آزمایش 3

طاها موسوی 98243058

نیلوفر مرادی جم 97243063

گروه 2

سوالات تحلیلی:

1- انواع وقفه را نام برده و با یکدیگر مقایسه کنید. اگر اتفاق افتادن چندین درخواست وقفه دارای اشتراکی زمانی باشد، عملکرد سیستم در M4-Cortex و قبل از آن را توضیح داده و با هم مقایسه کنید. اگر دو درخواست وقفه همزمان رخ دهند، سیستم به چه صورت عمل خواهد کرد؟

4 نوع وقفه (interrupt) داریم:

اول) نوع اول interrupt requests است که آسنکرون می‌باشد و به کدی که در آن زمان اجرا می‌شود ارتباط ندارد. در این نوع وقفه یک سیگنال به پروسسور ارسال می شود تا فوری فرآیند فعلی متوقف شود و روال interrupt handler اجرا شود.

دوم) نوع دوم non-maskable interrupt است که همانند وقفه نوع اول می‌باشد با این تفاوت که قابل چشم پوشی نیست. این نوع وقفه معمولا در شرایطی ایجاد می‌شود که یک مشکل غیر قابل حل در سیستم هست.

سوم) نوع سوم Exceprtion ها و مشکلاتیند که توسط هسته پردازنده ایجاد شدند. این وقفه‌ها مربوط به اجرا شدن اینستراکشن های خاصیند همانند مثلا وقتی overflow رخ می‌دهد.

چهارم) نوع چهارم آن نیز systick timer است که وقفه هایی با زمان اجرای مشخص ایجاد می‌کند. سورس کلاک آن مانند cortex-m cpu است.

وقفه هایی متفاوت priority متفاوت دارند که در صورتی که همزمان رخ دهند، پردازنده هر کدام که عدد priorityکوچکتری داشته باشد را در الویت قرار می‌دهد.

2 - تفاوت روش سرکشی و وقفه چیست؟

اول) سرعت: در سرکشی باید دقیق چک کنیم فرآیند موردنظر اتفاق افتاده یا نه و بنابرین سرعت کمتری دارد اما در وقفه فرآیند سخت‌افزاری است پس سرعتش بیشتر است.

دوم) بهره زمانی و بهره وری: در سرکشی هر چه پاسخ سریع‌تری بخواهیم، باید بیشتر چکش کنیم و بنابرین وقت زیادی می‌گیرد اما وقفه به دلیل سرعت بالا، بهره وری زمانی بهتری دارد. همچنین برخلاف سرکشی، کد وقفه تنها در زمان مورد نیاز اجرا می‌شود پس اجرای تکراری نداریم.

سوم) هزینه توسعه: اگر در یک فرآیند بزرگ وقفه بخواهیم، بدلیل وابستگی اجرای وقفه به بسیاری از فرآیند های دیگر، امکان توسعه چند بخش به صورت همزمان وجود ندارد و در مقیاس بزرگ به کار بسیار زیادی نیازمندیم. اما در وقفه هر بخش از کد به صورت غیر وابسته به بقیه اجزای کد اجرا می شود و امکان توسعه همزمان بخش های مختلف وجود دارد و هزینه توسعه آن نیز کاهش می یابد.

Polling
use software to check it

- Slow - need to explicitly check to see if switch is pressed
- Wasteful of CPU time - the faster a response we need, the more often we need to check
- Scales badly - difficult to build system with many activities which can respond quickly. Response time depends on all other processing.

Interrupt
Use HW to detect event and run ISR

- Efficient - code runs only when necessary
- Fast - hardware mechanism
- Scales well
  - ISR response time doesn't depend on most other processing.
  - Code modules can be developed independently

3- جا به جایی بردار وقفه در M4-Cortex را توضیح دهید. مزیت آن چیست؟

بردار وقفه در واقع حافظه‌ای می‌باشد که آدرس ابتدای ISR در آن ذخیره می‌شود. جابه‌جایی بردار وقفه نیز با یک رجیستر قابل برنامه‌ریزی با نام Vector Table Offset Register انجام می‌شود.

