Prekidi(Interrupts)

- Prekidi su važne karakteristike svih mikrokontrolera I za svaki mikrokontroler potrebno je upoznati se sa arhitekturom prekida.
- Prekid je spoljašnji ili interni (npr. tajmer) događaj koji zahteva od CPU-a da se zaustavi ono što trenutno izvršava i odmah skoči da izvrši drugi kod.
- Kod na koju CPU skače poznat je kao Interrupt Service Routine (ISR).
- Prekidi su asinhroni događaji, CPU ne zna kada će doći do prekida u programu tako da se može pojaviti u bilo kom trenutku.

Prekidi(Interrupts)

- Vreme između generisanja zahteva za prekid i ulazak u ISR poznato je kao 'interrupt latency', .
- Niža vrednost je uvek bolja jer to pokazuje da je odziv sistema brz.
- Kada dođe do prekida, CPU pamti lokaciju sledeće instrukcije koju je trebalo da izvrši tako što se programski brojač čuva u registru ili memorijskoj lokaciji koja se obično se zove stek (stack).

Prekidi(Interrupts)

- CPU se vraća na ovu instrukciju i nastavlja normalan rad nakon završetka koda u ISR i povratka iz prekida.
- Pored čuvanja brojača programa, CPU takođe može da skladišti važne vrednosti registra, kao što su registar statusa CPU-a, zastavice i drugi registri od značaja.

Prekidi(Interrupts)

- Zahtev za prekidom koji je pokrenuo prekid takođe može biti onemogućen tako da se dalji prekidi iz istog izvora ne prepoznaju sve dok se trenutni koji se izvrava ne zavrsi.

Prekidi(Interrupts)

- U mikrokontroleru, CPU može biti dizajniran da odgovori na veliki broj eksternih i internih prekida, na primer u opsegu od 10 do preko 100.
- Svaki izvor prekida ima namenski memorijski prostor gde se očekuje da se ISR kod nalazi.

Prekidi(Interrupts)

- U nekim procesorima, više izvora prekida dele istu memorijsku lokaciju za svoje ISR.
- Tada je na korisničkom programu da odredi stvarni izvor prekida.
- Prekidi su uvek onemogućeni kada se procesor pokrene.

Prekidi(Interrupts)

- Sledeći uslovi moraju biti ispunjeni pre nego što CPU prihvati prekid:
- prekid iz traženog izvora mora biti omogućen,
- izvor mora da generiše zahtev za prekid,
- globalni prekidi procesora moraju biti omogućeni.

Prekidi(Interrupts)

- Eksterni prekidi se obično javljaju kada je pin porta spušten na nisko (tj. na opadajućoj ivici) ili visoko (tj. na rastućoj ivici) ili ako postoji promena u stanju pina porta (od niskog do visokog ili od visokog do niskog).
- Interni tajmerski prekidi se javljaju na primer kada se tajmer prepuni.

Prekidi(Interrupts)

- Izvori prekida obično imaju dodeljene prioritete.
- Prekid višeg prioriteta može da zaustavi izvršavanje prekida nižeg prioriteta i preuzme CPU.
- Kada se prekid viseg prioriteta zavrsi, ISR nižeg prioriteta može da se nastavi.

Prekidi(Interrupts)

- Važno je da ISR kod treba da troši što je moguće manje vremena procesora.
- Ovo je posebno vazno kada je u programu omogućeno više prekida.
- Ako ISR traje dugo, onda to može da izazove odloženi odgovor na druge prekide kojima je možda potreban brz odgovor.

Prekidi(Interrupts)

- Prekidi se mogu maskirati softverski tako da zahtevi iz takvih izvora ne mogu biti prihvaćeni.
- Neki procesori takođe imaju izvore nemaskiranog prekida (NMI) koji se ne može softverski maskirati ili onemogućiti.

Prekidi(Interrupts)

- Različiti registri unutar CPU-a moraju biti konfigurisani pre eksternog ili internog prekida da bi prekid bio prihvacen od strane CPU.
- Svaka familija miktokontrolera ima specificne karatkeristike prekida koje je potrebno prouciti da bi se pravilno konfigurisali prekidi.

Prekidi(Interrupts)

- Na uređaju
- Svaki izvor prekida ima poseban bit za omogućavanje.
- Bit mora biti setovan da bi se prekid prihvatio.
- Svaki izvor prekida ima poseban bit zastavicu (flag bit).
- Hardver postavlja ovaj bit kada se dogodi zahtev za prekid.
- Ovaj bit mora biti obrisan u ISR softverski.

Prekidi(Interrupts)

- U CPU-u
- Zahtev za prekid se prima preko ugnežđenog vektorskog kontrolera prekida (NVIC).
- Zahtev za prekidom najvišeg prioriteta se šalje CPU-u.
- Globalni bit za omogućavanje prekida mora biti omogućen u PRIMASK registru, na primer.
- Nivo prioriteta izvora prekida koji se zahteva mora biti viši od osnovnog prioriteta BASEPRI.

