

➤ **Nezbytná literatura pro základní informace**

- **RM0368 - Reference manual_F401.pdf**
- **Cortex-M4_Generic User Guide.pdf**
- **STM32F401RE.pdf**
- **STM32F4xx_Clock_Configuration_V1.1.0.xls**
- **MDK5-getting-started.pdf**

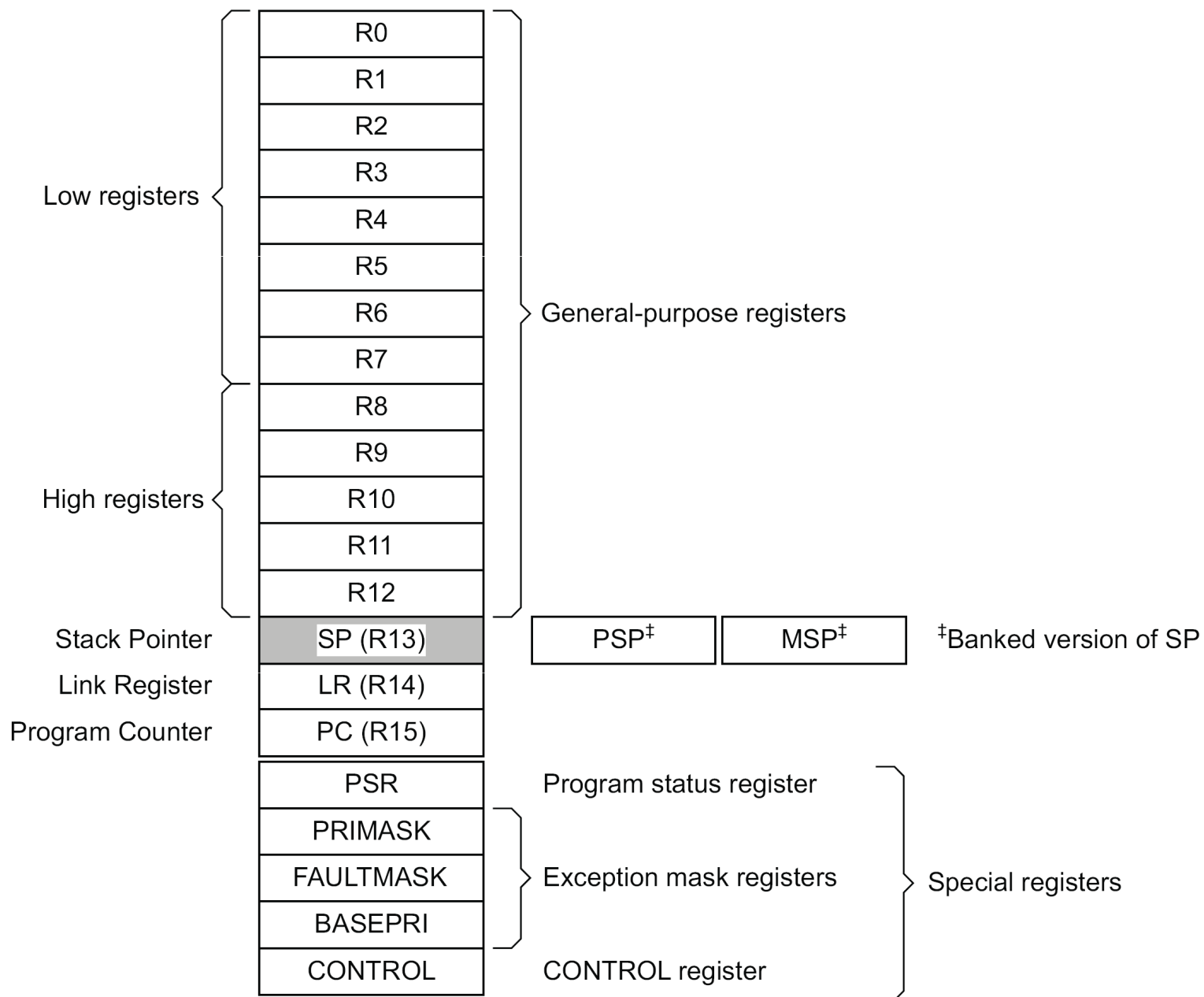
Pro podrobnější studium

- **M4_Technical Reference Manual.pdf**
- **Timers_STM32F4.pdf** nebo **Timers_STM32L4.pdf**
- **Clock configuration_Application note_AN3988.pdf**
- **STM32F4 HAL and LL drivers_Manual_UM1725.pdf**

Pro práci v JSA (Assembleru)

- **Cortex-M3_M4F Instruction Set.pdf**
- **DUI0473C_Using_the_ARM_Assembler_0.pdf**
- **DUI0489C_Arm_Assembler_Reference.pdf**

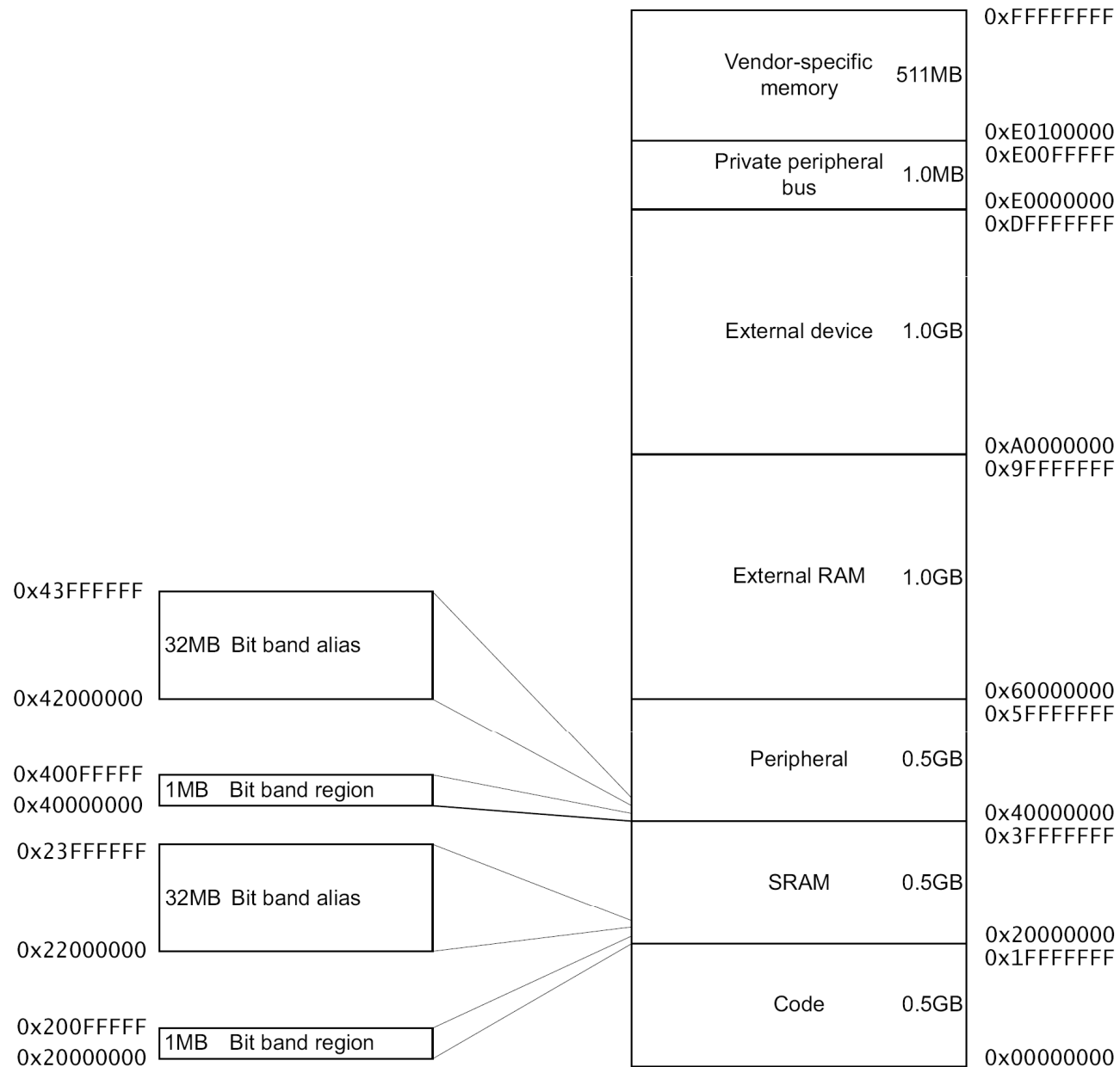
REGISTRY ARM M3 A M4 A JEJICH FUNKCE



REGISTRY NA PROCESORECH ARM

- **R0–R12: všeobecně použitelné registry.** Některé 16-ti bitové instrukce Thumb® mají přístup pouze ke registrům R0 až R7.
- **R13: Ukazatel zásobníku (stack pointer)** – dva ukazatele
 - **MSP** defaultní - operačním systémem a obsluha přerušení.
 - **PSP** používaný v aplikacích. Oba zarovnány mod4 (.align 4) na dvě slova nebo adresu.
- **R14: Link Register.** - jednoúrovňový zásobník
- **R15: Program Counter** - adresa čteného bytu programu (instrukce). Oproti jiným procesorům je přístupný.
- **Speciální registry** - mají speciální funkci ovlivněnou speciálními instrukcemi.
 - **xPSR** – Program status registr - **příznaky**
 - **PRIMASK, FAULTMASK a BASEPRI** - Interrupt Mask registers,
 - **CONTROL** – řídicí registr.

ROZLOŽENÍ ADRESOVÉHO PROSTORU ARM M3 A M4



PŘÍZNAKY CORTEX M3 A M4

xPSR (Program Status Register) – je rozdělený na tři stavové registry:

- ❖ Stavový registr aplikačního programu (APSR)
- ❖ Stavový registr přerušení (IPSR)
- ❖ Stavový registr prováděného programu (EPSR)

Stavové registry mohou být přístupné společné nebo odděleně pomocí instrukcí MSR a MRS. Při společném přístupu se používá označení xPSR. Registry EPSR a IPSR mohou být pouze čteny (MRS), registr APSR může být měněn instrukcí MSR.

