

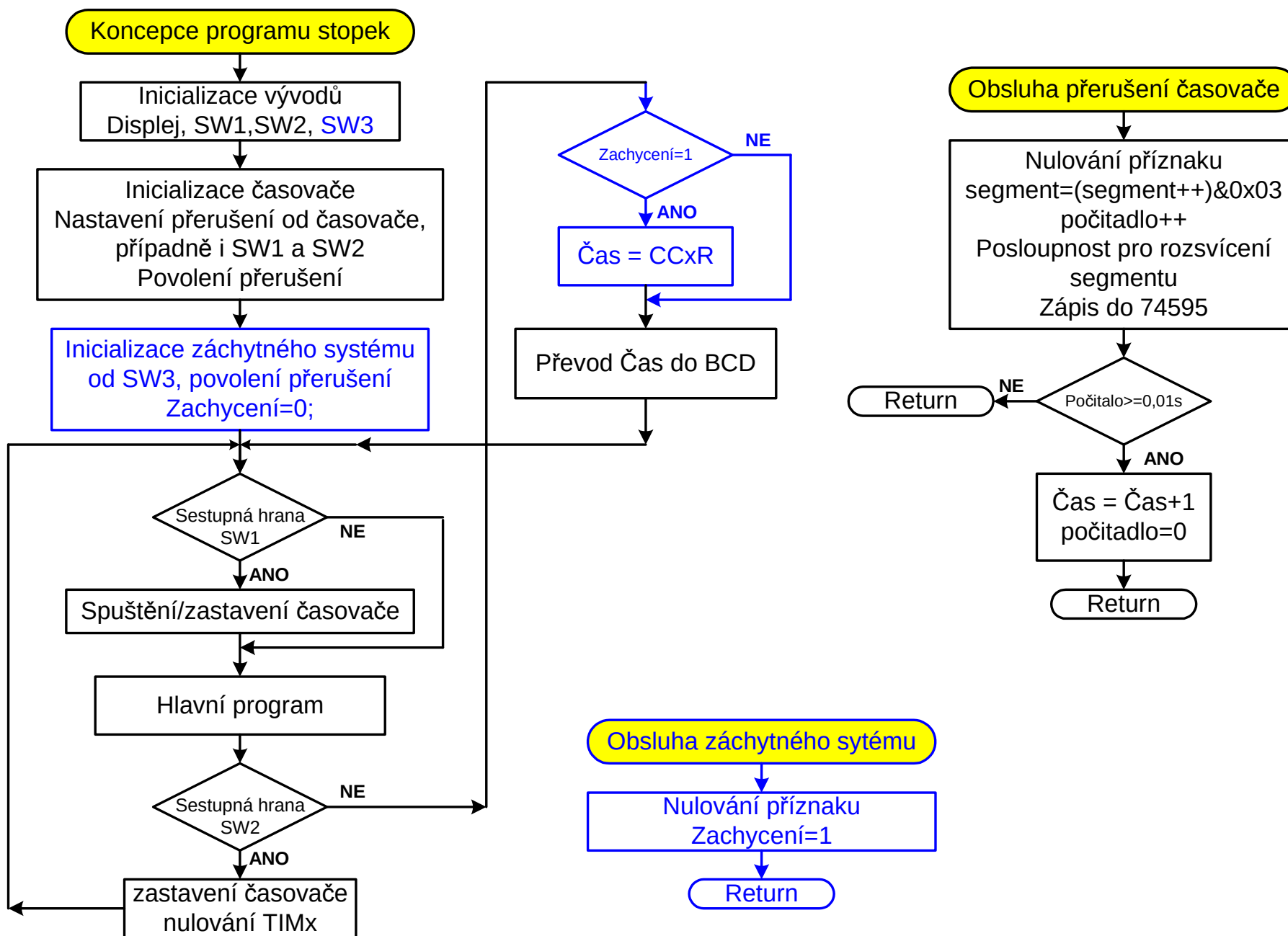
Dynamický displej LED

*Navrhněte v jazyce C program realizující stopky na 4 místném displeji s využitím časovače. Časový údaj bude ve tvaru #X,XX [s]. Pokud údaj bude menší jak 10s, nechť nula na nejvyšším řádu nesvítí. Při inicializaci V/V použijte knihovnu **Nastaveni_GPIO.c**. K řízení stopek využijte dvou tlačítek, kde jedno bude spouštět a zastavovat stopky a druhé je bude nulovat. Měření času realizujte pomocí časovače.*

Bonus

K měření času využijte výhody záchytného systému časovače - Input capture mód. To znamená, že třetí tlačítko bude představovat vstupní signál do záchytného systému. Po každém zmáčknutí bude průběžný zachycený čas zobrazován na displeji (stopky stále běží).

