0. Sau đó ta chuyển trạng thái trở về CURRENT.
+ Nếu ta nhấn nút DEC, thì ta sẽ chuyển đến trạng thái SUB, khi đó giá trị "counter" sẽ giảm đi 1 đơn vị nếu "counter" > 0. Còn nếu "counter" == 0, giá trị tiếp theo là 9. Sau đó ta chuyển trạng thái trở về CURRENT.

- Các trạng thái RESET_STATE, ADD và SUB khi thực hiện xong sẽ trở về trạng thái CURRENT mà không cần điều kiện gì.

Your report: Trình bày hiện thực của hàm dùng để hiển thực máy trạng thái ở trên. Thông thường, hàm này sẽ được gọi trong main().

```
void fsm_simple_buttons_run(){
    //TODO
}
```

Program 1.1: Hiện thực máy trạng thái

Từ máy trạng thái ở trên, em hiện thực hàm fsm_simple_buttons_run() có source code như sau:

```
void fsm_simple_buttons_run() {
  switch(status) { // Xet bien the hien trang thai hien gio
    cua finite state machine

  case INIT: // INIT
    display7SEG(counter); // Hien thi gia tri counter ra
   LED 7 doan
    status = CURRENT; // Trang thai tiep theo: CURRENT
    break;

  case CURRENT: // CURRENT
    display7SEG(counter); // Hien thi gia tri counter ra
   LED 7 doan
    if (isButtonPressed(0) == 1) { // Neu nut RES duoc nhan
```

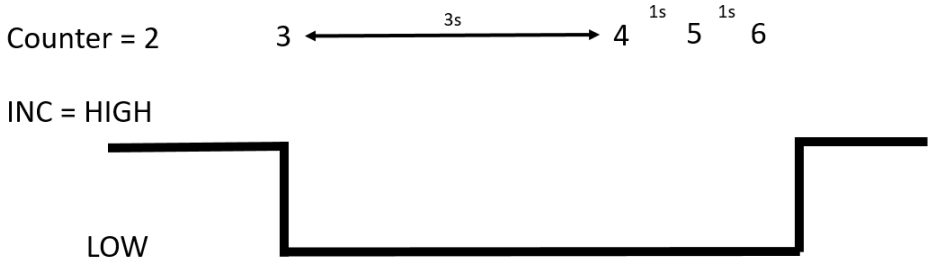
```
    , chuyen trang thai status sang RESET_STATE
      status = RESET_STATE;
    }
    if (isButtonPressed(1) == 1) { // Neu nut INC duoc nhan
    , chuyen trang thai status sang ADD
      status = ADD;
    }
    if (isButtonPressed(2) == 1) { // Neu nut DEC duoc nhan
    , chuyen trang thai status sang SUB
      status = SUB;
    }
    break;

  case RESET_STATE: // RESET_STATE
    counter = 0;  // Reset "counter" ve 0
    display7SEG(counter); // Hien thi gia tri counter ra
    LED 7 doan
    status = CURRENT; // Chuyen trang thai "status" ve
    CURRENT
    break;

  case ADD: // ADD
    counter++;  // Neu gia tri "counter" la 9, thi gia tri
    "counter" tiep theo se la 0, nguoc lai tang gia tri "
    counter"
    if (counter > 9) counter = 0;
    display7SEG(counter); // Hien thi gia tri counter ra
    LED 7 doan
    status = CURRENT; // Chuyen trang thai "status" ve
    CURRENT
    break;

  case SUB: // SUB
    counter--;  // Neu gia tri "counter" la 0, thi gia tri
    "counter" tiep theo se la 9, nguoc lai giam gia tri "
    counter"
    if (counter < 0) counter = 9;
    display7SEG(counter); // Hien thi gia tri counter ra
    LED 7 doan
    status = CURRENT; // Chuyen trang thai "status" ve
    CURRENT
    break;

  default:
    break;
  }
}
```

Program 1.2: Source code function fsm_simple_buttons_run()

2.3 State machine Step 2 - 2 điểm

Trong phần này, việc nhấn giữ (long-press) cho nút INC và DEC được thêm vào trong dự án. Đối với nút nhấn, sự kiện long-press xảy ra khi nó được nhấn giữ liên tục sau 3 giây.