	31	30	29	28	27	26:25	24	23:20	19:16	15:10	9	8	7	6	5	4:0
APSR	N	Z	C	V	Q											
IPSR												Exception number				
EPSR						ICI/IT	T				ICI/IT					

Registr stavových příznaků xPSR má 32 bitů, z nichž 5 nejvyšších bitů je pro začátek nejdůležitějších. Umístění příznaků v xPSR je zobrazeno na obrázku a jejich význam je následující:

	31	30	29	28	27	26:25	24	23:20	19:16	15:10	9	8	7	6	5	4:0	
xPSR	N	Z	C	V	Q	ICI/IT	T				ICI/IT		Exception number				

N – Negativní nebo menší než je nastaven jestliže po instrukci je do bitu b31 přenesena hodnota 1.

PŘÍZNAKY CORTEX M3 A M4

Z – Nula – nastaven aritmetickou nebo instrukcí porovnání.

C – Carry/Borrow - nastaven při přenosu z bitu b31 do bitu b32 nebo výpůjčce. Pro instrukce **ADC**, **ADD** a **CMN** je produktem přetečení. Pro **CMP**, **SBC** a **SUB**, je bit nastaven při výpůjčce. Při instrukcích posunu se do něj přenáší poslední bit.

V - Příznak přetečení (Overflow) indikuje přetečení při aritmetické operaci s čísly se znaménkem. Kdy součet záporných čísel je kladný nebo součet kladných čísel je záporný.

Q – Indikace saturace je nastaven, dojde-li při aritmetice se saturací k dosažení maximální nebo minimální hodnoty.

ICI/IT – slouží ke správnému vykonání instrukce IF-THEN, při které dojde k přerušení

T – Thumb mód=log.1

Exception number – indikuje, které přerušení procesor zpracovává

ZÁKLADNÍ ROZDÍLY CORTEX M3, M4 PROTI 8051, AVR atd.

V jazyce C i assembleru musíme před programem main:

- ❖ Vložit soubor s adresami symbolických názvů registrů procesoru např. `system_STM32F4xx.c`
- ❖ Vložit soubor se zdroji přerušení, nastavením ukazatele SP a programem se skokem na main - `startup_STM32F4xx.s`
- ❖ **Nastavit konfigurační bity μ P – Boot, zdroj hodin, WD, JTAG**
- V MAIN je nejdříve potřeba:
 - ❖ Případně nastavit nový hodinový kmitočet procesoru
 - ❖ Nastavit hodinový kmitočet pro sběrnice APB1 a APB2, ke kterým jsou připojeny Vámi používané periferie.
 - ❖ **Nakonfigurovat používané vstupně/výstupní vodiče 16 bitových bran GPIOA, GPIOB, až GPIOK, atd. podle možností procesoru.**
- **Nastavit ostatní periferie, povolit přerušovací systém, atd.**
- **Vytvářet požadovaný program**

Udělat pro procesor ARM ♣, ♣ a ♣ (všechno), AVR ♣, ♣ a ♣, 8051 jen ♣

Start procesoru – uvedení do definovaného počátečního stavu, který zjišťuje nulování μP (tzv. Reset).

Reset může být vyvolán

- Log.0 na vývodu NRST – externí reset
- Oknovým watchdogem WWDG při podtečení
- Nezávislým watchdogem IWDG dosažením konce čítání
- Programovým nulováním SW reset
- Obvodem pro kontrolu napájecího napětí a spotřeby
 - Nastavením **Standby mode** při nRST_STDBY=0
 - Nastavením **Stop mode** při nRST_STOP=0
 - Aktivací obvodu POR/PDR nebo BOR (prahové úrovně)

Konfiguraci hodinového signálu a jeho přivedení k potřebným jednotkám zajišťuje skupina registrů

RCC (Reset and Clock Control)

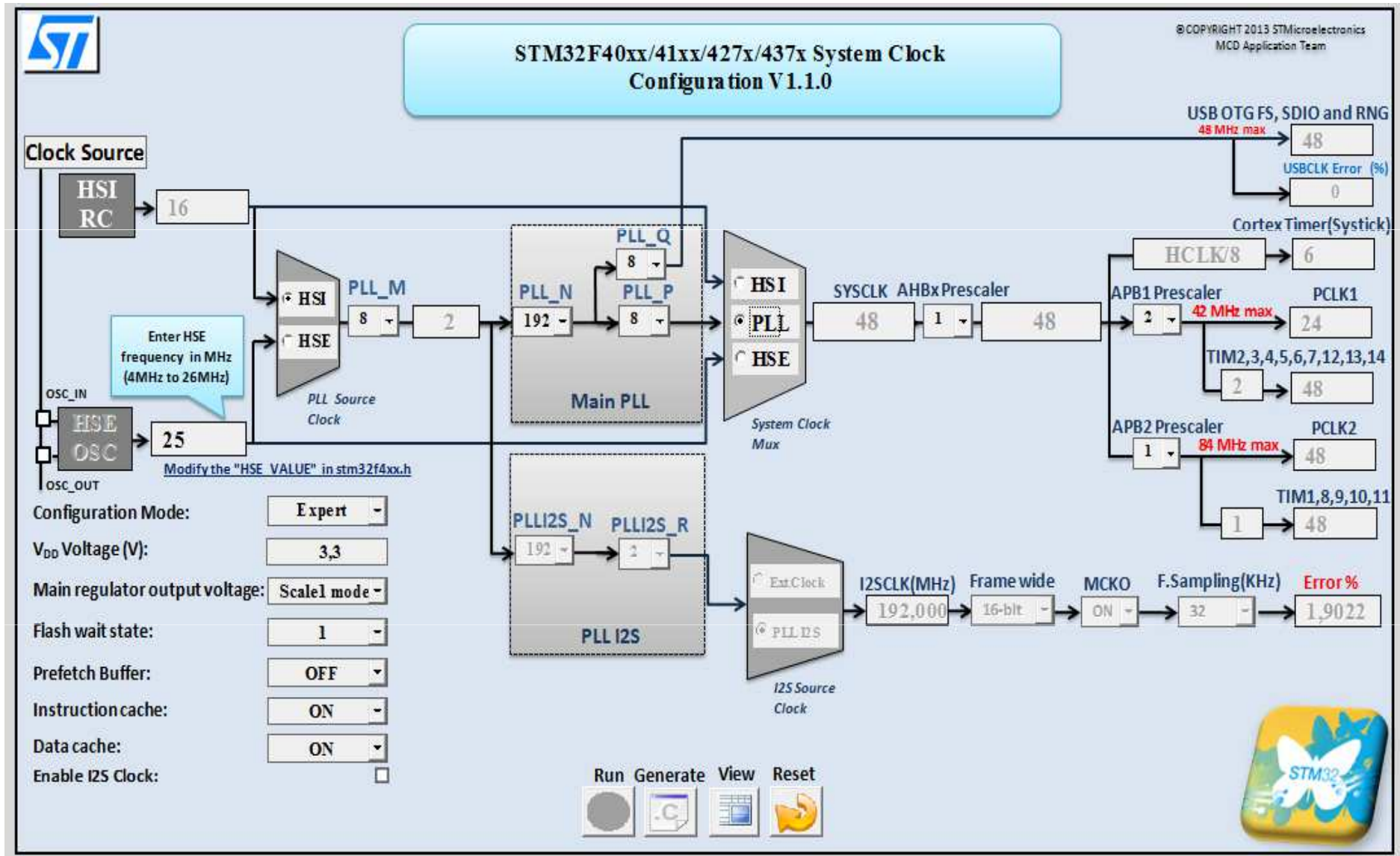
Hodinový signál – se konfiguruje nejen pro samotný procesor a přenos po sběrnicích, ale ke každé periférii, kterou budeme v aplikaci potřebovat (rozdíl vůči 8051, AVR, HC11, atd.).

Zdroje hodinového signálu:

- **HSI** RC oscilátor – defaultní kompenzovaný interní zdroj
- **HSE** interní oscilátor
 - S externím krystalovým/keramickým rezonátorem
 - S externím hodinovým signálem
- **PLL** fázový závěs řízený HSI nebo HSE
- Sekundární zdroje hodin
 - 32 kHz low-speed internal RC (LSI RC) pro WDT
 - 32.768 kHz low-speed external crystal (LSE crystal) pro RTC.

