گزارش آزمایشگاه

رشته مهندسی برق

گزارشکار سری چهارم آزمایشگاه ریزپردازنده و مدارهای واسطه

نگارش

امیرحسین منصوری

مهشاد اکبری سریزدی - ۹۹۲۳۰۹۳

آنوشا شریعتی -۹۹۲۳۰۴۱

استاد راهنما

مهندس ذکی‌زاده

آبان ۱۴۰۲ شمسی

آزمایش سوم - بخش ۲ :

در این بخش به پیاده سازی مانند قسمت قبل اما با دستوراتی متفاوت برپایه تعریف شده بودن رجیستر ها می پردازیم . در moder هر پین دو بیتی می باشد و بر اساس آن و دستور کار syntax های مربوط به یک کردن هر بیت را برای moder و BSRR به صورت زیر پیاده سازی کردیم. با توجه به بخش اول آزمایش بیت های ۱۲ تا ۱۵ رجیستر BSRR و MODER را یک میکنیم زیرا led های به پایه های ۱۲ تا۱۵پریفرال GPIOD متصل هستند .

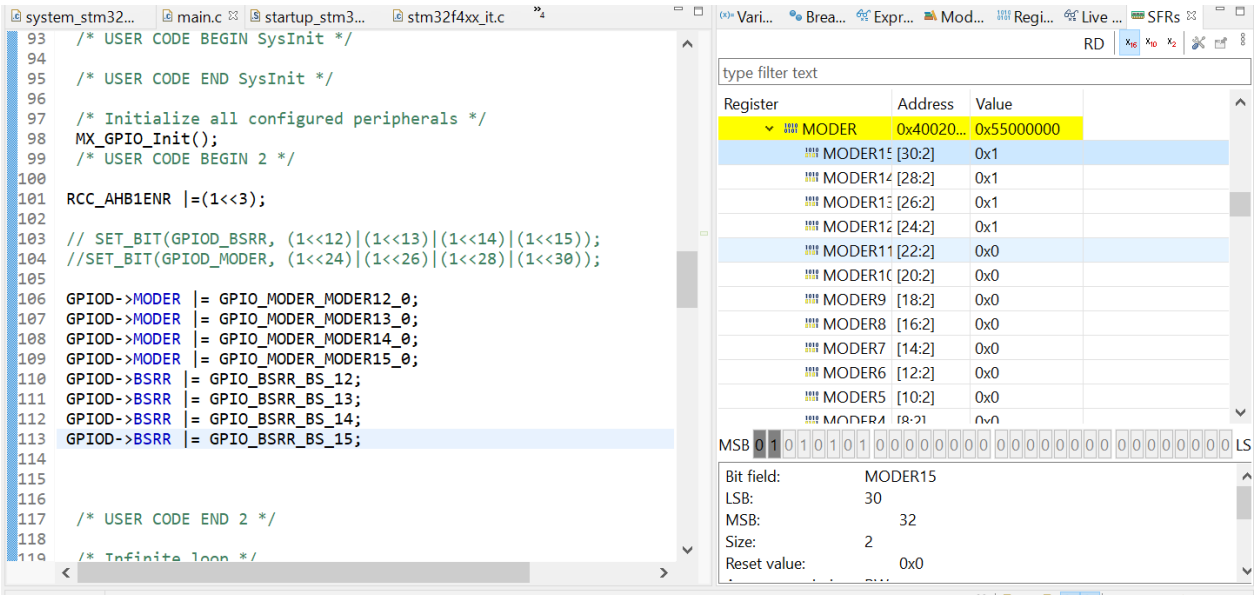
system_stm32... | main.c | startup_stm3... | stm32f4xx_it.c

```
 93   /* USER CODE BEGIN SysInit */
 94
 95   /* USER CODE END SysInit */
 96
 97   /* Initialize all configured peripherals */
 98   MX_GPIO_Init();
 99   /* USER CODE BEGIN 2 */
100
101  RCC_AHB1ENR |=(1<<3);
102
103  // SET_BIT(GPIOD_BSRR, (1<<12)|(1<<13)|(1<<14)|(1<<15));
104  //SET_BIT(GPIOD_MODER, (1<<24)|(1<<26)|(1<<28)|(1<<30));
105
106  GPIOD->MODER |= GPIO_MODER_MODER12_0;
107  GPIOD->MODER |= GPIO_MODER_MODER13_0;
108  GPIOD->MODER |= GPIO_MODER_MODER14_0;
109  GPIOD->MODER |= GPIO_MODER_MODER15_0;
110  GPIOD->BSRR |= GPIO_BSRR_BS_12;
111  GPIOD->BSRR |= GPIO_BSRR_BS_13;
112  GPIOD->BSRR |= GPIO_BSRR_BS_14;
113  GPIOD->BSRR |= GPIO_BSRR_BS_15;
114
115
116
117   /* USER CODE END 2 */
118
119   /* Infinite loop */
```

در تصویر بالا هم صحت کد را مشاهده می کنیم که بیت ها را به درستی یک کرده است وled ها روشن شدند.