موارد استفاده :

- برنامه هایی که boot loader دارند
- ذخیره کردن برنامه ها در رم
- تغییر داینامیک بردار وقفه در حافظه ای

4 – در هنگام وقوع وقفه چه اطلاعاتی در LR قرار میگیرد؟ از زمان وقوع وقفه (با فرض عدم وجود وقفه ای دیگر) تا زمان شروع اجرای اولین روتین سرویس وقفه، حداکثر چند سیکل CPU طور میکشد؟

در هنگام وقفه به همراه LR ، EXC_RETURN کد ذخیره شده که CPU آن را تولید می‌کند. در این کد اطلاعاتی همانند این که رجیسترها از کدام STACK POINTER(SP) باید خوانده شوند و این که به کدام مد باید برگردیم (مثلا هنگام مد هندلر یا نخ در حال اجرا(ترد)) و با بیت های 4 تا 7 نیز این که از واحد FLOATING POINT استفاده کرده ایم یا نه، ذخیره می شود.

تنها 12 سایکل از زمان درخواست exception تا اجرای اولین اینستراکشن را می‌دهند.

رفرنس های سوالات تحلیلی:

- کلاس درس و اسلاید های درسی

دستور کار:

طبق صورت سوال باید یک ماشین حساب طراحی میکردیم.

موارد مورد استفاده:

