# Ekološko snovanje elektronskih naprav

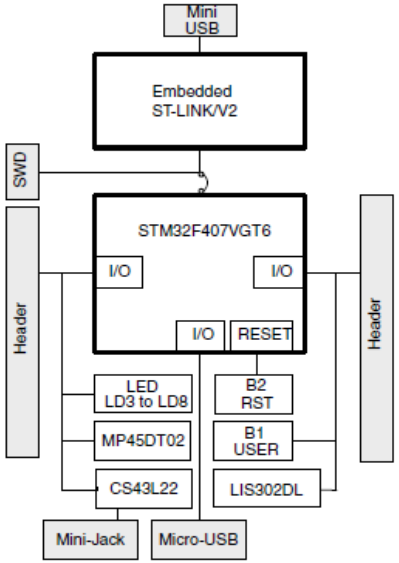
## ENOTA 9: Mikrokrmilniški sistemi 2.del

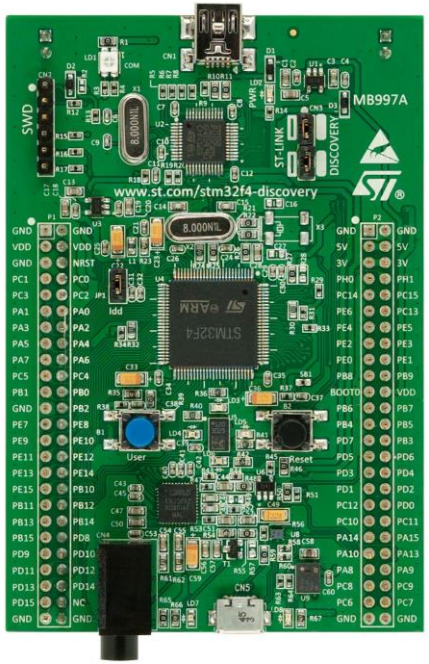
Funded by the Erasmus+ Programme of the European Union

Ecoinnovation Skills for European Designers, Project number: 562573-EPP-1-2015-1-SI-EPPKA2-SSA.

# Uporaba mikrokrmilniške plošče SMT32F4discovery

- Razvojna plošča **STM32F4Discovery** je nizko cenovna in zmogljiva mikrokrmilniša platforma, primerna za hiter razvoj vgrajenih aplikacij in testiranje zmogljivega mikrokontrolerja ARM STM32F407VG z vgrajeno FPU enoto (FPU – floating point unit, enota za računanje s števili s plavajočo vejico). Razvojna plošča vsebuje mikrokrmilni ARM STM32F407VG, JTAG programator ST-link V2, pospeškometer, audio kodek, mikrofon, 4 kontrolne led diode, stikalo in mikro-USB priključek.






Funded by the Erasmus+ Programme of the European Union

# Uporaba mikrokrmilniške plošče SMT32F4discovery

- Jedro mikrokrmilnik STM32F407 bazira na jedru Cortex-M4 in lahko teče na maksimalni frekvenci 168MHz. Jedro Cortex-M4 ponuja 32bit procesorsko enoto ter strojni FPU. FPU je namenjen za digitalno procesiranje signal in ima funkcionalnost DSP procesorja. Pri maksimalni frekvenci doseže 210 DMIPS-ov (DMIPS- 'Dyrestone- Milion Instructions per Second'). Prav tako krmilnik vsebuje velik nabor perifernih naprav.
  - 2x USB OTG (On the Go).
  - Audio fazno zaklenjeno zanko (PLL- 'Phase lock loop').
  - Vsebuje 15 komunikacijskih vmesnikov (4 x UART,  2 x USART, 3 x SPI, 3 x I2C, 2x CAN, SDIO).
  - Vsebuje 2x 12bit DAC in 3 x 12bit ADC hitrosti 2.4Msps.
  - Ima 17 časovnikov. Vsi časovniki so 16biti razen dveh, ki sta 32bitna.
  - Vmesnik za dodatne pomnilniške sisteme, SRAM, NAND itd..




Funded by the
Erasmus+ Programme
of the European Union

# Nastavitev diskretnih vhodov in izhodov-GPIO

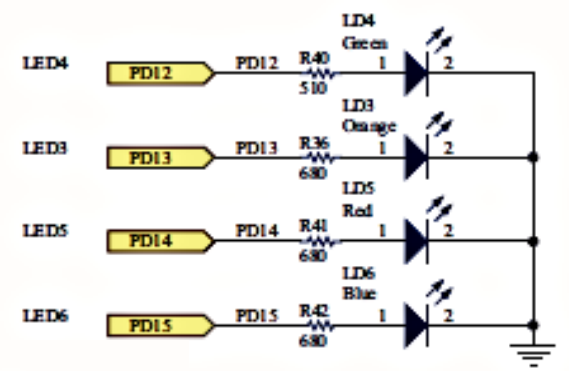
- Nastavitev digitalnega izhoda: Pini mikrokontrolerja so razporejeni po skupinah, ki jih imenujemo porti in so označeni s PA,PB,PC,PD,PE. Kjer vsaka skupina zajema porte od 0-15 (npr. PA0-PA15 itd.). Nastavitev GPIO modula in vseh pripadajočih funkcionalnosti je potrebno v projekt vključiti knjižnico ***stm32fxxx_gpio.c***.

