

Osnove mikroprocesorske elektronike

Vaja 11: Operacijski sistem FreeRTOS

Uporaba operacijskega sistema zelo olajša programiranje, kadar se mora hkrati izvajati več opravil, vendar ni primerna za zelo majhne mikrokrmilnike ali aplikacije, kjer mora biti odzivnost na nivoju mikrosekund.

Mikroprocesor ATmega324 je ravno dovolj velik, da je operacijski sistem FreeRTOS še uporaben.

Naloga:

Igrico "Ujemi voluharja" napišite še enkrat, tokrat z uporabo FreeRTOS.

Specifikacija igre:

Igra naj ima le enostaven način delovanja - vsakič se prikaže en voluhar. Igra ima 7 nivojev težavnosti.

Igro zaženemo s pritiskom na katerokoli tipko. Dokler čakamo na pritisk tipke, vse diode utripajo dvakrat na sekundo. Na LCD-ju piše "Pritisni tipko za začetek".

Dokler igra traja, naj se na LCD-ju prikazuje nivo težavnosti in trenutne točke.

Ko je igre konec, napišemo "KONEC" in izpišemo dosežene točke. LED diode potujejo od obeh skrajnih koncev proti sredini. Ko pritisnemo katerokoli tipko, preidemo na začetek.

Domača naloga:

FreeRTOS nastavite tako, da bo ločljivost sistemske ure 0,1 ms (definicija "configTICK_RATE_HZ" v datoteki "FreeRTOSConfig.h").

Ustvarite globalno spremenljivko "char LED[8]={1,2,4,8,16,32,64,100};"

Napišite opravilo (task) "vPWM_LED", ki bo imelo najvišjo prioriteto od vseh opravil. To opravilo naj se izvede enkrat vsakih 100 μ s (uporabite "vTaskDelayUntil") in glede na pretečeni čas in vrednost v "LED[i]" prižge ali ugasne vsako izmed LED diod.

Napišite še eno opravilo z malo nižjo prioriteto, ki vsakih 100 ms določi nove vrednosti globalne spremenljivke "LED".

Dodatno: Ker dve opravili dostopata do iste globalne spremenljivke, je treba poskrbeti, da ne bi opravilo z višjo prioriteto prekinilo dostopa opravila z nižjo prioriteto. To najlažje rešite tako, da namesto, da bi komunicirali preko globalne spremenljivke, uporabite Queue (nekaj podobnega kot krožni medpomnilnik). Uporaba za vse potrebne funkcije je najbolje opisana na strani z opisom funkcije "xQueueReceive()": "FreeRTOS \rightarrow API Reference \rightarrow Queues \rightarrow xQueueReceive()".

Osnovna navodila:

1. Z Atmel studiem ustvarite nov projekt.
2. Na uradni strani FreeRTOS poiščite arhiv (zip) zadnje izdaje operacijskega sistema in ga prenesite na svoj računalnik.
3. Na spletni strani FreeRTOS poiščite navodila za pripravo novega projekta z operacijskim sistemom – v meniju na levi strani "FreeRTOS \rightarrow More Advanced... \rightarrow Creating a new project".
4. V podpoglavju "Source Files" je napisan seznam izvornih datotek, ki jih potrebujete za minimalno konfiguracijo operacijskega sistema. Datoteka "port.c" je odvisna od procesorja. Za MIŠKO-ta jo dobite na echo-tu, skupaj s še dvema deklarcijskima datotekama. Za upravljanje

spomina potrebujemo najenostavnejši način, zato izberite datoteko "heap_1.c". Vse te datoteke skopirajte v mapo svojega projekta.

5. V podpoglavju "Header files" je napisano, v kateri mapi arhiva najdete deklaracijske datoteke. Celo mapo "include" skopirajte v mapo svojega projekta. V mapi "...portable..." je le ena deklaracijska datoteka ("portmacro.h"), ki jo dobite na echo-tu skupaj z datoteko "port.c". Tudi to skopirajte v mapo svojega projekta.
6. V Atmel studiu nastavite mape v katerih naj išče deklaracijske datoteke:
 - Izberite "Project → ImeVasegaProjekta Properties...".
 - Na levi strani novega okna poiščite razdelek "Toolchain".
 - Na seznamu kategorij izberite kategorijo "AVR/GNU C Compiler → Directories"
 - Dodajte glavno mapo projekta (mapa, v kateri je main.c, portmacro.h, itd.) in mapo "includes", ki ste jo prej skopirali v glavno mapo.
7. V projekt dodajte vse ".c" datoteke, ki ste jih skopirali v mapo.
8. V "main.c" prepišite primer programa, ki utripa z dvema LED diodama:

```
#include <avr/io.h>
#include "FreeRTOS.h"
#include "task.h"

void vBlinkLED1( void * pvParameters );
void vBlinkLED2( void * pvParameters );
void Init_IO();    //to funkcijo skopirajte iz prejsnjih projektov

int main(void)
{
    TaskHandle_t task1, task2;

    Init_IO();    //inicializacije
    //Ustvari opravila
    xTaskCreate(vBlinkLED1,"LD1",configMINIMAL_STACK_SIZE,NULL,tskIDLE_PRIORITY, &task1);
    xTaskCreate(vBlinkLED2,"LD2",configMINIMAL_STACK_SIZE,NULL,tskIDLE_PRIORITY, &task2);
    //pozeni program
    vTaskStartScheduler();

    while (1) {PORTB = 0xFF;}    //do tukaj program naj ne bi nikoli prišel
}

void vBlinkLED1( void * pvParameters )
{
    for ( ;; )
    {
        PORTB ^= 0x01;
        vTaskDelay(500);    //ta nacin generira približno frekvenco
    }
}

void vBlinkLED2( void * pvParameters )
{
    TickType_t LastTick = xTaskGetTickCount();
    while(1)
    {
        PORTB ^= 0x02;
        vTaskDelayUntil(&LastTick,100);    //ta nacin generira točno frekvenco -
    }    // ce ne upostevamo trepetanja (jitter)
}
```

Dodatna navodila:

- Opis funkcij, ki jih ponuja operacijski sistem, najdete na spletni strani v meniju "FreeRTOS → API reference".
Na voljo so samo funkcije, ki so izbrane v konfiguracijski datoteki FreeRTOSConfig.h. Za nekatere funkcije je treba skopirati tudi dodatne datoteke. Podrobna navodila za konfiguracijsko datoteko najdete v meniju "FreeRTOS → More Advanced... → FreeRTOSConfig.h".
- Kratak seznam najosnovnejših funkcij in makrojev:
 - xTaskCreate
 - vTaskStartScheduler
 - vTaskDelay
 - vTaskDelayUntil
 - taskYIELD
 - taskENTER_CRITICAL
 - taskEXIT_CRITICAL
 - taskDISABLE_INTERRUPTS
 - taskENABLE_INTERRUPTS
 - xQueueCreate
 - xQueueSendToBack
 - xQueueReceive
 - uxQueueMessagesWaiting
 - uxQueueSpacesAvailable
 - xSemaphoreCreateMutex
 - xSemaphoreTake
 - xSemaphoreGive