Časovniki

Vgrajeni sistemi

Rok Češnovar

Časovniki (timers)

- Omogočajo
  - izvajanje operacij ob natančnih intervalih
  - merjenje časa
  - generiranje PWM signala
- STM32F7 & časovniki
  - 2 naprednejša časovnika (advanced)
    - TIM1 & TIM8
  - **10 splošno-namenskih časovnikov (general-purpose)**
    - TIM2-5, TIM9-14
  - 2 osnovna časovnika (basic)
    - TIM6, TIM7

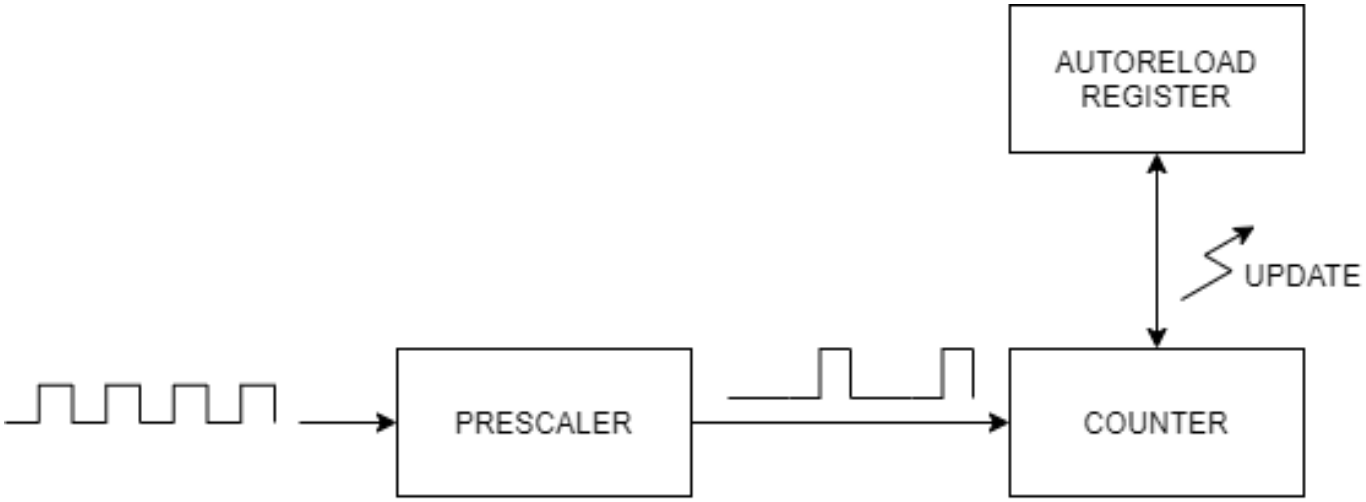
Frekvence vhodnih ur časnovikov

- Nastavljive ločeno glede na vodilo časovnika
  - APB1
    - TIM2-7, TIM12-14
  - APB2
    - TIM1,TIM8, TIM9-11

