Časovniki v STM32F4 (Cortex-M)

Na tokratni vaji se bomo spoznali s časovniki, ki se uporabljajo za merjenje časa, izvajanje opravil v natančnih intervalih, programske zakasnitve, generiranje in prepoznavanje signalov s pulzno-dolžinsko modulacijo (PWM) ter še mnogo ostalih pogostih opravil povezanih s časom.

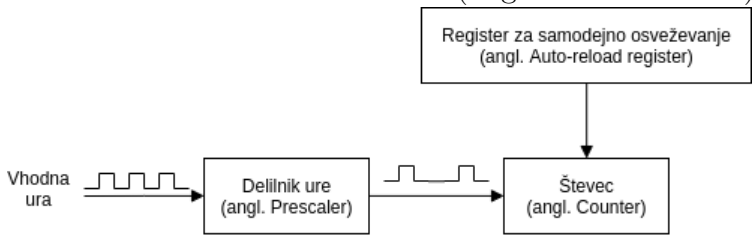
V splošnem imajo mikrokrmilniki STM32 tri različne tipe časovnikov: osnovne (angl. basic timers), splošno-namenske (angl. general-purpose timers) ter napredne časovnike (angl. advanced timers). Na tokratni vaji se bomo spoznali s principi delovanja splošno-namenskih časovnikov.

Mikrokrmilnik STM32F407, ki ga uporabljamo na vajah, ima na voljo 14 časovnikov, ki so označeni s predpono TIM ter številsko oznako. Časovnika TIM6 in TIM7 sta osnovnega tipa, časovnika TIM1 in TIM8 spadata med napredne časovnike, preostale (TIM2-TIM5, TIM9-TIM14) pa štejemo med splošno-namenske časovnike. Slednji se sicer med sabo še dodatno razlikujejo, vendar le v podrobnostih kot je velikost števcev, število kanalov in podobno. Koncept delovanja, ki je predstavljen v nadaljevanju, pa je enak za vse.

Osnovno delovanje časovnika

Osnovni blok splošno-namenskih časovniko (angl. time-base unit) je prikazan na sliki 1. Glavna komponenta vsakega časovnika je števec, ki zna ob pozitivni urini fronti na vhodu prištevati ali odštevati stanje števca. Števec je v večini časovnikov 16-biten, razen pri TIM2 in TIM5, kjer je 32-biten. Frekvenca s katero šteje števec je določena z vhodno uro ter vrednostjo v delilniku ure (angl. prescaler). Slednji namreč deli frekvenco vhodne ure pred vhodom v števec ter števec tako upočasni.

Slika 1: Osnovni blok časovnika (angl. time-base unit)

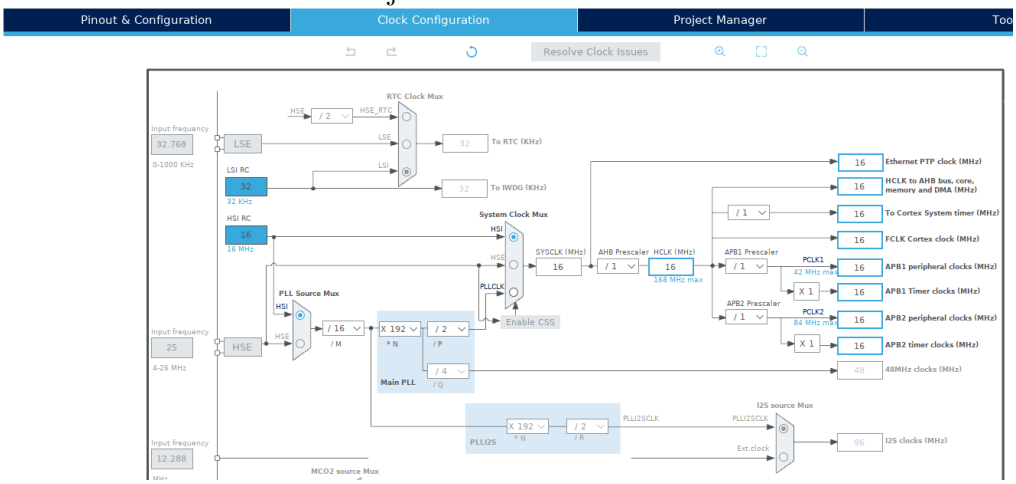


Vhodna ura časovnika je običajno ura vodila na katerem je priklopljen časovnik, lahko pa uporabimo tudi zunanjo ura, ki jo priklopimo na GPIO

