

Osnove mikroprocesorske elektronike

Vaja 4: Projekt "blinky" z registri

Namen te vaje je, da preizkusite, kako je mogoče *upravljati mikrokrmilnik neposredno preko registrov*. Pri tej vaji še vedno ostajamo pri projektu "blinky", le da bomo tokrat utripanje LEDice sprogramirali *brez pomoči funkcij HAL knjižnice*. Tako boste hkrati tudi dobili boljši občutek o tem, kakšno vlogo ima HAL knjižnica pri programiranju mikrokrmilnikov.

Priprava na vajo

1. Obnovite znanje o uporabi bitnih operacij in maskiranja.

To znanje boste namreč potrebovali, ko bo potrebno *manipulirati s posameznimi biti v registrih mikrokrmilnika*.

Pomagajte si z gradivom "[Bitni in logični operatorji v jeziku C](#)". Za rešitev te vaje je ključno gradivo, ki je podano na straneh od 5 do 29.

2. Preučite, katere registre mikrokrmilnika bo potrebno uporabiti, če želimo priključek (tj. pin), kamor je priključena LEDica LED0 uporabiti kot izhod. Preučite tudi, katere bite teh registrov bo potrebno nastavljati ter kakšne morajo biti njihove vrednosti.

Vse potrebne informacije si zabeležite v tabelo, da vam bodo prišle prav pri implementaciji potrebne programske kode na vajah.

| Kaj želimo nastaviti? | Kateri register uporabimo? | Kateri bit je potrebno nastaviti? | Kakšna mora biti vrednost tega bita/bitov? |
|---|----------------------------|-----------------------------------|--|
| omogočiti uro portu GPIOF | | | |
| nastaviti vrsto digitalnega izhoda na "push-pull" izhod | | | |
| onemogočiti "pull-up/pull-down" upore | | | |
| nastaviti hitrost izhoda na "low" | | | |
| nastaviti pin kot digitalni izhod | | | |
| nastaviti stanje dig. izhoda na 1 | | | |
| nastaviti stanje dig. izhoda na 0 | | | |
| invertirati stanje dig. izhoda (angl. toggle) | | | |

Namig: če želite invertirati stanje digitalnega izhoda, morate poznati *trenutno* stanje digitalnega izhoda. Kateri register hrani *trenutno* stanje digitalnega izhoda? In še: ali lahko taisti register uporabite tudi za določitev *novega invertiranega* stanja?

Namig: invertiranje posameznega bita v registru lahko elegantno dosežete z bitno XOR operacijo.

Naloge vaje

1. **Pripravite nov projekt "blinky", kjer pa onemogočite avtomatsko generiranje HAL kode za digitalne vhode/izhode.**

Pri tem "blinky" projektu bomo preprečili, da bi CubeMX avtomatsko generiral programsko kodo za inicializacijo digitalnega izhoda s HAL funkcijo `MX_GPIO_Init()`, saj želimo to kodo implementirati sami.

Namig: v grafičnem vmesniku CubeMX pojdite pod zavihek "Project Manager" in izberite rubriko "Advanced Settings", kjer boste v podoknu "Generated Function Calls" našli možnost za izklop generiranja kode ("Generate code") za funkcijo `MX_GPIO_Init()`. Več v poglavju z namigi.

2. **Definirajte prototipe funkcij**, ki jih nameravamo implementirati za upravljanje LED0:

```
void LED0_GPIO_init(void);  
void LED0_on(void);  
void LED0_off(void);  
void LED0_toggle(void);
```

Prototipe definirajte znotraj `main.c` datoteke znotraj sekcije

```
Private function prototypes
```

3. **Implementirajte funkcijo `LED0_GPIO_init()`**, kjer poskrbite za naslednje stvari:

1. omogočite uro za port GPIOF
2. nastavite začetno stanje digitalnega izhoda za krmiljenje LED0 na nizko stanje (tj. LED0 bo na začetku po inicializaciji ugasnjena)
3. nastavite vrsto digitalnega izhoda na "push-pull" izhod
4. za digitalni izhod onemogočite "pull-up" in "pull-down" upore
5. nastavite hitrost digitalnega izhoda na "nizko" (angl. low)
6. nastavite LED0 priključek kot digitalni izhod

Programsko kodo umestite v sekcijo

```
Private user code
```

Namig: zgornje korake, ki si potrebni za inicializacijo GPIO pina, uporabite kot komentarje znotraj funkcije `LED0_GPIO_init()`. Pod vsakim takim komentarjem nato spišite programsko kodo, ki je nujna za implementacijo posameznega koraka. Tako hitro zagotovite nujno potrebno dokumentacijo vaše programske kode, hkrati pa imate vodilo pri programiranju.

