گزارش دستور کار شماره 3

فربد فولادی – 98243045

عرفان رفیعی اسکویی - 98243027

در این گزارش از ما نوشتن یک رشته کاراکتری بر روی LCD و طراحی یک ماشین حساب خواسته شده بود. سیستمی طراحی کنید که نام خانوادگی شما و همگروهیتان را روی خط)ردیف(اول LCD بنویسد و در خط دوم اعداد و دستور محاسباتی را از keypad دریافت کند و نتیجه حاصل را بر روی نمایشگر به نمایش درآورد. در هر مرحله، رقم وارد شده از هر عدد، باید بر روی صفحه نمایش نشان داده شود (مشابه با ماشین حساب حالت استاندارد ویندوز). دو عدد PushButton استفاده کنید و سیستم را طوری طراحی کنید که با هر بار فشردن یک کلید مقدار یک واحد از عدد حاصل کم شود و با هر بار فشردن کلید دوم یک واحد به مقدار اضافه شود.

در ابتدا RS و RW و EN را با مقادیر اولیه تعریف میکنیم. سپس توابع لازم را تعریف می کنیم که در ادامه توضیح داده ایم.

```
#define RS 0x20 /* PB5 mask for reg select */
#define RW 0x40 /* PB6 mask for read/write */
#define EN 0x80 /* PB7 mask for enable */
```

delayMs(int n) برای ایجاد delay است.

```
/* delay n milliseconds (16 MHz CPU clock) */
void delayMs(int n)
{
  int i;
  for (; n > 0; n--)
    for (i = 0; i < 3195; i++)
      __NOP();
}
```

LCD_command یک کامند میگیره و در ابتدا RS و RW را صفر می کنیم و کامند را در ODR قرار می دهیم و EN را high میکنیم. و command های 1 و 2 به delay 1.64ms نیاز دارند و مابقی به 40ms .

```
void LCD_command(unsigned char command)
{
  GPIOB->BSRR = (RS | RW) << 16; /* RS = 0, R/W = 0 */
  GPIOC->ODR &= ~0x00FF;        /* clear data bus */
  GPIOC->ODR |= command;        /* put command on data bus */
  GPIOB->BSRR = EN;          /* pulse E high */
  delayMs(0);
  GPIOB->BSRR = EN << 16; /* clear E */

  if (command < 4)
    delayMs(2); /* command 1 and 2 needs up to 1.64ms */
  else
    delayMs(1); /* all others 40 us */
}
```

در تابع LCD_DATA مقدار rs را 1 و rw را 0 میکنیم و همچنین ODR مان را clear میکنیم تا data ورودی را در ان قرار دهیم .

```
void LCD_data(char data)
{
  GPIOB->BSRR = RS;    /* RS = 1 */
  GPIOB->BSRR = RW << 16; /* R/W = 0 */
  GPIOC->ODR &= ~0x00FF;  /* clear data bus */
  GPIOC->ODR |= data;    /* put data on data bus */
  GPIOB->BSRR = EN;    /* pulse E high */
  GPIOB->BSRR = EN;    /* pulse E high */
  delayMs(0);
  GPIOB->BSRR = EN << 16; /* clear E */
  delayMs(1);
}
```

در تابع LCD_RESET ابتدا LCD_command(1) را صدا میزنیم تا محتویات نمایشگر ریست شود و در ادامه با صدا زدن LCD_DATA با مقادیر ورودی حروف فامیلی خود، فامیلی خودمان را در LCD چاپ میکنیم و سپس با command 0xC0 نشانگر به اولین مکان از دومین سطر منتقل میکنیم.

```
void LCD_reset(void)
{
  LCD_command(1);
  LCD_data('R');
  LCD_data('a');
  LCD_data('f');
  LCD_data('e');
  LCD_data('e');
  LCD_data('-');
  LCD_data('F');
  LCD_data('o');
  LCD_data('o');
  LCD_data('l');
  LCD_data('a');
  LCD_data('d');
  LCD_data('i');
  LCD_command(0xC0);
}
```

