

# Wybrane listingi z opisu ćwiczenia 3

Strona 60

```
RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO , ENABLE); // włącz taktowanie AFIO
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE); // włącz taktowanie GPIO
A
RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1, ENABLE); // włącz taktowanie USAR
T1
```

```
GPIO_InitTypeDef GPIO_InitStructure;

// Pin nadawczy należy skonfigurować jako "alternative function, push-pull"
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOA, &GPIO_InitStructure);

// Pin odbiorczy należy skonfigurować jako wejście "pływające"
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOA, &GPIO_InitStructure);
```

```
USART_InitTypeDef USART_InitStructure;
USART_InitStructure.USART_BaudRate = 19200;
USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
USART_InitStructure.USART_WordLength = USART_WordLength_9b;
USART_InitStructure.USART_Parity = USART_Parity_Even;
USART_InitStructure.USART_StopBits = USART_StopBits_1;
USART_InitStructure.USART_Mode = USART_Mode_Tx | USART_Mode_Rx;

USART_Init(USART1, &USART_InitStructure);
USART_ITConfig(USART1, USART_IT_RXNE, DISABLE);
USART_ITConfig(USART1, USART_IT_TXE, DISABLE);
```

---

Strona 61

```
NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQn;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
```

```
NVIC_Init(&NVIC_InitStructure);
```

```
USART_Cmd(USART1, ENABLE);
```

---

Strona 63

```
MB_SendRequest(103, FUN_WRITE_SINGLE_COIL, write_single_coil_3, 4);
```

```
uint8_t write_single_coil_3[] = {0x00, 0x03, 0xFF, 0x00};
```

```
uint8_t *resp;  
uint16_t resplen;  
MB_RESPONSE_STATE respstate;  
respstate = MB_GetResponse(103, FUN_WRITE_SINGLE_COIL, &resp, &resplen, 1000);
```

```
MB_SendRequest(103, FUN_READ_DISCRETE_INPUTS, read_discrete_input_4, 4);
```

```
uint8_t read_discrete_input_4[] = {0x00, 0x04, 0x00, 0x01};
```

```
uint8_t *resp;  
uint16_t resplen;  
MB_RESPONSE_STATE respstate;  
respstate = MB_GetResponse(103, FUN_READ_DISCRETE_INPUTS, &resp, &resplen, 1000);
```

---

Strony 65-66

```
void USART1_IRQHandler(void) {  
    if( USART_GetITStatus(USART1, USART_IT_RXNE) ){  
        USART_ClearITPendingBit(USART1, USART_IT_RXNE);  
        SetCharacterReceived(true);  
    }  
    if( USART_GetITStatus(USART1, USART_IT_TXE) ){  
        USART_ClearITPendingBit(USART1, USART_IT_TXE);  
        SetCharacterReadyToTransmit();  
    }  
}
```

```
void Communication_Put(uint8_t ch){
```

```
    USART_SendData(USART1, ch);
}

uint8_t Communication_Get(void){
    uint8_t tmp = USART_ReceiveData(USART1);
    SetCharacterReceived(false);
    return tmp;
}

void Communication_Mode(bool rx, bool tx){
    USART_ITConfig(USART1, USART_IT_RXNE, rx?ENABLE:DISABLE);
    USART_ITConfig(USART1, USART_IT_TXE , tx?ENABLE:DISABLE);
}

void Enable50usTimer(void){
    TIM_ITConfig(TIM4, TIM_IT_Update, ENABLE);
}

void Disable50usTimer(void){
    TIM_ITConfig(TIM4, TIM_IT_Update, DISABLE);
}
```