ŘEŠENÍ ÚLOHY 3 - VÝVOJOVÝ DIAGRAM , BONUS = MODRÁ



INICIALIZACE ČASOVAČE TIM2 A OBSLUHA JEHO PŘERUŠENÍ

Inicializace časovače a ověření volání jeho přerušovací rutiny.

1. Pro dlouhé časové intervaly pomocí diody LED
2. Pro krátké časové intervaly pomocí osciloskopu

```
void TIM2_Inicializace(void)
{
//  TIM2 je připojen ke sběrnici APB1, která běží na 24MHz
//  frekvence před TIM2 je zdvojnásobena PLL
RCC->APB1ENR |= RCC_APB1ENR_TIM2EN; // Povolení hodinového signálu pro TIM2
//  nebo setbit(RCC->APB1ENR, 0);
TIM2->PSC = XXXX; // Nastavení dělicího poměru XXXX předděliče
TIM2->ARR = YYYY; // Nastavení hodnoty YYYY přednastavení ARR
TIM2->DIER |= TIM_DIER_UIE; // Povolení přerušování od TIM2
NVIC_EnableIRQ(TIM2_IRQn); // Povolení přerušování od TIM2 v kontroléru NVIC
TIM2->CR1 |= TIM_CR1_CEN; // Spuštění časovače TIM2
// nebo setbit(TIM2->CR1, 0);
TIM2->SR &= ~TIM_SR_UIF; // Nulování příznaku události časovače TIM2
// nebo clearbit(TIM2->SR, 0);
}

void TIM2_IRQHandler() // Obslužná rutina přerušování TIM2
{
TIM2->SR &= ~TIM_SR_UIF; // Nulování příznaku události od časovače TIM2
...
... // Realizace obsluhy pro každé přetečení
... // časovače TIM2
}
```

Úloha se skládá z dílčích modulů s využitím přerušení od časovače TIMx a modifikace přerušení tlačítka SW1 a SW2.

Budeme muset vytvořit:

- ❖ Inicializaci časovače TIMx, který vytvoří časovou základnu nejen pro 0,01s, ale též pro přepínání jednotlivých segmentovek.
- ❖ Přerušovací rutinu časovače TIMx, v které po každém přerušení provedeme rozsvícení další segmentovky. Po N přerušení (záleží na periodě časovače TIMx), kdy uplyne čas 0,01s, zvětšíme počítadlo setin vteřiny.
- ❖ Vytvoříme podprogram pro odeslání stavu segmentovky (ekvivalent řadiče v FPGA) a doplníme jej o nastavení bitu určujícího, pro kterou segmentovku je určen.
- ❖ Upravíme inicializaci externího přerušení modrého tlačítka z úlohy 1 na tlačítko SW1.
- ❖ Přerušovací rutinu tlačítka upravíme tak, aby na první zmáčknutí se stopky rozběhly a na druhé zastavily.

ŘEŠENÍ ÚLOHY 3

- ❖ Vytvoříme proměnnou nebo proměnné pro časový údaj
- ❖ V závislosti na volbě proměnné nebo pole vytvoříme převod časového údaje do dekadické soustavy.
- ❖ Vytvoříme generátor znaků, který pro každé číslo vytvoří hodnotu odpovídající zobrazovanému znaku (ekvivalent převodníku BCD na 7 segmentů).

Podprogram pro přesun a rozsvícení zvoleného segmentu

Podprogram převezme hodnotu z generátoru znaků doplněnou o pozici segmentu a bit po bitu nasune do posuvných registrů 595. Po každém přivedeném bitu na vstup 595 generuje hodinový impulz.

o Nastavování datových bitů na vstup 595

- **Vysunovat bity do příznaku** přenosu a jeho testováním rozhodnout, zda na sériový vstup přivedeme 1 nebo 0.

- ♣ `hodnota=hodnota+hodnota;`

- ♣ `if (carry) setbit(GPIOA->ODR,9); else clrbit(GPIOA->ODR,9);`

Nebezpečná realizace – pracujeme-li s proměnnou pole (hodnota příznaku C se změní, při výpočtu adresy výsledku).