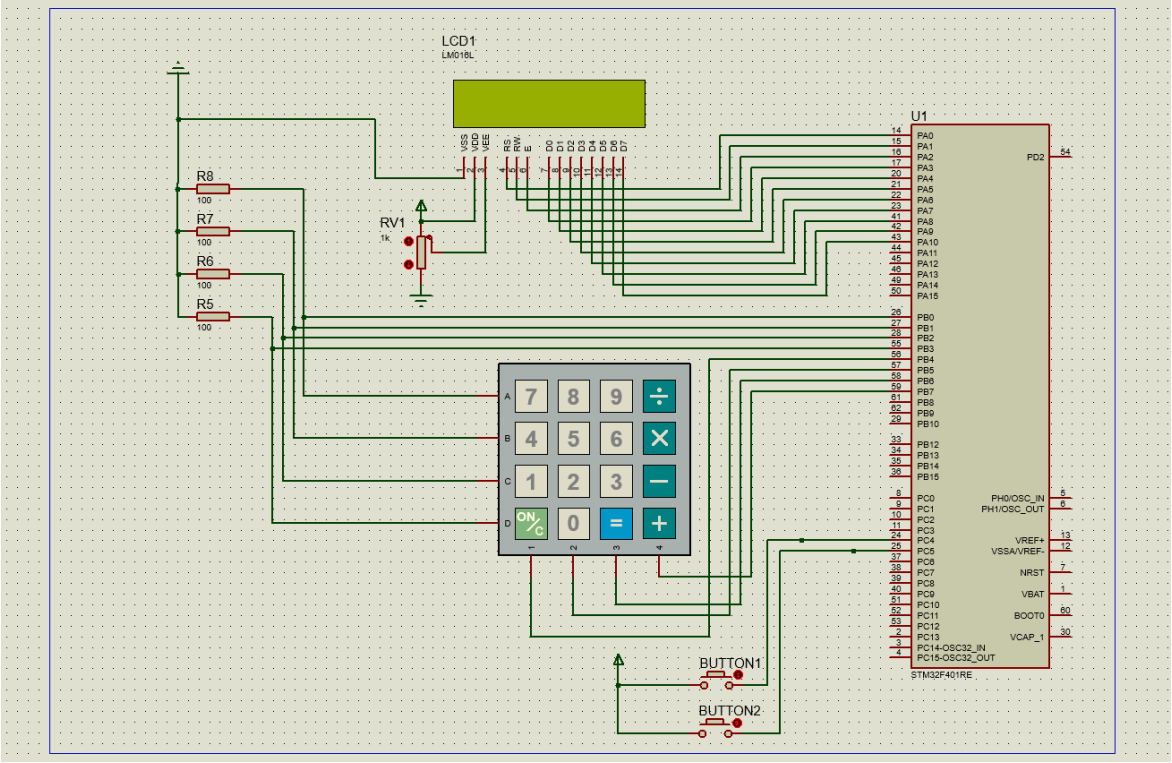
Lcd

Keypad

Stm32f401 microcontroller

buttons

برای این بخش از قالب کد ارائه شده در گروه درس استفاده کردیم و یکسری روتین های موردنظرمان را به آن می باشد که keypad هم مختص GPIOB .تخصیص داده ایم LED را به GPIOA اضافه کردیم. پین های 0 تا 10 در چهارپین اول ورودی به میکرو و چهارپین دوم را خرو جی از میکرو قرار داده ایم.



در ادامه لازم است کد c برای میکروکنترلر را طراحی کنیم و فایل هگز آن را برای اجرای برنامه بدهیم.

تعریف متغیرها و توابع:

```
#include <stm32f4xx.h>
#define RS 0x01     // Pin mask for reg select
#define RW 0x02     // Pin mask for read/write
#define EN 0x04     // Pin mask for enable
#define MASK(x) (1UL << (x))

volatile static long long int firstOperand = 0 , secondOperand = 0 ,result = 0 ,isFirstOPExist = 0;
volatile static unsigned int zero_counter = 0;
volatile static char operation = '0';
volatile static unsigned int equalClicked = 0;
volatile static unsigned int LCD_line = 0;
volatile static int numIncBtn1Cick = 0, numOfDecCick = 0;

void delay(int n);
void printNames(void);
void LCD_command(unsigned char command);
void LCD_data(char data);
void setGPIOConfig(void);
void enableInterrupts(void);
void initializingLCD(void);
void initializingPorts(void);
long long int reverse(long long int);
void LCD_display(long long int );
void calculateResult(void);
void EXTI0_IRQHandler(void);
void EXTI1_IRQHandler(void);
void EXTI2_IRQHandler(void);
void EXTI3_IRQHandler(void);
void increseResult(long long int num);
void decreaseResult(long long int num);
void clearSecondRow(void);

```

FirstOperand و secondOperand دو عملوند ما در عملیات ها هستند. وقتی که یک اپراتور را وارد می کنیم متغیر isFirstOpExist یک میشود و می فهمیم شماره دوم باید وارد شود .متغیر equalClicked اگر یک بشود باید نتیجه را چاپ کنیم.

توابع افزایش و کاهش نتیجه:

```
void increseResult(long long int num){
  num += numIncBtn1Cick;
  clearSecondRow();
  LCD_display(num);
  numIncBtn1Cick = 0 ;
  result = num;
}

void decreaseResult(long long int num){
  num -= numOfDecCick;
  clearSecondRow();
  LCD_display(num);
  numOfDecCick = 0;
  result = num;
}
```

تابع محاسبه:

```
void calculateResult(void){
  switch(operation){
        case '*':result = firstOperand * secondOperand;break;
        case '+':result = firstOperand + secondOperand;break;
        case '/':result = firstOperand / secondOperand;break;
        case '-':result = firstOperand - secondOperand;break;
      }
      //result is the first number for continue typing or calculating with it
      firstOperand = result;
      isFirstOPExist = 0;
      secondOperand = 0;
      equalClicked = 0;

      LCD_command(0xC0);
      for(int i =0 ; i<16;i++){
        LCD_data(' ');
      }
      LCD_display(result);
}
```

با توجه به نوع عملیات، آن را آنجام داده و در result میریزیم و سپس با تابع lcd_display آن را نمایش میدهیم.