آزمایش سوم - بخش۳:

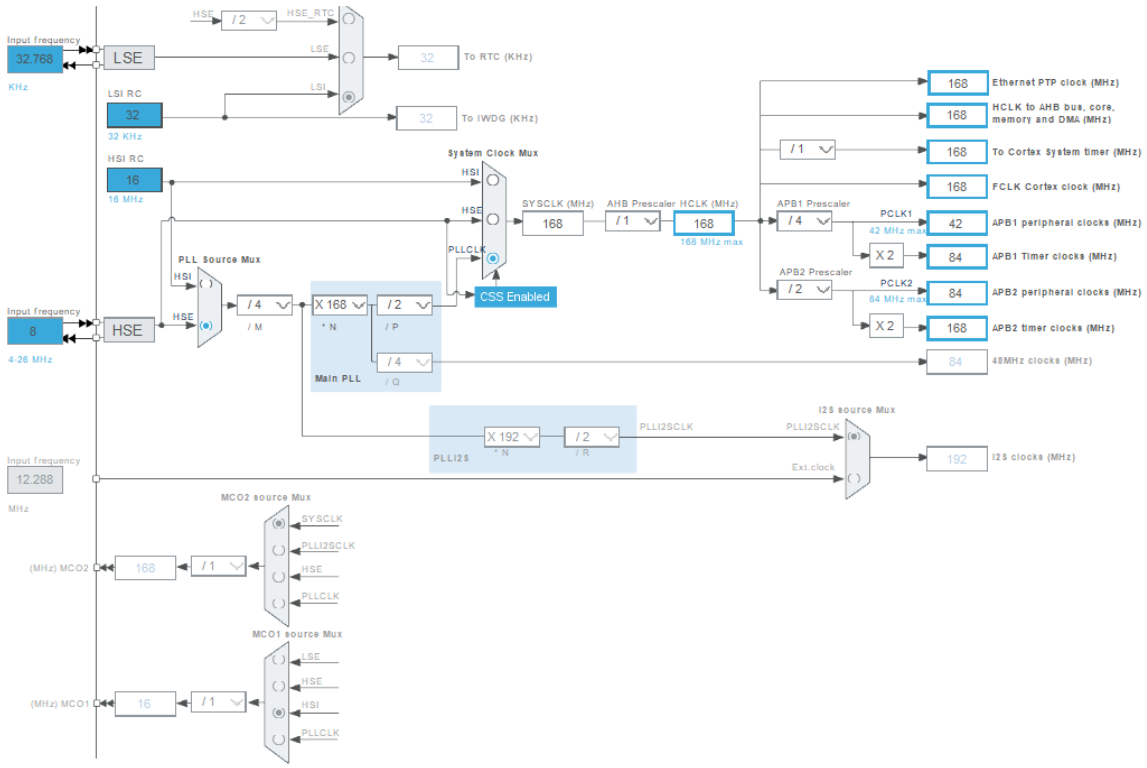
در این جا به تغییر مقدار رجیستر ها در پنجره SFR به صورت زیر پرداختیم و با تغییر دادن بیت های مربوط به پایه های LED تغییرات را در روشن و خاموش شدن LED های مختلف روی برد مشاهده کردیم .



