دانشگاه شهید بهشتی

دانشکده مهندسی و علوم کامپیوتر

گزارش پروژه پایانی ریزپردازنده و اسمبلی

علیرضا چقامیرزایی - ۹۷۲۴۳۰۱۷

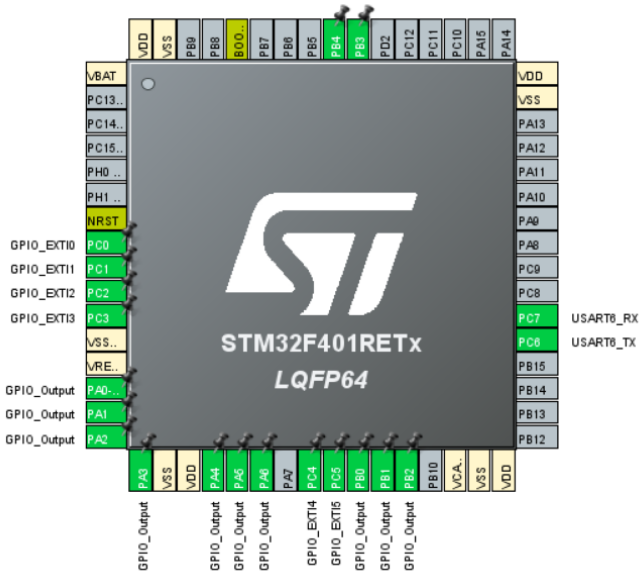
سارا صارمی – ۹۷۲۴۳۰۴۵

۱ گره مرکزی – HAL

۱.۱ جدول پین‌ها

PIN	FUNCTION
PA0..PA6	Seven Segment
PB0..PB3	LEDs
PB4	MAX487
PC0, PC1	Device Select Switch
PC2..PC5	Buttons
PC6	USART6_TX
PC7	USART6_RX

۱.۲ وضعیت پین‌ها در CubeMX



۱.۳ نمایش سون‌سگمنت

مشابه آزمایش دوم که در طول ترم انجام دادیم، نمایش باینتری متناظر با هر عدد را برای نمایش روی سون سگمنت تعریف کرده‌ایم و تابعی برای تبدیل اعداد به این نمایش باینری نوشته‌ایم.

```
int returnHexNumber(int decimalNumber)
{
    if (decimalNumber == 1)
        return number_1;
    else if (decimalNumber == 2)
        return number_2;
```

```
    else if (decimalNumber == 3)
        return number_3;
    return number_0;
}
```

در اینجا ۴ حالت مختلف داریم که بسته به وضعیت DIP Switch مشخص می‌شود. با زدن کلیدهای DIP Switch یک وقفه خارجی ایجاد می‌شود که در تابع مربوطه هال این وقفه مدیریت شده‌است. به این ترتیب که با AND کردن input های پین‌های c با 11 متوجه می‌شویم در کدامیک از حالت‌های مشخص شده هستیم و عدد حالت مربوطه را روی سون‌سگمنت نمایش می‌دهیم.

```
    volatile int sel = GPIOC->IDR & 3;
    if (sel == 1)
    {
        temp = 0;
        switch_select = 1;
    }
    else if (sel == 2)
    {
        temp = 0;
        switch_select = 2;
    }
    else if (sel == 3)
    {
        temp = 0;
        switch_select = 3;
    }
    else
    {
        temp = 0;
        switch_select = 0;
    }

    temp |= returnHexNumber(switch_select);
```

۱.۴ مدیریت وضعیت LEDها

برای مدیریت وضعیت LEDها یک آرایه دوبعدی سه در چهار تعریف کردیم که ۳ تعداد گره‌ها و ۴ تعداد LEDهای هر گره است. هنگامی که یک کلید فشرده می‌شود، دو متغیر switch_select برای تعیین گره و led_select برای تعیین LED مورد نظر تعیین می‌شوند. سپس در صورتی که LED مشخص شده روشن باشد، وضعیتش به خاموش تغییر پیدا می‌کند و برعکس. به عنوان مثال در قطعه کد زیر کلید مربوط به LED اول زده شده.

```
if (sel & MASK(2))
   {
       led_select = 0;
       if (slaves_leds[switch_select][led_select] == 1)
           slaves_leds[switch_select][led_select] = 0;

       else
           slaves_leds[switch_select][led_select] = 1;
   }
```

در main با استفاده از توابع HAL وضعیت LEDهای مربوط به گره انتخاب شده را روی پین‌های مربوطه می‌نویسیم.

```
       GPIOA->ODR = 0;
       GPIOA->ODR |= temp;

       HAL_UART_Receive(&huart6, Rx_data, 4, 1000); // receive 4 bytes of
data
       check_slaves();

       HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, slaves_leds[switch_select][0]);
       HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1, slaves_leds[switch_select][1]);
       HAL_GPIO_WritePin(GPIOB, GPIO_PIN_2, slaves_leds[switch_select][2]);
       HAL_GPIO_WritePin(GPIOB, GPIO_PIN_3, slaves_leds[switch_select][3]);
```