برای چاپ کردن دو روتین داریم یک روتین هرچه ورودی بگیرد را چاپ می‌کند (lcd_data) و دیگری برای نمایش اعداد است که در ادامه دومی آمده و سپس اولی.

```
void LCD_display(long long int num){
   LCD_command(0xC0);
   delay(13);
   if(num==0){
     LCD_data('0');
     return;
   }else if(num < 0){
     LCD_data('-');
     num *= -1;

   }
   long long int reversedNumber= reverse(num);
   for(;reversedNumber > 0;reversedNumber/=10){
        LCD_data(reversedNumber%10 + '0');
     }
     while(zero_counter > 0){
        LCD_data('0');
        zero_counter--;
     }


}
```

در روتین display_LCD به کمک جدول زیر ابتدا کامند پاکسازی صفحه نمایشگر صدا زده می شود و سپس عدد ورودی بعنوان پارامتر را نمایش می دهیم

.روتین اول هم data_LCD است که هرچه ورودی بگیرد چاپ می کند و ما از آن برای نمایش حروف و کاراکتر ها بهره جسته ایم.

```
void LCD_data(char data) {
    // RS = 1
    GPIOA->ODR |= RS;
    // R/W = 0
    GPIOA->ODR &= ~(RW);
    GPIOA->ODR &= (0X007);
    // put data on data bus
    GPIOA->ODR |= data << 3;
    // pulse EN high
    GPIOA->ODR |= EN;
    delay(13);
    // clear EN
    GPIOA->ODR &= ~(EN);
    delay(13);
}
```

روتین دیگری داریم برای ارسال کامند های طبق جدول کامند های زیر

ردیف	دستور	معادل هگزادسیمال
۱	نمایش در یک سطر با آرایه‌های ۵×۷ در مد هشت بیتی	0x30
۲	نمایش در دو سطر با آرایه‌های ۵×۷ در مد هشت بیتی	0x38
۳	نمایش در یک سطر با آرایه‌های ۵×۷ در مد چهار بیتی	0x20
۴	نمایش در دو سطر با آرایه‌های ۵×۷ در مد چهار بیتی	0x28
۵	مد ورود داده‌ها	0x06
۶	خاموش کردن نشانگر و نمایشگر بدون پاک شدن محتویات RAM	0x08
۷	روشن کردن نشانگر و نمایشگر	0x0E
۸	روشن کردن نمایشگر بدون روشن کردن نشانگر	0x0C
۹	نمایش اطلاعات با نشانگر چشمک‌زن	0x0F
۱۰	شیفت دادن همه اطلاعات در حال نمایش به سمت چپ	0x18
۱۱	شیفت دادن همه اطلاعات در حال نمایش به سمت راست	0x1C
۱۲	انتقال نشانگر به سمت چپ به مقدار یک کاراکتر	0x10
۱۳	انتقال نشانگر به سمت راست به مقدار یک کاراکتر	0x14
۱۴	پاک کردن کامل نمایشگر به همراه محتویات RAM	0x01
۱۵	انتقال نشانگر به اولین مکان از اولین خط	0x80
۱۶	انتقال نشانگر به اولین مکان از دومین خط	0xC0

```
403
404 void LCD_command(unsigned char command) {
405     // RS = R/W = 0
406     GPIOA->ODR &= ~(RS|RW);
407     GPIOA->ODR &= (0X007);
408     // putting command on data bus
409     GPIOA->ODR |= command << 3;
410     // pulse EN high and clear it
411     GPIOA->ODR |= EN;
412     delay(13);
413     GPIOA->ODR &= ~(EN);
414     //The first and second commands just only  needs more delay
415     if (command < 4)
416         delay(26);
417     else
418         delay(13);
419 }
420
```