pin. V nadaljevanju predpostavljamo, da je vhodna ura časovnika ura vodila. Vsi časovniki so povezani na vodili APB1 in APB2. Na slednjem, na katerega so povezani časovniki TIM1, TIM8, TIM9 in TIM12, je najvišja možna frekvenca 168 MHz, kar je tudi glavna frekvenca mikrokrmilnika. Na vodilu APB1, na katerega so povezani preostali časovniki, pa je najvišja možna frekvenca polovica glavne frekvence mikrokrmilnika (84 MHz).

V praznem projekt, kreiranem v STM32CubeIDE, sta obe omenjeni frekvenci nastavljeni na 16MHz. Spreminjate ju z dvoklikom na `.ioc` datoteko vašega projekta. V odprtem oknu nato izberete `Clock Configuration`. Ta vam odpre diagram prikazan na sliki 2. S spreminjamnjem vrednosti `APB1 Timer clocks` ter `APB1 Timer clocks` lahko spremenite frekvence vhodne ure časovnikov. Primeri v nadaljevanju predpostavljajo, da je vhodna frekvenca časovnika 16 MHz.

Slika 2: Določanje frekvenc vhodnih ur časovnikov.



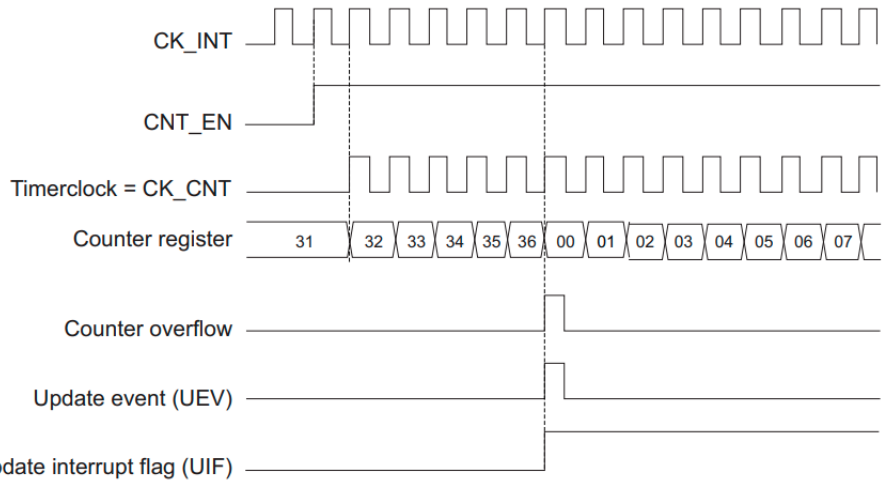
Vhodna ura časovnika je, kot je vidno na sliki 1 vhod v delilnik ure (angl. prescaler). Ta deli vhodno frekvenco z vrednostmi od 1 do 65536 (največje predstavljivo nepredznačeno 16-bitno število + 1). Če je delilnik ure nastavljen na vrednost 0, to pomeni, da se ura deli z 1, torej je frekvenca štetja enaka uri vodila. Če je delilnik ure nastavljen na vrednost 1, je frekvenca štetja polovica vhodne frekvence, in tako naprej do 65536, ki pomeni štetje s frekvenco $f = f_{vhod}/65537$. V splošnem je torej frekvenca enaka $f = f_{vhod}/(n+1)$, kjer n predstavlja vrednost delilnika ure.

Števec lahko prišteva oziroma šteje navzgor (angl. upcounting mode),

odšteva oziroma šteje navzdol (angl. downcounting mode) ali pa šteje navzgor in navzdol (angl. up/down counting ali center-aligned mode). V primeru štetja navzgor števec šteje od 0 do vključno vrednosti, ki je definirana v registru za samodejno osveževanje (angl. auto-reload register - ARR). Nato začne ponovno šteti od 0. V primeru štetja navzdol je začetno stanje števca enako ARR registru, ko doseže vrednost 0 pa začne ponovno šteti pri vrednosti definirani v registru ARR. V primeru štetja navzgor/navzdol števec šteje od 0 do ARR in nato nazaj od ARR do 0.