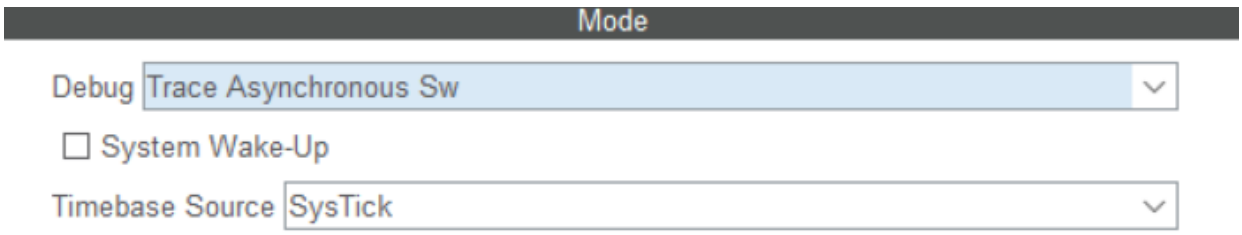
آزمایش سری ۴ :

برای شروع کار مدل پردازنده (STM۳۲F۴۰۷VGT۶) از منوی MCU انتخاب می‌کنیم.

سپس برای اینکه بتوانیم از میکرو استفاده کنیم و از مزایای Debugging آن بهره‌مند شویم باید تغییراتی را در کلاک دستگاه اعمال کنیم. برای تنظیم HSE دستگاه از یک نوسان‌ساز خارجی با فرکانس‌کاری ۸ MHz استفاده می‌کنیم، سپس تامین کلاک پردازنده را از PLL تعیین کرده و منبع آنرا HSE قرار می‌دهیم. حداکثر فرکانس‌کاری این پردازنده ۱۶۸ MHz می‌باشد. با قراردادن فرکانس‌کاری سایر تغییرات به صورت اتوماتیک انجام خواهد شد.



پس از این برای بهره مندی از امکانات دیباگینگ، مد دیباگ را به Trace Asynchronous Sw تغییر می‌دهیم.



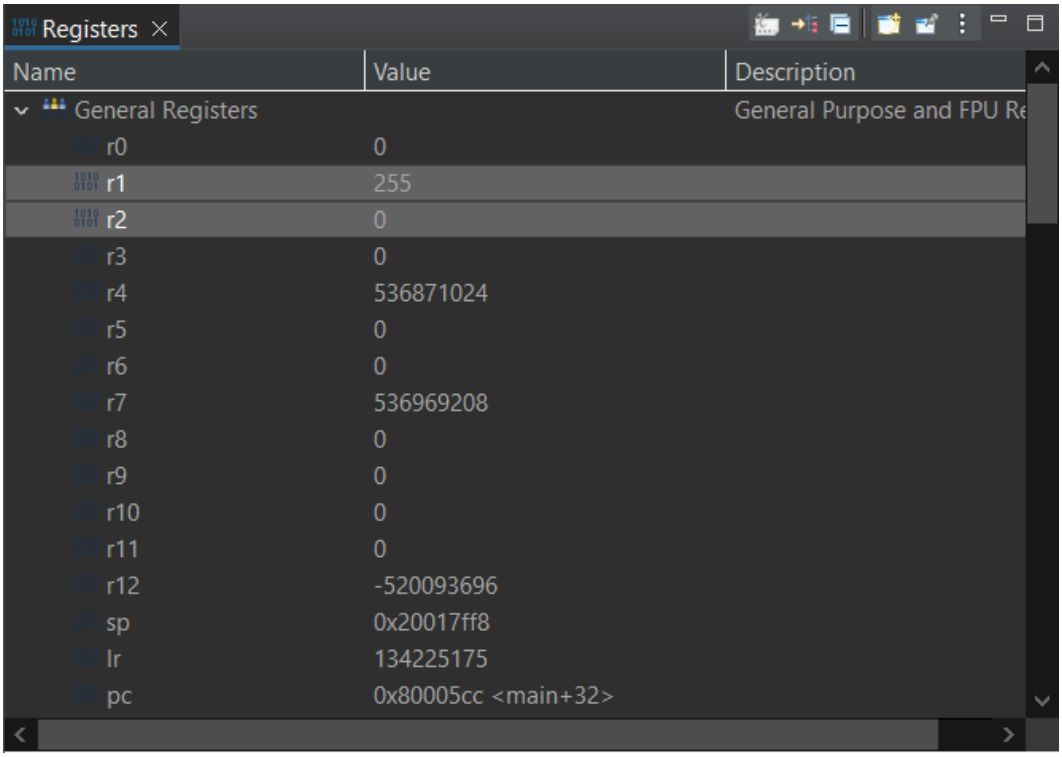
حال که اعمال مقدماتی را انجام دادیم، کافی است گزینه Generate-code را فشار دهیم تا سایر تنظیمات توسط نرم‌افزار انجام شود.