مقداردهی اولیه به پورت ها:

```
}

void setGPIOConfig(){
    RCC->AHB1ENR|=RCC_AHB1ENR_GPIOAEN; /* Initialize needed GPIOs and set ports mode appropriately  */
    GPIOA->MODER = 0x00555555;
    GPIOA->OTYPER = 0x0000;
    GPIOA->PUPDR = 0x00000000;
    GPIOA->ODR = 0x00000000;
    RCC->AHB1ENR|=RCC_AHB1ENR_GPIOBEN;
    GPIOB->MODER = 0x00005500;
    RCC->AHB1ENR|=RCC_AHB1ENR_GPIOCEN;
    GPIOC->MODER = 0x00000000;
    RCC->APB2ENR |= RCC_APB2ENR_SYSCFGEN;
    //CONNECTING from PB0 tO PB3, to ICR
    SYSCFG->EXTICR[0] |= SYSCFG_EXTICR1_EXTI0_PB;
    SYSCFG->EXTICR[0] |= SYSCFG_EXTICR1_EXTI1_PB;
    SYSCFG->EXTICR[0] |= SYSCFG_EXTICR1_EXTI2_PB;
    SYSCFG->EXTICR[0] |= SYSCFG_EXTICR1_EXTI3_PB;
    SYSCFG->EXTICR[1] |= SYSCFG_EXTICR2_EXTI4_PC;
    SYSCFG->EXTICR[1] |= SYSCFG_EXTICR2_EXTI5_PC;
}

void enableInterrupts(){ // ENABLING INTERRUPTS
    EXTI->IMR |= (MASK(0));
    EXTI->IMR |= (MASK(1));
    EXTI->IMR |= (MASK(2));
    EXTI->IMR |= (MASK(3));
    EXTI->IMR |= (MASK(4));
    EXTI->IMR |= (MASK(5));
    EXTI->RTSR |= (MASK(0));
    EXTI->RTSR |= (MASK(1));
    EXTI->RTSR |= (MASK(2));
    EXTI->RTSR |= (MASK(3));
    EXTI->RTSR |= (MASK(4));
    EXTI->RTSR |= (MASK(5));
  __enable_irq();
    NVIC_ClearPendingIRQ(EXTI0_IRQn);
    NVIC_ClearPendingIRQ(EXTI1_IRQn);
    NVIC_ClearPendingIRQ(EXTI2_IRQn);
    NVIC_ClearPendingIRQ(EXTI3_IRQn);
    NVIC_ClearPendingIRQ(EXTI4_IRQn);
    NVIC_ClearPendingIRQ(EXTI9_5_IRQn);
    NVIC_EnableIRQ(EXTI0_IRQn);
    NVIC_EnableIRQ(EXTI1_IRQn);
    NVIC_EnableIRQ(EXTI2_IRQn);
    NVIC_EnableIRQ(EXTI3_IRQn);
    NVIC_EnableIRQ(EXTI4_IRQn);
    NVIC_EnableIRQ(EXTI9_5_IRQn);
}
void initiallzingPorts(void) {
    setGPIOConfig();
    enableInterrupts();
}
```