Konfiguraci umožňují **RCC Registry**

KONFIGURACE HODINOVÉHO SYSTÉMU PROCESORU F401



KONFIGURACE HODINOVÉHO SYSTÉMU PROCESORU F401

```
// SystemCoreClockConfigure: configure SystemCoreClock using HSI (HSE is not populated on Nucleo board)
```

```
void SystemCoreClockConfigure(void)
{
    //RCC->CR |= ((uint32_t)RCC_CR_HSION);           // Enable HSI
    while ((RCC->CR & RCC_CR_HSIRDY) == 0);        // Wait for HSI Ready
    RCC->CFGR = RCC_CFGR_SW_HSI;                    // HSI is system clock
    while ((RCC->CFGR&RCC_CFGR_SWS)!=RCC_CFGR_SWS_HSI); // Wait for HSI used as system clock

    FLASH->ACR = FLASH_ACR_PRFTEN;                // Enable Prefetch Buffer
    FLASH->ACR |= FLASH_ACR_ICEN;                  // Instruction cache enable
    FLASH->ACR |= FLASH_ACR_DCEN;                  // Data cache enable
    FLASH->ACR |= FLASH_ACR_LATENCY_5WS;          // Flash 5 wait state

    RCC->CFGR |= RCC_CFGR_HPRE_DIV1;               // HCLK(sběrnice AHB) = SYSCLK
    RCC->CFGR |= RCC_CFGR_PPRE1_DIV4;              // APB1 = HCLK/4    MAX 42MHz
    RCC->CFGR |= RCC_CFGR_PPRE2_DIV2;              // APB2 = HCLK/2    MAX 84MHz

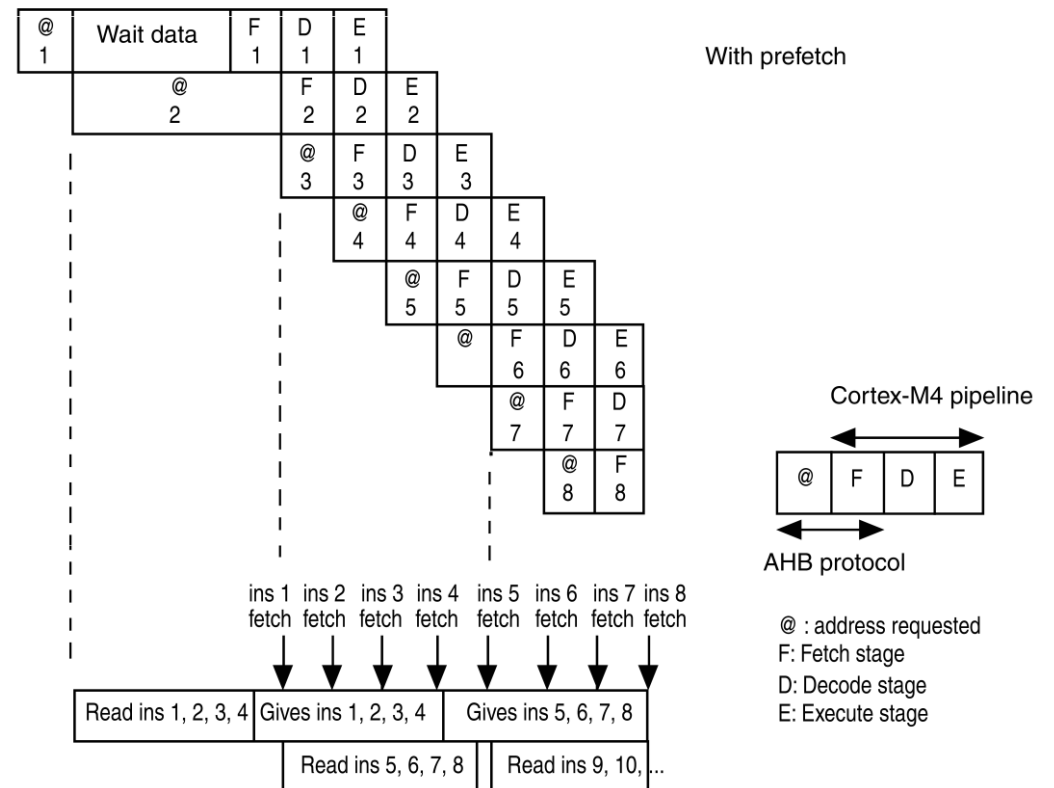
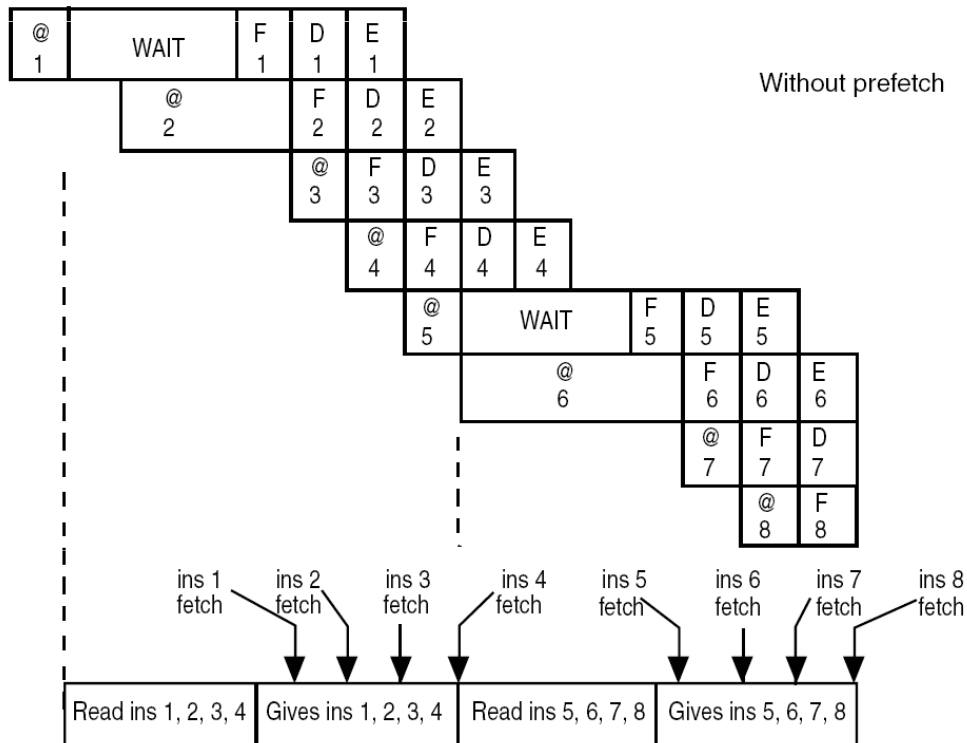
    RCC->CR &= ~RCC_CR_PLLON;                       // Disable PLL
    // PLL configuration: VCO = HSI/M * N, Sysclk = VCO/P
    RCC->PLLCFGR = ( 16ul |                          // PLL_M = 16
                   (384ul << 6) |                  // PLL_N = 384
                   ( 3ul << 16) |                  // PLL_P = 8
                   (RCC_PLLCFGR_PLLSRC_HSI) |      // PLL_SRC = HSI
                   ( 8ul << 24) );                 // PLL_Q = 8

    RCC->CR |= RCC_CR_PLLON;                         // Enable PLL
    while((RCC->CR & RCC_CR_PLLRDY) == 0) __NOP(); // Wait till PLL is ready
    RCC->CFGR &= ~RCC_CFGR_SW;                        // Select PLL as system clock source
    RCC->CFGR |= RCC_CFGR_SW_PLL;
    while ((RCC->CFGR & RCC_CFGR_SWS) != RCC_CFGR_SWS_PLL); // Wait till PLL is system
                                                                // clock src
}
}
```

SYSTEM ČTENÍ PROGRAMU Z PAMĚTI FLASH U PROCESORŮ F4

```

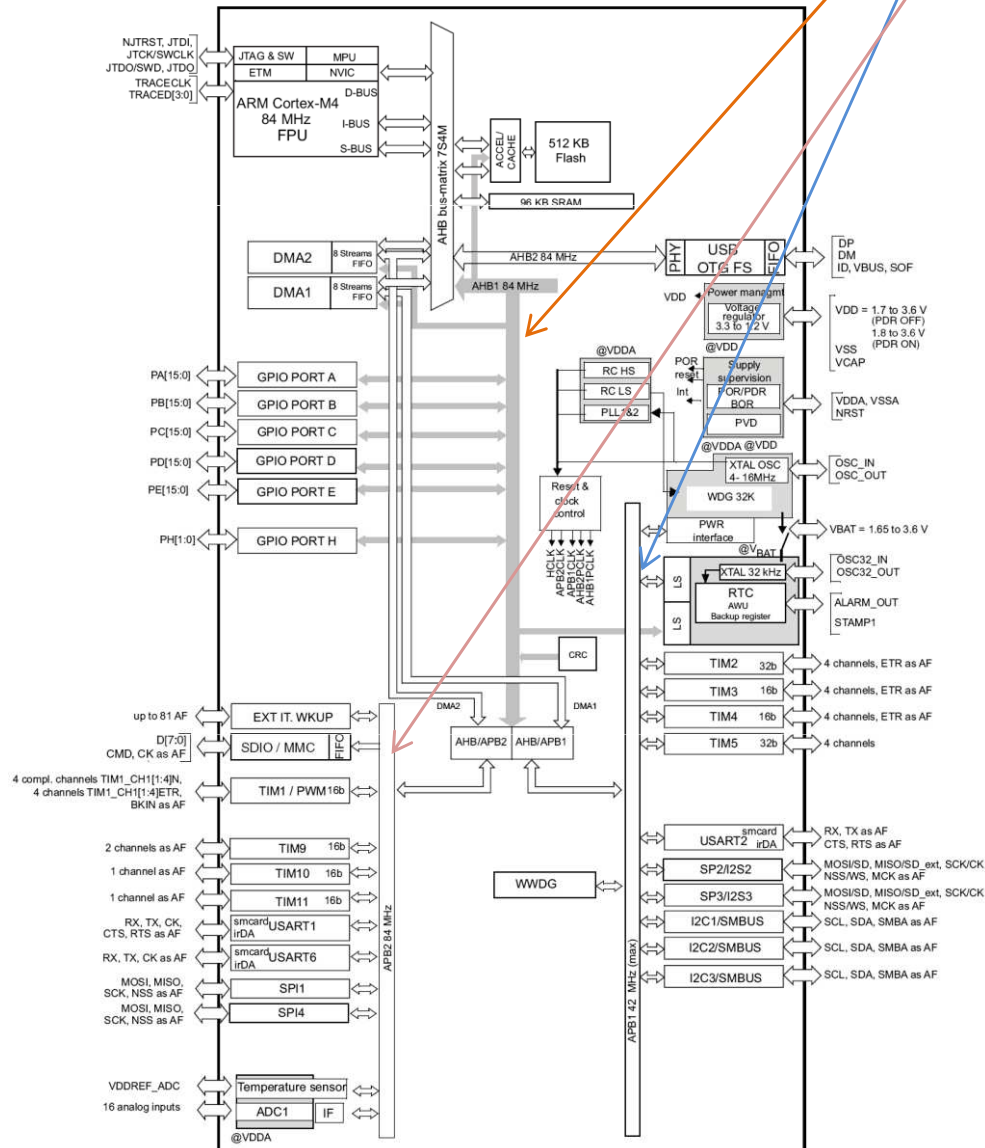
FLASH->ACR = FLASH_ACR_PRFTEN; // Enable Prefetch Buffer
FLASH->ACR |= FLASH_ACR_ICEN; // Instruction cache enable
FLASH->ACR |= FLASH_ACR_DCEN; // Data cache enable
FLASH->ACR |= FLASH_ACR_LATENCY_5WS; // Flash 5 wait state
    
```



KONFIGURACE HODINOVÉHO SIGNÁLU PRO SBĚRNICE

```

RCC->CFGR | = RCC_CFGR_HPRE_DIV1; // HCLK(sběrnice AHB) = SYSCLK
RCC->CFGR | = RCC_CFGR_PPRE1_DIV4; // APB1 = HCLK/4    MAX 42MHz
RCC->CFGR | = RCC_CFGR_PPRE2_DIV2; // APB2 = HCLK/2    MAX 84MHz
    
```



KONFIGURACE BITŮ V REGISTRECH

Nastavování bitů v konfiguračních registrech

```
#define setbit(reg, bit)    ((reg) |= (1U << (bit)))
```

```
#define clearbit(reg, bit) ((reg) &= (~(1U << (bit))))
```

```
#define togglebit(reg, bit) ((reg) ^= (1U << (bit)))
```

```
#define getbit(reg, bit)   (((reg) & (1U << (bit))) >> (bit))
```

Jednoduchý způsob \Rightarrow nutné zjišťování polohy bitu v registru, nutná kontrola při přechodu na jiný procesor ARM. Pro každý procesor existuje soubor (zde NUCLEO401)

stm32f401xe.h

obsahující symbolické názvy registrů a jeho bitů. Řada registrů je koncipována jako **struktury** zajišťující odstup adres jednotlivých registrů. Při použití symbolických názvů ARM ST elektronik nemusíme **ověřovat** umístění jednotlivých bitů v konfiguračních registrech při přechodu na jiný procesor.