Khi một sự kiện nhấn giữ được phát hiện, giá trị của counter liên tục thay đổi mỗi 1 giây. Ví dụ, giá trị hiện tại của counter là 2. Khi nút INC được nhấn, ngay lập tức giá trị của nó sẽ tăng lên 3. Tuy nhiên, khi INC tiếp tục được giữ, 3 giây sau, sự kiện long-press xảy ra, giá trị của counter tăng lên 4. Từ lúc này, giá trị của counter sẽ tăng lên 1 mỗi giây, cho đến khi nút INC được thả ra. Giá trị của biến counter này được minh họa như hình bên dưới.

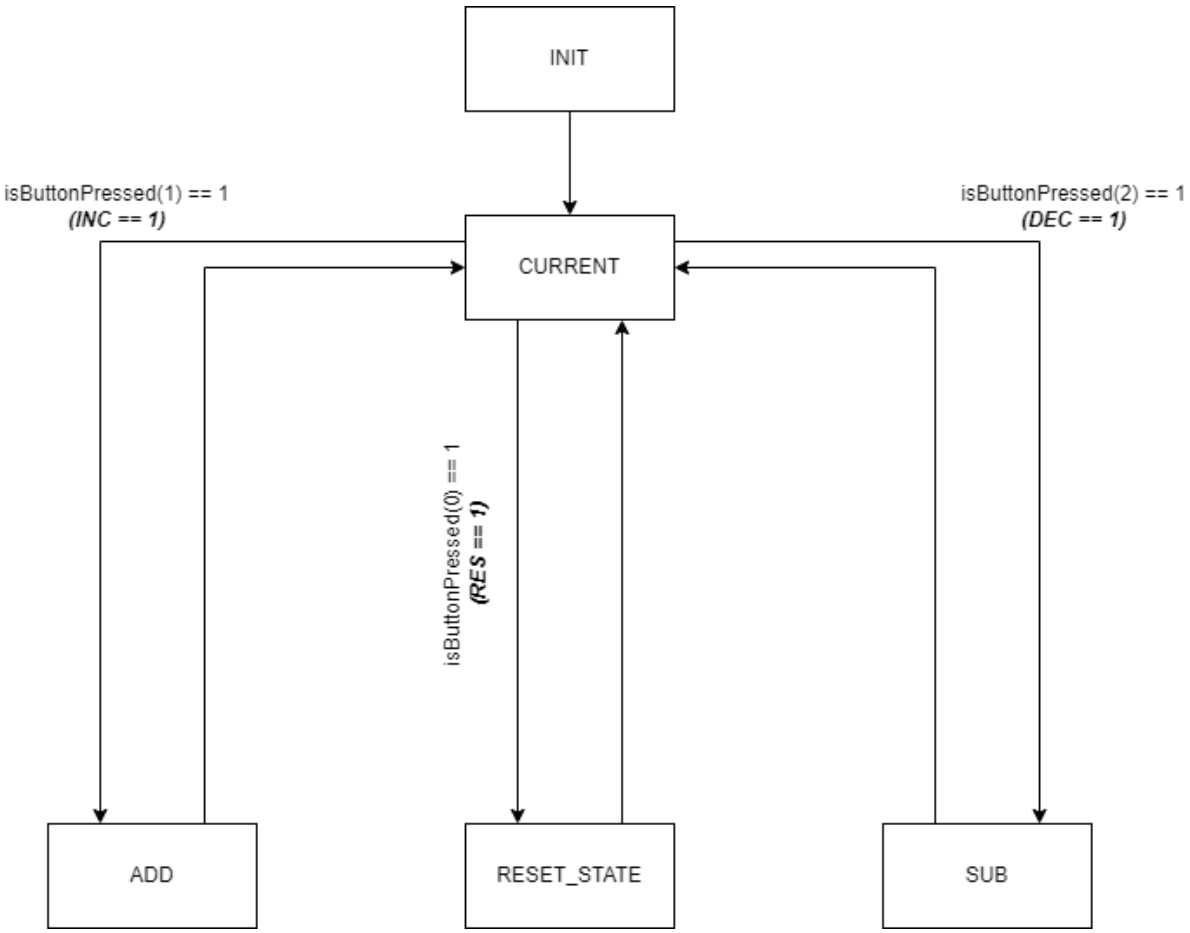


Hình 1.4: *Long press behavior for INC button*

Hành vi của nút DEC sẽ ngược lại với nút INC. Giá trị của biến counter sẽ xoay vòng khi nó đến 0 hoặc 9.

