

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

ZADANIE K PREDMETU VNORENÉ RIADIACE
SYSTÉMY

Ústav: Robotika a kybernetika

Dňa: Bratislava 3.11.2016

Vypracoval: Peter Kmeť, Miroslav Kohút

Cvičenie: Streda 15:00

Repozitár: https://github.com/Chickenman222/vrs_cv2

Časť 1: Ovládanie s priamym prístupom k registrom

Úloha 1: Ovládanie LED PA5, nastavenie GPIO periférie a príslušného portu.

Podľa zadania sme aktivovali Clock na GPIO perifériách:

```
RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOA, ENABLE);  
RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOC, ENABLE);
```

Nastavenie GPIO mode => OUT

```
GPIOA->MODER |= (1<<10);  
GPIOA->MODER &= ~(uint32_t)(1<<11);
```

Nastavenie Output type => PushPull

```
GPIOA->OTYPER &= ~(uint32_t)(1<<5);
```

Nastavenie PullUp PullDown => UP

```
GPIOA->PUPDR |= (1<<10);  
GPIOA->PUPDR &= ~(uint32_t)(1<<11);
```

Nastavenie GPIO speed => Very high speed

```
GPIOA->OSPEEDR |= (1<<10);  
GPIOA->OSPEEDR |= (1<<11);
```

Nastavenie konkrétneho registra sme spravili presne pre konkrétny bit pomocou bitových operácií, aby sme neovplyvnili iné hodnoty registra.

Nastavenie LED bliká použitím všetkých troch možností BSRR ODR a aj toggle ODR.

```
//ODR control  
GPIOA->ODR |= (1 << 5);  
for(i = 0; i<hranica;i++){  
}  
GPIOA->ODR &= ~(uint32_t)(1<<5);  
for(i = 0; i<hranica;i++){  
}  
  
//BSRR control  
GPIOA->BSRRH |= (1 << 5);  
for(i = 0; i<100000;i++);  
GPIOA->BSRRL |= (1 << 5);  
for(i = 0; i<100000;i++);  
//ODR XOR control  
  
GPIOA->ODR ^= (1 << 5);  
for(i = 0; i<100000;i++);
```

Name	Value	Description
General Registers		General Purpose and FPU Register Gr...
r0	1	
r1	99999	
r2	1073872896	
r3	0	
r4	0	
r5	0	
r6	0	
r7	536952808	
r8	0	
r9	0	
r10	0	
r11	0	
r12	0	
sp	0x20013fe8	
lr	134218183	
pc	0x800026c <main+180>	
xpsr	-2130706432	
PRIMASK	0	
BASEPRI	0	
FAULTMASK	0	
CONTROL	0	
MSP	536952808	
PSP	0	

Obr.1 Stav registrov LED nesvieti

Name	Value	Description
General Registers		General Purpose and FPU Register Gr...
r0	1	
r1	99999	
r2	1073872896	
r3	32	
r4	0	
r5	0	
r6	0	
r7	536952808	
r8	0	
r9	0	
r10	0	
r11	0	
r12	0	
sp	0x20013fe8	
lr	134218183	
pc	0x800022e <main+118>	
xpsr	-2130706432	
PRIMASK	0	
BASEPRI	0	
FAULTMASK	0	
CONTROL	0	
MSP	536952808	
PSP	0	

Obr.2 Stav registrov: LED svieti

Úloha 2. Nastavenie Pinu PC13 ako tlačidlo, snímanie stavu stlačenia

Nastavenie GPIO mode => IN

```
GPIOC->MODER &= ~(uint32_t) (1<<26);  
GPIOC->MODER &= ~(uint32_t) (1<<27);
```

Nastavenie Output type => PushPull

```
GPIOC->OTYPER &= ~(uint32_t) (1<<13);
```

Nastavenie PullUp PullDown => NOPULL

```
GPIOC->PUPDR &= ~(uint32_t) (1<<26);  
GPIOC->PUPDR &= ~(uint32_t) (1<<27);
```

