

گزارشکار جلسه پنجم ۲۰ آبان

آزمایشگاه سیستم‌های ریزپردازنده و مدارهای واسطه گروه ۳

آنوشا شریعتی ۹۹۲۳۰۴۱

مهشاد اکبری سریزدی ۹۹۲۳۰۹۳

امیرحسین منصوری

آزمایش سری ۴:

بخش سوم: آزمایش روشن کردن LED که در سری ۳ انجام شد را با **assembly** تکرار کنید.

در ابتدا با دستور EQU، آدرس پایه رجیستر RCC\_AHB۱ که مربوط به کلاک است را تعریف میکنیم و با دستورات MOV و MOVT مقدار آن را در رجیستر عمومی ۲۰ ذخیره میکنیم. برای یک کردن بیت سوم این رجیستر مقدار هگز ۰۰۰۸ ۰۰۰۰ را ابتدا درون متغیری به نام ۱value ریخته و سپس به رجیستر عمومی ۲۱ منتقل میکنیم. در نهایت این مقدار را با دستور Str در خانه با آدرس تعریف شده در رجیستر ۲۰ به همراه آفست ۰X۳۰ ذخیره میکنیم.

//part2

```
".EQU clk_adrs1,0X3800\n"
".EQU clk_adrsh,0X4002\n"
".EQU value1l ,0X0008\n"
".EQU value1h,0X0000\n"
"MOV r0 , clk_adrs1\n"
"MOVT r0 , clk_adrsh\n"
"MOV r1 , value1l\n"
"MOVT r1 , value1h\n"
"STR R1 , [R0 , #0X30]\n"
```

در ادامه آدرس رجیسترهای moder و bsrr را با دستور EQU. تعریف کرده و به ترتیب در رجیسترهای r2 و r4 میریزیم. برای فعال کردن مود خروجی باید بیت‌های ۳۰ و ۲۸ و ۲۶ و ۲۴ رجیستر moder را ۱ کرد. برای انجام این کار عدد هگز ۰X۵۵۰۰۰۰۰۰ را در متغیری به نام value\_mdr ذخیره کرده و به رجیستر عمومی r3 منتقل میکنیم. سپس مقدار این رجیستر را در خانه با آدرس ذخیره شده در r2 و آفست ۰X۰۰ ذخیره میکنیم.

```
".EQU GD_ADRSh, 0X4002\n"//
".EQU GD_ADRS1, 0X0C00\n"
"MOV R2 , GD_ADRS1\n"
"MOVT R2 , GD_ADRSh\n"
".EQU value_mdr1 ,0X0000\n"
".EQU value_mdrh,0X5500\n"
"MOV R3 , value_mdr1\n"
"MOVT R3 , value_mdrh\n"
"STR R3 , [R2 ,#0X00 ]\n"
```

برای یک کردن بیت‌های ۱۲ و ۱۳ و ۱۴ و ۱۵ رجیستر bsrr هم عدد هگز ۰X۰۰۰۰ f۰۰۰ را در متغیر value\_bsrr ریخته و به رجیستر r4 انتقال داده سپس مقدار این رجیستر را در خانه با آدرس ذخیره شده در r2 و آفست ۰X۱۸ ذخیره میکنیم.

```
".EQU value_bsrr1 ,0Xf000\n"
".EQU value_bsrrh,0X0000\n"
"MOV R4 , value_bsrr1\n"
"MOVT R4 , value_bsrrh\n"
"STR R4 , [R2 ,#0X18 ]\n"
```

توجه شود که کدهای بالا درون تابع \_\_asm() و پیش از حلقه تکرار میکروکنترلر نوشته شده است.

در ادامه با ران کردن برنامه و وارد شدن به مود دیباگ مقادیر رجیستر ها را در قسمت SFR مشاهده میکنیم. و همچنین با روشن شدن LED ها متوجه میشویم برنامه به درستی کار میکند.

The screenshot shows the Keil uVision IDE interface. The main editor displays assembly code for initializing the GPIOC peripheral:

```
//part2  
.EQU clk_adrs1,0X3800\n //rcc adrs  
.EQU clk_adrhs,0x4002\n".EQU value1l ,0x0008\n //idr adrs  
.EQU value1h,0x0000\nMOV r0 , clk_adrs1\nMOVT r0 , clk_adrhs\nMOV r1 , value1l\nMOVT r1 , value1h\nSTR R1,[R0,#0X30]\n".EQU GD_ADRSh, 0x4002\n//gpiod adrs  
.EQU GD_ADRS1, 0x0c00\nMOV R2 , GD_ADRS1\nMOVT R2 , GD_ADRSH\n.EQU value_mdr1 ,0x0000\n //moder adrs  
.EQU value_mdrrh,0x5500\nMOV R3 , value_mdr1\nMOVT R3 , value_mdrrh\nSTR R3 ,[R2,#0x00 ]\n.EQU value_bsrl1 ,0xf000\n //bsrr  
.EQU value_bsrhh,0x0000\nMOV R4 , value_bsrl1\nMOVT R4 , value_bsrhh\nSTR R4 ,[R2,#0x18 ]\n
```