Report: Trình bày toàn bộ máy trạng thái cho đến bước này.

Vì ở step 2 em sẽ cải tiến các hàm hiện thực trong file button.c, nên em không tạo ra trạng thái mới. Hệ thống em thực hiện ở step 2 sẽ có máy trạng thái giống như máy trạng thái ở step 1 như hình dưới:



Hình 1.5: *Report Step 2 - Finite State Machine*

Report: Sinh viên trình bày một hàm chính dùng để hiện thực các trạng thái mới. Các thay đổi trên máy trạng thái cũ không cần phải trình bày ở đây.

Mặc dù không có trạng thái mới nào nhưng cơ chế của máy trạng thái step 2 sẽ có sự thay đổi so với máy trạng thái ở step 1 mà cụ thể ở cơ chế xử lí nút nhấn ở file button.c.

Ý tưởng của em về cơ chế nhấn đè sẽ là set biến TimeOutForKeyPress đối với mỗi nút nhấn là 3s khi mới bắt đầu nhấn đè. Sau đó nếu sau 3s mà nút nhấn vẫn còn đè thì cờ buttonlong_flag sẽ tick lên bằng 1, báo hiệu sự kiện nhấn đè. Sau đó nếu vẫn còn nhấn đè thì TimeOutForKeyPress sẽ được set mỗi 1 giây để có thể xét trường hợp cho các biến counter có thể thay đổi giá trị trong quá trình nhấn đè.

Ở đây do có nút RES khi nhấn bình thường hay nhấn đè đều sẽ không có sự khác biệt khi chức năng duy nhất của nó là reset biến counter về 0 cho nên ở đây em sẽ lập trình nhấn tương tự cho cả 3 nút nhấn RES, INC và DEC.

Cụ thể code có sự thay đổi như sau:

File button.h:

```c
#include "main.h"
#include "software_timer.h"


#define NORMAL_STATE SET
#define PRESSED_STATE RESET

#define NUM_OF_BUTTONS 3  // Number of Buttons to use
#define TIMEOUTFORLONGKEYPRESS 3000/TIME_CYCLE  // So chu ki dem nguoc cho 3s (thoi gian dieu kien cho longkeypress)

int isButtonPressed(int index);

void getKeyInput();
```