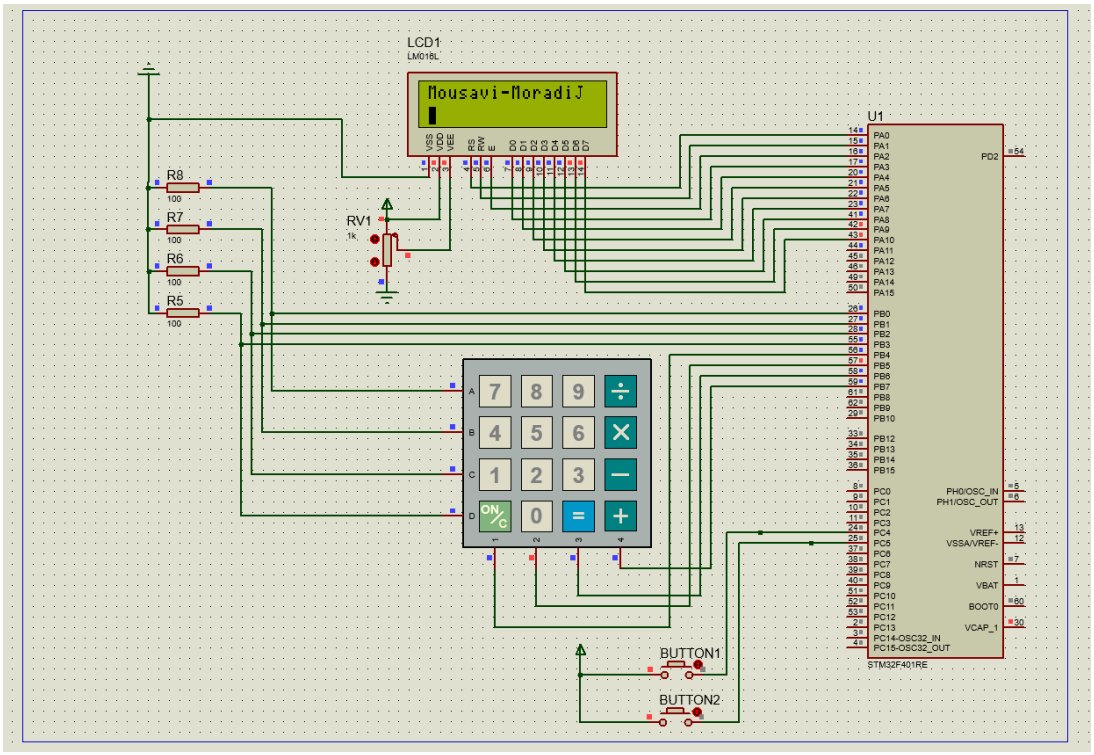
هر سری با کد زیر یکی از خطوط عمودی صفحه کلید فعال می شوند (طوری تنظیم شده سرعت فعال/غیرفعالسازی که با یکبار فشرده شدن اینتراپت ایجاد شود)

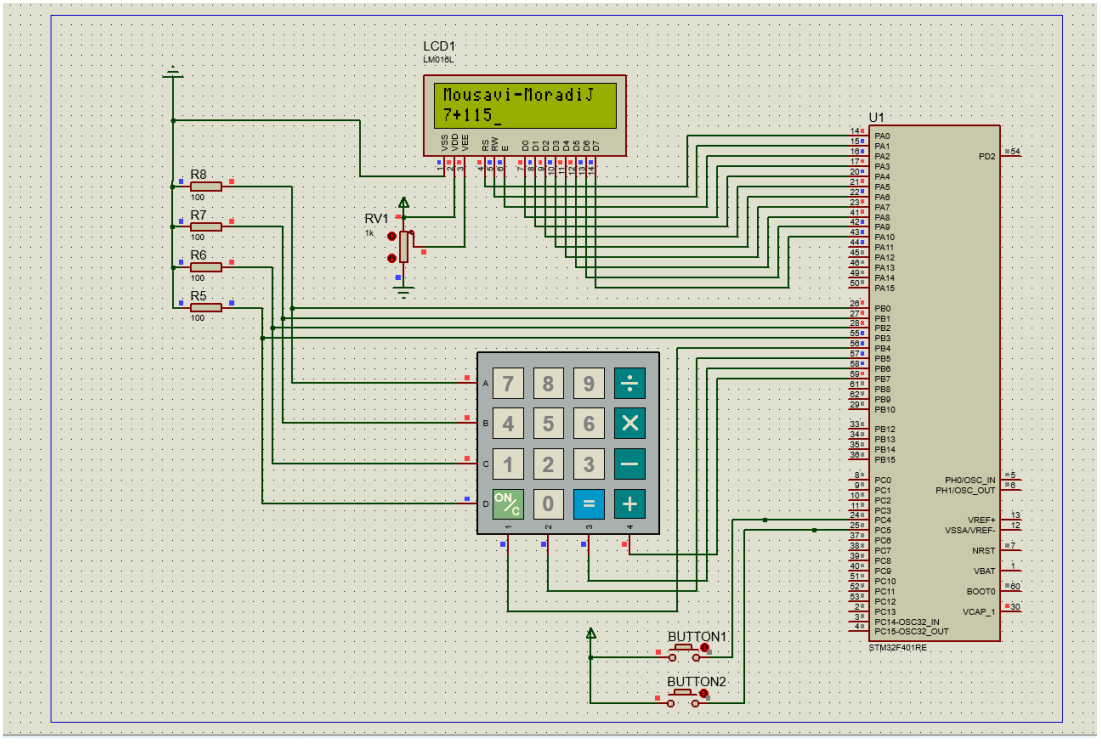
```
int main(void) {
  initializingLCD();
  printNames();
  LCD_command(0xC0); // Go to second Line
  while(1) {

        GPIOB->ODR &= 0xF; //activating proper columns
        LCD_line = (LCD_line+1)%4;
        GPIOB->ODR |= MASK(LCD_line)<<4;
        if(equalClicked == 1){
            calculateResult();
        }
        delay(25);
    }
}
```