- **Testováním a maskováním bitů čísla**

- ♣ `for (i=0; i<16; i++)`

- `{ if (hodnota&0x8000)!=0) setbit(GPIOA->ODR,9);`
`else clearbit(GPIOA->ODR,9); hodnota+=hodnota; }`

- ♣ `v cyklu s proměnnou j if ( hodnota &(1<< j))`
`if (hodnota&(0x8000>>i))!=0)`

ŘEŠENÍ ÚLOHY 3

- o **Doplnění bitu, určujícího svítící segmentovku.** Budou-li spodní čtyři bity nulové můžeme:

- **Bit přidat operací OR, EX-OR nebo aritmetickým součtem pomocí**

- ♣ 1<<která ♣ pole napětí [4]={ 0x1, 0x2, 0x4, 0x8};
- ♣ if -else ♣ switch

Obdobným způsobem přidáme k třetí segmentovce desetinnou tečku

- o Po nastavení datového bitu na vstup 74HC595 realizujeme hodinový impulz. **Pozor na počáteční stav na vývodu PA.8** (nebezpečí vynechání jednoho impulsu CLK).

- ♣ `clrbit(GPIOA->ODR,8); setbit(GPIOA->ODR,8);`

- o **Po nasunutí celé posloupnosti** je potřeba realizovat přepis do paralelního registru.

- ♣ `clrbit(GPIOB->ODR,5); setbit(GPIOB->ODR,5);`

Poznámka: Na vývodech typu otevřený kolektor se slabou log.1, není vhodné **dlouhodobě setrvávat ve stavu log.1.**

ŘEŠENÍ ÚLOHY 3

Proměnná(é) pro časový údaj a jejich modifikace každých 0,01s.

➤ **Pole pro jednotlivé cifry** unsigned char cifra[4] = {0, 0, 0, 0};

```
♣ if (++cifry[0]>9)
    { cifry[0]=0; if (++cifry[1]>9)
        { cifry[1]=0; if (++cifry[2]>9)
            { cifry[2]=0; if (++cifry[3]>9) cifry[3]=0; }}}}
```

➤ **Proměnná čas**, v které čtveřice bitů realizuje jednu cifru

```
♣ ++cas; if ((cas&0x000F)>9)
    { cas=cas&0xFFFF0+0x0010;
      if ((cas&0x00F0)>0x0090)
        { cas=cas&0xFF0F+0x0100;
          if ((cas&0x0F00)>0x0900)
            { cas=cas&0xF0FF+0x1000;
              if ((cas&0xF000)>0x9000) cas=cas&0xFFFF; }}}}
```


ŘEŠENÍ ÚLOHY 3

- **Proměnná čas představuje binární číslo** (= počet setin vteřiny), která bude převedena

- ☐ převod pomocí dekadické předkorekce

- ♣ `kolik=++cas;`
`for (i=0; i<16; i++)`
`{ korekce=0;`
`if ((kolik&0x000F0000)>=0x00050000) korekce|=0x00030000;`
`if ((kolik&0x00F00000)>=0x00500000) korekce|=0x00300000;`
`if ((kolik&0x0F000000)>=0x05000000) korekce|=0x03000000;`
`if ((kolik&0xF0000000)>=0x50000000) korekce|=0x30000000;`
`kolik=kolik+korekce;`
`kolik=kolik+kolik;`
`}`
`cifra[3]=(kolik&0xF0000000)>>28;`
`cifra[2]=(kolik&0x0F000000)>>24;`
`cifra[1]=(kolik&0x00F00000)>>20;`
`cifra[0]=(kolik&0x000F0000)>>16;`

ŘEŠENÍ ÚLOHY 3

□ pomocí dělení a operace modulo

♣ `cas++; cifra[3]=cas/1000; pomoc=cas % 1000; cifra[2]=pomoc/100;
pomoc= pomoc % 100; cifra[1]=pomoc/10; cifra[0]= pomoc % 10;`

nebo

♣ `cas++; cifra[3]=cas /1000; pomoc=cas-cifra[3]*1000;
cifra[2]=pomoc/100; pomoc=pomoc-cifra[2]*100;
cifra[1]=pomoc/10; cifra[0]=pomoc % 10;`

Generátor znaků

➤ Pole o deseti prvcích určujících na jednotlivých bitech zda segment svítí (log.0) nebo ne (log.1).

♣ `Znaky[10]={0xC0, 0xF9, 0xA4, atd. } // Hodnota pro znak 0, 1, 2,
atd.`

16-bitová hodnota pro přesun do kaskády 74HC595 např.