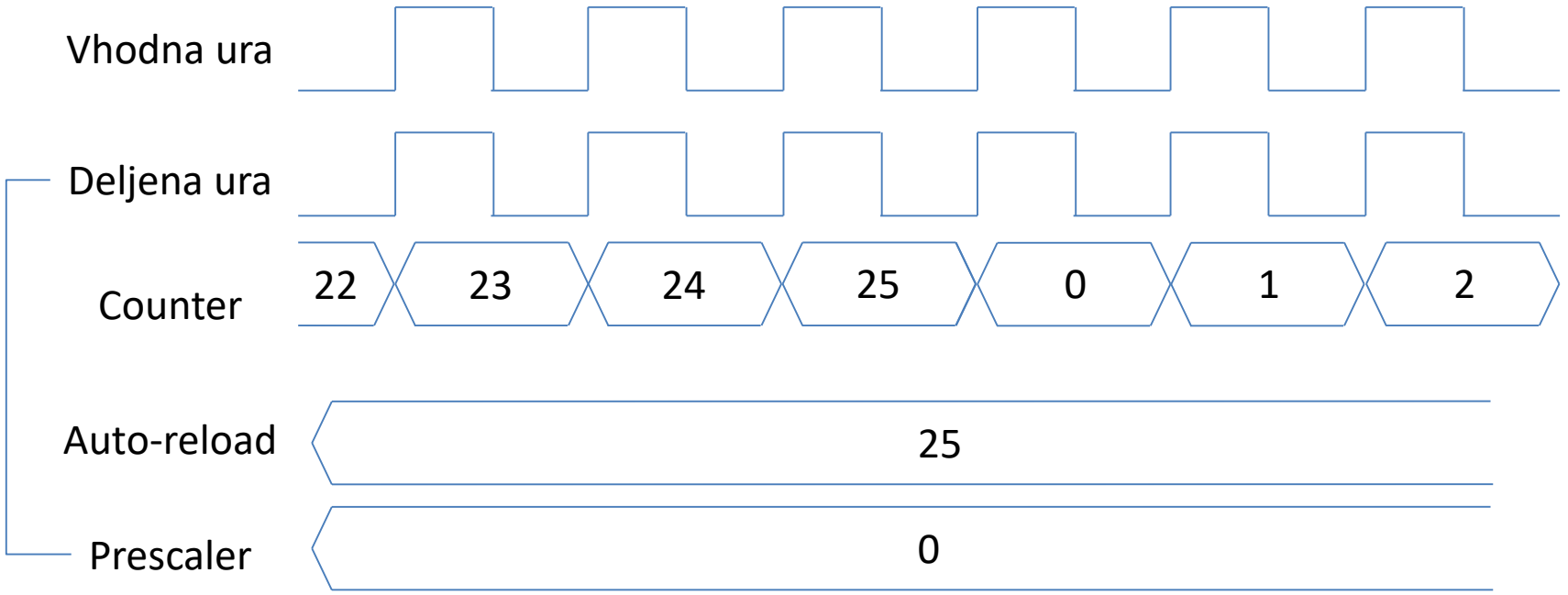
Osnovna funkcija

- Štetje v **counter** registru
  - 16 biten register
  - štetje navzgor, navzdol ali navzgor & navzdol
- Frekvenca štetja je odvisna od izbire vhodne ure in nastavitve **prescaler**-ja
  - prescaler je delilnik ure
    - vrednost registra = 0 -> ni deljenja
    - vrednost registra = 1 -> polovična frekvenca ure
    - vrednost registra = 2 -> frekvenca ure zmanjšana na tretjino
- Štetje se resetira, ko prištejemo do vrednosti, ki je zapisana v **auto-reload** registru (ARR) – UPDATE dogodek