File button.c:

```c
#include "button.h"

// Khoi tao cac bien de xu li chong rung, xu li tin hieu, xu li nut nhan o hai che do (nhan thuong va nhan giu)
int KeyReg0[NUM_OF_BUTTONS] = {NORMAL_STATE};
int KeyReg1[NUM_OF_BUTTONS] = {NORMAL_STATE};
int KeyReg2[NUM_OF_BUTTONS] = {NORMAL_STATE};
int KeyReg3[NUM_OF_BUTTONS] = {NORMAL_STATE};

// Khoi tao cac flag cho tung button: button_flag (nhan thuong) va buttonlong_flag (nhan giu)
int button_flag[NUM_OF_BUTTONS] = {0};
int buttonlong_flag[NUM_OF_BUTTONS] = {0};

// Khoi tao cac bien dem chu ki de xac dinh hai su kien co the xay ra cua nut nhan
int TimeOutForKeyPress[NUM_OF_BUTTONS] = {
   TIMEOUTFORLONGKEYPRESS};

int isButtonPressed(int index) {
  if (button_flag[index] == 1) {
    button_flag[index] = 0;
    return 1;
  }
  return 0;
}

void subKeyProcess(int index){
  button_flag[index] = 1;
}

void getKeyInput() {
```

```
29  for (int i = 0; i < NUM_OF_BUTTONS; i++) {
30    KeyReg0[i] = KeyReg1[i];  // Xu li chong rung
31    KeyReg1[i] = KeyReg2[i];
32
33    // Doc cac tin hieu tu cac nut nhan
34    if (i == 0) {
35      KeyReg2[i] = HAL_GPIO_ReadPin(RES_GPIO_Port, RES_Pin)
    ;
36    }
37    else if (i == 1) {
38      KeyReg2[i] = HAL_GPIO_ReadPin(INC_GPIO_Port, INC_Pin)
    ;
39    }
40    else if (i == 2) {
41      KeyReg2[i] = HAL_GPIO_ReadPin(DEC_GPIO_Port, DEC_Pin)
    ;
42    }
43
44    // Xu li tin hieu
45    if ((KeyReg1[i] == KeyReg0[i]) && (KeyReg1[i] ==
    KeyReg2[i])){
46      if (KeyReg3[i] != KeyReg2[i]){
47        KeyReg3[i] = KeyReg2[i];
48        if (KeyReg2[i] == PRESSED_STATE){
49          subKeyProcess(i); // Kich timer_flag[i] = 1;
50          if (buttonlong_flag[i] == 0) {
51            TimeOutForKeyPress[i] = TIMEOUTFORLONGKEYPRESS;
     // Set thoi gian 3s (dieu kien cho su kien longkeyPress
     dien ra)
52          }
53          else {
54            TimeOutForKeyPress[i] = 1000 / TIME_CYCLE;
    // Set thoi gian 1s (dieu kien cho nhung lan bien "
    counter" se tang gia tri (chi su dung cho button INC va
    DEC))
55          }
56        }
57      } else {
58        TimeOutForKeyPress[i]--;
59        // Neu sau khoang thoi gian TimeOutForKeyPress lan
    dau tien (3s) ma van con nhan giu button, thi
    buttonlong_flag[i] se bang 1, nguoc lai se bang 0
60        if (TimeOutForKeyPress[i] == 0){
61          KeyReg3[i] = NORMAL_STATE;
62          buttonlong_flag[i] = 1;
63        }
64      }
65    }
66    else {
```