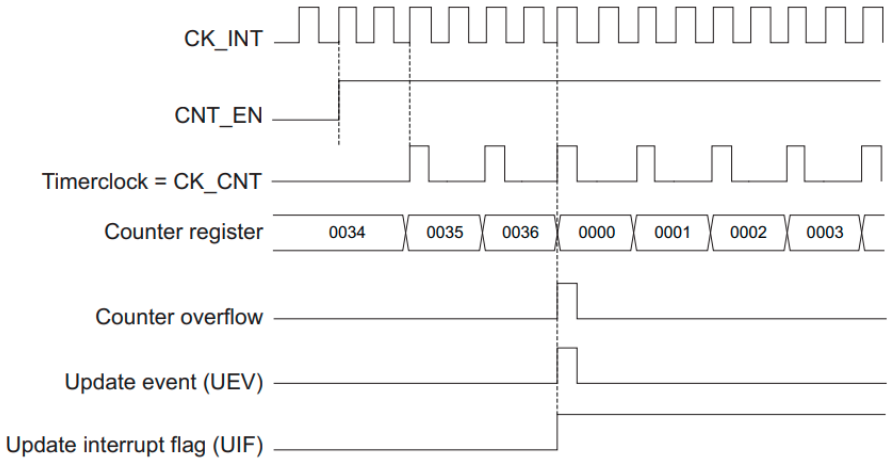
Sliki 3 in 4 prikazujeta dva diagrama štetja navzgor pri različnih vrednosti v delilniku ure (0 in 1). V obeh primerih je vrednost registra za samodejno osveževanje (ARR registra) 36. `CK_INT` predstavlja vhodno uro časovnika, `CK_CNT` predstavlja uro štetja, `Counter register` pa predstavlja stanje števca. Na diagramih lahko vidite tudi, da se ob ponovnem štetju z 0 pojavi tako imenovani update dogodek (angl. update event). Ob update dogodku se postavi zastavica (angl. flag), če želimo ob tem dogodku lahko prožimo tudi prekinitev. Prav tako se ob temu dogodku osveži nastavitev za ARR register. Če v program med izvajanjem želimo spremeniti vrednost ARR registra, se bo ta sprememba zgodila šele ob naslednjem update dogodku in ne takoj. Podobno velja za spremembe količnika za deljenje ure.

Slika 3: Štetje navzgor brez deljenja ure in ARR = 36.



Do sedaj opisane funkcionalnosti so vse kar ponujata osnovna časovnika TIM6 in TIM7. V nadaljevanju si poglejmo še dodatne funkcionalnosti, ki jih omogočajo splošno-namenski časovniki.

Slika 4: Štetje navzgor s polovično frekvenco in ARR = 36.



Kanali za zajem in primerjavo (angl. capture-/compare channels)

Vsi časovniki razen TIM6 in TIM7 imajo v časovniku na voljo kanale, ki omogočajo dodatne funkcionalnosti nad vhodnimi signali ter avtomatsko generiranje izhodnih signalov. Časovniki TIM1, TIM2, TIM3, TIM4, TIM5 in TIM8 imajo po 4 takšne ločene kanale, časovnika TIM9 in TIM12 imata oba po 2 kanala, preostali splošno-namenski časovniki pa po enega.

Na tokratni vaji se bomo osredotočili predvsem na izhodni del kanalov. Med vhodne funkcionalnosti, ki jih na tej vaji ne bomo spoznali, med drugim spadajo merjenje časa med dvema prehodoma na vhodni liniji, prepoznavanje frekvence ter dolžine pulza v signalu generiranem s pulzno-širinsko modulacijo (PWM).