♣ `hodnota = (znaky[cifra[která]]<<8) | napětí[která]; // místo OR může
// být +, EX-OR`

Časová základna

- V přerušovací rutině zajistíme přepínání segmentovek a časový interval 0,01s. Frekvence přepínání segmentovek musí být minimálně 72[Hz]. Pro snadné vytvoření intervalu 0,01s volíme přerušení každých 2,5μs.
- Vytvoříme si **počítadlo** určující počet zavolání přerušovací rutiny.
 - ♣ počítadlo++;
která=počítadlo&0x03; // každá cifra zobrazena 100x za vteřinu
if (která==0) cas++; // přičtení setiny vteřiny
// výpočet hodnoty --- v přerušení nebo hlavním programu
// odeslání hodnoty do 74HC595 --- v přerušení nebo hlavním programu

Záchytný systém

- V manuále je potřeba zjistit vývody, které tuto funkci umožňují. Vybraný vývod musíme nakonfigurovat na **alternativní funkci**.

ZÁCHYTNÝ SYSTÉM - INICIALIZACE

Zároveň musíme v registrech AFRL nebo AFRH definovat potřebnou alternativní funkci (např. AFIO2) takto:

```
PIN_ALT_PP_Initialize (GPIOB,0);           // Inicializace tlačítka SW3 - vstup
                                           // záchytného systému TIM3-CH3
GPIOB->AFR[0] |= GPIO_AFRL_AFRL0 & 0x00000002; // AFRL0 na AFIO2  PB.0
```

```
void ZACHYTNY_Inicializace()                // Odvozena z časovače TIM3
{
    RCC->APB1ENR |= RCC_APB1ENR_TIM3EN;      // Povolení hodin časovače TIM3
    TIM3->CR1 |= TIM_CR1_ARPE|TIM_CR1_URS;    // Reset 0000, CKD=00, ARPE-buffred,
                                           // CMS=00, DIR=UP, OPM=0, USR=jen
                                           // přetečení, UDIS=enable
                                           // CEN=zatím nepovoleno

    TIM3->CCMR2 |= TIM_CCMR2_CC3S_0;          // Aktivace kanálu 3 na vstup TI3
    TIM3->CCER &= ~(TIM_CCER_CC3NP | TIM_CCER_CC3P); // Nuťování bitu CC3NP a CC3P
    TIM3->CCER |= TIM_CCER_CC3P;              // Reakce na sestupnou hranu
    TIM3->SMCR |=TIM_SMCR_SMS&0x0;           // Interní hodiny přímo do předdělice
    TIM3->CCER |= TIM_CCER_CC3E;              // Povolení zachycení citace TIM3
    TIM3->SR &= ~TIM_SR_UIF;                  // Nuťování příznaku události
                                           // od časovače TIM3

    TIM3->SR &= ~TIM_SR_CC3IF;                // Nuťování příznaku zachycení v kanálu 3
    TIM3->CCMR2 |= TIM_CCMR2_CC3S&0x01;      // IC3 mapovaný na TI3
    TIM3->DIER |= TIM_DIER_CC3IE;            // Povolení přerušování při zachycení CC3
    TIM3->CR1 |= TIM_CR1_CEN;                 // Povolení časovače TIM3
}
```

VÝVODY OBSAZENÉ DESTIČKOU DISPLEJE A JEJICH POUŽITELNOST

Indikační diody				
Vývod	Funkce	Vstup	Výstup	Alternativní Funkce
PA-5	Led D1	ANO	ANO	ANO
PA-6	Led D2	ANO	ANO	ANO
PA-7	Led D3	ANO	ANO	ANO
PB-6	Led D4	ANO	ANO	ANO
Tlačítka				
PA-1	SW-1	ANO	NE	ANO (vstupní)
PA-4	SW-2	ANO	NE	ANO (vstupní)
PB-0	SW-3	ANO	NE	ANO (vstupní)
Reset	SW-4	ANO	NE	NE
Displej				
PA-9	Sériová data	ANO	ANO	ANO
PA-6	Hodinový signál	ANO	ANO	ANO
PA-7	Zápis posloupnosti	ANO	ANO	ANO
Zvuk				
PB-3	Spínání repro	NE	ANO	ANO
Analogový vstup				
PA-0	Vstup děliče napětí	ANO	ANO (pozor)	ANO

Záchytný systém AF – PA-6, PB-0, PB-4, PC-6 (AFIO2)

IC1IOS (AFIO14) – PA-0, PA-4, PA-8, PC-0, PC-4, atd. dle tabulky IC1IOS