KONFIGURACE VÝVODŮ S POMOCÍ STM32F401XE.H

Konstrukce přístupu k registrům jednotlivých bran procesoru

```
#define PERIPH_BASE          0x40000000U /*!< Peripheral base address in the alias region
#define AHB1PERIPH_BASE    (PERIPH_BASE + 0x00020000U)
#define GPIOA_BASE        (AHB1PERIPH_BASE + 0x0000U)
#define GPIOB_BASE        (AHB1PERIPH_BASE + 0x0400U)
// atd.
#define GPIOA                ((GPIO_TypeDef *) GPIOA_BASE)

typedef struct
{
  __IO uint32_t MODER;      /*!< GPIO port mode register,           Address offset: 0x00 */
  __IO uint32_t OTYPER;    /*!< GPIO port output type register,      Address offset: 0x04 */
  __IO uint32_t OSPEEDR;   /*!< GPIO port output speed register,     Address offset: 0x08 */
  __IO uint32_t PUPDR;     /*!< GPIO port pull-up/pull-down register, Address offset: 0x0C */
  __IO uint32_t IDR;       /*!< GPIO port input data register,       Address offset: 0x10 */
  __IO uint32_t ODR;       /*!< GPIO port output data register,      Address offset: 0x14 */
  __IO uint32_t BSRR;      /*!< GPIO port bit set/reset register,    Address offset: 0x18 */
  __IO uint32_t LCKR;      /*!< GPIO port configuration lock register, Address offset: 0x1C */
  __IO uint32_t AFR[2];    /*!< GPIO alternate function registers,   Address offset: 0x20-0x24 */
} GPIO_TypeDef;
```

KONFIGURACE VÝVODŮ S POMOCÍ STM32F401XE.H

Dál následuje označení jednotlivých bitů registrů např. **MODER** 2 bit

```
#define GPIO_MODER_MODE1_Pos    (2U)
#define GPIO_MODER_MODE1_Msk    (0x3U << GPIO_MODER_MODE1_Pos) /*!< 0x0000000C */
#define GPIO_MODER_MODE1
#define GPIO_MODER_MODE1_0      (0x1U << GPIO_MODER_MODE1_Pos) /*!< 0x00000004 */
#define GPIO_MODER_MODE1_1      (0x2U << GPIO_MODER_MODE1_Pos) /*!< 0x00000008 */
```

Registr **IDR** 3 bit

```
#define GPIO_IDR_ID2_Pos        (2U)
#define GPIO_IDR_ID2_Msk        (0x1U << GPIO_IDR_ID2_Pos) /*!< 0x00000004 */
#define GPIO_IDR_ID2
#define GPIO_IDR_IDR_2          GPIO_IDR_ID2
```

Registr **ODR** 2 bit

```
#define GPIO_ODR_OD1_Pos        (1U)
#define GPIO_ODR_OD1_Msk        (0x1U << GPIO_ODR_OD1_Pos) /*!< 0x00000002 */
#define GPIO_ODR_OD1
#define GPIO_ODR_ODR_1          GPIO_ODR_OD1
```


KONFIGURACE A OVLÁDÁNÍ LED NA BRÁNĚ GPIOA VÝVOD PA5

```
// PODPROGRAM PRO INICIALIZACI A NÁSLEDNĚ ROSVÍCENÍ a ZHASNUTÍ LED NA BRÁNĚ PA5

#include "stm32f4xx.h" // Device header
#define LED 5

int32_t LED_Initialize (void) // void možno nahradit proměnou
{
    RCC->AHB1ENR |= (1ul << 0); // Povolení hodinového signálu pro GPIOA
    // Nastavení vývodu PA.5 (Zelena LED) na výstup push-pull
    // bez upnutí k napájení nebo zemi
    GPIOA->MODER    &= ~((3ul << 2*LED)); // Stav po nulování (rušení předchozího
stavu)
    GPIOA->MODER    |= ((1ul << 2*LED)); // Vystup
    GPIOA->OTYPER    &= ~((1ul << LED)); // Push-Pull
    GPIOA->OSPEEDR   &= ~((3ul << 2*LED)); // Rušení předchozího stavu
    GPIOA->OSPEEDR   |= ((1ul << 2*LED)); // Medium speed
    GPIOA->PUPDR     &= ~((3ul << 2*LED)); // Bez Pull DOWN i Pull UP
    return (0);
}

int32_t LED_On (uint32_t num)
{
    if (num < 16)
        { GPIOA->BSRR |= (1ul << LED); }
    // nebo GPIOA->BSRRL |= (1ul << LED);
    return (0);
}

int32_t LED_Off (uint32_t num)
{
    if (num < 16)
        { GPIOA->BSRR |= (1ul << LED)<<16; }
    // nebo GPIOA->BSRRH |= (1ul << LED);
    return (0);
}
```

KONFIGURACE A ČTENÍ STAVU TLAČÍTKA NA VÝVODU PC13

```
// PODPROGRAM PRO INICIALIZACI A NÁSLEDNĚ ZJIŠTĚNÍ STAVU TLAČÍTKA NA VÝVODU PC13

#include "stm32f4xx.h" // Device header

int32_t Buttons_Initialize (void)
{
    RCC->AHB1ENR |= (1ul << 2); // Povolení hodinového signálu pro GPIOC
                                // V registru AHB1ENR nastaven bit 2
                                // (počítáno od 0)

    // Nastavení PC.13 (Modré tlačítko) na vstup, bez upnutí k napájení nebo zemi
    GPIOC->MODER    &= ~(3ul << 2*13); // Vstup
    GPIOC->OSPEEDR  &= ~(3ul << 2*13); // Rušení předchozího stavu
    GPIOC->OSPEEDR |= (1ul << 2*13); // Medium speed
    GPIOC->PUPDR    &= ~(3ul << 2*13); // Bez upínacích odporu
    return (0);
}

uint32_t Buttons_GetState (void)
{
    if ((GPIOC->IDR & (1ul << 13)) == 0) return(1);
    else return(0);
}
```

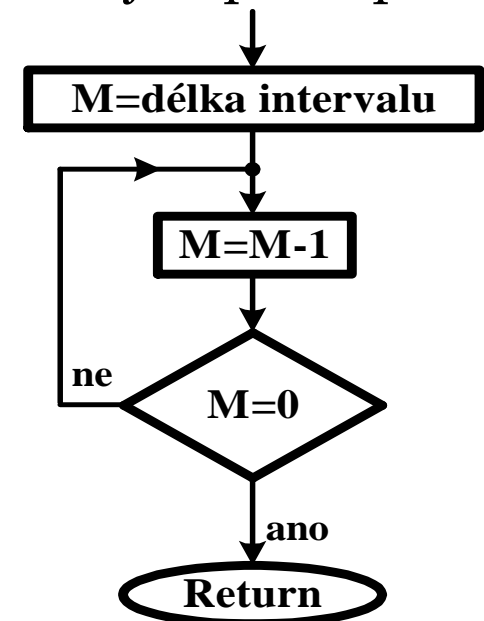
PROGRAMOVĚ GENEROVANÉ ZPOŽDĚNÍ

Vytvoření potřebného zpoždění můžeme realizovat

- Monostabilním obvodem (číslico-analogové řešení)
- Čítačem počítajícím impulzy hodinového signálu (čisté číslicové řešení). V procesorech jsou proto (čítače/časovače) podporované přerušovacím systémem
- Zpožděním vytvořeným dobou trvání programu viz. obrázek.

Zpoždění realizujeme pomocí podprogramu, v kterém nastavíme hodnotu odpovídající požadovanému zpoždění. Ta je postupně dekrementována/ inkrementována za pomoci instrukcí. Stabilita programovém řešení zpoždění je dána stabilitou synchronizačního hodinového signálu procesoru za předpokladu, že:

- Instrukce trvají vždy stejně dlouhou dobu
- Procesor nevykonává jinou činnost, než je vlastní generování zpoždění. Např. se neobjeví obsluha přerušování.