۱.۵ ارتباط سریال USART

بیت‌ها در سه فریم طبق خواسته صورت پروژه فرستاده می‌شوند. فریم اول مربوط به Physical address است که مشخص می‌کند کدام slave انتخاب شده است. فریم دوم register است که LED فشار داده‌شده در آن فرستاده می‌شود و در فریم سوم که دیتاست وضعیت LED مشخص شده در گره

مورد نظر فرستاده می‌شود. با استفاده از تابع هال UART_transmit این سه فریم فرستاده می‌شوند. پیام‌ها در این برنامه تبدیل به کرکتر می‌شوند و فرستاده می‌شوند.

```
    if (led_select != -1)
    {

        char phy_address = switch_select;
        char register_led_number = led_select;
        char data = slaves_leds[switch_select][led_select];
        sprintf(send_bits, "%c%c%c", phy_address, register_led_number, data);
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, GPIO_PIN_SET);
        HAL_UART_Transmit(&huart6, send_bits, sizeof(send_bits), 100);
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4, GPIO_PIN_RESET);
    }
```

۲ گره CMSIS

۲.۱ جدول پین‌ها

PIN	FUNCTION
PA0	**MAX487**
PA2	**USART2_TX**
PA3	**USART2_TX**
PC0..PC3	**Buttons**
PC4..PC7	**LEDs**

برای دکمه‌ها از وقفه خارجی استفاده شده‌است مانند آزمایش‌های پیشین، تنظیمات مربوط به وقفه خارجی در تابع irq0_init() انجام شده‌است.

```
void irq0_init()
{
    SYSCFG->EXTICR[0] |= SYSCFG_EXTICR1_EXTI0_PC;
    EXTI->IMR |= (MASK(0));
    EXTI->RTSR |= (MASK(0));
    __enable_irq();
    NVIC_SetPriority(EXTI0_IRQn, 0);
    NVIC_ClearPendingIRQ(EXTI0_IRQn);
    NVIC_EnableIRQ(EXTI0_IRQn);
}
```

سپس برای پین مربوط به هر کدام از کلیدها تابع وقفه خارجی مربوطه پیاده‌سازی شده که در صورت فشرده شدن مقدار متناظر با LED مورد نظر را در آرایه led_array تنظیم می‌کند. خط GPIO->ODR = 1 در واقع برای اجازه نوشتن در باس است که بعد از فرستادن داده 0 فرستاده می‌شود. قطعه کد زیر مربوط به کلید و LED اول است.

```
void EXTI0_IRQHandler(void)
{

    EXTI->PR |= MASK(0);
    NVIC_ClearPendingIRQ(EXTI0_IRQn);

    if (GPIOC->IDR & MASK(0))
    {
        GPIOA->ODR = 1;
        leds_array[0] ^= 1;
        send_data(0);
        GPIOA->ODR = 0;
        update_leds();
    }
}
```

۲.۲ تابع send_data(led_num)

با توجه به این که از slave در master می‌خواهیم بنویسیم، مقدار physical address را برابر ۱ قرار می‌دهیم. در فریم دوم طبق پروتکل داده شده، بیت هفتم برای Read یا Write است در نتیجه برای slave در حالت read قرار داریم و باید بیت هفتم ۱ باشد. پس مقدار فریم دوم را با ۱۲۸ جمع می‌کنیم که در باینری بیت هفتم را برابر یک قرار دهد. در این فریم شماره LED مورد نظر نیز فرستاده می‌شود. در نهایت در فریم سوم وضعیت LEDهای اسلیو قرار می‌گیرد.

```
void send_data(int led_num)
{
    volatile char phy_address = 1;
    volatile char register_led = led_num + 128;
    volatile char data_led = leds_array[led_num];
    UART_Transmit(phy_address);
    UART_Transmit(register_led);
    UART_Transmit(data_led);
}
```

۲.۳ تابع update_leds()

این تابع با توجه به آرایه‌ی led_arrays و پین‌های مربوط به LEDها، خروجی پین‌های C را مشخص می‌کند که منجر به خاموش و روشن شدن LEDهای مورد نظر می‌شود.

```
void update_leds()
{
    volatile int selected = 0;
    selected |= leds_array[0] << 4;
    selected |= leds_array[1] << 5;
    selected |= leds_array[2] << 6;
    selected |= leds_array[3] << 7;
    GPIOC->ODR = 0;
    GPIOC->ODR |= selected;
}
```