بخش اول:

در این بخش ابتدا با استفاده از دستور MOV برای انتقال عدد هگزادسیمال ۰xFF به داخل رجیسترهای عمومی یک و دو استفاده می‌کنیم. و سپس محتویات رجیستر دوم را دوازده بار شیفت حسابی می‌دهیم. نکته قابل توجه در این بخش، قراردادن کدهای اسمبلی پیش از حلقه تکراری میکروکنترلر است.

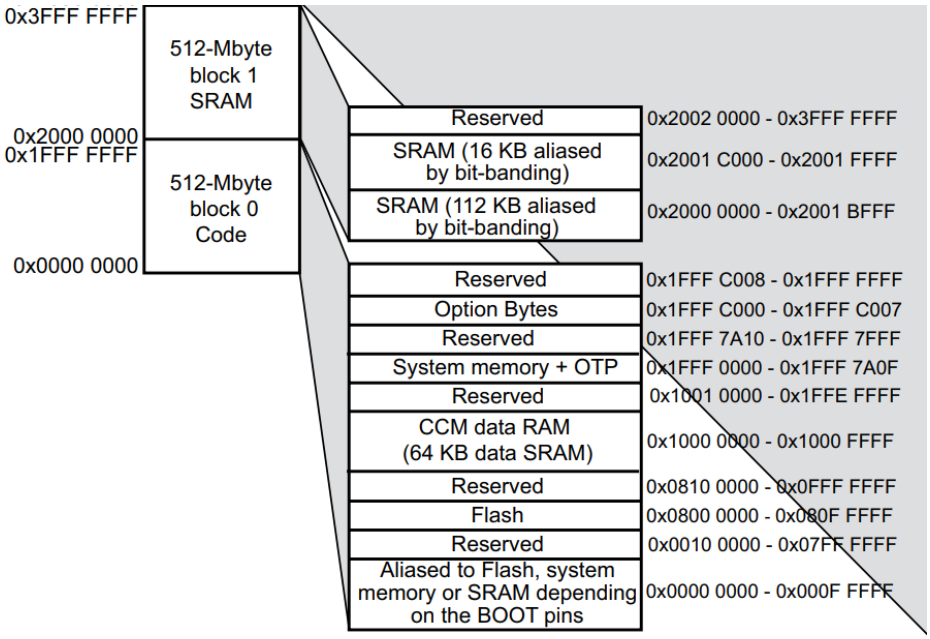
```
/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_USART2_UART_Init();
/* USER CODE BEGIN 2 */
__asm(
  "MOV R1, #0xFF \n"
  "MOV R2, #0xFF \n"
  "ASR R2, #12   \n"
);

/* USER CODE END 2 */
```

خروجی رجیسترها در مد دیباگ مطابق زیر است:



بخش دوم:

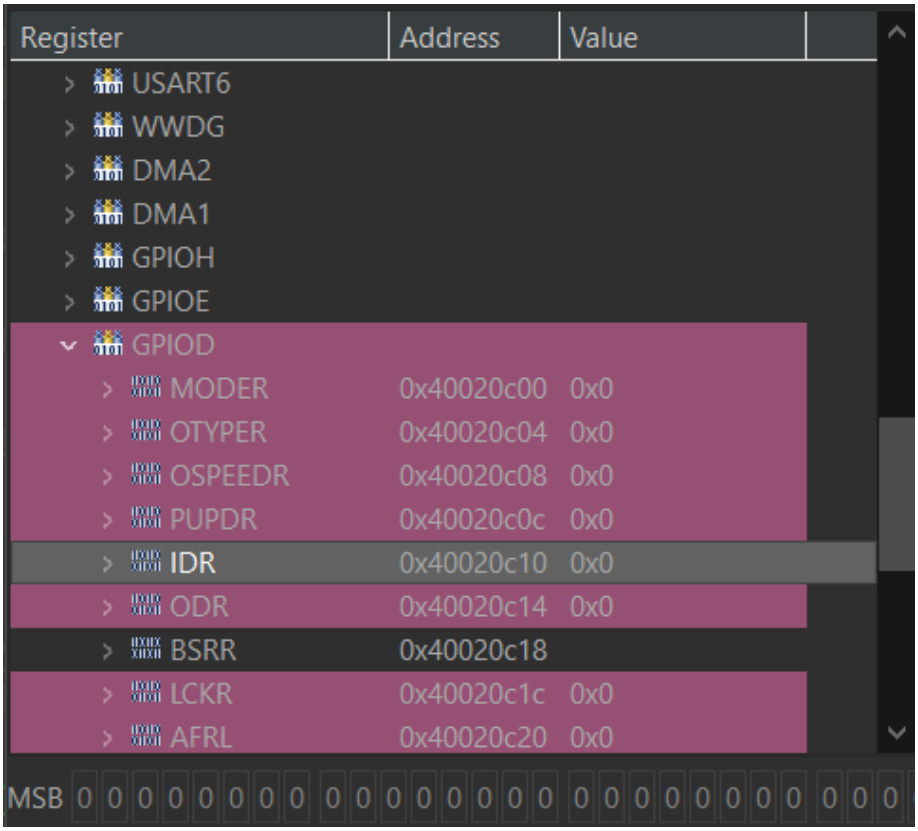
در این بخش ابتدا به ساختار SRAM های موجود در برد موجود در آزمایشگاه می‌پردازیم. به طورکلی ریزپردازنده مذکور دارای ۲ حافظه اصلی با ظرفیت‌های ۱۶KB و ۱۱۲KB، یک باس برای ارتباط با SRAM های خارجی احتمالی و یک SRAM فرعی تحت عنوان BKP-SRAM می‌باشد. براساس دیتاشیت پردازنده، آدرس SRAM مطابق زیر است:



