به نام خدا

گزارش کار آزمایشگاه ریزپردازنده

آزمایش ۸

مدرس : مهندس بی طالبی

تارا برقیان

مهرشاد سعادتی نیا

نیم سال اول ۰۱-۱۴۰۰

فهرست

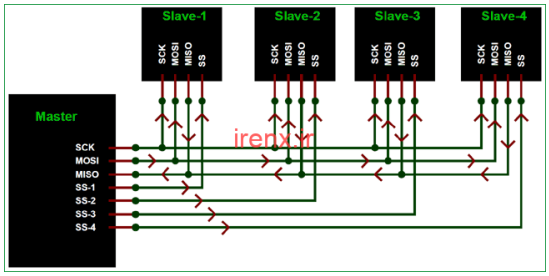
مقدمه :

در این آزمایش هدف آشنایی با مفاهیم و دستوارت UART وMAX7221 و انتقال مناسب داده بود. مفاهیم مربوط به ADC نیز دوره شد.

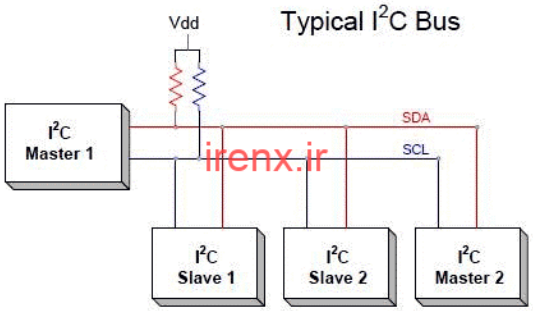
پاسخ سوالات تحلیلی:

- در واقع مخفف Universal Asynchronous Serial Receiver and Transmitter هست اما Usart مخفف Universal Synchronous and Asynchronous Serial Receiver and Transmitterهست، پس نتیجه میگیریم که USART همون UART هست که قابلیت ارتباط سریال سنکرون رو هم اضافه کرده است. تفاوت مهمتر این است که اگر از usart نرم افزاری استفاده شود دیگر نمیتوان از وقفه های آن استفاده کرد.

-
  - SPI : رابط محیطی سریال (SPI) یک رابط همزمان است که به شما امکان می دهد تا چندین میکروکنترلر SPIبه هم متصل شوند. درSPI ، سیم هایی جداگانه برای داده و خط ساعت مورد نیاز است. همچنین ساعت در جریان داده گنجانده نشده است و باید به عنوان یک سیگنال جداگانه ارائه شود SPI .ممکن است به عنوان masterیا بصورت slave پیکربندی شود. چهار سیگنال اساسیSPI (MISO ، MOSI، SCKو(SS ، Vcc و Ground بخشی از ارتباطات داده هستند. بنابراین برای ارسال و دریافت داده از slave یا master به ۶ سیم نیاز دارد. از نظر تئوری، SPIمی تواند تعداد نامحدودی slave داشته باشد. ارتباط داده ها در ثبت کننده های SPI پیکربندی می شود SPI .می تواند حداکثر سرعت ۱۰ مگابیت بر ثانیه را ارائه دهد و برای ارتباط دادهایی با سرعت بالا ایده آل است.

