Sprawozdanie

Systemy wbudowane

Ćwiczenie 3:

Przerwania i timer systemowy.

Wykonanie:

Busłowski Tomasz

Suchwałko Tomasz

Skrouba Kamil

Zawadzka Magdalena

(Grupa PS3)

Prowadzący zajęcia: **dr inż. Adam Klimowicz**

SW, semestr VI, 04-03-2017, Wydział Informatyki, Politechnika Białostocka

Zakres Materiału

1. Kontroler przewań NVIC.
2. Timer systemowy SysTick
3. Wykorzystanie przerwań od timera systemowego i portów GPIOx.

Zadania do wykonania

1. Napisz funkcję opóźnienia void Delay (uint32_t ms) wykorzystującą timer SysTick, gdzie ms – wartość opóźnienia w milisekundach. Wykorzystaj ją np. do migania diodą LED.

2. Napisz program, który miga 8 diodami w taki sposób, że pierwsza miga z częstotliwością 2Hz, a każda następna miga z częstotliwością o 20% wyższą niż poprzednia.

3. Napisz program, który wykorzystując przerwania od wyprowadzeń zewnętrznych steruje miganiem diod LED w następujący sposób: Każdy z przycisków SW0 – SW4 włącza lub wyłącza miganie danej diody w następujący sposób: pierwsze naciśnięcie – dioda świeci, drugie dioda miga, trzecie – dioda gaśnie itd. Częstotliwość migania diod powinna być regulowana przy pomocy joysticka (np. góradół) w zakresie od 1 do 4Hz ze skokiem co 0,5 Hz (7 wartości).

Zadanie 1

Treść:
Napisz funkcję opóźnienia void Delay (uint32_t ms) wykorzystującą timer SysTick, gdzie ms – wartość opóźnienia w milisekundach. Wykorzystaj ją np. do migania diodą LED.

Realizacja:
Połączyliśmy kabelkami port jednej z diód z portem GPIOB_PB0. Z treści przykładu 1 z instrukcji do zajęć dowiedzieliśmy się, że aby przerwanie występowało co 1 milisekundę, należy zdefiniowaną SysTick_Frequency o wartości 9000000 (9MHz) podzielić przez 1000. Napisaliśmy funkcję delay(uint32_t ms) która jako parametr przyjmuje wartość w milisekundach określającą odstępy pomiędzy kolejnymi przerwaniami(Rysunek 1).

```
void Delay (uint32_t ms)
{
  if (SysTick_Config((SysTick_Frequency / 1000) * ms))
  {
    while(1);
  }
  SysTick_CLKSourceConfig(SysTick_CLKSource_HCLK_Div8);
}
```

Rysunek 1

Nasza funkcja main, w której ustalamy, że przerwania będą się wykonywały co 2sekundy(2000ms) oraz funkcja SysTick_Handler, w której zmieniamy stan diody:

```
int main(void)
{
  RCC_Config();
  GPIO_Config();

  Delay(2000);

  while (1)
  {

  };

  return 0;
}
```

Rysunek 2 (funkcja main)

```
void SysTick_Handler(void)
{
  GPIO_WriteBit(GPIOB,GPIO_Pin_0,(BitAction)(1-GPIO_ReadOutputDataBit(GPIOB,GPIO_Pin_0)));
}
```

Rysunek 3(SysTick_Handler)

Zadanie 2

Treść:
Napisz program, który miga 8 diodami w taki sposób, że pierwsza miga z częstotliwością 2Hz, a każda następna miga z częstotliwością o 20% wyższą niż poprzednia.