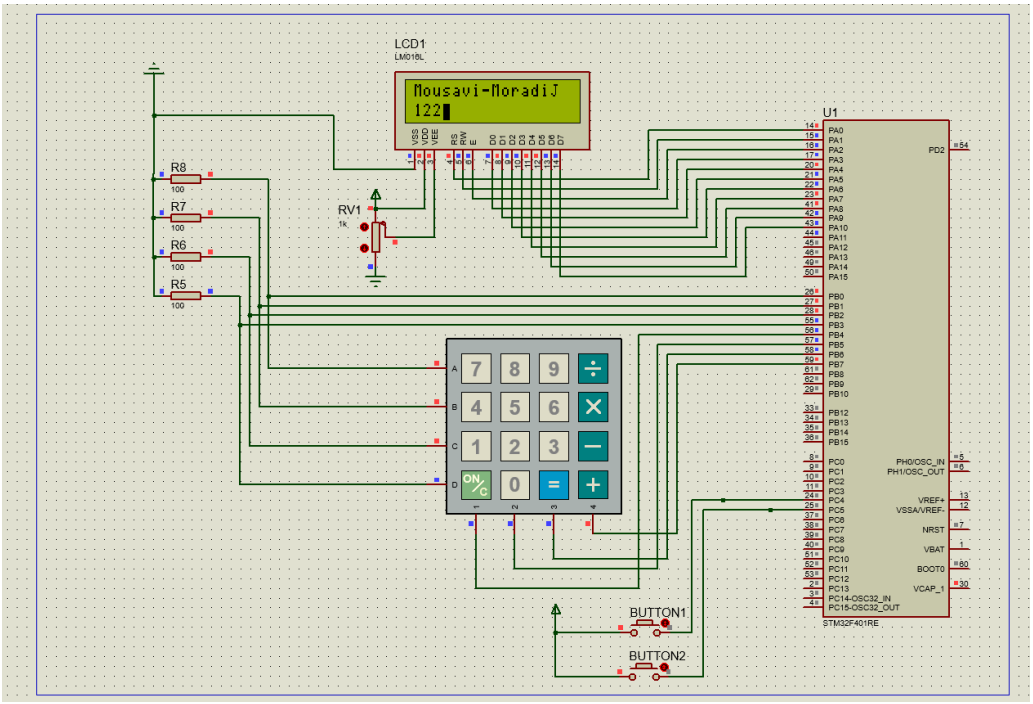
و بسته به اینکه کدام سطر (یکی از کلید های این سطر) فشرده شده به روتین interrupt مختص ان می رویم و در آنجا می فهمیم که کدام کلید فشرده شده به طور مثال اگر یکی از کلید های سطر اول را بفشاریم وارد روتین زیر می شویم.