Nastavitev GPIO modula kot izhod:

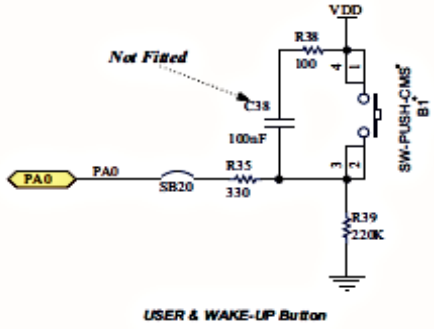
```
GPIO_InitTypeDef GPIO_InitStructure; //Struktura

//GPIO out
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_12| GPIO_Pin_13|
                              GPIO_Pin_14 | GPIO_Pin_15;//Konfiguracija pinov 12-15
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;           // Definicija izhoda
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;       // Hitrost modula (2/10/50 MHz)
GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;    //  push / pull (opposed to open d
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL; // pullup / pulldown upor ni aktiv
GPIO_Init(GPIOD, &GPIO_InitStructure);                  // Nastavitev porta D
```

Nastavitev GPIO modula kot vhod:

```
//GPIO IN
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;         //  konfiguracija PINA 0
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN;      // GPIO kot vhod
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz; // Hitrost GPIO modula(2/10/50 MHz)
GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;    // push/pull (opposed to open drain)
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL; // pullup / pullup upor ni aktiven
GPIO_Init(GPIOA, &GPIO_InitStructure);           // Nastavitev porta A
```




LEDs




USER & WAKE-UP Button

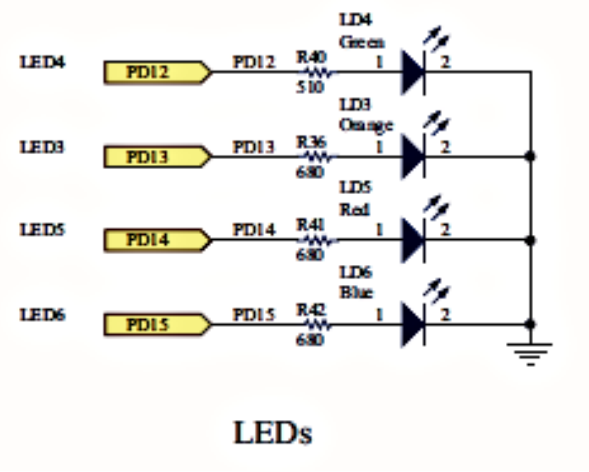


Funded by the Erasmus+ Programme of the European Union

# Nastavitev diskretnih vhodov in izhodov-GPIO

Primer programa:

```
//MACRO
#define LED1           GPIO_Pin_12
#define LEDon(LED)    GPIO_SetBits(GPIOD, LED)
#define LEDoff(LED)   GPIO_ResetBits(GPIOD, LED)
#define Button        GPIO_ReadInputDataBit(GPIOA,GPIO_Pin_0)

void RCC_Configuration(void);
void GPIO_Configuration(void);
int  key=0;


int main(void)
{

    //Nastavitev MCU-ja in GPIO
  RCC_Configuration();
  GPIO_Configuration();

    //Programm
  GPIO_SetBits(GPIOD, GPIO_Pin_12);
  GPIO_SetBits(GPIOD, GPIO_Pin_13);
  GPIO_ResetBits(GPIOD, GPIO_Pin_14);
  GPIO_ResetBits(GPIOD, GPIO_Pin_15);

    //Infinite loop
    while(1)
    {

            key = GPIO_ReadInputDataBit(GPIOA,GPIO_Pin_0); //Read input bit(Button value)

                    if(key == 1)
                    {
                            GPIO_SetBits(GPIOD, GPIO_Pin_14);
                            GPIO_SetBits(GPIOD, GPIO_Pin_15);
                            LEDon(LED1); //MACRO function
                     }else
                     {
                            GPIO_ResetBits(GPIOD, GPIO_Pin_14);
                            GPIO_ResetBits(GPIOD, GPIO_Pin_15);
                            LEDoff(LED1); //MACRO function
                     }

                }//Infinite loop

}//Konec MAIN
```



LEDs



Funded by the
Erasmus+ Programme
of the European Union

# UART komunikacija

- Razvojna plošča vsebuje 4 internih USART module, ki se nahajajo na sledečih portih:
  - UART 1 - Rx: PA10 Tx: PA9
  - **UART 2 - Rx: PA3 Tx: PA2**
  - UART 3 - Rx: PB11 Tx: PB10
  - UART 4 - Rx: PA1 Tx: PA0

Nastavitev GPIO pina za UART2:

```
   // UART2 GPIO
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2| GPIO_Pin_3;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOA, &GPIO_InitStructure);

/* Connect USART pins to AF */
GPIO_PinAFConfig(GPIOA, GPIO_PinSource2, GPIO_AF_USART2); // USART2_TX
GPIO_PinAFConfig(GPIOA, GPIO_PinSource3, GPIO_AF_USART2); // USART2_RX


}
```

Nastavitev GPIO pina za UART:

```
void USART2_Configuration(void)
{
    USART_InitTypeDef USART_InitStructure;


  //USART2 configuration
  USART_InitStructure.USART_BaudRate = 115200;
  USART_InitStructure.USART_WordLength = USART_WordLength_8b;
  USART_InitStructure.USART_StopBits = USART_StopBits_1;
  USART_InitStructure.USART_Parity = USART_Parity_No;
  USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
  USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
  USART_Init(USART2, &USART_InitStructure);
  USART_Cmd(USART2, ENABLE);
  USART_ITConfig(USART2, USART_IT_RXNE, ENABLE); // Receive Interrupt enable

}
```