در تابع LCD_INIT صرفا initializeهای اولیه را انجام میدهیم با command هایی که در دستور کار به ما داده شده بود و یک تابع port_init هم صدا زده میشود که پورت های بورد را متناسب با توضیحاتی که در دستور کار بود به keypad و lcd متصل میکند.

```
/* initialize port pins then initialize LCD controller */
void LCD_init(void)
{
  PORTS_init();

  delayMs(30); /* initialization sequence */
  LCD_command(0x30);
  delayMs(10);
  LCD_command(0x30);
  delayMs(1);
  LCD_command(0x30);

  LCD_command(0x38); /* set 8-bit data, 2-line, 5x7 font */
  LCD_command(0x06); /* move cursor right after each char */
  LCD_command(0x01); /* clear screen, move cursor to home */
  LCD_command(0x0F); /* turn on display, cursor blinking */
}
```

```
void PORTS_init(void)
{
  RCC->AHB1ENR |= 0x06; /* enable GPIO B/C clock */
  RCC->APB2ENR |= 0x4000; /* enable SysConfig clock */

  /* PB5 for LCD R/S */
  /* PB6 for LCD R/W */
  /* PB7 for LCD EN */
  GPIOB->MODER = 0x00005400; /* set pin output mode */
  GPIOB->PUPDR = 0x000A5400; /* set pin output mode */
  GPIOB->BSRR = 0x00C00000;  /* turn off EN and R/W */

  /* PC0-PC7 for LCD D0-D7, respectively. */
  GPIOC->MODER = 0x00555555; /* set pin output mode */
  GPIOC->PUPDR = 0xAA000000; /* set pin output mode */

  SYSCFG->EXTICR[3] = 0x2222;
  SYSCFG->EXTICR[2] = 0x0011;
  EXTI->IMR = 0xF300;
  EXTI->RTSR = 0xF300;
  NVIC_EnableIRQ(EXTI9_5_IRQn);
  NVIC_EnableIRQ(EXTI15_10_IRQn);
  __enable_irq();
}
```

دو تابع EXTI15_10_IRQHandler و EXTI9_5_IRQHandler داریم که یکی برای keypad ماشین حساب و دیگری برای دو push button است . تابع EXTI15_10_IRQHandler به این صورت عمل میکند که مقدار current key را متناسب با IDR(input data register) که متناسب با column که در صفحه lcd است را نمایش میدهد و همچنین متناسب با current key که بدست میاید

calculator_state را update میکند و همچنین مقدار op را مساوی با current key قرار میدهد و متناسب با calculator_state و current key مقادیر num1 و num2 را اپدیت میکند.

و EXTI9_5_IRQhandler در این تابع اگر calculator state ما برابر با show result بود متناسب با حاصل exit اگر button اول را فشار بدهیم مقدار result یک عدد اضافه میشود و اگر دکمه دوم را فشار داده باشیم یک عدد کم میشود و در نهایت تابع LCD_PRINT_RESULT را صدا میرنیم که در ان با کمک تابع lcd data تک تک کاراکتر هارا در idr قرار میدهیم تا در lcd به نمایش گذاشته شود.

```c
void EXTI9_5_IRQHandler(void)
{
  if (calculator_state == show_result)
  {
    LCD_reset();
    if (EXTI->PR & 0x0100)
    {
      result++;
      EXTI->PR |= 0x0100;
    }
    if (EXTI->PR & 0x0200)
    {
      result--;
      EXTI->PR |= 0x0200;
    }
    LCD_print_result();
  }
  else
  {
    EXTI->PR |= 0x0300;
  }
  NVIC_ClearPendingIRQ(EXTI9_5_IRQn);
}
```

```c
void LCD_print_result(void)
{
  char charValue[16];
  sprintf(charValue, "%d", result);
  int i = 0;
  while (charValue[i] != '\0')
  {
    LCD_data(charValue[i]);
    i++;
  }
}
```