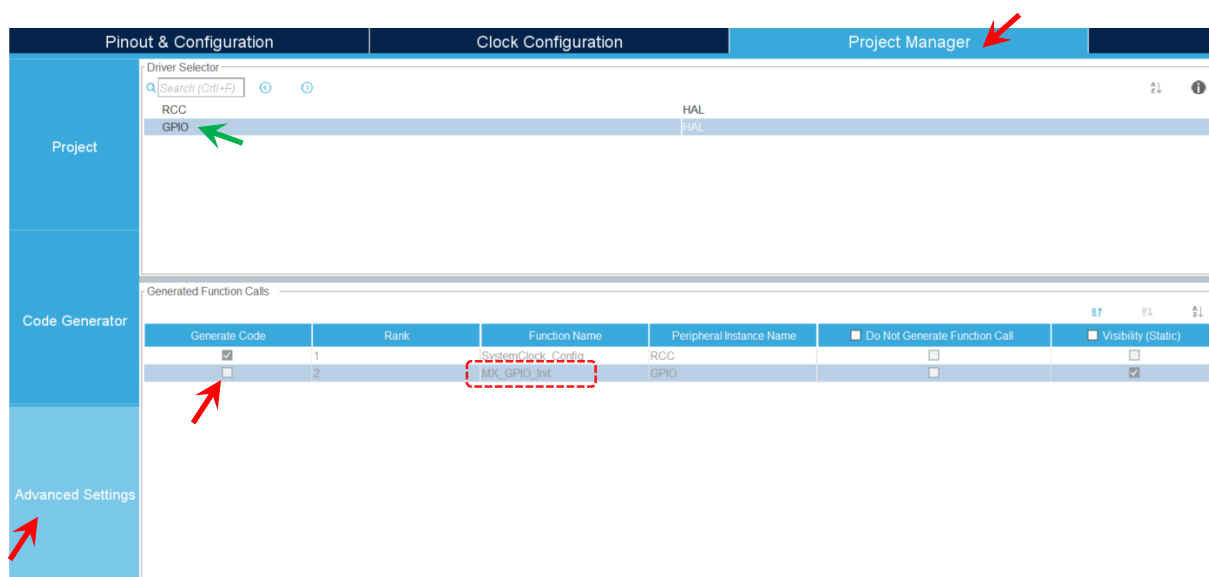
Implementirajte še funkcije, ki bodo *upravljale s stanjem* LEDice LED0.

4. **Implementirajte funkcijo `LED0_on()`**, kjer poskrbite za *vklop* LEDice LED0.
5. **Implementirajte funkcijo `LED0_off()`**, kjer poskrbite za *izklop* LEDice LED0.
6. **Implementirajte funkcijo `LED0_toggle()`**, kjer poskrbite *spremembo stanja* LEDice LED0.

Namigi

Izklop avtomatskega generiranja programske kode s HAL knjižnico

V grafičnem vmesniku CubeMX pojdite pod zavihek "Project Manager" in izberite rubriko "Advanced Settings", kjer boste v podoknu "Generated Function Calls" našli možnost za izklop generiranja kode ("Generate code") za funkcijo `MX_GPIO_Init()`. Poskrbite še, da imate izbrani GPIO modul (zelena puščica).



Slika 1 - podrobnejši napotki za izklopitev avtomatskega generiranja kode za funkcijo `MX_GPIO_Init()`

Opis ključnih korakov procedure naj postane komentar, ki vas vodi

Zelo ugodno je, če proceduro, ki jo programirate, na kratko povzamete s kratkimi opisi ključnih korakov. Opisi teh korakov nato lahko postanejo komentarji, ki vas vodijo pri implementaciji celotne procedure. Poglejte idejo spodaj. Tako dobite komentarje, ki vas po eni strani vodijo pri implementaciji, po drugi strani pa služijo kot opis in dokumentacija vaše procedure.