Realizacja:
Podłączyliśmy 8 portów diód z portami GPIOB(PB0-PB7). Mieliśmy kilka koncepcji wykoania tego zadania, jednak finalnie nasza wersja zadania wygląda następująco: zdecydowaliśmy się na jeden wspólny counter dla wszystkich diód. Wyliczyliśmy matematycznie, że jeżeli dioda0 ma migać z częstotliwością 2MHz a kolejne 20% szybciej od poprzedniej diody, to zmiany stanów kolejnych diód będa następowały kolejno co: 558ms, 670ms, 804ms, 965ms, 1158ms, 1388ms, 1666ms i 2000ms. Wyliczyliśmy NWW tych wartości, aby poruszać się w ograniczonych wartościach naszego countera. Aby zwiększyć szansę, że w trakcie trwania jednego przerwania program zdąży wykonać wszystkie instrukcje warunkowe, to przerwanie następuje co 10ms i counter jest zwiększany o 10. Do wywołania przerwania co 10ms użyliśmy funkcji napisanej przez nas w ramach zadania 1.

```
void SysTick_Handler(void)
{
    if(counter%560 == 0)     { changeDiodeStatus(7); }
    if(counter%670 == 0)     { changeDiodeStatus(6); }
    if(counter%800 == 0)     { changeDiodeStatus(5); }
    if(counter%960 == 0)     { changeDiodeStatus(4); }
    if(counter%1160 == 0)    { changeDiodeStatus(3); }
    if(counter%1390 == 0)    { changeDiodeStatus(2); }
    if(counter%1670 == 0)    { changeDiodeStatus(1); }
    if(counter%2000 == 0)    { changeDiodeStatus(0); }

    if(counter == 2085646912398000)    { counter = 0; }

    counter+=10;
}
```

Rysunek 4

```
void changeDiodeStatus(uint32_t nr)
{
  switch (nr)
  {
    case 0:
      GPIO_WriteBit(GPIOB,GPIO_Pin_0,(BitAction)(1-GPIO_ReadOutputDataBit(GPIOB,GPIO_Pin_0)));
      break;

    case 1:
      GPIO_WriteBit(GPIOB,GPIO_Pin_1,(BitAction)(1-GPIO_ReadOutputDataBit(GPIOB,GPIO_Pin_1)));
      break;

    case 2:
      GPIO_WriteBit(GPIOB,GPIO_Pin_2,(BitAction)(1-GPIO_ReadOutputDataBit(GPIOB,GPIO_Pin_2)));
      break;

    case 3:
      GPIO_WriteBit(GPIOB,GPIO_Pin_3,(BitAction)(1-GPIO_ReadOutputDataBit(GPIOB,GPIO_Pin_3)));
      break;

    case 4:
      GPIO_WriteBit(GPIOB,GPIO_Pin_4,(BitAction)(1-GPIO_ReadOutputDataBit(GPIOB,GPIO_Pin_4)));
      break;

    case 5:
      GPIO_WriteBit(GPIOB,GPIO_Pin_5,(BitAction)(1-GPIO_ReadOutputDataBit(GPIOB,GPIO_Pin_5)));
      break;

    case 6:
      GPIO_WriteBit(GPIOB,GPIO_Pin_6,(BitAction)(1-GPIO_ReadOutputDataBit(GPIOB,GPIO_Pin_6)));
      break;

    case 7:
      GPIO_WriteBit(GPIOB,GPIO_Pin_7,(BitAction)(1-GPIO_ReadOutputDataBit(GPIOB,GPIO_Pin_7)));
      break;
  }
}
```

Rysunek 5

Zadanie 3

Treść:
Napisz program, który wykorzystując przerwania od wyprowadzeń zewnętrznych steruje miganiem diod LED w następujący sposób: Każdy z przycisków SW0 – SW3 włącza lub wyłącza miganie danej diody w następujący sposób: pierwsze naciśnięcie – dioda świeci, drugie dioda miga, trzecie – dioda gaśnie itd. Częstotliwość migania diod powinna być regulowana przy pomocy joysticka (np. góradół) w zakresie od 1 do 4Hz ze skokiem co 0,5 Hz (7 wartości).

Realizacja:

Połączyliśmy porty diód do odpowiednich portów GPIOB, porty przycisków oraz joysticka do odpowiednich portów GPIOA. Przerwania wewnętrzne ustawiamy na wykonywanie co ½ sekundy. W funkcji konfiguracyjnej kontroler przerwań NVIC_Config(void) inicjalizujemy strukturę i ustawiamy adres tablicy wektorów.