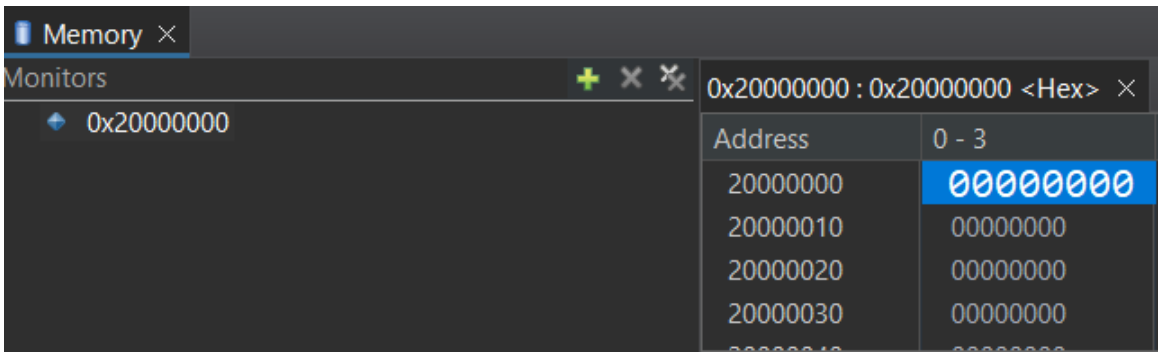
در ادامه ابتدا با استفاده از دستور equ. آدرس‌های‌پایه SRAM و GPIOD را تعریف کرده، سپس مقادیرفوق را در رجیسترهای اول و چهارم قرار می‌دهیم تا از آنها برای آدرس‌دهی استفاده کنیم. می‌دانیم آدرس نسبی رجیستر GPIOD_IDR مقدار هگزادسیمال ۰x۱۰ می‌باشد. با استفاده از آدرس پایه که در رجیستر اول بارگذاری شده و آدرس نسبی، مقدار رجیستر مذکور را در رجیستر صفرم بارگذاری می‌کنیم. در پایان مقدار رجیستر فوق را در SRAM که آدرس پایه آن در رجیستر چهارم ذخیره شده است ذخیره می‌کنیم.

```
__asm(
        ".EQU  SR_ADDH, 0x2000      \n"
        ".EQU  SR_ADDL, 0x0000      \n"
        ".EQU  GD_ADDH, 0x4002      \n"
        ".EQU  GD_ADDL, 0x0C00      \n"
        " MOV  R4     , SR_ADDL     \n"
        " MOVT R4     , SR_ADDH     \n"
        " MOV  R1     , GD_ADDL     \n"
        " MOVT R1     , GD_ADDH     \n"
        " LDR  R0     , [R1,#0x10]  \n"
        " STR  R0     , [R4]        \n"
);
```

در مرحله بعد مقدار رجیستر مذکور را ابتدا از SFR می‌بینیم:



همچنین مقدار فوق که همان صفر است را می‌توان از خانه اول SRAM نیز مشاهده کرد:



با توجه به اینکه در این بخش هنوز پایه خاصی تنظیم نشده است، نتیجه فوق منطقی به نظر می‌رسد.