Izhodni primerjalni način delovanja (angl. output compare mode)

Osnovna funkcionalnosti časovnika, ki se ukvarja z izhodom, se imenuje output compare mode. Vsak kanal za zajem in primerjavo (angl. capture-/compare channel) ima svoj register za primerjavo (angl. compare register). Ko števec časovnika doseže vrednost, ki je zapisana v tem registru, se kanal lahko odzove. Privzeti odziv je postavljanje zastavic CC1, CC2, CC3 in CC4

– vsaka zastavica pripada enemu izmed štirih možnih kanalov. Ob postavljanju omenjenih zastavic se lahko proži tudi prekinitev ali neposredno nastavi stanje poljubnega GPIO pina. Vsak output compare kanal ima namreč fiksno določen GPIO pin, ki ga lahko ob odzivu kanala vsakič nastavi na logično vrednost 0, logično vrednost 1 ali pa negira trenutno logično vrednost na tem pinu (angl. toggle). V uporabniškem priročniku je jasno zapisano, kateri GPIO pin pripada kateremu izmed kanalov časovnikov. Na sliki 5 lahko pri pinu PD12 vidite, da ima zapisano oznako `TIM4_CH1`. To pomeni, da je ta pin povezan na prvi kanal TIM4. Preostali trije kanali istega časovnika so, kot lahko vidite, povezani na pine PD13, PD14 in PD15. Na naši razvojni plošči so ti pini, kot bi sedaj že morali vedeti, povezani na LED diode na razvojni plošči.

Slika 5: GPIO pini kanalov časovnika.

PD12	FSMC_A17/ TIM4_CH1/ USART3_RTS
PD13	FSMC_A18/ TIM4_CH2
PD14	FSMC_D0/ TIM4_CH3
PD15	FSMC_D1/ TIM4_CH4
PE0	TIM4_ETR/ FSMC_NBL0/ DCMI_D2
PE1	FSMC_NBL1/ DCMI_D3
PE2	TRACECLK/ FSMC_A23/ ETH_MII_TXD3
PE3	TRACED0/ FSMC_A19
PE4	TRACED1/ FSMC_A20/ DCMI_D4
PE5	TRACED2/ FSMC_A21/ TIM9_CH1/ DCMI_D6
PE6	TRACED3/ FSMC_A22/ TIM9_CH2/ DCMI_D7

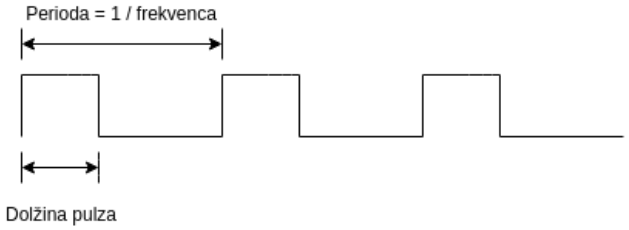
Odziv na kanal je lahko tudi proženje akcije druge naprave. Praktičen primer uporabe te funkcije je procesiranje zvočnih signalov, saj lahko s to funkcijo dosežemo vzorčenje vhodnega signala ob zelo natančnih intervalih.

Generiranje signala s pulzno-širinsko modulacijo

Pulzno-širinska modulacija (angl. pulse-width modulation), ali na kratko PWM, je način kodiranja informacije z dolžino (širino) pulza. PWM si-

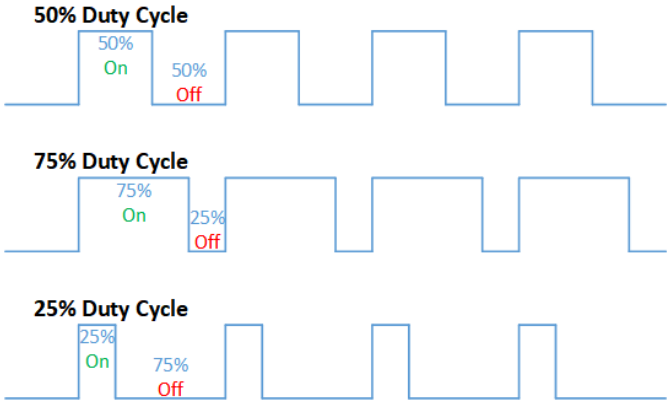
gnali imajo konstanto frekvenco in spreminjajočo dolžino pulza (glej sliko 6). Vsaka dolžina pulza ima lahko poseben pomen.

Slika 6: Duty cycle PWM signala.