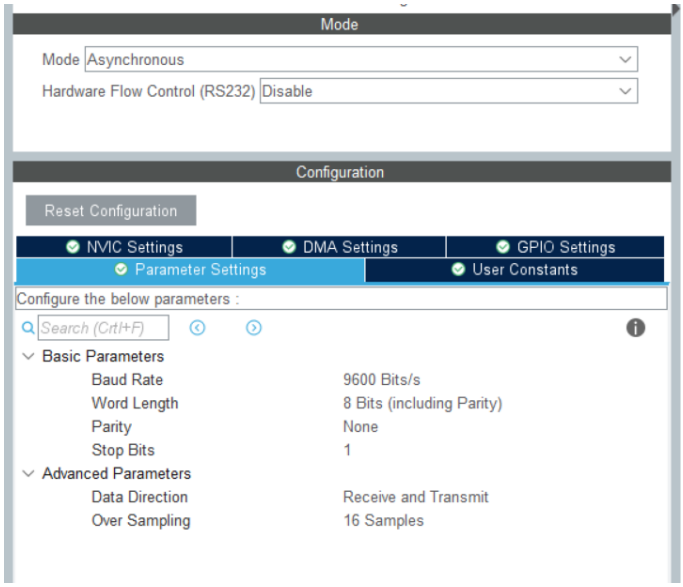


- مدار مجتمع اینتر(I2C) ، دو خط ارتباطی بین IC یا ماژول های مختلف است که آن دو خط SDA (Serial Data Line)و SCL (Serial Clock Line) هستند. هر دو خط باید با استفاده از مقاومت کششی به منبع مثبت متصل شوند .I2C می تواند سرعتی تا ۴۰۰ Kbpsرا ارائه دهد و از سیستم آدرس دهی ۱۰ بیتی یا ۷ بیتی برای هدف قرار دادن یک دستگاه خاص در گذرگاه i2c استفاده می کند تا بتواند ۱۰۲۴ دستگاه را بهم متصل کند. این ارتباط از طول محدودی برخوردار است و برای ارتباطات روی صفحه ایده آل است. راه اندازی شبکه های I2C آسان است، زیرا فقط از دو سیم استفاده می کنند و دستگاه های جدید به سادگی می توانند به دو خط مشترک I2C متصل شوند. همانندSPI ، میکروکنترلرها معمولاً دارای پین های I2Cبرای اتصال هر دستگاه I2C هستند.



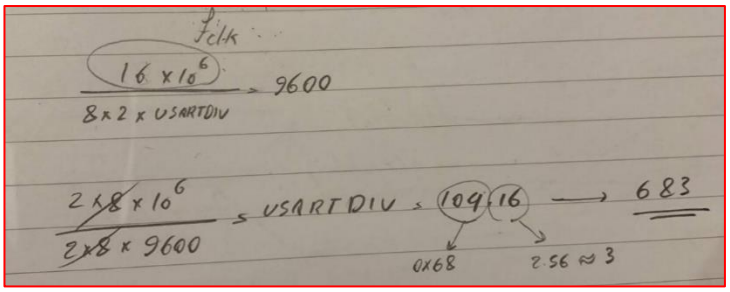
سوالات کدی:

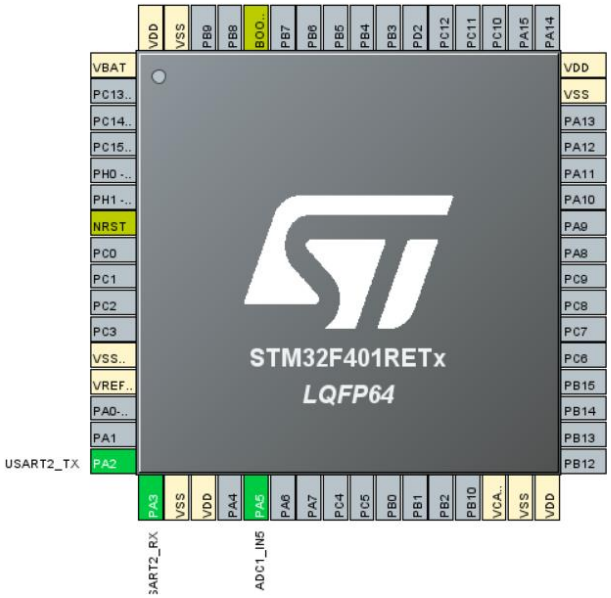
در ابتدا با کمک ویدیوهایی که در منابع آمده تنظیمات میکروها را با کمکcube انجام دادیم که تصاویر را در زیر قرار میدهم.



طبق تمرین قبلی کانال ۵ adc هم فعال کردیم.

External Trigger Conversion Edge	None
Rank	1
Channel	Channel 5
Sampling Time	28 Cycles





برای میکرو دوم نیز تنظیمات CMSISای بود که به توضیح میپردازیم. میکرو اول یک بار با HAL و یک بار با CMSIS زده شده است. فقط موقع تنظیم br با هال،

جلوتر نمیرفت و مجبور شدیم این خط را msis

کنیم تا کار کند.

```
huart2.Instance = USART2;
//huart2.Init.BaudRate = 9600;
huart2.Init.WordLength = UART_WORDLENGTH_8B;
huart2.Init.StopBits = UART_STOPBITS_1;
huart2.Init.Parity = UART_PARITY_NONE;
huart2.Init.Mode = UART_MODE_TX_RX;
huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
huart2.Init.OverSampling = UART_OVERSAMPLING_16;
```

```
MX_GPIO_Init();
MX_USART2_UART_Init();
USART2->BRR = 0x0683;    /* 960
MX_ADC1_Init();
/* USER CODE BEGIN 2 */
```

```
UART_Transmit(adcValue_i);
/* USER CODE END WHILE */

//HAL_UART_Transmit(&huart2, (uint8_t *)"2", 1, 100);
```