Tajmerksi prekidi

- Tajmerski prekidi na STM32F103 mikroprocesorima omogućavaju izvršavanje određenih funkcija u pravilnim vremenskim intervalima.
- Ovo je korisno za aplikacije koje zahtevaju tačno vremensko merenje, generisanje PWM-a, ili brojačke operacije.

- STM32F103 ima više tajmera (TIM1, TIM2, TIM3, itd.), koji su opremljeni brojačima, prescaler-ima i režimima rada.
- Tajmeri mogu raditi u režimu pulsno-širinske modulacije (PWM), merenja ulaznih signala, brojača ili kao osnovni tajmeri za generisanje prekida.

- Prescaler: Delitelj takta, kojim se smanjuje ulazni takt (npr. APB1 ili APB2 takt) kako bi se prilagodio brojaču tajmera.
- ARR (Auto-Reload Register): Postavlja maksimalnu vrednost do koje brojač tajmera broji pre nego što se resetuje.

- Tajmer generiše interrupt kada dostigne vrednost u ARR-u.
- Prekid se koristi za izvršavanje funkcija na svakom završetku brojačkog ciklusa.

Konfiguracija tajmera u STM32CubeMX

- U STM32CubeMX i postavi se odgovarajući tajmer (npr. TIM2).
- Postavi se Clock Source na Internal Clock.
- Konfiguriše se Prescaler i Counter Period da odgovaraju željenom vremenskom intervalu.
- Na primer, za 1 Hz interval (1 sekunda):
- Prescaler = (SystemClock / DesiredClock) - 1.
- ARR = DesiredClock / DesiredFrequency.
- Omogući Update Interrupt za tajmer.

- Nakon konfiguracije u STM32CubeMX, generiše se kod u STM32CubeIDE.
- Kod će automatski uključivati HAL biblioteku za upravljanje tajmerom.
- Kod za inicijalizaciju tajmera i aktivaciju prekida:

  HAL_TIM_Base_Start_IT(&htim2); // Startuje tajmer i omogućava prekide

- Ovo se obično postavlja u main() funkciji.

- Funkcija za rukovanje prekidom se generiše u stm32f1xx_it.c. Na primer, za TIM2:

```
void TIM2_IRQHandler(void) { HAL_TIM_IRQHandler(&htim2);  // Poziva HALfunkciju
                                                          //za rukovanje prekidima

}
```

- Unutar stm32f1xx_hal_tim.c, koristi se funkcija HAL_TIM_PeriodElapsedCallback:

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {

  if (htim->Instance == TIM2) {

  // Kod koji se izvršava na svakom prekidu

 HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_13); // Blinka LED-om

 }

}
```

- U stm32f1xx_hal_msp.c, HAL inicijalizacija automatski uključuje konfiguraciju NVIC-a za tajmer, ali moze I ručno da se doda:

```
HAL_NVIC_SetPriority(TIM2_IRQn, 0, 0);  // Postavlja prioritet
HAL_NVIC_EnableIRQ(TIM2_IRQn);          // Omogućava prekid
```

Primer koda za bljeskanje LED-a svakih 500 ms

```
#include "main.h"

TIM_HandleTypeDef htim2;

void SystemClock_Config(void);

static void MX_GPIO_Init(void);

static void MX_TIM2_Init(void);

int main(void) {
    HAL_Init();
    SystemClock_Config();
    MX_GPIO_Init();
    MX_TIM2_Init();
    // Start tajmera sa prekidima
    HAL_TIM_Base_Start_IT(&htim2);
```

Primer koda za bljeskanje LED-a svakih 500 ms

```
// Inicijalizacija TIM2
static void MX_TIM2_Init(void) {
    htim2.Instance = TIM2;
    htim2.Init.Prescaler = 8000 - 1;  // Podešava takav prescaler za 1ms tick (za 8 MHz clock)
    htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
    htim2.Init.Period = 500 - 1;     // 500 ms interval
    htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
    if (HAL_TIM_Base_Init(&htim2) != HAL_OK) {
        Error_Handler();
    }
}
```

Primer koda za bljeskanje LED-a svakih 500 ms

```
// GPIO inicijalizacija za LED
static void MX_GPIO_Init(void) {
    GPIO_InitTypeDef GPIO_InitStruct = {0};