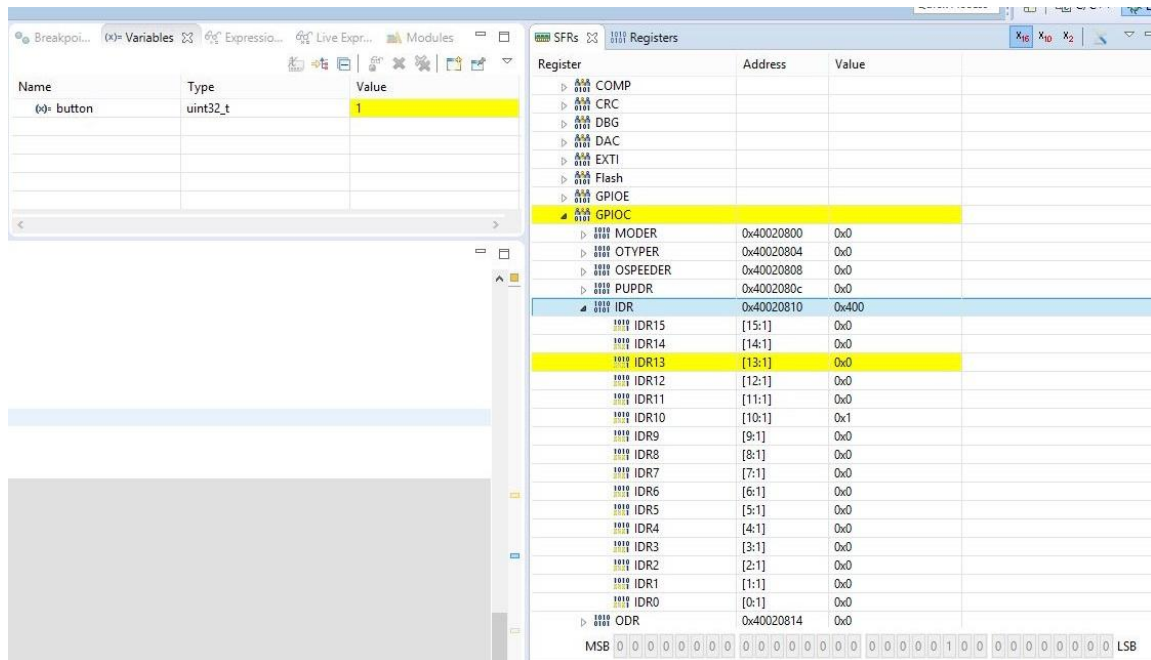
Prepínanie premennej pre Button

```
BUTTON = !(GPIOC->IDR & (uint32_t) (1<<13));
```

The screenshot displays the STM32CubeIDE interface. On the left, the 'Variables' window shows a variable named 'button' of type 'uint32_t' with a value of 0. On the right, the 'Registers' window shows the GPIOC register set. The 'IDR' register is highlighted, showing a value of 0x2400. The bit field for pin PC13 (IDR13) is highlighted, showing a value of 0x1.

Register	Address	Value
COMP		
CRC		
DBG		
DAC		
EXTI		
Flash		
GPIOE		
GPIOC		
MODER	0x40020800	0x0
OTYPER	0x40020804	0x0
OSPEEDER	0x40020808	0x0
PUPDR	0x4002080c	0x0
IDR	0x40020810	0x2400
IDR15	[15:1]	0x0
IDR14	[14:1]	0x0
IDR13	[13:1]	0x1
IDR12	[12:1]	0x0
IDR11	[11:1]	0x0
IDR10	[10:1]	0x1
IDR9	[9:1]	0x0
IDR8	[8:1]	0x0
IDR7	[7:1]	0x0
IDR6	[6:1]	0x0
IDR5	[5:1]	0x0
IDR4	[4:1]	0x0
IDR3	[3:1]	0x0
IDR2	[2:1]	0x0
IDR1	[1:1]	0x0
IDR0	[0:1]	0x0
ODR	0x40020814	0x0

Obr. 3. Stav registrov: Nestlačené tlačidlo



Obr. 4. Stav registrov: Stlačené tlačidlo

Úloha 3:

Rozdelili sme ju na 3 časti jednotlivé časti kódu môžete nájsť v commitoch na githube.

Úloha 3.1

Ledka bliká delay je nastavený for cyklom.

```
uint64_t hranica = 500000;
GPIOA->ODR |= (1 << 5);
for(i = 0; i<hranica;i++){
}
GPIOA->ODR &= ~(uint32_t) (1<<5);
for(i = 0; i<hranica;i++){
}
```

Úloha 3.2

Program sleduje tlačidlo a zobrazuje ho na LED plus funkcia na filtrovanie prekmitov pri zmene stavu tlačidla.

Funkcia na filtrovanie

```
uint32_t button_control() {
    while(sum_good_values < 20){
        button_current = !(GPIOC->IDR & (uint32_t) (1<<13));
        if(button_previous == button_current)
            sum_good_values++;
    }
}
```

```

        else
            sum_good_values = 0;
            button_previous = !(GPIOC->IDR & (uint32_t)(1<<13));
            for(i = 0; i<10;i++);
        }
        sum_good_values =0;
        return button_current;
    }
}

```

Blikanie led

```

if(button_control())
    GPIOA->ODR |= (1 << 5);

else
    GPIOA->ODR &= ~(uint32_t)(1<<5);