Frekvenca in dolžina pulza sta edini lastnosti, ki jih moramo določiti PWM signalom. Dolžino pulza pogosto opišemo s tako imenovanim "duty cycleom", ki je zapisan v odstotkih in pomeni kolikšen delež periode je signal aktiven. Če je signal celotno periodo neaktiven, je duty cycle 0%. Petdesetodstotni duty cycle pa pomeni, da je signal aktiven polovico periode. Trije primeri so prikazani na sliki 7.

Slika 7: Duty cycle PWM signala.



PWM signali se pogosto uporabljajo za krmiljenje servo motorjev. Servo motorji na primer zahtevajo PMW signal s frekvenco 50 Hz, kjer pulz dolžine 1 ms pomeni, da je motor obrnjen v eno smer, 2 ms pa pomeni, da je motor obrnjen v drugo smer.

Drug primer uporabe PWM signalov je zmanjšanje svetilnosti LED diod (angl. dimming). Pri stoodstotnem duty cyclu bo LED dioda svetila s polno svetilnostjo. Če pa duty cycle zmanjšamo, bo to za naše oko vidno kot da LED dioda sveti z zmanjšano svetilnostjo.

Knjižnica HAL in časovniki za STM32F4

Pred inicializacijo časovnika moramo tako, kot vedno do sedaj, prižgati uro periferne naprave. Za časovnike to storimo z `__HAL_RCC_TIMx_CLK_ENABLE()`.
Pred tem v datoteki `Inc/stm32f4xx_hal_conf.h` odkomentirajte `HAL_TIM_MODULE_ENABLED`.

Inicializacija osnovnega bloka

Za inicializacijo osnovnega bloka časovnika določimo smer štetja, vrednost količnika delilnika ure ("prescalerja") ter periodo štetja (vrednost v ARR registru). V strukturi v kateri določimo nastavitve, izberemo tudi časovnik. Če želimo uporabiti zgolj osnovni blok časovnika (time-base unit) inicializiramo časovnik s funkcijo `HAL_TIM_Base_Init()`. Če želimo uporabiti časovnik za output compare, inicializiramo z `HAL_TIM_OC_Init()`. `HAL_TIM_PWM_Init()` uporabimo, če hočemo časovnik uporabiti za generiranje PWM signalov. Primer inicializacije je prikazan spodaj:

```
1  __HAL_RCC_TIM3_CLK_ENABLE();
2
3  TIM_HandleTypeDef timer;
4  timer.Instance = TIM3;
5  timer.Init.CounterMode = TIM_COUNTERMODE_UP;
6  timer.Init.Period = 1000-1; // ARR
7  timer.Init.Prescaler = 16000-1;
8  HAL_TIM_Base_Init(&timer);
9  // ali HAL_TIM_OC_Init(&timer);
10 // ali HAL_TIM_PWM_Init(&timer);
```

Prescaler je v tem primeru nastavljen na 16000-1. To pomeni, da je frekvenca štetja 1000 Hz. `Period` določa vrednost ARR registra. S temi nastavitvami se bo update dogodek pripetil vsako sekundo. Za nastavljene smeri štetja je možen nabor vrednosti:

- `TIM_COUNTERMODE_UP`

- `TIM_COUNTERMODE_DOWN`
- `TIM_COUNTERMODE_CENTERALIGNED1`
- `TIM_COUNTERMODE_CENTERALIGNED2`
- `TIM_COUNTERMODE_CENTERALIGNED3`

Inicializacija output compare kanala

Za osnovno delovanje output compare kanala zadostuje, da mu nastavimo vrednost compare registra. Primer takšne inicializacije je prikazan spodaj. `Pulse` je v tem primeru vrednost compare registra. Po spodnji inicializaciji in zagonu časovnika bi lahko brali stanje zastavice CC1 ali pa dodali, da se ob postavljanju zastavice CC1 proži prekinitev.

```
1  TIM_OC_InitTypeDef OC_channel;
2  OC_channel.Pulse = 500;
3  HAL_TIM_OC_ConfigChannel(&timer, &OC_channel, TIM_CHANNEL_1);
```