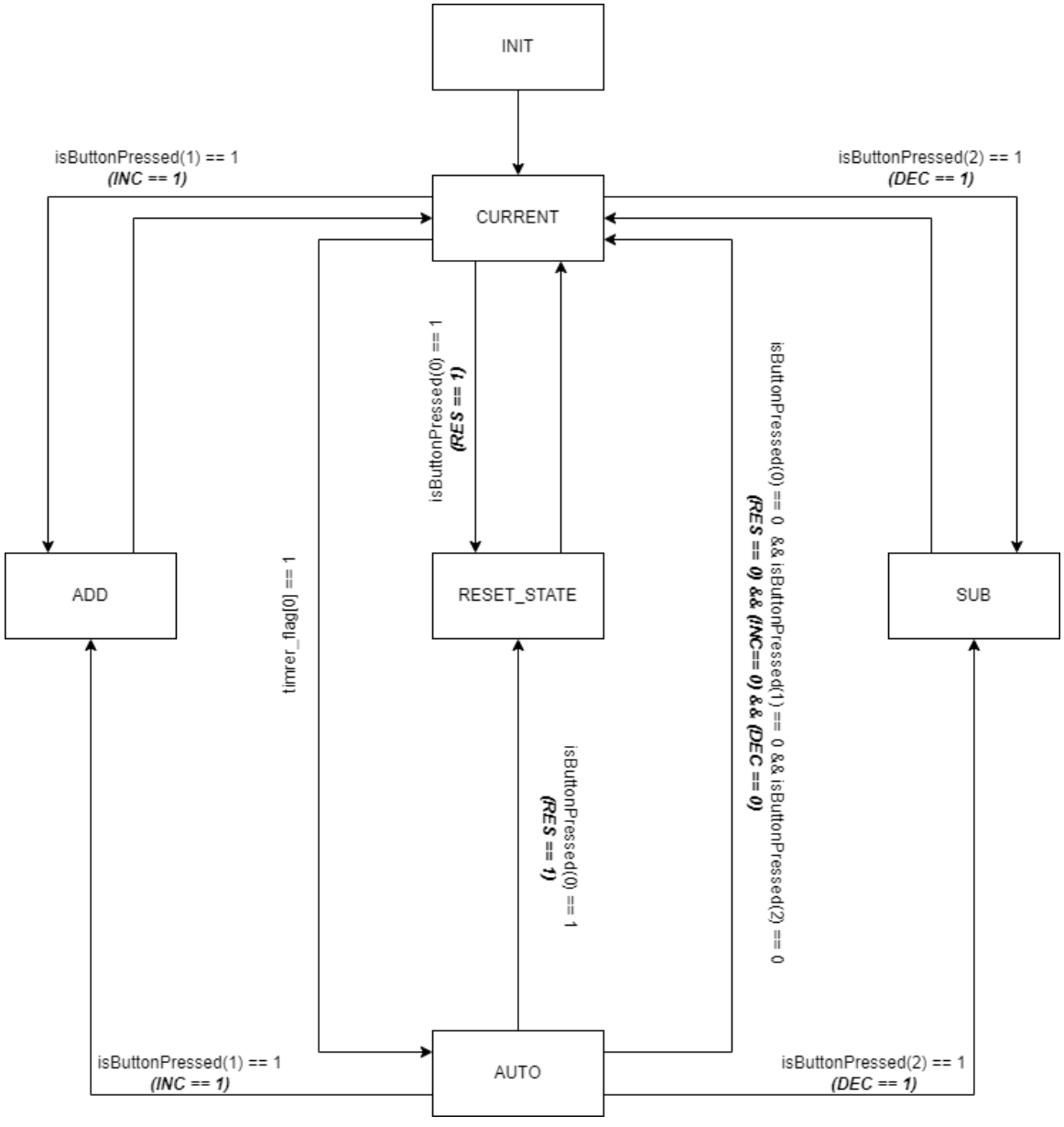
```
67          buttonlong_flag[i] = 0;
68        }
69      }
70  }
```

2.4 State machine Step 3 - 2 points

Cuối cùng, khi không có nút nào được nhấn, hệ thống tự động đếm lùi biến counter và dừng lại khi nó bằng 0. Sau đó, nếu nút INC hoặc DEC được nhấn, trạng thái của hệ thống lại quay về một trong các trạng thái đã thiết kế trước đó.

Your report: Trình bày toàn bộ máy trạng thái, khi sự kiện time-out 10s được thêm vào.

Hệ thống em thực hiện ở step 3 sẽ có máy trạng thái như hình dưới:



Hình 1.6: Report Step 3 - Finite State Machine

Giải thích sơ lược về máy trạng thái trên:

- Ngoài những trạng thái cũ đã có sẵn ở step 2 đã được trình bày ở trên, em sẽ bổ sung vào máy trạng thái trạng thái AUTO sẽ có cơ chế như sau:

+ Em sẽ đặt một setTimer có thời gian là 10s để đếm ngược thời gian mà hệ thống không tác động bởi nút nhấn nào.

+ Các trạng thái ADD, RESET_STATE hay SUB đều không có chiều hướng về trạng thái AUTO do khi đó nút vừa mới nhấn, khi nút mới nhấn thì ta lập tức setTimer thời gian lại là 10s.

+ Trạng thái duy nhất có thể chuyển tới trạng thái AUTO là trạng thái CURRENT. Ở đây nếu như timer_flag dành cho quá trình tĩnh này báo tín hiệu 1, thì ta sẽ chuyển sang trạng thái AUTO.

+ Ở trạng thái AUTO ta sẽ trừ biến counter mỗi 1 giây cho đến khi counter == 0 thì giữ nguyên biến counter như vậy. Để có thể thay đổi giá trị biến counter mỗi giây như thế thì sau khi qua trạng thái AUTO ta sẽ setTimer lại chỉ còn là 1s. Khi đã thay đổi biến counter xong thì trạng thái AUTO sẽ trở về trạng thái CURRENT để chờ cho sự kiện tiếp theo xảy ra.