Uporaba funkcije za pošiljanje podatkov:

```
USART_Send_Str(USART2, "Hello from StmF4Discovery!\n\r" );
```



Funded by the Erasmus+ Programme of the European Union

# AD pretvorba

- Razvojna plošča omogoča 16 zunanjih analognih vhodov  ter dva notranja (senzor temperature in napetost baterije). Mikrokrmilnik vsebuje tri ločene AD pretvornike, kjer  vsak AD1-3 ima ločeno nastavljivo resolucijo od 6, 8, 10 do 12bit-ov (Privzeta vrednost AD je 12bitov). Tri AD pretvornike je možno programsko konfigurirati na različne GPIO enote in pine s dodano knjižnico **stm32fxxx_adc.c**
- Imena portov in pinov, kjer se nahajajo AD-pretvorniki:

| **Channel** | **ADC1** | **ADC2** | **ADC3** | |
|---|---|---|---|---|
| APB | 2 | 2 | 2 | |
| ADC Channel 0 | PA0 | PA0 | PA0 | |
| ADC Channel 1 | PA1 | PA1 | PA1 | |
| ADC Channel 2 | PA2 | PA2 | PA2 | |
| ADC Channel 3 | PA3 | PA3 | PA3 | |
| ADC Channel 4 | PA4 | PA4 | PF6 | |
| ADC Channel 5 | PA5 | PA5 | PF7 | |
| ADC Channel 6 | PA6 | PA6 | PF8 | |
| ADC Channel 7 | PA7 | PA7 | PF9 | |
| ADC Channel 8 | PB0 | PB0 | PF10 | |
| ADC Channel 9 | PB1 | PB1 | PF3 | |
| ADC Channel 10 | | PC0 | PC0 | PC0 |
| ADC Channel 11 | | PC1 | PC1 | PC1 |
| ADC Channel 12 | | PC2 | PC2 | PC2 |
| ADC Channel 13 | | PC3 | PC3 | PC3 |
| ADC Channel 14 | | PC4 | PC4 | PF4 |
| ADC Channel 15 | | PC5 | PC5 | PF5 |



Funded by the Erasmus+ Programme of the European Union

# AD pretvorba

## Primer nastavitve ADC-pretvornika:

```
/ ADC configuration
void ADC_init()
{
   GPIO_InitTypeDef        GPIO_InitStructure;
   ADC_InitTypeDef         ADC_InitStructure;
   ADC_CommonInitTypeDef   ADC_CommonInitStructure;

   RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1  | RCC_APB2Periph_ADC2 ,  ENABLE);

    //Setup GIPO pins
  GPIO_InitStructure.GPIO_Pin   = GPIO_Pin_1 | GPIO_Pin_4 | GPIO_Pin_5;
  GPIO_InitStructure.GPIO_Mode  = GPIO_Mode_AN;
  GPIO_InitStructure.GPIO_PuPd  = GPIO_PuPd_NOPULL;
  GPIO_Init(GPIOA, &GPIO_InitStructure);


    //Common settings of ADC
  ADC_CommonInitStructure.ADC_DMAAccessMode    = ADC_DMAAccessMode_Disabled;
  ADC_CommonInitStructure.ADC_Mode             = ADC_Mode_Independent ;
  ADC_CommonInitStructure.ADC_Prescaler        = ADC_Prescaler_Div4;
    ADC_CommonInitStructure.ADC_TwoSamplingDelay = ADC_TwoSamplingDelay_10Cycles;
  ADC_CommonInit(&ADC_CommonInitStructure);


    //Common settings of ADC
  ADC_InitStructure.ADC_Resolution             = ADC_Resolution_12b;
  ADC_InitStructure.ADC_ScanConvMode           = DISABLE;
  ADC_InitStructure.ADC_ContinuousConvMode     = DISABLE;
  ADC_InitStructure.ADC_ExternalTrigConvEdge   = ADC_ExternalTrigConvEdge_None;
  ADC_InitStructure.ADC_DataAlign              = ADC_DataAlign_Right;
  ADC_InitStructure.ADC_NbrOfConversion        = 1;

    //Connect GPIO to the ADC
  ADC_Init(ADC1, &ADC_InitStructure);
  ADC_Init(ADC2, &ADC_InitStructure);



   //Enable ADC1 and ADC2
  ADC_Cmd(ADC1, ENABLE);
  ADC_Cmd(ADC2, ENABLE);

}
```

## Funkcija za branje ADC-pretvornika:

```
//READ ADC function
uint16_t Read_ADC(ADC_TypeDef* ADCx, uint8_t channel)
{
        uint16_t timeout = 0xFFF;

            ADC_RegularChannelConfig(ADCx, channel, 1, ADC_SampleTime_28Cycles);

            /* Start software conversion */
            ADCx->CR2 |= (uint32_t)ADC_CR2_SWSTART;

            /* Wait till done */
            while (!(ADCx->SR & ADC_SR_EOC)) {
                if (timeout-- == 0x00) {
                    return 0;
                }
            }

            /* Return result */
            return (uint16_t)ADCx->DR;
}
```



Funded by the
Erasmus+ Programme
of the European Union

# Pulzno širinska modulacija - PWM

- Pulzna širinska modulacija PWM ('Pulse Width Modulation') je način modulacije s katero spreminjamo prevajalno razmerej ('Duty Cycle') pri konstantni frekvenci signala. Pulzno širinska modulacija je vezana na časovnik (Timer) mikrokrmilnika.
- Vsak časovnik je lahko fizično vezan na GPIO, kjer pri nastavitvi PWM-a z izbiro časovnika ('Timer') in kanala ('Channel'), določimo izhodni pina na portu A,B,C,D,E.