Če želimo, da ob odzivu output compare kanal spremeni tudi stanje na pripadajočem izhodnem GPIO pinu moramo določiti še nastavitev `OCMode`, kjer lahko nastavimo, da kanal izhodni pin postavi na aktivno vrednost (`TIM_OCMODE_ACTIVE`), neaktivno logično vrednost (`TIM_OCMODE_INACTIVE`) ali pa stanje pina invertira (`TIM_OCMODE_TOGGLE`). Z nastavitvijo `OCPolarity` moramo še določiti ali je aktivna logična vrednost ničla `TIM_OCPOLARITY_LOW` ali enica `TIM_OCPOLARITY_HIGH`. Običajno je slednja nastavitev tista, ki smo je bolj vajeni. Primer obeh zgoraj opisanih opcij je prikazan spodaj. Poleg spodnje kode je potrebno ustrezno inicializirati tudi GPIO pin, ki pripada izbranem output compare kanalu časovnika. Več o nastavitvi GPIO pina je zapisano v nadaljevanju.

```
1  TIM_OC_InitTypeDef OC_channel;
2  OC_channel.OCMode = TIM_OCMODE_TOGGLE;
3  OC_channel.OCPolarity = TIM_OCPOLARITY_HIGH;
4  OC_channel.Pulse = 500;
5  HAL_TIM_OC_ConfigChannel(&timer, &OC_channel, TIM_CHANNEL_1);
```

Inicializacija PWM izhoda

Frekvenco PWM signala določimo z nastavitvijo količnika v delilniku vhodne ure ter nastavitvi ARR registra. To torej določimo ob inicializaciji osnovnega bloka. Frekvenca PWM signala je enaka frekvenci update dogodkov. Dolžino pulza pa določimo v compare kanalu z nastavitvijo compare registra. Če je njegova vrednost 0 gre za 0% duty cycle. 100% duty cycle dobimo, če compare register nastavimo na vrednost registra ARR. Kot pri inicializaciji output compare kanala tudi tu določimo katero logično vrednost ima signal, ko je aktiven. Primer inicializacije PWM izhoda je prikazan v primeru spodaj. Poleg spodnje inicializacije je zopet potrebno inicializirati tudi GPIO pin.

```
1  TIM_OC_InitTypeDef PWM_channel;
2  PWM_channel.OCMode = TIM_OCMODE_PWM1;
3  PWM_channel.OCPolarity = TIM_OCPOLARITY_HIGH;
4  PWM_channel.Pulse = 500;
5  HAL_TIM_PWM_ConfigChannel(&timer, &PWM_channel, TIM_CHANNEL_1)
     ;
```

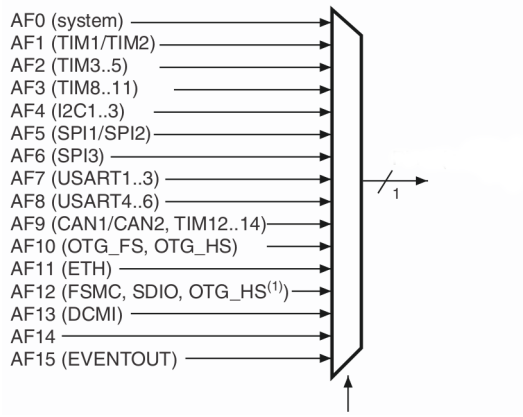
Način delovanja `TIM_OCMODE_PWM1` je običajen PWM signal, kot je prikazan na sliki 6, kjer je prvi del periode pulz, temu pa sledi neaktiven del periode. Pri `TIM_OCMODE_PWM2` je pulz na koncu periode.

Inicializacija GPIO

GPIO pine, ki jih uporablja časovnik, moramo inicializirati podobno, kot smo to storili, ko smo jih inicializirali za GPIO. Glavna razlika je v tem, da jim tokrat kot način delovanja moramo določiti način alternativne funkcije (angl. alternate function). To pomeni, da s pini ne bomo upravljali mi (pisali oz. brali njihovega stanja), ampak bo pine uporabljala neka druga naprava, v tem primeru časovnik. Za vsakega izmed pinov lahko uporabimo eno izmed 16 možnih alternativnih funkcij. Slika 8 prikazuje alternativne funkcije v mikrokrmilnikih STM32F407. Kot vidite, moramo v primeru, da uporabljamo TIM4, pri GPIO pinih uporabiti 2. alternativno funkcijo (konstanta `GPIO_AF2_TIM4`). Podobno velja, kadar delamo s časovniki TIM3 ali TIM5. V primeru uporabe

`Drivers/STM32F4xx_HAL_Driver/Inc/stm32f4xx_hal_gpio_ex.h` je datoteka, kjer najdemo seznam nastavitev za alternativne funkcije. Izsek nekaterih pomembnih vrednosti iz datoteke je prikazan spodaj.