Oprócz tego w tej samej funkcji konfigurujemy przerwania:

EXTI0:

Wybieramy model grupowania przerwań: NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1);

Wybieramy konfigurowany IRQ: NVIC_InitStructure.NVIC_IRQChannel = EXTI0_IRQn;

Wybieramy priorytet grupowy: NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 1;

Wybieramy podpriorytet: NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;

Włączamy obsługę IRQ: NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStructure);

Ustawiamy źródło przerwania:
GPIO_EXTILineConfig(GPIO_PortSourceGPIOA, GPIO_PinSource0);

Konfigurujemy przerwania EXTI0 na linię 0: EXTI_InitStructure.EXTI_Line = EXTI_Line0;

Ustawiamy generowania przerwania (a nie zdarzenia):
EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;

Ustawiamy wyzwalanie przerwania zboczem opadającym (wciśnięcie przycisku):
EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;

Włączamy przerwanie: EXTI_InitStructure.EXTI_LineCmd = ENABLE; 8
EXTI_Init(&EXTI_InitStructure);

Tak samo konfigurujemy:

- EXTI1 – wyzwalane przez pin 1 portu GPIOB (połączony z LED1)
- EXTI2 – wyzwalane przez pin 2 portu GPIOB (połączony z LED2)
- EXTI3 – wyzwalane przez pin 3 portu GPIOB (połączony z LED3)
- EXTI9_5 – wyzwalane przez pin 5 portu GPIOB (połączony z wyjściem S joysticka)
- EXTI15_10 – wyzwalane przez pin 10 portu GPIOB (połączony z wyjściem N joysticka)

W pliku stm32f10x.h należy zaimplementować wnętrze funkcji, które są wykonywane przy wciskaniu wyżej wymienionych przycisków:

- void EXTI0_IRQHandler(void) – wywoływana przy wciśnięciu przycisku SW0
- void EXTI1_IRQHandler(void) – wywoływana przy wciśnięciu przycisku SW1
- void EXTI2_IRQHandler(void) – wywoływana przy wciśnięciu przycisku SW2
- void EXTI3_IRQHandler(void) – wywoływana przy wciśnięciu przycisku SW3
- void EXTI9_5_IRQHandler(void) – wywoływana przy wciśnięciu przycisku S joysticka
- void EXTI15_10_IRQHandler(void) – wywoływana przy wciśnięciu przycisku N joysticka

Na początku deklarujemy zmienne, które będą zliczać naciśnięcia przycisków SW0 – SW3:

```
int nacisniecia0 = 0;
int nacisniecia1 = 0;
int nacisniecia2 = 0;
int nacisniecia3 = 0;
```

Oraz zmienne, które określają, czy dana dioda powinna migać:

```
int miganie0 = 0;
int miganie1 = 0;
int miganie2 = 0;
int miganie3 = 0;
```

Implementacje funkcji EXTI0 – EXTI3 pokażemy na przykładzie EXTI0:

void EXTI0_IRQHandler(void) { if (EXTI_GetITStatus(EXTI_Line0) != RESET){

Inkrementujemy licznik naciśnięć:

nacisniecia0++;

Jeśli jest to pierwsze naciśnięcie – anulujemy miganie diody oraz zapalamy daną diodę:

```
if(nacisniecia0 == 1)
{
	miganie0 = 0;
	GPIO_WriteBit(GPIOB, GPIO_Pin_0, 1);
}
```

Jeśli jest to drugie naciśnięcie – włączamy miganie diody:

```
if(nacisniecia0 == 2)
{
	miganie0 = 1;
}
```

Jeśli jest to trzecie naciśnięcie – anulujemy miganie diody, gasimy daną diodę i zerujemy licznik zliczający naciśnięcia:

```
if(nacisniecia0 == 3)
{
	miganie0 = 0;
	GPIO_WriteBit(GPIOB, GPIO_Pin_0, 0);
	nacisniecia0 = 0;
}
```

Zerujemy flagę obsługi przerwania:

```
			EXTI_ClearITPendingBit(EXTI_Line0);
	}
}
```