| PWM Module Pinouts for ST | | | | |
|---|---|---|---|---|
| | **Channel 1** | **Channel 2** | **Channel 3** | **Channel 4** |
| **Timer1** | _GPIO_MODULE_TIM1_CH1_PA8<br>_GPIO_MODULE_TIM1_CH1_PE9 | _GPIO_MODULE_TIM1_CH2_PA9<br>_GPIO_MODULE_TIM1_CH2_PE11 | _GPIO_MODULE_TIM1_CH3_PA10<br>_GPIO_MODULE_TIM1_CH3_PE13 | _GPIO_MODULE_TIM1_CH4_PA11<br>_GPIO_MODULE_TIM1_CH4_PE14 |
| **Timer2** | _GPIO_MODULE_TIM2_CH1_PA0<br>_GPIO_MODULE_TIM2_CH1_PA5 | _GPIO_MODULE_TIM2_CH2_PA1<br>_GPIO_MODULE_TIM2_CH2_PB3 | _GPIO_MODULE_TIM2_CH3_PA2<br>_GPIO_MODULE_TIM2_CH3_PB10 | _GPIO_MODULE_TIM2_CH4_PA3<br>_GPIO_MODULE_TIM2_CH4_PB11 |
| **Timer3** | _GPIO_MODULE_TIM3_CH1_PA6<br>_GPIO_MODULE_TIM3_CH1_PB4<br>_GPIO_MODULE_TIM3_CH1_PC6 | _GPIO_MODULE_TIM3_CH2_PA7<br>_GPIO_MODULE_TIM3_CH2_PB5<br>_GPIO_MODULE_TIM3_CH2_PC7 | _GPIO_MODULE_TIM3_CH3_PB0<br>_GPIO_MODULE_TIM3_CH3_PC8 | _GPIO_MODULE_TIM3_CH4_PB1<br>_GPIO_MODULE_TIM3_CH4_PC9 |
| **Timer4** | _GPIO_MODULE_TIM4_CH1_PB6<br>_GPIO_MODULE_TIM4_CH1_PD12 | _GPIO_MODULE_TIM4_CH2_PB7<br>_GPIO_MODULE_TIM4_CH2_PD13 | _GPIO_MODULE_TIM4_CH3_PB8<br>_GPIO_MODULE_TIM4_CH3_PD14 | _GPIO_MODULE_TIM4_CH4_PB9<br>_GPIO_MODULE_TIM4_CH4_PD15 |
| **Timer5** | _GPIO_MODULE_TIM5_CH1_PH10 | _GPIO_MODULE_TIM5_CH2_PA1<br>_GPIO_MODULE_TIM5_CH2_PH11 | _GPIO_MODULE_TIM5_CH3_PA2<br>_GPIO_MODULE_TIM5_CH3_PH12 | _GPIO_MODULE_TIM5_CH4_PA3<br>_GPIO_MODULE_TIM5_CH4_PI0 |
| **Timer8** | _GPIO_MODULE_TIM8_CH1_PC6<br>_GPIO_MODULE_TIM8_CH1_PI5 | _GPIO_MODULE_TIM8_CH2_PC7<br>_GPIO_MODULE_TIM8_CH2_PI6 | _GPIO_MODULE_TIM8_CH3_PC8<br>_GPIO_MODULE_TIM8_CH3_PI7 | _GPIO_MODULE_TIM8_CH4_PC9<br>_GPIO_MODULE_TIM8_CH4_PI2 |
| **Timer9** | _GPIO_MODULE_TIM9_CH1_PA2<br>_GPIO_MODULE_TIM9_CH1_PE5 | _GPIO_MODULE_TIM9_CH2_PA3<br>_GPIO_MODULE_TIM9_CH2_PE6 | | |
| **Timer10** | _GPIO_MODULE_TIM10_CH1_PB8<br>_GPIO_MODULE_TIM10_CH1_PF6 | | | |
| **Timer11** | _GPIO_MODULE_TIM11_CH1_PB9<br>_GPIO_MODULE_TIM11_CH1_PF7 | | | |
| **Timer12** | _GPIO_MODULE_TIM12_CH1_PB14<br>_GPIO_MODULE_TIM12_CH1_PH6 | _GPIO_MODULE_TIM12_CH2_PB15<br>_GPIO_MODULE_TIM12_CH2_PH9 | | |
| **Timer13** | _GPIO_MODULE_TIM13_CH1_PA6<br>_GPIO_MODULE_TIM13_CH1_PF8 | | | |
| **Timer14** | _GPIO_MODULE_TIM14_CH1_PA7<br>_GPIO_MODULE_TIM14_CH1_PF9 | | | |