در تابع زیر تنظیمات اولیه را انجام دادیم مثلا ۲ بیت استاپ داریم ، alternate فانکشن مناسب را با کمک دیتاشیت فعال کردیم و رجیسترهارا برای Rx Tx مقدار مناسب دادیم. (محاسبه brrهم در عکس بالا میبینید.)

```c
void UART_init(){
  RCC->AHB1ENR |= RCC_AHB1ENR_GPIOAEN; /* enable GPIOA clock */
  RCC->APB1ENR |= RCC_APB1ENR_USART2EN;              /* enable USART2 clock */

    /* Configure PA2,PA3 for USART2_TX */
  GPIOA->OSPEEDR |= 0x80; //10 00 00 00
  GPIOA->OSPEEDR |= 0x20; //00 10 00 00

  GPIOA->AFR[0] &= ~0x0FF00;
  GPIOA->AFR[0] |= 0x07700; /* alt7 for USART2 */
  GPIOA->MODER  &= ~0x00F0;
  GPIOA->MODER  |= 0x00A0; // enable alternate function for PA3,PA2   10 10 00 00
  USART2->BRR = 0x0683;    /* 9600 baud @ 16 MHz */
  USART2->CR1 |= 0x0008; /* enable Tx, 8-bit data */
  USART2->CR1 |= 0x0004; /* enable Rx, 8-bit data */
  USART2->CR2 |= (2 << 12);    /* 1 stop bit*/
  USART2->CR3 = 0x0000;    /* no flow control */
  USART2->CR1 |= 0x2000;   /* enable USART2 */

}
```

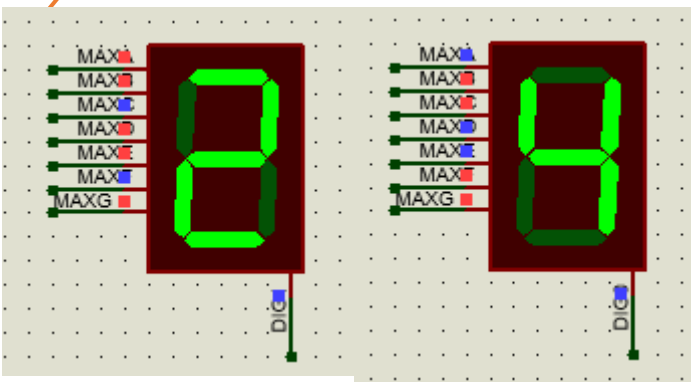
در این بخش عدد ۲رقمی دریافت شده را به دو بخش مقسوم علیه و باقی مانده تبدیل میکنیم.

و در دو مرحله برای میکرو دوم میفرستیم.

```c
void UART_Transmit(int digit){
  int big = digit / 10;
  int small = digit % 10;
  USART2->DR = big + '0';
  while(!READ_BIT(USART2->SR, USART_SR_TC)){}
  USART2->DR = small + '0';
  while(!READ_BIT(USART2->SR, USART_SR_TC)){}
}
```

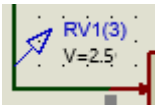
در حلقه ی اصلی برنامه نیز عدد مناسب با کمک adc گرفته و تبدیل میشود. سپس در ۱۰ ضرب کردیم تا قسمت اعشاری هم به صحیح تبدیل شود. با کمک تابعuart انتقال دادیم.

```c
  while (1)
  {
    HAL_ADC_Init(&hadc1);
    HAL_ADC_Start(&hadc1);
    if (HAL_ADC_PollForConversion(&hadc1, 10) == HAL_OK)
    {
      adcValue_f = HAL_ADC_GetValue(&hadc1);
      adcValue_f = (adcValue_f * 5) / 4095;
      adcValue_f = adcValue_f * 10;

    }
    adcValue_i = (int) adcValue_f;

    UART_Transmit(adcValue_i);
```



عدد ۲.۵ را به تقریب خوبی

به شکل ۲.۴ نشان میدهد.