The screenshot displays a video player showing a C code snippet for configuring the clock on an STM32F4 microcontroller. The code is titled "STEP 4 : Configure PRESCALARS". The code includes comments for each step of the configuration process. Red arrows point from the code to text annotations on the right side of the video player.

```
5
6
7 void SysClockConfig (void)
8 {
9     /***** STEPS FOLLOWED *****/
10
11     1. ENABLE HSE and wait for the HSE to become Ready
12     2. Set the POWER ENABLE CLOCK and VOLTAGE REGULATOR
13     3. Configure the FLASH PREFETCH and the LATENCY Related Settings
14     4. Configure the PRESCALARS HCLK, PCLK1, PCLK2
15     5. Configure the MAIN PLL
16     6. Enable the PLL and wait for it to become ready
17     7. Select the Clock Source and wait for it to be set
18
19     *****/
20
21     // 1. ENABLE HSE and wait for the HSE to become Ready
22     RCC->CR |= RCC_CR_HSEON;
23     while (!(RCC->CR & RCC_CR_HSERDY));
24
25     // 2. Set the POWER ENABLE CLOCK and VOLTAGE REGULATOR
26     RCC->APB1ENR |= RCC_APB1ENR_PWREN;
27     PWR->CR |= PWR_CR_VOS;
28
29     // 3. Configure the FLASH PREFETCH and the LATENCY Related Settings
30     FLASH->ACR = FLASH_ACR_ICEN | FLASH_ACR_DCEN | FLASH_ACR_PRFTEN | FLASH_ACR_LATENCY_SWS;
31
32     // 4. Configure the PRESCALARS HCLK, PCLK1, PCLK2
33     // AHB PR
34     RCC->CFGR |= RCC_CFGR_HPRE_DIV1;
35
36     // APB1 PR
37     RCC->CFGR |= RCC_CFGR_PPRE1_DIV4;
38
39     // APB2 PR
40     RCC->CFGR |= RCC_CFGR_PPRE2_DIV2;
41
42
43
44 int main (void)
45 {
46
```

STEP 4 : Configure PRESCALARS

potrebni koraki za inicializacijo periferije (povzetek procedure)

opis koraka procedure

potrebna koda za implementacijo tega koraka

#1. Intro to STM32F4 Register Based Programming || Clock Setup || LED Blinking || NO HAL

Slika 2 - povzetke ključnih korakov pri inicializaciji uporabite kot komentarje, pod katerimi nato implementirate potrebno programsko kodo (vir: [youtube](#))

Dodatna pojasnila

Še vedno si pomagamo z ARM knjižnicami

S tem, ko smo izklopili avtomatsko generiranje kode za funkcijo `MX_GPIO_Init()`, smo se odločili, da za delo z digitalnimi vhodi in izhodni ne bomo uporabili funkcij iz *HAL knjižnice*. Vendar pa to ne pomeni, da si pri programiranju ne bomo pomagali z že obstoječimi knjižnicami!

Čeprav bomo digitalni izhod *upravljali neposredno preko registrov mikrokrmilnika*, nam bodo pri tem še vedno pomagale *CMSIS knjižnice za delo z mikrokrmilnikom STM32G474QE*. Namreč, poskrbele bodo, da bomo lahko zelo elegantno dostopali do registrov, ki upravljajo z GPIO periferijo. Za osvežitev koncepta si pogledajte sliko s konzultacij spodaj.

OSNOVE MIKROPROCESORSKE ELEKTRONIKE
44

Konzultacije arhitekture 2021/2022

Primer STM32 CMSIS

- > Binaries
- > Includes
- > Core
- > Drivers
 - > CMSIS
 - > Device
 - > STM32G4xx
 - > Include
 - stm32g474xx.h
 - stm32g4xx.h
 - system_stm32g4xx.h
 - Source
 - Templates
 - License.md
 - LICENSE.txt
 - > Include
 - cmsis_armcc.h
- > Core
 - > Inc
 - > Src
 - adc.c
 - ColorSpaces.c
 - crc.c
 - dac.c
 - dma.c
 - fdcan.c
 - fmc.c
 - gpio.c
 - i2c.c
 - joystick.c
 - main.c
 - misc.c

```

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 */
    __IO uint32_t BRR;           /*!< GPIO Bit Reset register,      Address offset: 0x28  */
} GPIO_TypeDef;

#define PERIPH_BASE      (0x40000000UL) /*!< Peripheral base address */
#define GPIOC_BASE      (AHB2PERIPH_BASE + 0x08000UL)
#define AHB2PERIPH_BASE (PERIPH_BASE + 0x08000000UL)

#include "stm32g474xx.h"

...
GPIOC->BSRR = 0x00000001;
    
```

Slika 3 –CMSIS knjižnice za delo z mikrokrmilnikom STM32G474QE bodo poskrbele za elegantno dostopanje do registrov mikrokrmilnika v smislu dela s strukturami.