Funded by the
Erasmus+ Programme
of the European Union

# Pulzno širinska modulacija - PWM

- Primer nastavitve PWM signala na portu PD14 in PD15 in TIMER 4.

```
//PWM konfiguracija
void PWM_Configuration(void)
{

    GPIO_InitTypeDef GPIO_InitStructure;
    TIM_TimeBaseInitTypeDef  TIM_TimeBaseStructure;
    TIM_OCInitTypeDef  TIM_OCInitStructure;


    //PWM on pin PD14 PD15
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_14 | GPIO_Pin_15; // konfiguiramo I/O
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;       // GPIO kot izhod (Alternate
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;  // Hitrost  GPIO modula (2/10/50)
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;      // push / pull
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;     // pullup / pullup
    GPIO_Init(GPIOD, &GPIO_InitStructure);

      /* Connect pins PD14 and PD15 on timer TIM4 */
   GPIO_PinAFConfig(GPIOD, GPIO_PinSource14, GPIO_AF_TIM4); // PD14 on Channel 3
   GPIO_PinAFConfig(GPIOD, GPIO_PinSource15, GPIO_AF_TIM4); // PD14 on Channel 4
```

```
   /* TIM4 clock enable */
RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM4, ENABLE);

  /* Time base configuration 10kHz */
TIM_TimeBaseStructure.TIM_Period = 8399; // ARR+1=(TIM3 clock/PWM_frequncy)=(84MHz/10kHz)=8400
TIM_TimeBaseStructure.TIM_Prescaler = 0;
TIM_TimeBaseStructure.TIM_ClockDivision = 0;
TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;

TIM_TimeBaseInit(TIM4, &TIM_TimeBaseStructure);


/* PWM1 Mode configuration: Channel3 */
TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
TIM_OCInitStructure.TIM_Pulse = 4200; //Duty 50%
TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;
TIM_OC3Init(TIM4, &TIM_OCInitStructure);
TIM_OC3PreloadConfig(TIM4, TIM_OCPreload_Enable);


  /* PWM1 Mode configuration: Channel4 */
  TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
TIM_OCInitStructure.TIM_Pulse = 0; //Duty 0%
TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;
TIM_OC4Init(TIM4, &TIM_OCInitStructure);
TIM_OC4PreloadConfig(TIM4, TIM_OCPreload_Enable);


  TIM_ARRPreloadConfig(TIM4, ENABLE);

 /* TIM4 enable counter */
 TIM_Cmd(TIM4, ENABLE);
 }
```



Funded by the Erasmus+ Programme of the European Union

# Časovne prekinitvene rutine

- **Nastavitev prekinitve na časovniku 3 (TIM3)**

Periode časovnika je enaka:

$$Period = (TimerX\_frequency / TimerX\_prescaler + 1)^{-1} \cdot TimerX\_perioda$$

V našem primeru uporabljamo skalirni faktor 2, kar pomeni, da je osnovna frekvenca časovnika enaka polovici glavne ure MCU-ja. Osnovna frekvenca ure časovnikov znaša 84MHz.

- **PRIMER**

Nastavimo periodo časovnika na 10ms. Izberemo vrednost TimerX_prescaler=209, TimerX_preioda=4000;

$$Peridoa = (TimerX\_frequency / TimerX\_prescaler + 1)^{-1} \cdot TimerX\_perioda$$
$$= (84MHz / 209 + 1)^{-1} \cdot 4000 = 0.01s$$



Funded by the Erasmus+ Programme of the European Union

# Časovne prekinitvene rutine

- Primer nastavitve prekinitev časovnika TIMER 3 (@10ms):

```
/**************************************************************************
Timer3 configuration
**************************************************************************/
void TIM_Configuration(void)
{

    TIM_TimeBaseInitTypeDef  TIM_TimeBaseStructure;



     /* TIM3 clock enable */
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);

    /* Time base configuration */
    TIM_TimeBaseStructure.TIM_Period = 4000;     //Timer3 period =   ((TIM3_clock_freq)
((84MHz)/(209+1))^-1 * 4000=10ms
    TIM_TimeBaseStructure.TIM_Prescaler =209;
    TIM_TimeBaseStructure.TIM_ClockDivision = 0;
    TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;   //Counter mode up
    TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);


    TIM_ClearFlag(TIM3, TIM_FLAG_Update);   //Clear flag
    TIM_ITConfig(TIM3, TIM_IT_Update, ENABLE); //Interrupt on auto reload register
    TIM_Cmd(TIM3, ENABLE); //Enable timer
}
```

## Nastavitev prioritete:

```
/**************************************************************************
 NVIC-vector
**************************************************************************/
void NVIC_Configuration(void)
{
        NVIC_InitTypeDef NVIC_InitStruct;

        NVIC_PriorityGroupConfig(NVIC_PriorityGroup_0);
      //TIM3 Priority
         NVIC_InitStruct.NVIC_IRQChannel = TIM3_IRQn;
         NVIC_InitStruct.NVIC_IRQChannelPreemptionPriority = 0;
         NVIC_InitStruct.NVIC_IRQChannelSubPriority = 0;
         NVIC_InitStruct.NVIC_IRQChannelCmd = ENABLE;
         NVIC_Init(&NVIC_InitStruct);

 }
```