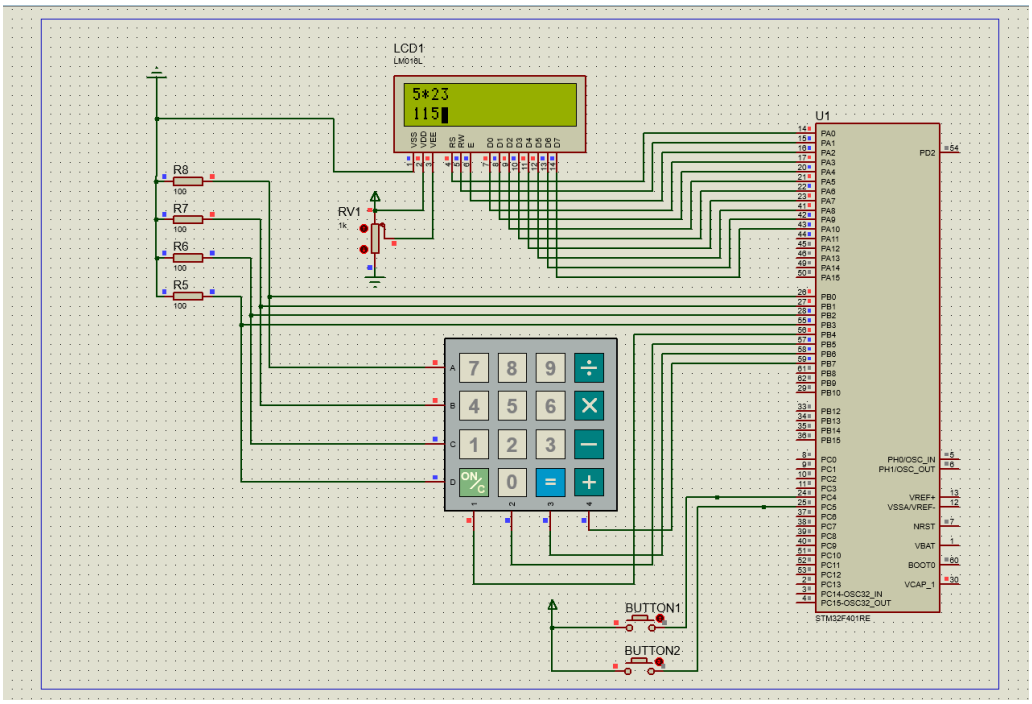
```
void EXTI0_IRQHandler(void){

  //clearing pending bits
  EXTI->PR |= MASK(0);
  NVIC_ClearPendingIRQ(EXTI0_IRQn);
  switch(LCD_line){
    case 0:{
      LCD_data('7');
      if(isFirstOPExist == 0){
        firstOperand *= 10;
        firstOperand += 7;
      }
      else{
        secondOperand *= 10;
        secondOperand += 7;
      }
      break;
    }
    case 1:{
    LCD_data('8');
      if(isFirstOPExist == 0){
        firstOperand *= 10;
        firstOperand += 8;
      }
      else{
        secondOperand *= 10;
        secondOperand += 8;
      }
      break;
    }
    case 2:{
    LCD_data('9');
      if(isFirstOPExist ==0){
        firstOperand *= 10;
        firstOperand += 9;
      }
      else{
        secondOperand *= 10;
        secondOperand += 9;
      }
      break;
    }
    case 3:{
      LCD_data('/');
      isFirstOPExist = 1;
      operation = '/';
      break;
    }
  }
}
```

در ادامه نتیجه کار را در پروتئوس مشاهده می‌کنیم:









رفرنس دستور کار:

کلاس درس و اسلاید های درسی

دیتا شیت و رفرنس منوآل