PROGRAMOVĚ GENEROVANÉ ZPOŽDĚNÍ V JAZYCE C A ASEMLERU

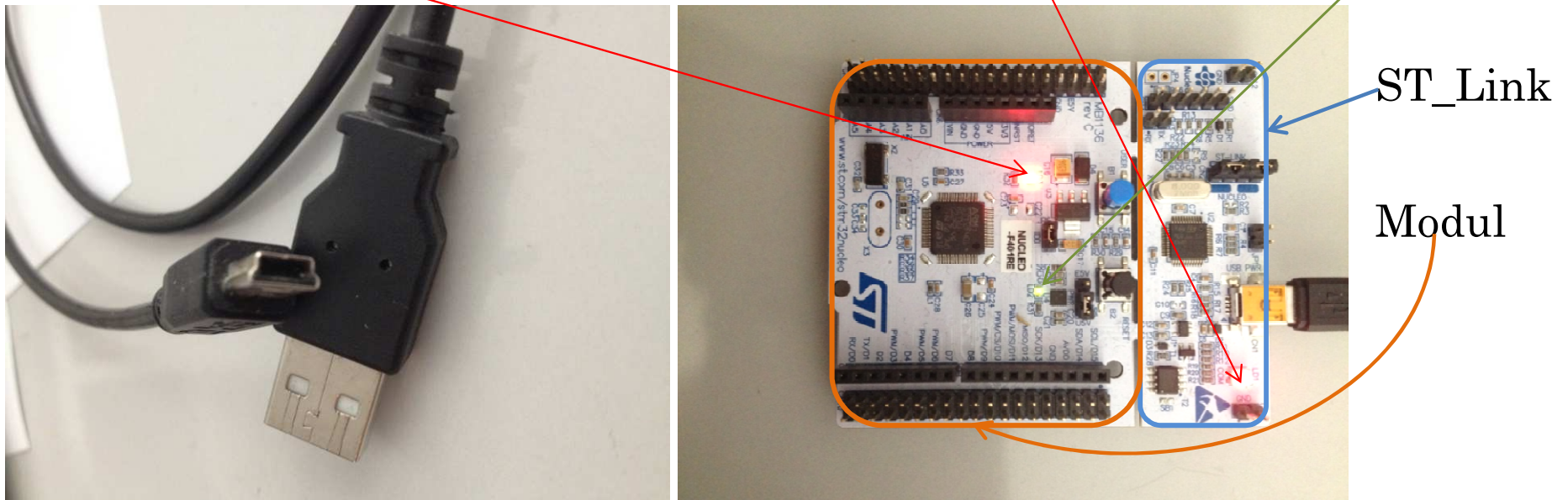
```
const uint32_t doba_zpozdeni = 0x22800; // nutno vyzkouset dobu trvání. Změna hodnoty
// může způsobit v překladu použití jiné instrukce
// a díky tomu dojde ke skokové změně zpoždění.

void Delay (uint32_t pocet_ms)
{
    uint32_t pocet, i;
    for (i = 0; i < pocet_ms; i++)
    {
        for (pocet = 0; pocet < doba_zpozdeni; pocet++) i=i;
    }
}

;*****
;* Jméno funkce          DELAY
;* Popis                Softwarové zpoždění procesoru
;* Mění stav registrů   R0 a R3
;* Vstup                R0 = počet opakování cyklu zpoždění
;* Vystup               Žádný
;* Komentář             Podprogram zpozdí průběh vykonávání programu
;*****
DELAY                ; Navěstí začátku podprogramu
                    PUSH    {LR}    ; Uložení hodnoty návratové adresy - LR do zásobníku
WAIT1
                    LDR     R3, =doba ; Vložení konstanty doba pro prodlevu do R3
WAIT                SUBS    R3, R3, #1 ; Odečtení 1 od R3,tj. R3 = R3 - 1 a nastavení příznakového
                    ; registru
                    BNE    WAIT    ; Skok na navěstí při nenulovosti R3 (skok dle příznaku)
                    SUBS    R0, R0, #1 ; Odečtení 1 od R0,tj. R0 = R0 - 1 a nastavení příznakového
                    ; registru
                    BNE    WAIT1   ; dokud není nula v R0, skočí na wait1
                    POP    {LR}    ; Návrat z podprogramu, obnovení hodnoty LR ze zásobníku
                    BX     LR      ; a návrat do hlavního programu
                    ; Nebo jednodušší varianta POP {PC} místo předchozích dvou řádků
                    POP    {PC}
;*****
                    END          ;Konec programu, jakýkoliv kód za tímto řádkem překladač nepřeloží
```

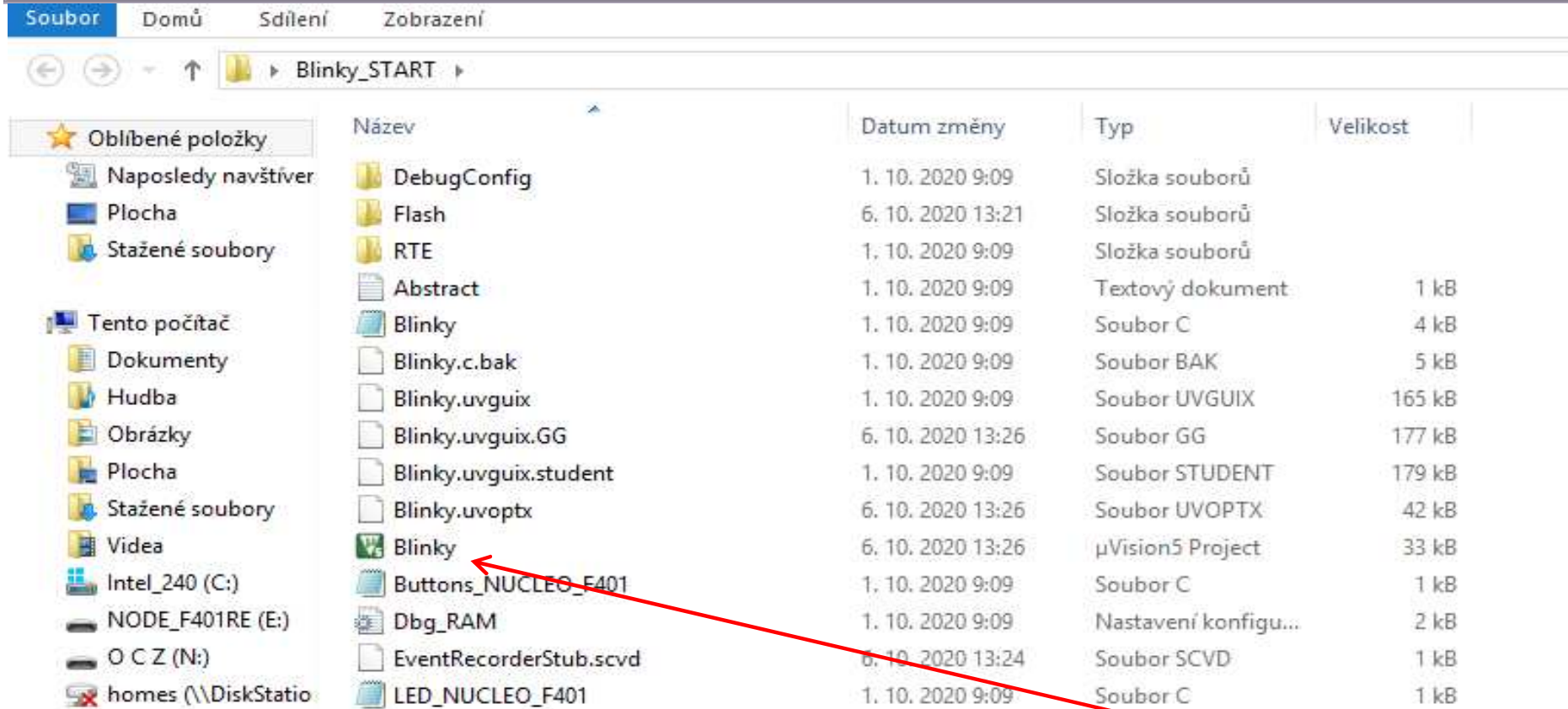
PRVNÍ KROKY K REALIZACI PROGRAMU

- ❖ Nainstalujeme si na PC program Keil uVision5 5.36 a STM32 ST-LINK Utility V3.8.0, který je k dispozici na Moodle.
- ❖ Propojíme kabelem PC s vývojovým modulem. Na modulu se rozsvítí červená dioda a v části ST-Link svítí dioda též červeně viz. obrázek. Zelená dioda může a nemusí svítit, záleží na posledním uloženém programu.



- ❖ Z Moodle si nakopírujeme Blinky_START.zip, rozbalíme direktorář Blinky_START umístíme na disk.
- ❖ Otevřeme direktorář Blinky_START, kde budou umístěny následující programy.

PRVNÍ KROKY K REALIZACI PROGRAMU



- ❖ Vývojové prostředí Keil uVision5 spustíme poklepáním na zelenou ikonku Blinky
- ❖ Pro seznámení se ze systémem a ověření funkčnosti bude potřeba doplnit programy Blinky.c, LED_NUCLEO_F401.c a Buttons_NUCLEO_F401.c. Dříve, než k tomu přistoupíte, zkontrolujeme propojení modulu s vývojovým prostředím.
- ❖ Po spuštění prostředí uvidíme obrazovku

PRVNÍ KROKY K REALIZACI PROGRAMU

The screenshot shows the µVision IDE interface. The top menu bar includes File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, and Help. The toolbar contains various icons for file operations and debugging. The Project pane on the left shows a tree view for 'Project: Blinky' with subfolders for 'Source Files' and 'Documentation'. The Source Files folder contains files like 'Blinky.c', 'Buttons_NUCLEO_F401.c', 'cmsis_armcc.h', 'cmsis_compiler.h', 'cmsis_version.h', 'core_cm4.h', 'LED_NUCLEO_F401.c', 'mpu_armv7.h', 'stdint.h', 'stdio.h', 'stm32f401xe.h', 'stm32f4xx.h', and 'system_stm32f4xx.h'. The main editor window displays the C code for 'Blinky.c', with the 'MAIN function' label highlighted in green. The code includes comments in Czech, such as '// Nekonecna smyčka' and '// Cteni tlacitka'. The Build Output pane at the bottom shows the compilation process, including warnings and errors.