۲.۴ ارتباط سریال USART

در این پردازنده USART2 فعال شده‌است که تنظیمات مربوط به راه‌اندازی در تابع USART2_init() آمده است. تابع USART2_recieve نیز برای دریافت سه فریم مورد نظر نوشته شده‌است.

```
char USART2_receive()
{
    while (!READ_BIT(USART2->SR, USART_SR_RXNE))
    {
    }
    physical_address = (int)(USART2->DR);
    while (!READ_BIT(USART2->SR, USART_SR_RXNE))
    {
    }
    register_led_number = (int)(USART2->DR);
    while (!READ_BIT(USART2->SR, USART_SR_RXNE))
    {
    }
    data = (int)(USART2->DR);
    return 't';
}
```

۳ گره 8086

۳.۱ وصل کردن LED و دکمه‌ها به 8086

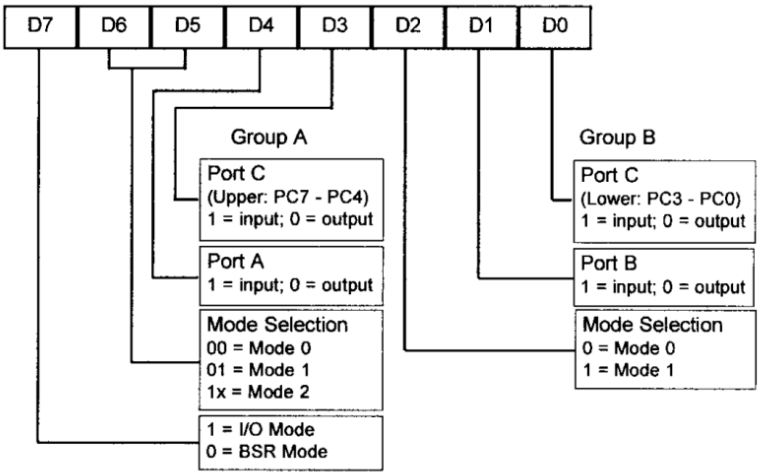
برای وصل کردن این ادوات به 8086 از دو قطعه دیگر استفاده شده‌است. قطعه اول latch-74HC373 است که دیتای CPU را لچ می‌کند و دیگری 8255A است که برای وصل کردن دیوایس‌های IO به 8086 استفاده می‌شود.

۳.۲ جدول مربوط به پین‌های 8255

PIN	Description
PA0 – PA3	LEDs
PB0 – PB3	Buttons
D0 – D7	AD bus (AD0-AD7)
RD	8086 RD
WR	8086 WR
A0	Latch Q1
A1	Latch Q2

۳.۳ کد اسمبلی برای روشن کردن LEDها

با توجه به این که LEDها به port A و کلیدها به port B وصل هستند، باید A را به عنوان output و B را به عنوان input تنظیم کنیم. با توجه به شکل زیر به صورت ۱۰۰۰۰۰۱۰ در می‌آید.



سپس در هر مرحله با چک کردن وضعیت پورت B به یکی از استیت‌های BTN0 تا BTN3 که مربوط به هر دکمه است می‌رویم. در این استیت‌ها خروجی مربوط به LED متناظر با باتن فشرده شده برابر یک می‌شود.

```
    CMP AL, 11111110B
    JZ BTN0
```

```
BTN0:
   OR CL, 00000001B
   JMP DISPLAY
```

```
DISPLAY:
   MOV AL, CL
   OUT DX,AL
   JMP START
```

۳.۴ ارتباط سریال USART

برای ارتباط سریال در 8086 می‌توانیم از قطعه 8251 استفاده کنیم. این قطعه داده را به صورت سریال دریافت کرده و به صورت موازی به پردازنده می‌فرستد و به طور مشابه داده را به صورت موازی از باس پردازنده دریافت کرده و به صورت سریال می‌فرستد. پین D[0..7] به پردازنده متصل می‌شوند تا داده را انتقال دهند. سیگنال‌های کنترلی را نیز به توجه به دیتاشیت باید تنظیم کنیم.

به عنوان مثال زمانی که بخواهیم 8086 به عنوان slave عمل کند و داده بخواند، می‌خواهیم داده از 8251 به پردازنده فرستاده شود در نتیجه سیگنال‌های کنترلی به صورت زیر درمی‌آیند.

$\overline{CS}$	C/$\overline{D}$	$\overline{RD}$	$\overline{WR}$	Operation
1	X	X	X	Invalid
0	0	0	1	data CPU< ----- 8251
0	0	1	0	data CPU ----- > 8251
0	1	0	1	Status word CPU < ----------8251
0	1	1	0	Control word CPU---------- > 8251

با توجه به دیتا دریافت شده از 8251 باید وضعیت LEDها را تغییر دهیم.

MAX487 ۴

این قطعه دارای دو پین RE و DE است برای مشخص کردن این که باس در اختیار چه کسی قرار دارد. پین DI برای زمانی است که می‌خواهیم بنویسیم و RO برای زمانی که می‌خواهیم بخوانیم. به عنوان مثال زمانی که slave می‌خواهد اطلاعاتی را از سمت master بخواند به RO متصل می‌شود.