+ Tuy nhiên nếu ở trạng thái AUTO mà bất kì nút nhấn nào được thực hiện thì ta sẽ lập tức chuyển tới trạng thái tương ứng đồng thời setTimer lại là 10s.

Report: Sinh viên trình bày một hàm chính dùng để hiện thực các trạng thái mới. Các thay đổi trên máy trạng thái cũ không cần phải trình bày ở đây.

Từ máy trạng thái ở trên, em sẽ cải tiến hàm fsm_simple_buttons_run() hoàn chỉnh hơn. Hàm có source code như sau:

```
1  void fsm_simple_buttons_run() {
2    switch(status) {  // Xet bien the hien trang thai hien
     gio cua finite state machine
3
4    case INIT: // INIT
5      display7SEG(counter);   // Hien thi gia tri counter ra
     LED 7 doan
6      status = CURRENT; // Trang thai tiep theo: CURRENT
7      break;
8
9    case CURRENT: // CURRENT
10     display7SEG(counter); // Hien thi gia tri counter ra
     LED 7 doan
11     if (isButtonPressed(0) == 1) {  // Neu nut RES duoc
     nhan, chuyen trang thai status sang RESET_STATE, dong
```

```
    thoi ta dat lai setTimer cho dieu kien chuyen sang trang
     thai AUTO sang 10s
12      status = RESET_STATE;
13      setTimer(0, 10000); // 10s
14    }
15    if (isButtonPressed(1) == 1) {  // Neu nut INC duoc
    nhan, chuyen trang thai status sang ADD, dong thoi ta
    dat lai setTimer cho dieu kien chuyen sang trang thai
    AUTO sang 10s
16      status = ADD;
17      setTimer(0, 10000); // 10s
18    }
19    if (isButtonPressed(2) == 1) {  // Neu nut DEC duoc
    nhan, chuyen trang thai status sang SUB, dong thoi ta
    dat lai setTimer cho dieu kien chuyen sang trang thai
    AUTO sang 10s
20      status = SUB;
21      setTimer(0, 10000); // 10s
22    }
23    if (timer_flag[0] == 1) { // Neu sau 10s khong co nut
    nao duoc nhan, ta chuyen sang trang thai AUTO
24      status = AUTO;
25    }
26    break;
27
28  case RESET_STATE: // RESET_STATE
29    counter = 0;  // Reset "counter" ve 0
30    display7SEG(counter);  // Hien thi gia tri counter ra
    LED 7 doan
31    status = CURRENT; // Chuyen trang thai "status" ve
    CURRENT
32    break;
33
34  case ADD: // ADD
35    counter++;    // Neu gia tri "counter" la 9, thi gia
    tri "counter" tiep theo se la 0, nguoc lai tang gia tri
    "counter"
36    if (counter > 9) counter = 0;
37    display7SEG(counter); // Hien thi gia tri counter ra
    LED 7 doan
38    status = CURRENT; // Chuyen trang thai "status" ve
    CURRENT
39    break;
40
41  case SUB: // SUB
42    counter--;  // Neu gia tri "counter" la 0, thi gia tri
    "counter" tiep theo se la 9, nguoc lai giam gia tri "
    counter"
43    if (counter < 0) counter = 9;
```

```
44      display7SEG(counter); // Hien thi gia tri counter ra
    LED 7 doan
45      status = CURRENT; // Chuyen trang thai "status" ve
    CURRENT
46      break;
47
48    case AUTO: // AUTO
49      if (counter > 0) counter--; // Neu gia tri "counter" >
    0, thi gia tri "counter" tiep theo se giam di 1 don vi
50        // Con neu "counter" == 0, th  "counter" se duoc giu
     nguyen
51      display7SEG(counter); // Hien thi gia tri counter ra
    LED 7 doan
52      setTimer(0,1000); // setTimer cho nhung lan tu dong
    thay doi "counter" tiep theo la 1s
53      status = CURRENT; // Neu trong qua trinh nay khong xay
    ra nhan nut nao, thi "status" se duoc chuyen ve CURRENT
54      if (isButtonPressed(0) == 1) {  // Neu nut RES duoc
    nhan, chuyen trang thai status sang RESET_STATE, dong
    thoi ta dat lai setTimer cho dieu kien chuyen sang trang
     thai AUTO thanh 10s
55        status = RESET_STATE;
56        setTimer(0, 10000);
57      }
58      if (isButtonPressed(1) == 1) {  // Neu nut INC duoc
    nhan, chuyen trang thai status sang ADD, dong thoi ta
    dat lai setTimer cho dieu kien chuyen sang trang thai
    AUTO thanh 10s
59        status = ADD;
60        setTimer(0, 10000);
61      }
62      if (isButtonPressed(2) == 1) {  // Neu nut DEC duoc
    nhan, chuyen trang thai status sang SUB, dong thoi ta
    dat lai setTimer cho dieu kien chuyen sang trang thai
    AUTO thanh 10s
63        status = SUB;
64        setTimer(0, 10000);
65      }
66      break;
67
68    default:
69      break;
70    }
71  }
```