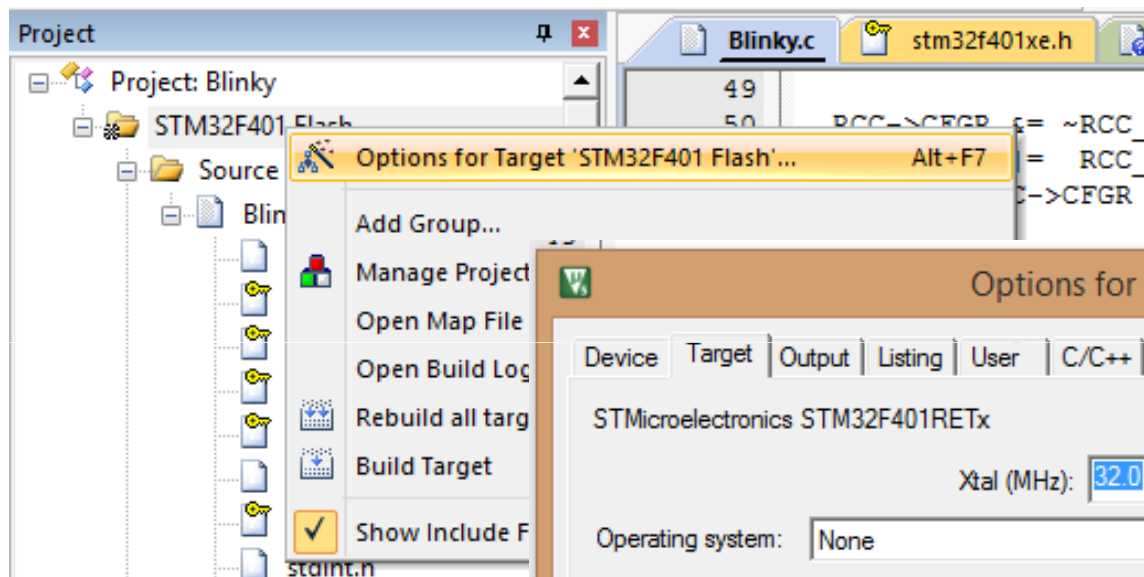
```
49
50 RCC->CFGR &= ~RCC_CFGR_SW; // Select PLL as system clock source
51 RCC->CFGR |= RCC_CFGR_SW_PLL;
52 while ((RCC->CFGR & RCC_CFGR_SWS) != RCC_CFGR_SWS_PLL); // Wait till PLL is system clock src
53 }
54
55 unsigned int output;
56 /*-----*/
57 MAIN function
58 /*-----*/
59 int main (void)
60 {
61     int32_t num = 5;
62     int32_t btns = 0;
63
64     SystemCoreClockSetHSI();
65     SystemCoreClockUpdate(); // Get Core Clock Frequency
66
67     LED_Initialize();
68     Buttons_Initialize();
69
70     while(1) // Nekonecna smyčka // Cteni tlacitka
71     {
72         btns = Buttons_GetState();
73         LED_On (num);
74     }
75 }
```

Build Output

```
{ uint32_t pocet, i;
Blinky.c(61): warning: #550-D: variable "btns" was set but never used
int32_t btns = 0;
Blinky.c: 3 warnings, 0 errors
assembling startup_stm32f401xe.s...
compiling system_stm32f4xx.c...
linking...
```

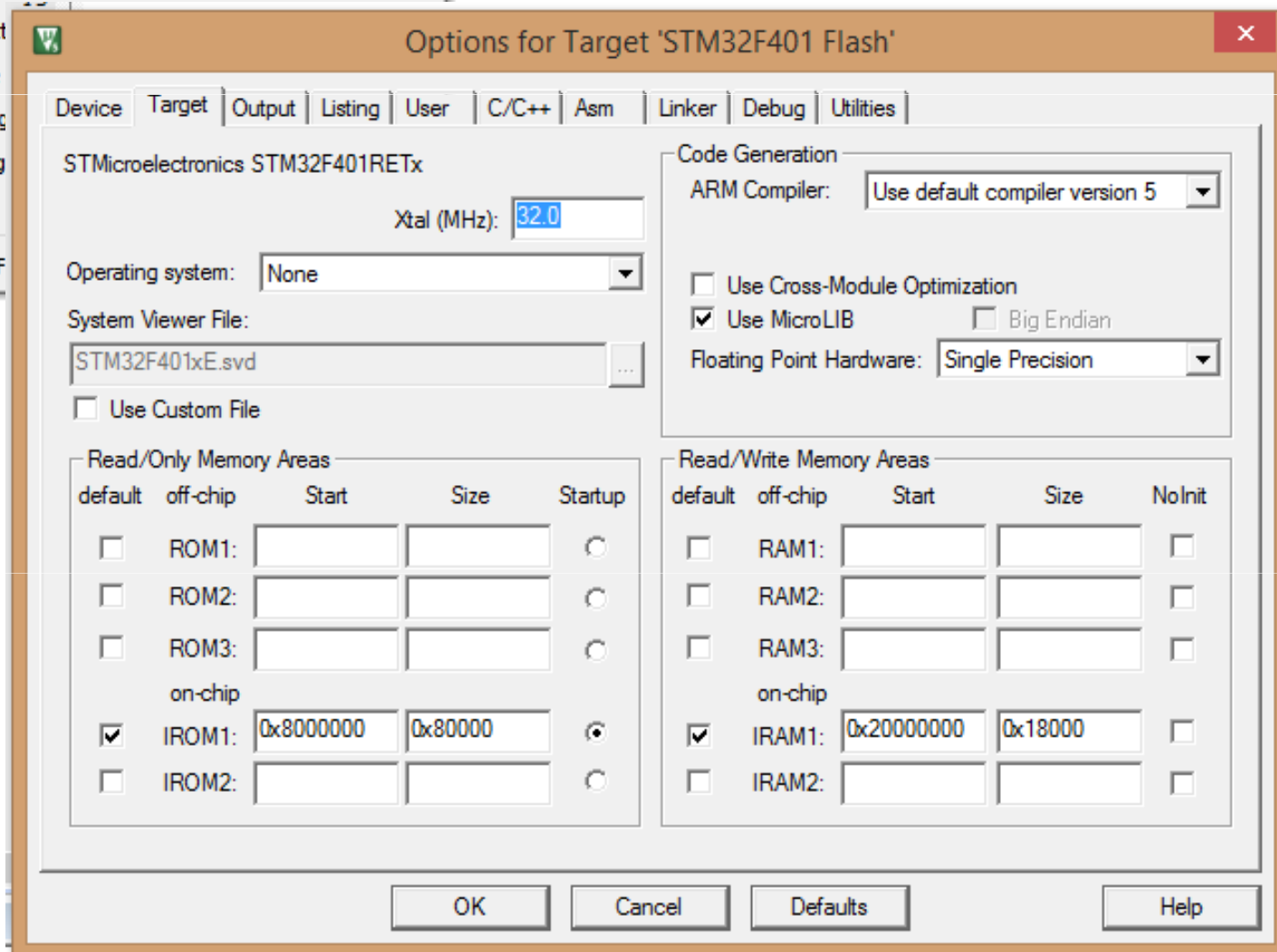
❖ Pravé tlačítko myši na název projektu

NEJDŮLEŽITĚJŠÍ NASTAVENÍ

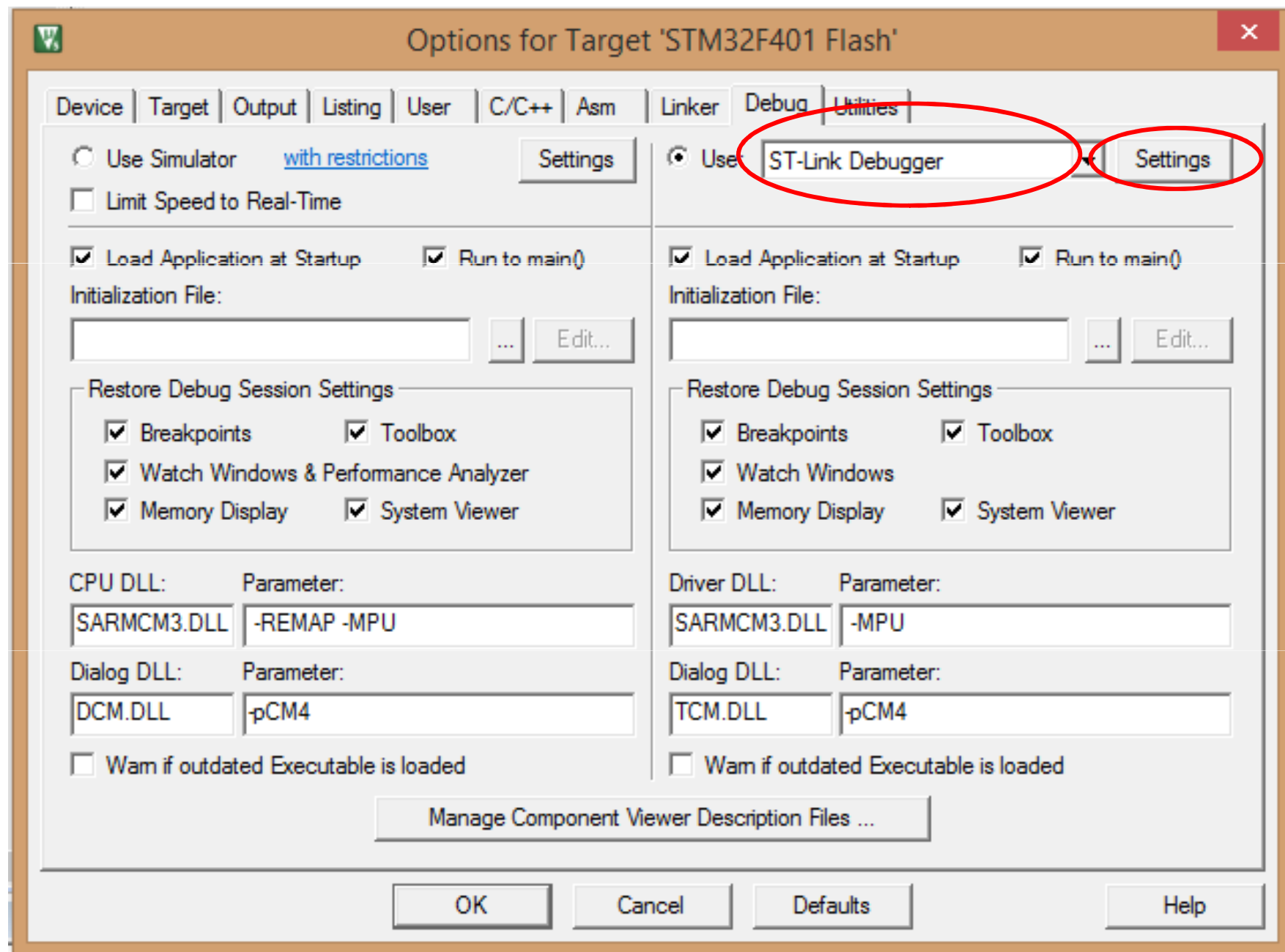


- ❖ Po vybrání položky Option for....
- ❖ Uvidíme okno

- ❖ Device – použitý procesor
- ❖ Target – hodinový kmitočet
- ❖ Debug – další stránka