## Funkcija prekinitve:

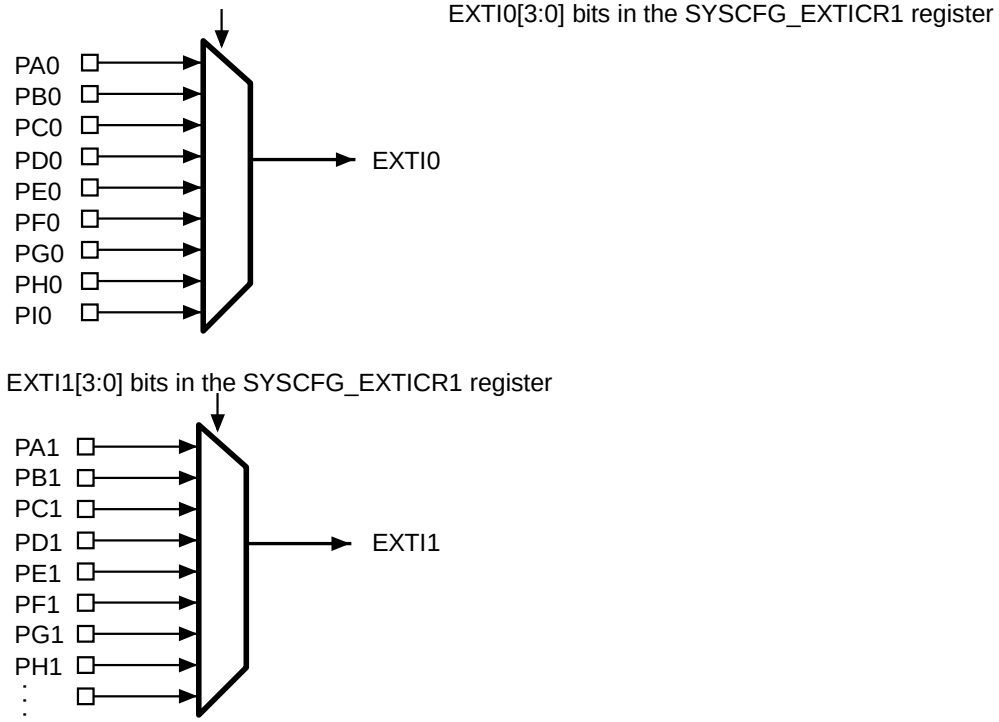
```
/**************************************************************
 Interrupt function
**************************************************************
void TIM3_IRQHandler(void)
{

    if (TIM_GetITStatus(TIM3, TIM_IT_Update) != RESET)
    {
           TIM_ClearITPendingBit(TIM3, TIM_IT_Update);


             GPIO_ToggleBits(GPIOD, GPIO_Pin_12);
           GPIO_ToggleBits(GPIOD, GPIO_Pin_13);


    }
}
```



Funded by the Erasmus+ Programme of the European Union

# Zunanja prekinitev

- Zunanje prekinitve, ki jih prožijo zunanje naprave, stikala, komunikacijski protokoli itd., je potrebno omogočiti z naborom vpisov v določene sistemske registre.
- Zunanje prekinitve so porazdeljene po enotah EXTI0 – EXTI15 in se programsko določijo v kontrolnem registru SYSCFG_EXTICRx.
- Vsaka EXTI0-15 pripada zaporedni številki porta 0-15.
- Zunanje prekinitve se lahko uporabljajo za merjenje frekvence, štetje pulzov, za merjenje časa med dogodki….






Funded by the Erasmus+ Programme of the European Union

# Zunanja prekinitev

- Primer zunanje prekinitve na pinu PA0:

```
/*******************************************************************************
 EXTI0 configuration
*******************************************************************************/
void EXTILine0_Configuartion(void)
{
       EXTI_InitTypeDef   EXTI_InitStructure;
       GPIO_InitTypeDef   GPIO_InitStructure;

        /* Enable SYSCFG clock */
     RCC_APB2PeriphClockCmd(RCC_APB2Periph_SYSCFG, ENABLE);

      //BUTTON
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;          //  konfiguracija PIN_A 0
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN;          //  GPIO kot vhod
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;    //  Hitrost  GPIO modula (2/10/50 MHz)
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;    //  push / pull (opposed to open drain)
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;    // pullup / pullup upor ni aktiven
    GPIO_Init(GPIOA, &GPIO_InitStructure);               // Nastavitev porta A

     /* Connect EXTI Line0 to PA0 pin */
    SYSCFG_EXTILineConfig(EXTI_PortSourceGPIOA, EXTI_PinSource0);

        /* Configure EXTI Line0 */
    EXTI_InitStructure.EXTI_Line = EXTI_Line0;
    EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
    EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Rising;
    EXTI_InitStructure.EXTI_LineCmd = ENABLE;
    EXTI_Init(&EXTI_InitStructure);

     /* Generate software interrupt: simulate a rising edge applied on EXTI0 line */
     EXTI_GenerateSWInterrupt(EXTI_Line0);

}
```

## Nastavitev prioritete:

```
/*******************************************************************************
 NVIC configuration for EXTI0
*******************************************************************************/
void NVIC_Configuration(void)
{

    NVIC_InitTypeDef NVIC_InitStruct;

    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_0);        //

   /* Enable and set EXTI Line0 Interrupt  */
  NVIC_InitStruct.NVIC_IRQChannel = EXTI0_IRQn;
  NVIC_InitStruct.NVIC_IRQChannelPreemptionPriority = 0;
  NVIC_InitStruct.NVIC_IRQChannelSubPriority = 0;
  NVIC_InitStruct.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Init(&NVIC_InitStruct);

}
```