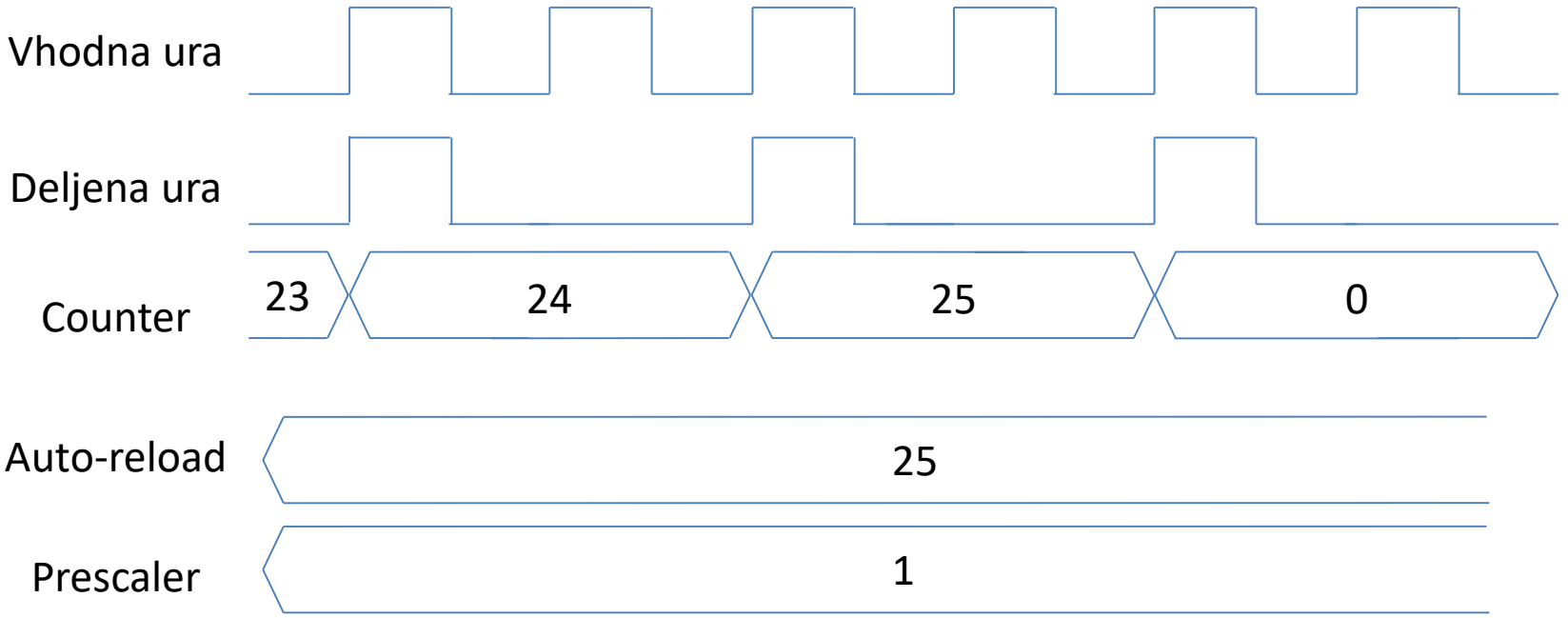
Osnovna funkcija



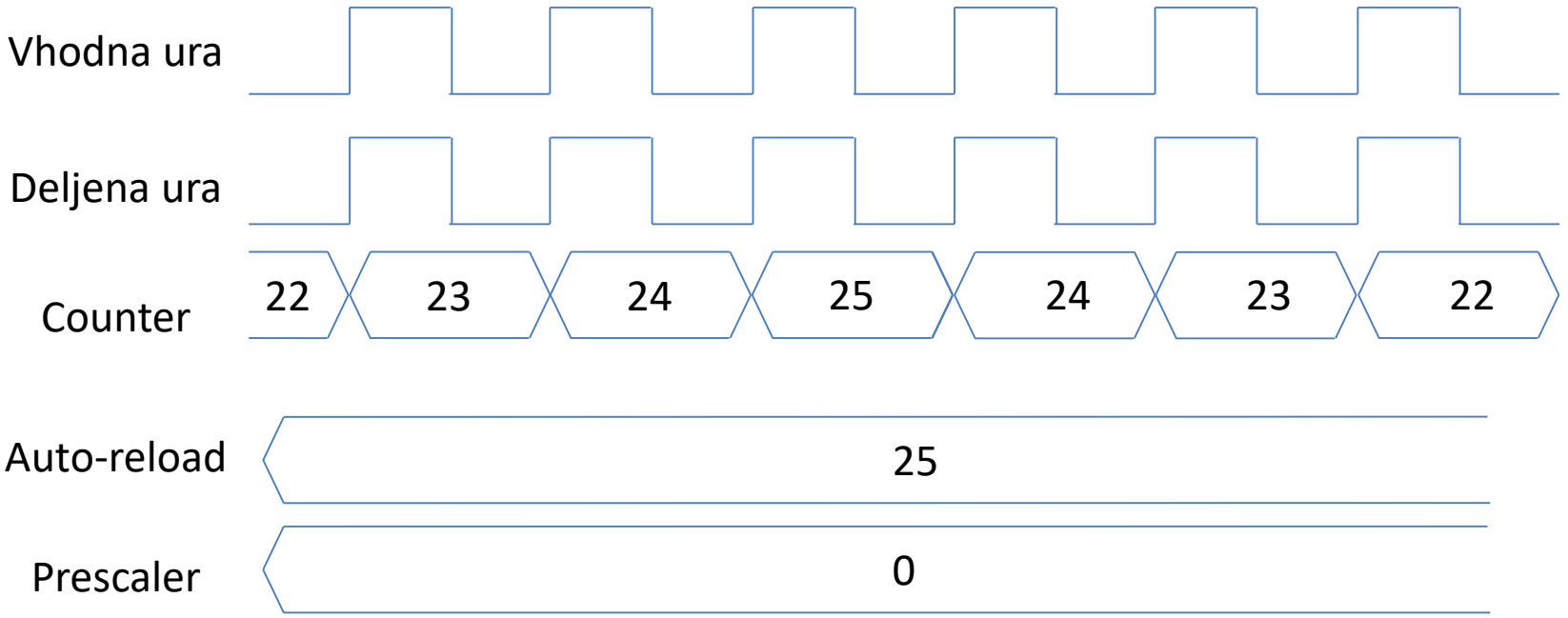
Časovni diagram – štetje navzgor



Časovni diagram – štetje navzgor



Časovni diagram štetje navzgor & navzdol



Izračun željenih frekvenc

- vhodni podatki:
  - vhodna ura časovnika – f_{IN}
  - prescaler
  - ARR
- frekvenca štetja – f_{count}