NEJDŮLEŽITĚJŠÍ NASTAVENÍ



NEJDŮLEŽITĚJŠÍ NASTAVENÍ

Cortex-M Target Driver Setup

Debug | Trace | Flash Download | Pack

Debug Adapter
Unit: **ST-LINK/V2-1**
 Shareable ST-Link

Serial Number:
066EFF575550897767162335

Version: HW: V2-1 FW: V2J29M18
 Check version on start

Target Com
Port: **SW**

Clock
Req: 10 MHz Selected: 0 MHz

SW Device

IDCODE	Device Name	Move
0x2BA01477	ARM CoreSight SW-DP (ARM Core	Up Down

Automatic Detection ID CODE:
 Manual Configuration Device Name:

IR len: AP: 0

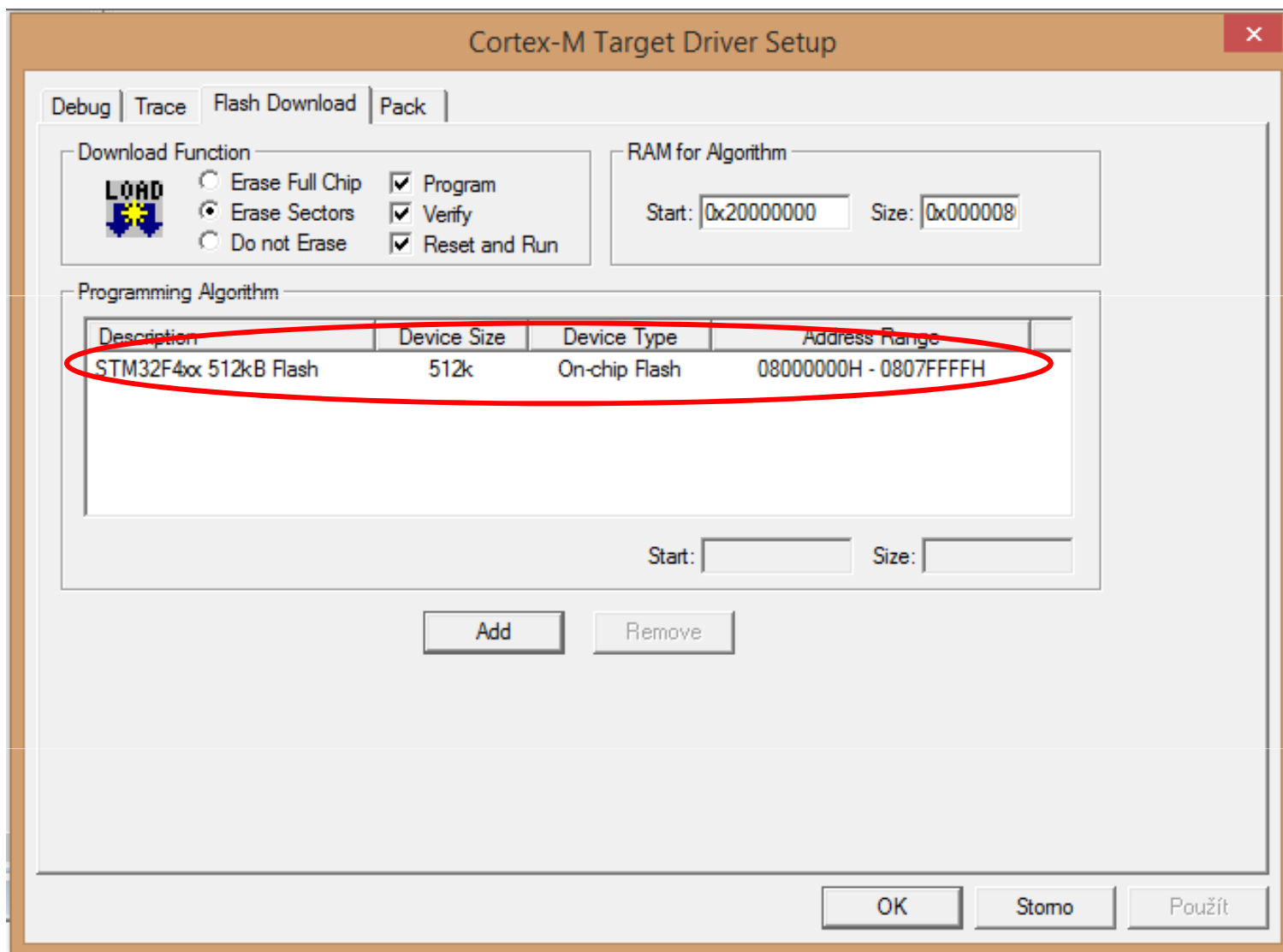
Debug

Connect & Reset Options
Connect: Normal Reset: Autodetect
 Reset after Connect Stop after Reset

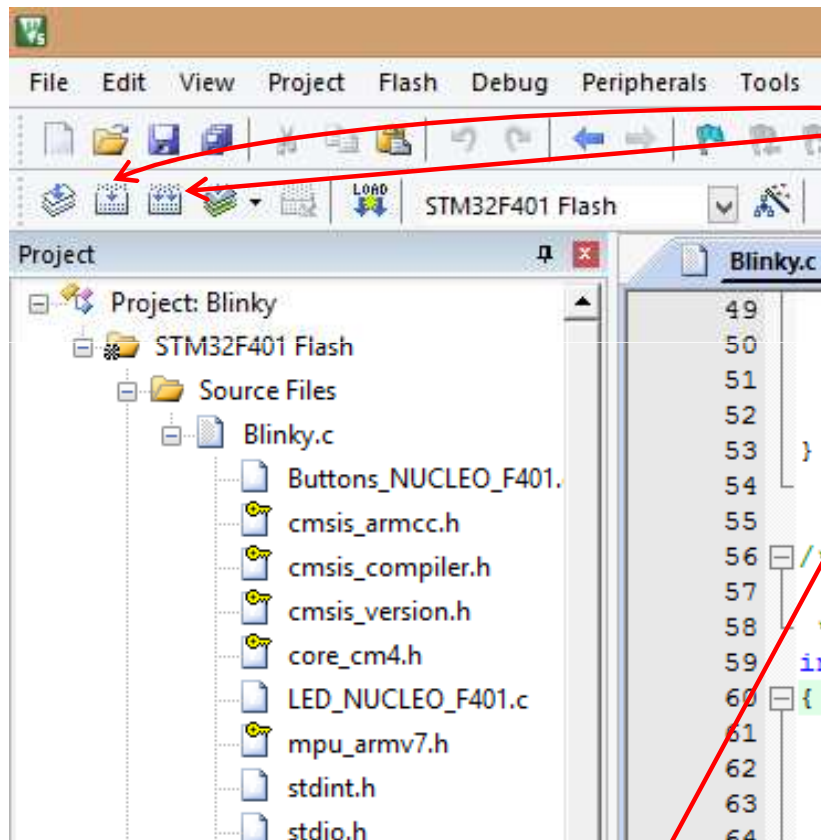
Cache Options
 Cache Code
 Cache Memory

Download Options
 Verify Code Download
 Download to Flash

NEJDŮLEŽITĚJŠÍ NASTAVENÍ



PŘEKLAD PROGRAMU

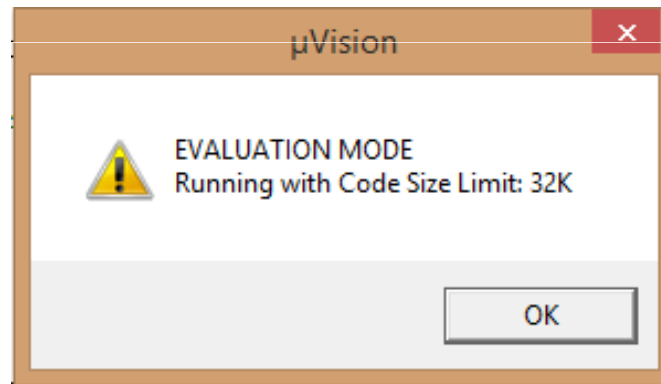


- ❖ Překlad programu zajistíme pomocí ikon
- ❖ Po překladu zkontrolovat zda je bez chyb, varování jsou většinou nevýznamná
- ❖ Code – velikost strojového kódu programu
- ❖ RO-data – velikost konstant v programu
- ❖ RW-data – velikost paměti pro proměnné

```
Build Output
Blinky.c: 3 warnings, 0 errors
assembling startup_stm32f401xe.s...
compiling system_stm32f4xx.c..
linking...
Program Size: Code=694 RO-data=462 RW-data=4 ZI-data=1028
".\Flash\Blinky.axf" - 0 Error(s), 3 Warning(s).
Build Time Elapsed: 00:00:03
<
```

SPUŠTĚNÍ PROGRAMU

- ❖ Spuštění programu zajistíme v záložce Debug – Start/Stop Debug Session nebo pomocí ikonky lupy s písmenem d. Je-li vše v pořádku, pak by se v levém dolním rohu měl na krátkou dobu objevit modrý proužek indikující přenos strojového kódu do vývojového modulu a následující zpráva.



- ❖ Po zmáčknutí OK přecházíme do okna Debug, které je na následující stránce. Pro začátek se spokojíme s následujícími okny:
 - ✓ Okno se zdrojovým programem nebo obsahem zvoleného souboru
 - ✓ Okno **Disassembly** s překladem řádku, na který ukážete ve zdrojovém programu.
 - ✓ Okno **Project** se soubory projektu nebo se stavem registrů **Registers**
 - ✓ Můžeme si aktivovat okno **Memory, Watch** a další.