Program 1.3: Source code function fsm_simple_buttons_run()

2.5 Led Blinky for Debugging - 1 điểm

Cuối cùng, trong nhiều dự án dựa trên vi điều khiển, có 1 LED luôn luôn chớp tắt mỗi giây, để giám sát hoạt động của hệ thống. Trong project này, LED nối với chân PA5 sẽ được dùng để hiện thực tính năng này.

Report: Sinh viên trình bày giải pháp của mình cho tính năng này. Giải pháp có thể rất đơn giản với vài dòng code hoặc thiết kế máy trạng thái và lập trình cho nó. Trong trường hợp thứ 2, sinh viên trình bày máy trạng thái trước khi trình bày phần hiện thực mã nguồn.

Về cơ chế Led_Blinky này em sẽ thực hiện giải pháp đơn giản là sử dụng một setTimer khác (trong sourcecode của em sẽ là setTimer[1] để phân biệt với setTimer[0] dùng cho quá trình máy trạng thái tĩnh không nhấn nút). Với setTimer này em sẽ đặt thời gian là 1s. Cứ sau 1s thì flag sẽ báo và đèn LED_RED sẽ toggle.

Source code hàm led_blinky sẽ như sau:

```
void led_blinky() {
  // Cu sau 1s, den RED_LED se toggle
  if (timer_flag[1] == 1) {
    setTimer(1, 1000);
    HAL_GPIO_TogglePin(RED_GPIO_Port, RED_Pin);
  }
}
```

Program 1.4: Source code led_blinky() function

Để đơn giản và gọn thì trong project em sẽ để hàm led_blinky này chung module với hàm fsm_simple_buttons_run().

2.6 Github và Demo

Link github cho dự án của sinh viên được trình bày ở đây, bao gồm tất cả các files trong dự án (configurations, header and source files). Link này là lần commit sau cùng trong project giữa kì của sinh viên.

Your github link

Link cho 1 video demo cũng sẽ được trình bày ở phần này.

Your video link

3 Bài tập thêm - Engineer mindset - 1 điểm

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

EXample:

1 suffixWithUnit(123) => 123

2 suffixWithUnit(1234) => 1.234 Kilo

3 suffixWithUnit(12345) => 12.345 Kilo

4 suffixWithUnit(1234567) => 1.234567 Mega

5 suffixWithUnit(12345678) => 12.345678 Mega

Prototype

```
string suffixWithUnit(double number) {
}
```

How would you solve them? Please share your thinking to solve this problem and provide your answer.

Ý tưởng thực hiện của em với vấn đề này như sau:

- Chuyển number từ double sang string.

- Tìm vị trí của dấu chấm "." (nếu có) và xóa dấu chấm đó, sau đó ta xác định vị trí trước vị trí đang trỏ tới thì đó chính là hàng đơn vị của số number.

- Thông qua độ lớn của số mà dùng vòng lặp while để vừa tìm số chữ số chuyển dịch, vừa tìm đơn vị hậu tố tương ứng.