## Funkcija prekinitve:

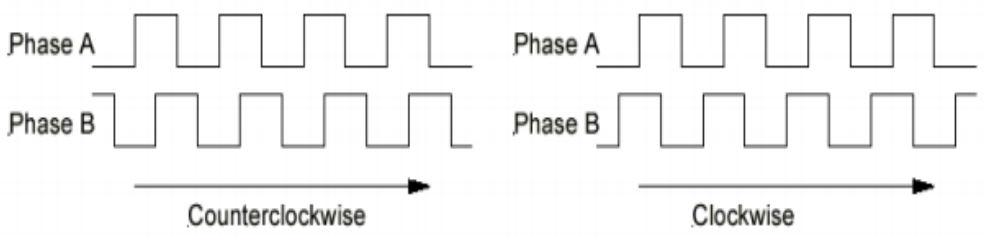
```
/*******************************************************************************
EXTI0 Interrupt function
*******************************************************************************/
void EXTI0_IRQHandler(void)
{

     if (EXTI_GetITStatus(EXTI_Line0) != RESET)
    {


            GPIO_ToggleBits(GPIOD, GPIO_Pin_12);
            GPIO_ToggleBits(GPIOD, GPIO_Pin_13);

          EXTI_ClearITPendingBit(EXTI_Line0);
    }
```

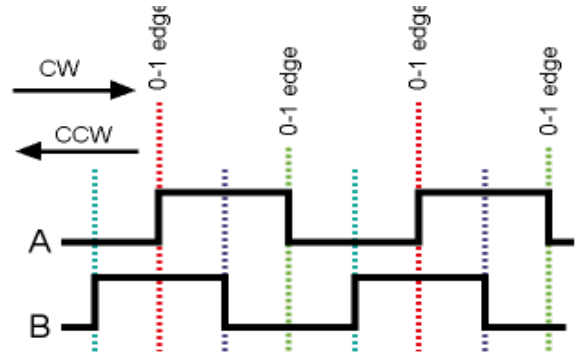


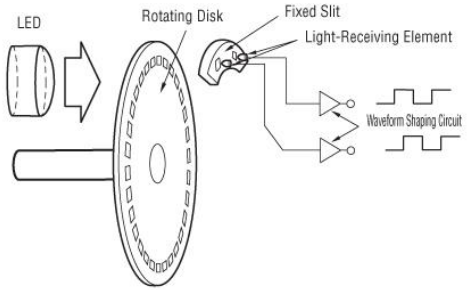
Funded by the Erasmus+ Programme of the European Union

# Inkrementalni dajalnik

- Inkrementalni dajalnik je naprava-senzor, ki zaznava premike ali zasuke sistema. Na ta način ločimo linearne in rotacijske inkrementalne dajalnike.
- Zelo pogosto rotacijske inkrementalne dajalnike srečamo, kot merilnike kota zasuka, kotne hitrosti motorja ali, kot kontrolne gumbe na elektronskih napravah.
- Princip merjenja z inkrementalnim dajalnikom sloni na principu štetja inkrementov, ki jih povzroči pomik merjenca.






Funded by the Erasmus+ Programme of the European Union

# Inkrementalni dajalnik

- Primer kode prikazuje nastavitev dajalnika na pinu PC6(A) in PC7(B) s časovnikom TIM3, ter prekinitvijo s 3200 preštetimi inkrementi.

```
/*******************************************************************************
ENCODER Configuration
*******************************************************************************/
void ENCODER_Configuration()
{

        TIM_TimeBaseInitTypeDef TIM3_TimeBaseStructure;
        GPIO_InitTypeDef GPIO_InitStructure;

        /*ENCODER PIN Configuration ( PC6 & PC7 )*/
        RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOC , ENABLE);
        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6 | GPIO_Pin_7; //PIN PC6-T1(A) PC7-T2(B)
        GPIO_InitStructure.GPIO_Mode =  GPIO_Mode_AF;
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
        GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_DOWN;  //Trigger only on + voltage
        GPIO_InitStructure.GPIO_OType= GPIO_AF_TIM3;
        GPIO_Init(GPIOC, &GPIO_InitStructure);

        /*Connect PIN PC6 and PC7 on TIM 3 (Channel 1&2)*/
  GPIO_PinAFConfig(GPIOC, GPIO_PinSource6, GPIO_AF_TIM3); //Enable GPIO PC6 To alternate
  GPIO_PinAFConfig(GPIOC, GPIO_PinSource7, GPIO_AF_TIM3); //Enable GPIO PC7 To alternate


        /*TIM3 setting*/
        TIM_DeInit(TIM3);
        RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);
        TIM3_TimeBaseStructure.TIM_Period = 3200;   //Number of encoder counts
        TIM3_TimeBaseStructure.TIM_Prescaler = 0;
        TIM3_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;
        TIM3_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
        //ENCODER MODE
      TIM_EncoderInterfaceConfig(TIM3,
                                 TIM_EncoderMode_TI12, //Count on both channel A in B
                                  TIM_ICPolarity_Rising,
                                  TIM_ICPolarity_Rising);
        TIM_TimeBaseInit(TIM3, &TIM3_TimeBaseStructure);


          TIM3->CNT=0; //Inital value of the encoder timer

        //Enable upadate flag
         TIM_ClearFlag(TIM3, TIM_FLAG_Update);
        //Timer interrupt enable, for one revolution
        TIM_ITConfig(TIM3, TIM_IT_Update, ENABLE);
        //Start timer TIM3
        TIM_Cmd(TIM3, ENABLE);
}
```



Funded by the Erasmus+ Programme of the European Union

# Inkrementalni dajalnik

## Nastavitev prioritete prekinitve:

```
/*******************************************************************************
*  Interrupt  priority configuration
*******************************************************************************/
void NVIC_Configuration(void)
{
        NVIC_InitTypeDef NVIC_InitStruct;


        //Timer 3 Priority, with encoder
        NVIC_InitStruct.NVIC_IRQChannel = TIM3_IRQn;
        NVIC_InitStruct.NVIC_IRQChannelPreemptionPriority = 0;
        NVIC_InitStruct.NVIC_IRQChannelSubPriority = 1;
        NVIC_InitStruct.NVIC_IRQChannelCmd = ENABLE;
        NVIC_Init(&NVIC_InitStruct);


}
```