OBRAZOVKA DEBUG PROSTŘEDÍ

The screenshot displays the µVision IDE interface during a debug session. The top menu bar includes File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, and Help. The Registers window on the left shows the state of various registers, with R15 (PC) at 0x08000294. The Disassembly window shows assembly code with a MOV instruction highlighted. The source code window shows the C code for main() with a breakpoint at line 60. The Watch window shows the variable 'num' with value 0x08000484. The Call Stack window shows the current function 'main' at address 0x00000000.

Register	Value
R0	0x08000295
R1	0x20000408
R2	0x00000000
R3	0x0800043D
R4	0x08000484
R5	0x08000484
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000408
R14 (LR)	0x08000425
R15 (PC)	0x08000294
xPSR	0x61000000

Name	Value	Type
num	0x08000484	int

Name	Location/Value	Type
main	0x00000000	int f()
num	0x08000484	auto - int
btms	<not in scope>	auto - int

Spuštění programu - F5, dioda ST Link – bliká a mění barvu. Krokování - F11 nebo F10. Nastavení hodin nekrokovat přeskočit na Breakpoint.

VOLNĚ POUŽITELNÉ GPIO_x VÝVODY, KTERÉ MŮŽEME KONFIGUROVAT

GPIOA – PA0 až PA12,

PA2 a PA3 - realizují sériový kanál využívaný ST Linkem

PA13, PA14, PA15 – realizují rozhraní JTAG/SWO

zápis do PA2, 3, 13, 14, 15 vede ke ztrátě komunikace s modulem

Odstranění je popsáno na následující stránce

GPIOB – PB0 až PB10, PB12 až PB15

GPIOC – PC0 až PC15

VÝVODY PŘEDURČENÉ PRO ALTERNATIVNÍ FUNKCE

USART2 – PA2, PA3 - nejsou propojeny na konektor

**A/D převodník (skupina A) – PA0, PA1, PA2, PA3, PA4÷7, PB1,
PB12÷15, PC0÷5**

Komparační systém kanál 1 – PA6, PB4, PC6

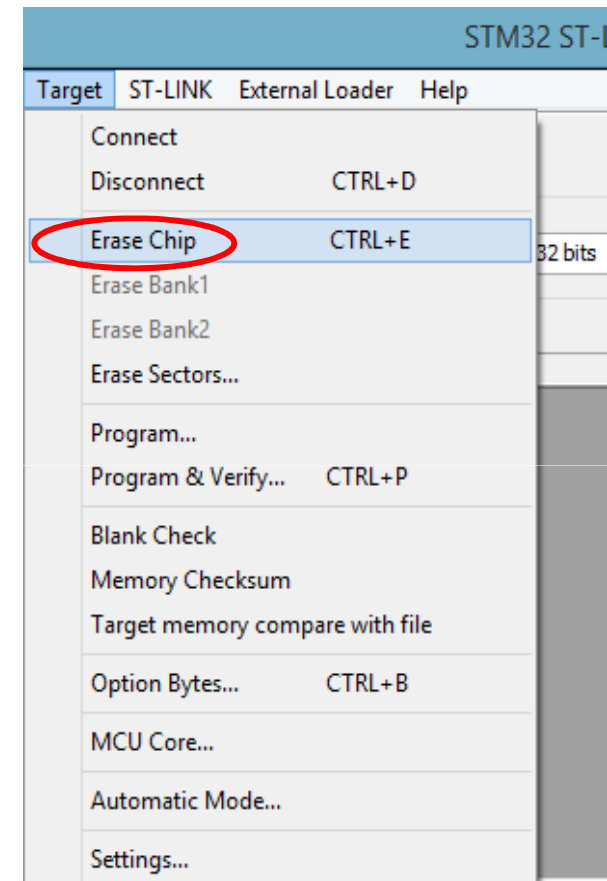
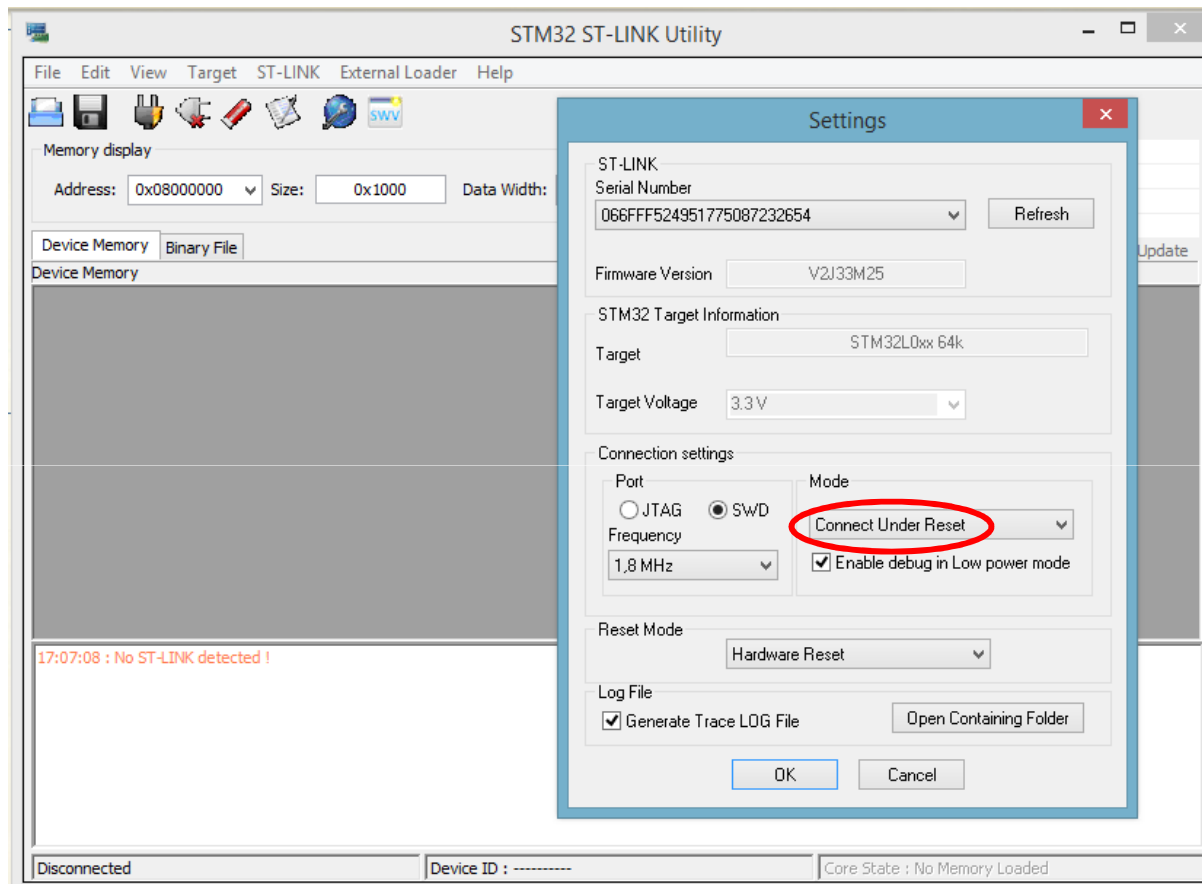
kanál 2 – PA7, PB5, PC7

Záchytný systém kanál 1 – PA6, PB4, PC6

OŽIVENÍ MODULU PO ŠPATNÉM ZÁPISU DO CELÉ BRÁNY PA

Nefungující modul NUCLEO v případech správně nainstalovaného vývojového prostředí Keil i ST-Link. K situaci dochází při zápisu do celé brány PA. **Odstranění:**

- ❖ Spustit ST link
- ❖ V položce *Target-Settings* nastavit variantu Connect Under Reset a OK
- ❖ Erase Chip



OŽIVENÍ MODULU PO ŠPATNÉM ZÁPISU DO BRÁNY PA

Následně by mělo okno ST Link vypadat takto:

The screenshot shows the STM32 ST-LINK Utility window. The title bar reads "STM32 ST-LINK Utility". The menu bar includes File, Edit, View, Target, ST-LINK, External Loader, and Help. The toolbar contains icons for file operations and connection. The "Memory display" section shows "Address: 0x08000000", "Size: 0x1000", and "Data Width: 32 bits". The "Device Memory @ 0x08000000" section is set to "Binary File" and "LiveUpdate" is disabled. The "Target memory, Address range: [0x08000000 0x08001000]" section contains a table with columns for Address, 0, 4, 8, C, and ASCII. The table shows a range of addresses from 0x08000000 to 0x080000B0, with all values in the 0, 4, 8, and C columns being 00000000. The ASCII column shows dots. The bottom status bar displays "Debug in Low Power mode enabled.", "Device ID:0x417", and "Core State : Live Update Disabled".

Address	0	4	8	C	ASCII
0x08000000	00000000	00000000	00000000	00000000
0x08000010	00000000	00000000	00000000	00000000
0x08000020	00000000	00000000	00000000	00000000
0x08000030	00000000	00000000	00000000	00000000
0x08000040	00000000	00000000	00000000	00000000
0x08000050	00000000	00000000	00000000	00000000
0x08000060	00000000	00000000	00000000	00000000
0x08000070	00000000	00000000	00000000	00000000
0x08000080	00000000	00000000	00000000	00000000
0x08000090	00000000	00000000	00000000	00000000
0x080000A0	00000000	00000000	00000000	00000000
0x080000B0	00000000	00000000	00000000	00000000

17:19:50 : Connected via SWD.
17:19:50 : SWD Frequency = 1,8 MHz.
17:19:50 : Connection mode : Connect Under Reset.
17:19:50 : Debug in Low Power mode enabled.
17:19:50 : Device ID:0x417
17:19:50 : Device flash Size : 64KBytes
17:19:50 : Device family :STM32L0xx 64k
17:23:31 : Flash memory erased.
17:23:58 : Flash memory erased.
17:24:22 : Flash memory is blank.

Debug in Low Power mode enabled. Device ID:0x417 Core State : Live Update Disabled