Slika 8: Mapiranje alternativnih funkcij.




```
#define GPIO_AF1_TIM1           ((uint8_t)0x01)
#define GPIO_AF1_TIM2           ((uint8_t)0x01)

#define GPIO_AF2_TIM3           ((uint8_t)0x02)
#define GPIO_AF2_TIM4           ((uint8_t)0x02)
#define GPIO_AF2_TIM5           ((uint8_t)0x02)

#define GPIO_AF3_TIM8           ((uint8_t)0x03)
#define GPIO_AF3_TIM9           ((uint8_t)0x03)
#define GPIO_AF3_TIM10           ((uint8_t)0x03)
#define GPIO_AF3_TIM11           ((uint8_t)0x03)
```

Spodnja koda prikazuje primer inicializacije pina PD12 za uporabo v napravi TIM4. Kot vidite je inicializacija podobna kot pri klasicnih GPIO pinih, glavna in pomembna razlika je zgolj v nastavitvi alternative funkcije.

```
__HAL_RCC_GPIOD_CLK_ENABLE();

GPIO_InitTypeDef init_structure;
init_structure.Pin = GPIO_PIN_12;
init_structure.Pull = GPIO_NOPULL;
init_structure.Speed = GPIO_SPEED_FREQ_LOW;

// Mode = alternativna funkcija, obicjano nacin push-pull
// ce potrebujemo open-drain nastavimo GPIO_MODE_AF_OD
init_structure.Mode = GPIO_MODE_AF_PP;

```

```
12 // dolocimo se katera naprava bo upravljala s pinom
13 init_structure.Alternate = GPIO_AF2_TIM4;
14
15 HAL_GPIO_Init(GPIOD, &init_structure);
```

Upravljanje časovnika

Po inicializaciji je potrebno časovnik še zagnati. Spodaj so prikazani primeri zagona časovnika, pri uporabi osnovnega bloka, output compare kanala ter generiranja PWM signala.

```
1  HAL_TIM_Base_Start(&timer);
2  HAL_TIM_OC_Start(&timer, TIM_CHANNEL_1);
3  HAL_TIM_PWM_Start(&timer, TIM_CHANNEL_2);
```

Med delovanjem časovnika lahko spreminjamo določene nastavitve. Najpogosteje spreminjamo vrednost compare registra ali količnika delilnika ure. Seznam funkcij in primeri uporabe so podani spodaj.

```
1  __HAL_TIM_SET_PRESCALER(&timer, 1000);
2  __HAL_TIM_SET_COUNTER(&timer, 1000);
3  __HAL_TIM_SET_AUTORELOAD(&timer, 1000);
4  __HAL_TIM_SET_COMPARE(&timer, TIM_CHANNEL_1, 1000);
```

Zastavice časovnikov

Časovniki imajo množico zastavic, s katerimi lahko spremljate njihovo stanje. Za nas je zanimivih spodnjih 5, ki označujejo update dogodek ter odzive vseh štirih možnih output compare kanalov.

```
1  TIM_FLAG_UPDATE
2  TIM_FLAG_CC1
3  TIM_FLAG_CC2
4  TIM_FLAG_CC3
5  TIM_FLAG_CC4
```

Zastavice lahko preverjamo s funckijo `__HAL_TIM_GET_FLAG()`, ki ji podamo kazalec na strukturo časovnika ter oznako zastavice. Enake argumente ima tudi funckija za brisanje zastavic `__HAL_TIM_CLEAR_FLAG()`.