  $f_{COUNT} = f_{IN} / (prescaler + 1)$
- frekvenca UPDATE dogodkov

  $f_{UPDATE} = f_{IN} / ((prescaler + 1)(ARR + 1))$

Primer 1

- Želimo da časovnik šteje s frekvenco 1kHz ter da se UPDATE dogodek, ki postavi zastavico, pojavlja s frekvenco 1HZ.
- Vhodna ura časovnika je 48MHz

$f_{COUNT} = f_{IN} / (prescaler + 1)$

$(prescaler + 1) = f_{IN} / f_{COUNT}$

$prescaler = f_{IN} / f_{COUNT} - 1$

$prescaler = 48\ MHz / 1kHz - 1$

$\underline{prescaler = 48000 - 1}$

Primer 1

$f_{UPDATE} = f_{IN} / ((prescaler + 1)\ (ARR + 1))$

$(prescaler + 1)\ (ARR + 1) = f_{IN} / f_{UPDATE}$

$(ARR + 1) = f_{IN} / (f_{UPDATE}\ (prescaler + 1))$

$ARR = f_{IN} / (f_{UPDATE}\ (prescaler + 1)) - 1$

$ARR = 48\ MHz / (1Hz\ (48000 - 1 + 1)) - 1$

$ARR = 48\ MHz / (1Hz\ (48000 - 1 + 1)) - 1$

$\underline{ARR = 1000 - 1}$

Primer

- Zanima nas zgolj frekvenca UPDATE dogodka. Ta naj bo 20Hz (perioda = 50ms).
- Vhodna ura časovnika je 48MHz

  $f_{UPDATE} = f_{IN} / ((prescaler + 1)\,(ARR + 1))$

  $(prescaler + 1)\,(ARR + 1) = f_{IN} / f_{UPDATE}$

  $(prescaler + 1)\,(ARR + 1) = 48MHz / 20Hz = 2{,}4\ MHz$

- Več rešitev

  a) prescaler = 2400 – 1, ARR = 1000 – 1

  b) prescaler = 1000 – 1, ARR = 2400 – 1

  c) prescaler = 500 – 1, ARR = 4800 – 1

  d) …

Inicializacija časovnika (na primeru TIM4)

1. Vklopimo uro časovnika

```
__HAL_RCC_TIM4_CLK_ENABLE();
```

2. Ustvarimo init strukturo in določimo vrednosti

```
TIM_HandleTypeDef timer4;
timer4.Instance = TIM4;
timer4.Init.CounterMode = TIM_COUNTERMODE_UP;
timer4.Init.Period = 1000 - 1;
timer4.Init.Prescaler = 48000 - 1;
HAL_TIM_Base_Init(&timer4);
```

Osnovna uporaba časovnika

- Vklop časovnika

HAL_TIM_Base_Start(TIM_HandleTypeDef*);

- Branje stanja zastavic

__HAL_TIM_GET_FLAG(TIM_HandleTypeDef*, zastavica)

- Brisanje zastavice

__HAL_TIM_CLEAR_FLAG(TIM_HandleTypeDef*, zastavica);

- 16 možnih zastavic
  - za nas bo trenutno zanimiva predvsem zastavica TIM_FLAG_UPDATE, ki označeju UPDATE dogodke

Osnovna uporaba časovnika

- Nastavi/beri vrednost števca

__HAL_TIM_SET_COUNTER(TIM_HandleTypeDef* , 1000);

__HAL_TIM_GET_COUNTER(TIM_HandleTypeDef*);

- Ponastavi vrednost ARR registra

__HAL_TIM_SET_AUTORELOAD(TIM_HandleTypeDef* , 1000);

- Vklop/Izklop časovnika

__HAL_TIM_ENABLE(TIM_HandleTypeDef*);

__HAL_TIM_DISABLE(TIM_HandleTypeDef*);

Naloga

- Realizirajte funkcijo za „zakasnitev“

```
void Delay(int ms)
```

- Realizirajte utripanje LED s tremi frekvencami
  - LED 1 – 1 Hz
  - LED 2 – 2 Hz
  - LED 3 – 4 Hz