در میکرو دوم هم هر دفعه مقدار را میخواند و روی سون سگمنت ها نمایش میدهد.

```
int main()
{

  SPI1_init();
  MAX_init();
  USART2_init ();

    while(1){
      USART2_receive();
      UART_Transmit(x);
      //int temp2 = x - '0';
      int x1 = x - '0';
      int y1 = y - '0';

      MAX_write(0x0200 + x);
      MAX_write(0x0100 + y);

  }
}
```

تنظیمات max با کمک اسلاید انجام شد. ابتدا روشن میشود و بیشترین intensity را ست کردیم و ۲ سون سگمنت داریم.

برای SPI هم با محاسبات مناسب که در دقیقا در اسلاید امده است عدد مناسب قرار دادیم.

```
void MAX_init(void){
   MAX_write(0xC01); //TURN ON MODULE
   MAX_write(0xA0F); //MAX INTENSITY
   MAX_write(0xB01);  //2 DIGITS
   MAX_write(0x9FF); //B DECODING
}

void SPI1_init(void)
{
  GPIO_init();
```

تنظیماتuart را بخش قبلی توضیح دادم و خیلی مشابه است با این تفاوت که این بخش اطلاعات را دریافت میکند.

```
void USART2_init (void) {
  RCC->AHB1ENR |= RCC_AHB1ENR_GPIOAEN;  /* enable GPIOA clock */
  RCC->APB1ENR |= 0x20000;              /* enable USART2 clock */
  /* Configure PA2 for USART2_TX */
  GPIOA->OSPEEDR |= 0x20;
  GPIOA->OSPEEDR |= 0x80;
  GPIOA->AFR[0] &= ~0x0FF00;
  GPIOA->AFR[0] |= 0x07700; /* alt7 for USART2 */
  GPIOA->MODER  &= ~0x00F0;
  GPIOA->MODER  |= 0x00A0; /* enable alternate function for PA2,PA3*/
  USART2->BRR = 0x0683; /* 9600 baud @ 16 MHz */
  USART2->CR1 |= 0x0008; /* enable Tx, 8-bit data */
  USART2->CR1 |= 0x0004; /* enable Rx, 8-bit data */
  //USART2->CR1 |= USART_CR1_TCIE;
  USART2->CR2 |= (2 << 12); /* 1 stop bit */
  USART2->CR3 = 0x0000; /* no flow control */
  USART2->CR1 |= 0x2000; /* enable USART2 */

}
```

در این بخش هم دو متغیر x,y که نشان دهنده اعداد روی سون سگمنت هستند مقدار دهی میشود.

```
char USART2_receive(){
  while (!READ_BIT(USART2->SR, USART_SR_RXNE)){}
  x = (uint8_t)(USART2->DR);
  while (!READ_BIT(USART2->SR, USART_SR_RXNE)){}
  y = (uint8_t)(USART2->DR);
  return 't';
}
```

در تابع max_write به دیوایس slave داده ارسال میکنیم ابتدا صبر میکنیم بافر خالی شود سپس اختیار slave را با کمک assert به دست میگیریم. سپس عملیات write کردن را انجام داده و مجدد disassert میکنیم.

در عملیات نوشتن در تابع write_operation هم ابتدا دیتا را ۸ تا به سمت راست شیفت میدهیم. سپس از ۸ بیت بودن دیتا با کمک &ff مطمئن میشویم و منتظر انجام عملیات میمانیم.

```
83  void assert(void){
84      GPIOA->BSRR = 0x00100000; //ACTIVE LOW ASSERT
85  }
86  void disassert(void){
87      GPIOA->BSRR = 0x00000010;
88  }
89
90  void write_operation(int16_t data){
91      SPI1->DR = data >> 8 ;  //  0....0(D11-D8)
92      while (!(SPI1->SR & SPI_SR_TXE)) {}
93      SPI1->DR = data & 0xFF;  // write command
94      while (SPI1->SR & SPI_SR_BSY) {}
95  }
96
97
98  void MAX_write(int16_t data) {
99
100     while (!(SPI1->SR & SPI_SR_TXE)) {}
101     assert();
102     write_operation(data);
103     disassert();
104
105 }
```

منابع:

- https://irenx.ir/learn/serial-communication-protocols/
- https://www.aparat.com/v/LQWUf