Prekinitve časovnikov

Časovniku lahko omogočimo, da ob katerikoli izmed zgornjih zastavic proži prekinitev. Prekinitev omogočimo z ukazom `__HAL_TIM_ENABLE_IT()`, ki mu podamo kazalec na strukturo časovnika ter oznako prekinitve. Oznake prekinitev so podobne imenom zastavic:

```
1  TIM_IT_UPDATE
2  TIM_IT_CC1
3  TIM_IT_CC2
4  TIM_IT_CC3
5  TIM_IT_CC4
```

Poleg vklopa prekinitve v časovniku, moramo prekinitve omogočiti tudi v prekinitvenem krmilniku, enako kot pri prejšnji vaj za zunanje prekinitve (EXTI). Primer vklopa prekinitev za časovnik TIM3 je podan spodaj.

```
1  HAL_NVIC_SetPriority(TIM3_IRQn, 0, 1);
2  HAL_NVIC_EnableIRQ(TIM3_IRQn);
```

Spodnja tabela vsebuje seznam vse prekinitvenih oznak ter imen prekinitveno servisnih programov za časovnike. Kot vidite ima vsak od naprednih časovnikov več prekinitvenih oznak, splošno namenski ter osnovna časovnika pa zgolj po eno. Časovniki TIM9-TIM14 si prekinitvene oznake delijo s TIM1 in TIM8. V prekinitveno servisnem programu preverjamo in brišemo prekinitvene zahteve s funkcijami za delo z zastavicami (kot pri prejšnji vaji).

Časovnik	Prekinitvene oznake	Imena PSP-jev
TIM1	TIM1_BRK_TIM9_IRQn, TIM1_UP_TIM10_IRQn, TIM1_TRG_COM_TIM11_IRQn, TIM1_CC_IRQn	TIM1_BRK_TIM9_IRQHandler, TIM1_UP_TIM10_IRQHandler, TIM1_TRG_COM_TIM11_IRQHandler, TIM1_CC_IRQHandler
TIM2	TIM2_IRQn	TIM2_IRQHandler
TIM3	TIM3_IRQn	TIM3_IRQHandler
TIM4	TIM4_IRQn	TIM4_IRQHandler
TIM5	TIM5_IRQn	TIM5_IRQHandler
TIM6	TIM6_DAC_IRQn	TIM6_DAC_IRQHandler
TIM7	TIM7_IRQn	TIM7_IRQHandler
TIM8	TIM8_BRK_TIM12_IRQn, TIM8_UP_TIM13_IRQn, TIM8_TRG_COM_TIM14_IRQn, TIM8_CC_IRQn	TIM8_BRK_TIM12_IRQHandler, TIM8_UP_TIM13_IRQHandler, TIM8_TRG_COM_TIM14_IRQHandler, TIM8_CC_IRQHandler
TIM9	TIM1_BRK_TIM9_IRQn	TIM8_UP_TIM13_IRQHandler
TIM10	TIM1_UP_TIM10_IRQn	TIM8_TRG_COM_TIM14_IRQHandler
TIM11	TIM1_TRG_COM_TIM11_IRQn	TIM8_CC_IRQHandler
TIM12	TIM8_BRK_TIM12_IRQn	TIM8_BRK_TIM12_IRQHandler
TIM13	TIM8_UP_TIM13_IRQn	TIM8_UP_TIM13_IRQHandler
TIM14	TIM8_TRG_COM_TIM14_IRQn	TIM8_TRG_COM_TIM14_IRQHandler

Naloga

Časovnik TIM4 nastavite tako, da bo generiral PWM signale za vse štiri LED diode. Ob zagonu naj 2 LED diodi po vaši izbiri (LED1 in LED2) svetita s polno svetilnostjo, preostali 2 (LED3 in LED4) pa naj bosta ugasnjeni (0% duty cycle).

Nato uporabite enega izmed preostalih časovnikov, da se bo vsakih 50 milisekund postavila zastavica. Ob postavljeni zastavici spreminjajte duty cycle za LED1 in LED2 tako da najprej zmanjšujete svetilnost dokler nista popolnoma ugasnjeni, nato pa jima svetilnost povečujte nazaj do polne svetilnost. Pri LED 3 in LED 4 duty cycle spreminjajte v obratni smeri, torej najprej ga povečujete, pri polni svetilnosti pa ga začnete zmanjševati. Takšno "neskončno" osciliranje LED diod je tudi končna rešitev naloge.