- Thêm dấu chấm vào vị trí mới cần chèn.

- Sau đó ta loại bỏ những giá trị số "0" thừa hoặc là dấu "." (nếu số đó là số tự nhiên).

- Cuối cùng ta nối string của số mới và hậu tố tìm được phía trên để trả về kết quả.

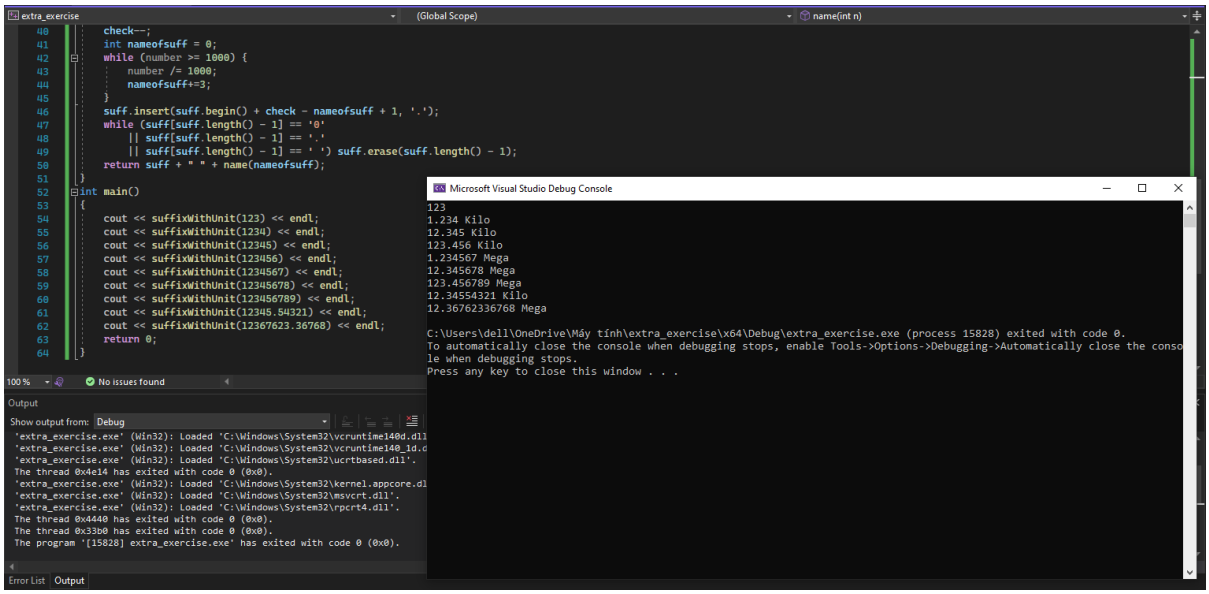
Source code như sau:

```
string name(int n) {
    switch (n) {
    case 0:
        return "";
    case 3:
        return "Kilo";
    case 6:
        return "Mega";
    case 9:
        return "Giga";
    case 12:
        return "Tera";
    case 15:
        return "Peta";
    case 18:
        return "Exa";
    case 21:
        return "Zeta";
    case 24:
        return "Yota";
    case 27:
        return "Bronto";
    case 30:
        return "Geop";
    default:
        return "No match!";
    }
}

string suffixWithUnit(double number) {
    string suff = to_string(number);
    int check = 0;
    while (suff[check] != '.') check++;
    suff.erase(suff.begin() + check);
    check--;
    int nameofsuff = 0;
    while (number >= 1000) {
        number /= 1000;
        nameofsuff+=3;
    }
    suff.insert(suff.begin() + check - nameofsuff + 1, '.')
   ;
    while (suff[suff.length() - 1] == '0'
        || suff[suff.length() - 1] == '.') suff.erase(suff.
   length() - 1);
```

```
44      return suff + " " + name(nameofsuff);
45  }
```

Program 1.5: Report source code Extra Exercise

Kết quả chạy thử trong Visual Studio:



Hình 1.7: Report Extra Exercise - Kết quả chạy thử trên Visual Studio