Analogicznie implementujemy pozostałe funkcje EXTI1 – EXTI3:

```
void EXTI1_IRQHandler(void)
{
	if (EXTI_GetITStatus(EXTI_Line1) != RESET){
	nacisniecia1++;
	if(nacisniecia1 == 1)
	{
		miganie1 = 0;
		GPIO_WriteBit(GPIOB, GPIO_Pin_1, 1);
	}
	if(nacisniecia1 == 2)
	{
		miganie1 = 1;
	}
```

```
        if(nacisniecia1 == 3)
        {
                miganie1 = 0;
                GPIO_WriteBit(GPIOB, GPIO_Pin_1, 0);
                nacisniecia1 = 0;
        }
        EXTI_ClearITPendingBit(EXTI_Line1);
        }
}

void EXTI2_IRQHandler(void)
{
        if (EXTI_GetITStatus(EXTI_Line2) != RESET)
        {
                nacisniecia2++;
                if(nacisniecia2 == 1)
                {
                        miganie2 = 0;
                        GPIO_WriteBit(GPIOB, GPIO_Pin_2, 1);
                }
                if(nacisniecia2 == 2)
                {
                        miganie2 = 1;
                }
                if(nacisniecia2 == 3)
                {
                        miganie2 = 0;
                        GPIO_WriteBit(GPIOB, GPIO_Pin_2, 0);
                        nacisniecia2 = 0;
                }
        EXTI_ClearITPendingBit(EXTI_Line2);
        }
}

void EXTI3_IRQHandler(void)
{
        if (EXTI_GetITStatus(EXTI_Line3) != RESET)
        {
                nacisniecia3++;
                if(nacisniecia3 == 1)
                {
                        miganie3 = 0;
                        GPIO_WriteBit(GPIOB, GPIO_Pin_3, 1);
                }
                if(nacisniecia3 == 2)
                {
                        miganie3 = 1;
                }
                if(nacisniecia3 == 3)
                {
```

```
                miganie3 = 0;
                GPIO_WriteBit(GPIOB, GPIO_Pin_3, 0);
                nacisniecia3 = 0;
            }
            EXTI_ClearITPendingBit(EXTI_Line3);
        }
}
```

Miganie diod zrealizujemy za pomocą przerwania wewnętrznego. W funkcji wywoływanej podczas przerwania sprawdzamy, która dioda ma włączone miganie i zmieniamy stan tej diody:

```
void SysTick_Handler(void)
{
if(miganie0 == 1)
  GPIO_WriteBit(GPIOB,GPIO_Pin_0,(BitAction)(1-GPIO_ReadOutputDataBit(GPIOB,GPIO_Pin_0)));

if(miganie1 == 1)
  GPIO_WriteBit(GPIOB,GPIO_Pin_1,(BitAction)(1- GPIO_ReadOutputDataBit(GPIOB,GPIO_Pin_1)));

if(miganie2 == 1)
  GPIO_WriteBit(GPIOB,GPIO_Pin_2,(BitAction)(1- GPIO_ReadOutputDataBit(GPIOB,GPIO_Pin_2)));

if(miganie3 == 1)
  GPIO_WriteBit(GPIOB,GPIO_Pin_3,(BitAction)(1- GPIO_ReadOutputDataBit(GPIOB,GPIO_Pin_3)));
}
```

Realizujemy jeszcze zmianę częstotliwości migania przy pomocy przerwań zewnętrznych odpowiadających przyciskom S i N joysticka. Zmienna globalna czestotliwosc określa, z jaką częstotliwością pomniejszoną dwa razy (Hz) ma być wykonywane przerwanie (na początku 4, czyli jakby 2, czyli co 0,5 sekundy). Podajemy dwa razy większą wartość, aby wciąż to była zmienna typu int.

```
int czestotliwosc = 4;
```

Częstotliwość ma być z zakresu od 1 do 4Hz, a skok co 0,5Hz. Przy wciśnięciu przycisku S joysticka zmniejszamy częstotliwość o 0,5Hz (czyli o 1 w naszym systemie razy dwa) i przekazujemy to do funkcji inicjalizującej przerwanie wewnętrzne:

```
void EXTI9_5_IRQHandler(void)
{
        if(czestotliwosc > 2)
        {
                czestotliwosc = czestotliwosc – 1;
                SysTick_Config((2000/czestotliwosc)*9000);
        }
}
```