    __HAL_RCC_GPIOC_CLK_ENABLE();

    GPIO_InitStruct.Pin = GPIO_PIN_13;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
}
```

Eksterni prekidi kod STM32F103

- Eksterni prekidi kod STM32F103 omogućavaju mikroprocesoru da reaguje na spoljne signale dolaskom prekida sa pina (GPIO).
- To je korisno za detekciju događaja kao što su pritiskanje tastera, promena stanja na senzoru ili komunikacija između uređaja.

GPIO kao izvor prekida

- Eksterni prekidi se konfigurišu putem EXTI linija (External Interrupt Lines).
- Svaki GPIO pin može biti povezan na odgovarajuću EXTI liniju (npr. PA0 na EXTI0)

  Prekid može biti aktiviran na osnovu sledećih događaja:

- Rising edge (promena sa 0 na 1).
- Falling edge (promena sa 1 na 0).
- Both edges (promena u oba smera).

GPIO kao izvor prekida

- NVIC (Nested Vectored Interrupt Controller)

  NVIC omogućava prioritet i rukovanje eksternim prekidima.

- ISR (Interrupt Service Routine)

- Funkcija koja se poziva prilikom prekida.

Konfiguracija u STM32CubeMX

- U STM32CubeMX i odabere se odgovarajući GPIO pin kao EXTI. Na primer:

- Postavi se PA0 kao GPIO_EXTI.

- Bira se vrsta trigera: Rising Edge, Falling Edge ili Both Edges.
- Omogući se NVIC prekid za EXTI liniju.
- Generiše kod i otvori u STM32CubeIDE. Kod će uključivati osnovnu inicijalizaciju.

Implementacija ISR funkcije

- Prekidna rutina za odgovarajuću EXTI liniju nalazi se u fajlu stm32f1xx_it.c. Na primer, za EXTI0:

```
void EXTI0_IRQHandler(void) {
    HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_0);
}
```

Implementacija ISR funkcije

- Callback funkcija se koristi za rukovanje događajem:

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) {
    if (GPIO_Pin == GPIO_PIN_0) {
        // Kod koji se izvršava kada se prekid dogodi
        HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_13); // Bljeskanje LED-a
    }
}
```

Implementacija ISR funkcije

- Postavi GPIO pin kao ulaz
- U main.c, GPIO pin mora biti postavljen kao ulaz.

```
GPIO_InitTypeDef GPIO_InitStruct = {0};

__HAL_RCC_GPIOA_CLK_ENABLE();

GPIO_InitStruct.Pin = GPIO_PIN_0;

GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;

GPIO_InitStruct.Pull = GPIO_NOPULL;

HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

HAL_NVIC_SetPriority(EXTI0_IRQn, 0, 0);  // Postavi prioritet

HAL_NVIC_EnableIRQ(EXTI0_IRQn);         // Omogući prekid
```

Primer: Prekid na tasteru (PA0) za blinkanje LED-a

- Postavi GPIO pin kao ulaz
- U main.c, GPIO pin mora biti postavljen kao ulaz.

```
GPIO_InitTypeDef GPIO_InitStruct = {0};

__HAL_RCC_GPIOA_CLK_ENABLE();

GPIO_InitStruct.Pin = GPIO_PIN_0;

GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;

GPIO_InitStruct.Pull = GPIO_NOPULL;

HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

HAL_NVIC_SetPriority(EXTI0_IRQn, 0, 0);  // Postavi prioritet

HAL_NVIC_EnableIRQ(EXTI0_IRQn);         // Omogući prekid
```

Primer: Prekid na tasteru (PA0) za blinkanje LED-a
Kod za main.c:

```
#include "main.h"

void SystemClock_Config(void);

static void MX_GPIO_Init(void);

int main(void) {

    HAL_Init();

    SystemClock_Config();

    MX_GPIO_Init();


    while (1) {

        // Glavna petlja

    }

}
```

Primer: Prekid na tasteru (PA0) za blinkanje LED-a
Kod za main.c:

```
// Callback funkcija za EXTI prekid
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) {
    if (GPIO_Pin == GPIO_PIN_0) {
        HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_13); // Bljeskanje LED-a
    }
}
```

Primer: Prekid na tasteru (PA0) za blinkanje LED-a
Kod za main.c:

```
static void MX_GPIO_Init(void) {
    GPIO_InitTypeDef GPIO_InitStruct = {0};
    // Inicijalizacija GPIOA (PA0 kao eksterni prekid)
    __HAL_RCC_GPIOA_CLK_ENABLE();
    GPIO_InitStruct.Pin = GPIO_PIN_0;
    GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING; // Rising edge
    GPIO_InitStruct.Pull = GPIO_NOPULL; // Bez povlačenja
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
    // Inicijalizacija GPIOC (PC13 za LED)
    __HAL_RCC_GPIOC_CLK_ENABLE();
    GPIO_InitStruct.Pin = GPIO_PIN_13;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
```

Primer: Prekid na tasteru (PA0) za blinkanje LED-a
Kod za main.c:

```
// Postavi NVIC za EXTI0
    HAL_NVIC_SetPriority(EXTI0_IRQn, 0, 0);
    HAL_NVIC_EnableIRQ(EXTI0_IRQn);
}
```

Napomena

- U aplikacijama sa više eksternih prekida vodi računa o konfliktima EXTI linija.
- Na STM32F103, jedna EXTI linija može biti povezana na više GPIO-ova (npr. PA0, PB0, PC0 koriste EXTI0).