## Funkcija prekinitve vsake 3200:

```
/*******************************************************************************
Encoder interrupt function, occur after 3200 increments
*******************************************************************************/
    void TIM3_IRQHandler(void)
    {

                    if (TIM_GetITStatus(TIM3, TIM_IT_Update) != RESET)
                    {
                            TIM_ClearITPendingBit(TIM3, TIM_IT_Update);



                    }
    }
```

## Branje enkoderja preko registra TIMER3:

```
encoder = TIM3->CNT; //Read encoder value
```



Funded by the Erasmus+ Programme of the European Union

# Inkrementalni dajalnik

- Primer kode prikazuje nastavitev dajalnika na pinu PC6(A) in PC7(B) s časovnikom TIM3, ter prekinitvijo s 3200 preštetimi inkrementi.

```
/*******************************************************************************
ENCODER Configuration
*******************************************************************************/
void ENCODER_Configuration()
{

        TIM_TimeBaseInitTypeDef TIM3_TimeBaseStructure;
        GPIO_InitTypeDef GPIO_InitStructure;

        /*ENCODER PIN Configuration ( PC6 & PC7 )*/
        RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOC , ENABLE);
        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6 | GPIO_Pin_7; //PIN PC6-T1(A) PC7-T2(B)
        GPIO_InitStructure.GPIO_Mode =  GPIO_Mode_AF;
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
        GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_DOWN;  //Trigger only on + voltage
        GPIO_InitStructure.GPIO_OType= GPIO_AF_TIM3;
        GPIO_Init(GPIOC, &GPIO_InitStructure);

        /*Connect PIN PC6 and PC7 on TIM 3 (Channel 1&2)*/                    }
  GPIO_PinAFConfig(GPIOC, GPIO_PinSource6, GPIO_AF_TIM3); //Enable GPIO PC6 To alternate
  GPIO_PinAFConfig(GPIOC, GPIO_PinSource7, GPIO_AF_TIM3); //Enable GPIO PC7 To alternate


        /*TIM3 setting*/
        TIM_DeInit(TIM3);
        RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);
        TIM3_TimeBaseStructure.TIM_Period = 3200;   //Number of encoder counts
        TIM3_TimeBaseStructure.TIM_Prescaler = 0;
        TIM3_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;
        TIM3_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
        //ENCODER MODE
       TIM_EncoderInterfaceConfig(TIM3,
                                  TIM_EncoderMode_TI12, //Count on both channel A in B
                                   TIM_ICPolarity_Rising,
                                   TIM_ICPolarity_Rising);
        TIM_TimeBaseInit(TIM3, &TIM3_TimeBaseStructure);


          TIM3->CNT=0; //Inital value of the encoder timer

        //Enable upadate flag
         TIM_ClearFlag(TIM3, TIM_FLAG_Update);
        //Timer interrupt enable, for one revolution
        TIM_ITConfig(TIM3, TIM_IT_Update, ENABLE);
        //Start timer TIM3
        TIM_Cmd(TIM3, ENABLE);
```

## Branje enkoderja preko registra TIMER3:

```
encoder = TIM3->CNT; //Read encoder value
```



Funded by the Erasmus+ Programme of the European Union

# Virtualni serijski port VCP

- V naslednjem primeru je zgled programski kode, ki prikazuje nastavitev USB-VCP vmesnika na razvojni plošči STM32 DiscoveryF407. Mikro-USB vtič, se nahaja na pinih PA11 in PA12, kot podatkovna linija (D- D+).
- Program opisuje prejemanje podatkov s končnim znakom '%' ter izpis prejete vsebine s pritiskom na gumb. S funkcijo USBD_Init() in argumentom USE_USB_OTG_FS inicializiramo USB-VCP vmesnik na pinih PA10,PA11,PA12,PA13.

```
USB_OTG_CORE_HANDLE USB_OTG_dev; //USB VCP structure

int main(void)
{
   uint8_t b;
   uint8_t i=0;
   char USB_read[100];

   //Nastavitev MCU-ja in GPIO
   RCC_Configuration();
   GPIO_Configuration();


    GPIO_SetBits(GPIOD, GPIO_Pin_12);  //  LED ON
    GPIO_SetBits(GPIOD, GPIO_Pin_13);  // LED ON


     /* Initialize USB VCP */
      USBD_Init(  &USB_OTG_dev,
                  USE_USB_OTG_FS
                  USB_OTG_FS_CORE_ID,
                  &USR_desc,
                  &USBD_CDC_cb,
                  &USR_cb);

   //Infinite loop
   while(1)
   {


         if(Button==1)//Send received data on the button press
         {
               if (TM_USB_VCP_Getc(&b) == TM_USB_VCP_DATA_OK) {


                     USB_read[i]=(char)b;
                     i++;

                        if(b=='%')//Terminal character
                         {
                              USB_read[i-1]=' ';
                              TM_USB_VCP_Puts(USB_read);TM_USB_VCP_Puts("\n\r"); //Send data

                                   for(int j=0;j<=i;j++) //Clear out buffer
                                   USB_read[j]=' ';
                              i=0; //Clear array index
                         }
               }
         }

   }//Infinite loop-END
}//MAIN - END
```



Funded by the Erasmus+ Programme of the European Union