A przy wciśnięciu przycisku N zwiększamy częstotliwość i również przekazujemy tą wartość do funkcji:

```
void EXTI15_10_IRQHandler(void)
{
	if(czestotliwosc < 8)
	{
		czestotliwosc = czestotliwosc + 1;
		SysTick_Config((2000/czestotliwosc)*9000);
	}
}
```

Kody źródłowe:

Zadanie 1

main.c

```
1.  #include "stm32f10x.h"
2.  #define SysTick_Frequency 9000000 // 9MHz
3.
4.  void GPIO_Config(void);
5.  void RCC_Config(void);
6.
7.
8.  void Delay (uint32_t ms)
9.  {
10.    if (SysTick_Config(SysTick_Frequency / 1000 * ms))
11.    {
12.       while(1);
13.    }
14.    SysTick_CLKSourceConfig(SysTick_CLKSource_HCLK_Div8);
15. }
16.
17.
18. int main(void)
19. {
20.    RCC_Config();
21.    GPIO_Config();
22.
23.    Delay(500);
24.
25.    while (1)
26.    {
27.       /*Tu nalezy umiescic glowny kod programu*/
28.    };
29.
30.    return 0;
31. }
32.
33.
34.
35.
36. void RCC_Config(void)
37. //konfigurowanie sygnalow taktujacych
38. {
39.   ErrorStatus HSEStartUpStatus;                    //zmienna opisujaca rezultat uruchomienia HSE
40.
41.   RCC_DeInit();                                 //Reset ustawien RCC
42.   RCC_HSEConfig(RCC_HSE_ON);                        //Wlaczenie HSE
43.   HSEStartUpStatus = RCC_WaitForHSEStartUp();       //Odczekaj az HSE bedzie gotowy
44.   if(HSEStartUpStatus == SUCCESS)
45.   {
46.     FLASH_PrefetchBufferCmd(FLASH_PrefetchBuffer_Enable);//
47.     FLASH_SetLatency(FLASH_Latency_2);              //ustaw zwloke dla pamieci Flash; zaleznie od taktowania rdzenia
48.                                   //0:<24MHz; 1:24~48MHz; 2:>48MHz
49.     RCC_HCLKConfig(RCC_SYSCLK_Div1);                //ustaw HCLK=SYSCLK
50.     RCC_PCLK2Config(RCC_HCLK_Div1);                 //ustaw PCLK2=HCLK
51.     RCC_PCLK1Config(RCC_HCLK_Div2);                 //ustaw PCLK1=HCLK/2
52.     RCC_PLLConfig(RCC_PLLSource_HSE_Div1, RCC_PLLMul_9); //ustaw PLLCLK = HSE*9 czyli 8MHz * 9 = 72 MHz
53.                                   //inne przykladowe konfiguracje:
54.                                    //RCC_PLLConfig(RCC_PLLSource_HSI_Div2, RCC_PLLMul_9); //ustaw PLLCLK =
    HSI/2*9 czyli 8MHz / 2 * 9 = 36 MHz
55.                                   //RCC_PLLConfig(RCC_PLLSource_HSE_Div2, RCC_PLLMul_9); //ustaw PLLCLK =
    HSE/2*9 czyli 8MHz / 2 * 9 = 36 MHz
56.     RCC_PLLCmd(ENABLE);                         //wlacz PLL
```

```
57.     while(RCC_GetFlagStatus(RCC_FLAG_PLLRDY) == RESET);  //odczekaj na poprawne uruchomienie PLL
58.     RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK);          //ustaw PLL jako zrodlo sygnalu zegarowego
59.     while(RCC_GetSYSCLKSource() != 0x08);               //odczekaj az PLL bedzie sygnalem zegarowym systemu
60.
61.     /*Tu nalezy umiescic kod zwiazany z konfiguracja sygnalow zegarowych potrzebnych w programie peryferiow*/
62.     RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB | RCC_APB2Periph_GPIOA | RCC_APB2Periph_AFIO, ENABL
    E);//wlacz taktowanie portu GPIO B
63.
64.   } else {
65.   }
66. }
67.
68.
69.
70. void GPIO_Config(void)
71. {
72.   //konfigurowanie portow GPIO
73.   GPIO_InitTypeDef  GPIO_InitStructure;
74.
75.   /*Tu nalezy umiescic kod zwiazany z konfiguracja poszczegolnych portow GPIO potrzebnych w programie*/
76.   GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1 ;
77.   GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
78.   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
79.   GPIO_Init(GPIOB, &GPIO_InitStructure);
80.
81.   GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1 ;
82.   GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
83.   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
84.   GPIO_Init(GPIOA, &GPIO_InitStructure);
85. }
```