```

Úloha 3.3

Kontrola nábežnej hrany a zmena stavu led stlačením tlačidla.

```

if(button_control() && !hrana_check){
    GPIOA->ODR ^= (1 << 5);
    hrana_check = 1;
}
else if(!button_control() && hrana_check){
    hrana_check = 0;
}

```

Časť 2: Vypracovanie časti jedna s využitím knižníc

Inicializovanie GPIO štruktúry a pridelenie Pin-u

```
GPIO_InitTypeDef GPIO_LED;  
GPIO_LED.GPIO_Pin = GPIO_Pin_5;
```

Nastavenie GPIO mode => OUT

```
GPIO_LED.GPIO_Mode = GPIO_Mode_OUT;
```

Nastavenie Output type => PushPull

```
GPIO_LED.GPIO_OType = GPIO_OType_PP;
```

Nastavenie PullUp PullDown => UP

```
GPIO_LED.GPIO_PuPd = GPIO_PuPd_UP;
```

Nastavenie GPIO speed => Very high speed

```
GPIO_LED.GPIO_Speed = GPIO_Speed_40MHz;
```

Inicializovanie portu A

```
GPIO_Init(GPIOA, &GPIO_LED);
```

Podobné nastavenie aj pre tlačidlo na pin-e 13 (Speed, PushPull, Mode...)

```
GPIO_InitTypeDef GPIO_BUTTON;  
GPIO_BUTTON.GPIO_Pin = GPIO_Pin_13;  
GPIO_BUTTON.GPIO_Mode = GPIO_Mode_IN;  
GPIO_BUTTON.GPIO_OType = GPIO_OType_PP;  
GPIO_BUTTON.GPIO_PuPd = GPIO_PuPd_NOPULL;  
GPIO_Init(GPIOC, &GPIO_BUTTON);
```

Úloha 1: Ovládanie led s využitím funkcie SET/Reset Bits

```
GPIO_SetBits(GPIOA, GPIO_LED.GPIO_Pin);  
for(i = 0; i<hranica; i++);  
GPIO_ResetBits(GPIOA, GPIO_Pin_5);  
for(i = 0; i<hranica; i++);
```

Úloha 1: Ovládanie led s využitím funkcie Toggle Bits

```
GPIO_ToggleBits(GPIOA, GPIO_Pin_5);  
for(i = 0; i<hranica; i++)
```

Úloha 2: Kontrola stavu tlačidla a uloženie do premennej

```
uint8_t BUTTON = 0;  
BUTTON = !GPIO_ReadInputDataBit(GPIOC, GPIO_BUTTON.GPIO_Pin);
```

Uloha 3, časť 3: LED bliká pomocou toggle funkcie v požadovanom intervale

Tu sa použila rovnaká funkcia `GPIO_ToggleBits()` ako v úlohe 1. Tam sme ledky

taktiež nechávali blikať v určenom intervale a následne pauzovali program a sledovali registre.

Úloha 3, časť 2: LED zmení svoj stav po stlačení tlačidla

Pre druhú a tretiu časť úlohy 3 bola vytvorená funkcia `button_control_library`, ktorá detekuje stlačenie tlačidla a odfiltráva prekmity. Je to len upravená funkcia `button_control`.

```
uint32_t button_control_library() {  
    while(sum_good_values < 20) {  
        button_current =  
!GPIO_ReadInputDataBit(GPIOC,GPIO_Pin_13);  
        if(button_previous == button_current)  
            sum_good_values++;  
        else  
            sum_good_values = 0;  
        button_previous =  
!GPIO_ReadInputDataBit(GPIOC,GPIO_Pin_13);  
        for(i = 0; i<10;i++);  
    }  
    sum_good_values =0;  
    return button_current;  
}
```

Príkazy na zmenu stavu led po stlačení tlačidla:

```
if(button_control_library() && !hrana_check){  
    GPIO_ToggleBits(GPIOA,GPIO_Pin_5);  
    hrana_check = 1;  
}  
else if(!button_control_library() && hrana_check){  
    hrana_check = 0;  
}
```

Úloha 3, časť 3: LED zrkadlí stav tlačidla

```
if( button_control_library()){  
    GPIO_SetBits(GPIOA,GPIO_LED.GPIO_Pin);  
}  
else{  
    GPIO_ResetBits(GPIOA,GPIO_LED.GPIO_Pin);  
}
```

Záver:

Úspešne sa nám podarilo splniť obe zadania a zariadenie sa správalo presne tak ako bolo požadované. Či už pomocou nadstavenia registrov manuálne, alebo využitím základných GPIO knižníc. Taktiež sme naprogramovali funkciu, ktorá filtruje prekmity pri zmene stavu tlačidla a v programe sme detekovali nábežnú hranu pri stláčaní tlačidla. Registre boli nastavené bez vplyvu na ostatné periférie.