To the right, the "Peripheral Register Window" is open, displaying information about the GPIOD peripheral:

- Type filter text:
- Register tree:
  - > GPIOF
  - > GPIOE
  - <> GPIOD
    - > MODER: Address 0x4002..., Value 0x55000000
    - > OTYPER: Address 0x4002..., Value 0x0
    - > OSPEEDR: Address 0x4002..., Value 0x0
    - > PUPDR: Address 0x4002..., Value 0x0
    - > IDR: Address 0x4002..., Value 0xf020
    - > ODR: Address 0x4002..., Value 0xf000
    - > BSRR: Address 0x4002..., Value 0x0
    - > LCKR: Address 0x4002..., Value 0x0
    - > AFRL: Address 0x4002..., Value 0x0
    - > AFRH: Address 0x4002..., Value 0x0
  - > GPIOC

Below the register tree, the MSB (Most Significant Bit) field is shown as a binary sequence: 0 1 0 1 0 1 0 1 0.

A detailed view of the selected MODER register is provided below:

Field	Value
Register:	MODER
Address:	0x4002c00
Value:	0x55000000
Size:	32
Reset value:	0x0
Reset mask:	0xFFFFFFFF

The screenshot displays the Keil uVision IDE with the assembly code for `stm32f4xx_hal.c` open in the main editor. The code defines constants for clock and address registers, then configures the GPIO pin (R0) as an output using the `GPIO_Init` function. The registers are configured with `GPIO_MODE_OUTPUT_PP`, `GPIO_SPEED_FREQ_LOW`, and `GPIO_PULLUP_DISABLE`. The pin is then driven high by setting `R0` to `0x30`.

On the right, the 'General Registers' window is open, showing the current state of the processor's registers. Register `r1` is highlighted, showing its value as `8`.

Name	Value	Description
General Registers		
r0	1073887232	General Purpose and...
r1	8	
r2	1073875968	
r3	1426063360	
r4	61440	
r5	0	
r6	0	
r7	537001976	
r8	0	
r9	0	
r10	0	
r11	0	
r12	0	

Below the register list, the 'Name : r1' section shows the following values:

- Hex: 0x8
- Decimal: 8
- Octal: 010
- Binary: 1000
- Default: 8

## آزمایش سری ۵:

بخش اول : تمرین جلسه قبل مربوط به روشن کردن LED را این بار در قالب فایل S. به صورت یک فایل اسمبلی جداگانه بنویسید.

یک فایل با پسوند S. ایجاد کرده و کد قسمت قبل را در آن کپی میکنیم. بالای برنامه خط یک و دو را اضافه میکنیم و در آخر برنامه نیز دستور BX LR که مقدار لینک رجیستر را ریترن میکند اضافه میکنیم.

```

1.global led_on
2.led_on:
3
4    .EQU clk_adrs,0x40023800 @ncc adrs
5    .EQU value1,0x00000008 @idr adrs
6    LDR r0, =clk_adrs
7    LDR r1, =value1
8    STR R1, [R0, #0x30]
9
10   .EQU GD_ADRS, 0x40020C00 @gpio adrs
11   LDR R2, =GD_ADRS
12   .EQU value_mdr, 0x55000000 @moder adrs
13   LDR R3, =value_mdr
14   STR R3, [R2, #0x00]
15
16   .EQU value_bsrr, 0x0000f000 @bsrr
17   LDR R4, =value_bsrr
18   STR R4, [R2, #0x18]
19
20   BX LR
  
```

سپس درون main.c اسم تابع نوشته شده به زبان اسمبلی که در اینجا led\_on است را با دستور extern تعریف کرده و داخل تابع وایل از آن استفاده میکنیم. و مشاهده میکنیم که برنامه به درستی کار میکند.

```

39
40 /* USER CODE END PM */
41
42 /* Private variables -----
43
44 /* USER CODE BEGIN PV */
45
46 /* USER CODE END PV */
47
48 /* Private function prototypes -----
49 void SystemClock_Config(void);
50 static void MX_GPIO_Init(void);
51 /* USER CODE BEGIN PFP */
52
53 /* USER CODE END PFP */
54
55 /* Private user code -----
56 /* USER CODE BEGIN 0 */
57 extern void led_on(void);
58 /* USER CODE END 0 */
59
60 /**
61  * @brief The application entry point.
62  * @retval int
63  */
  
```

```

84 /* USER CODE END SysInit */
85
86 /* Initialize all configured peripherals */
87 MX_GPIO_Init();
88 /* USER CODE BEGIN 2 */
89
90 /* USER CODE END 2 */
91
92 /* Infinite loop */
93 /* USER CODE BEGIN WHILE */
94 while (1)
95 {
96
97     /* USER CODE END WHILE */
98
99
100    while(1){
101        led_on();
102    }
103
104    /* USER CODE BEGIN 3 */
105 }
106 /* USER CODE END 3 */
107 }
108
  
```