Funkcja SysTick_Handler

```
1.  void SysTick_Handler(void)
2.  {
3.     GPIO_WriteBit(GPIOB,GPIO_Pin_0,(BitAction)(1-GPIO_ReadOutputDataBit(GPIOB,GPIO_Pin_0)));
4.  }
```

Zadanie 2

main.c

Tak jak w zadaniu 1. Jedynie funkcja Delay wywoływana z parametrem 10.

Dodane funkcja changeDiodeStatus oraz zmieniona funkcja handlera :

```
1.  void changeDiodeStatus(uint32_t nr)
2.  {
3.     switch (nr)
4.     {
5.        case 0:
6.           GPIO_WriteBit(GPIOB,GPIO_Pin_0,(BitAction)(1-GPIO_ReadOutputDataBit(GPIOB,GPIO_Pin_0)));
```

```
7.              break;
8.
9.          case 1:
10.             GPIO_WriteBit(GPIOB,GPIO_Pin_1,(BitAction)(1-GPIO_ReadOutputDataBit(GPIOB,GPIO_Pin_1)));
11.             break;
12.
13.         case 2:
14.             GPIO_WriteBit(GPIOB,GPIO_Pin_2,(BitAction)(1-GPIO_ReadOutputDataBit(GPIOB,GPIO_Pin_2)));
15.             break;
16.
17.         case 3:
18.             GPIO_WriteBit(GPIOB,GPIO_Pin_3,(BitAction)(1-GPIO_ReadOutputDataBit(GPIOB,GPIO_Pin_3)));
19.             break;
20.
21.         case 4:
22.             GPIO_WriteBit(GPIOB,GPIO_Pin_4,(BitAction)(1-GPIO_ReadOutputDataBit(GPIOB,GPIO_Pin_4)));
23.             break;
24.
25.         case 5:
26.             GPIO_WriteBit(GPIOB,GPIO_Pin_5,(BitAction)(1-GPIO_ReadOutputDataBit(GPIOB,GPIO_Pin_5)));
27.             break;
28.
29.         case 6:
30.             GPIO_WriteBit(GPIOB,GPIO_Pin_6,(BitAction)(1-GPIO_ReadOutputDataBit(GPIOB,GPIO_Pin_6)));
31.             break;
32.
33.         case 7:
34.             GPIO_WriteBit(GPIOB,GPIO_Pin_7,(BitAction)(1-GPIO_ReadOutputDataBit(GPIOB,GPIO_Pin_7)));
35.             break;
36.     }
37. }
38.
39.
40. void SysTick_Handler(void)
41. {
42.         if(counter%560 == 0)        {   changeDiodeStatus(7); }
43.         if(counter%670 == 0)        {   changeDiodeStatus(6);   }
44.         if(counter%800 == 0)        {   changeDiodeStatus(5);   }
45.         if(counter%960 == 0)        {   changeDiodeStatus(4);   }
46.         if(counter%1160 == 0)       {   changeDiodeStatus(3);   }
47.         if(counter%1390 == 0)       {   changeDiodeStatus(2);   }
48.         if(counter%1670 == 0)       {   changeDiodeStatus(1);   }
49.         if(counter%2000 == 0)       {   changeDiodeStatus(0);   }
50.
51.         if(counter == 2085646912398000)     {   counter = 0; }
52.
53.         counter+=10;
54. }
```