Vežba - FreeRTOS - 7

Idle Task/Hook, Assertions, Statistics

U ovoj vežbi ćemo videti na koji način možemo koristiti IDLE task. Videćemo kako se koriste Assertions u C-u. Na kraju će biti prikazan primer na kom se vidi statistika zauzetosti procesora.

1 Idle Task i Idle Hook

void vApplicationIdleHook (void);	Potrebno je implementirati funkciju sa ovakvim prototipom
Parametar: configUSE_IDLE_HOOK	Ovaj parametar u FreeRTOSConfig.h fajlu mora biti setovan na pdTRUE

Idle task

Procesor uvek mora imati nešto da izvršava - znači barem jedan task u svakom trenutku mora biti u mogućnosti da predje u *Running* stanje. Kako bi se ovaj uslov zadovoljio, FreeRTOS kreira Idle Task pri pozivu vTaskStartScheduler() funkcije. Ovaj task najčešće samo izvršava praznu petlju i uvek je spreman za izvršavanje. Idle task ima najmanji prioritet i bilo koji task koji postaje Ready će biti automatski izvršavan umesto Idle taska.

Idle Hook

Dodavanje funkcionalnosti u Idle Task je moguće preko Idle Hook-a (call-back funkcija). Ova funkcija se automatski poziva jednom po iteraciji Idle Taska.
Neke od čestih primena su:

- Izvršavanje nisko-prioritetnih zadataka ili kontinualnog računanja
- Merenje stepena zauzetosti procesora
- Stavljanje procesora u mod rada sa niskom potrošnjom

Rad sa Idle Hook funkcijom ima nekih limitacija:

1. Ne smeju se koristiti operacije koje suspenduju ili blokiraju.
2. Ako se koristi u kodu API funkcija vTaskDelete(), potrebno je da se iteracija Idle Taska izvršava u nekom razumnom vremenskom intervalu jer se tada oslobađaju resursi koji više nisu potrebni. To znači da bi Idle Hook funkcija trebala biti relativno kratka kako bi se efikasno oslobađali resursi

Primer korišćenja Idle Hook za merenje stepena zauzetosti procesora

- Primer Tajmer funkcije koja se periodično poziva i proverava koliko puta je iteriran Idle Task

```
static void checkIdleCountTimerFun(TimerHandle_t tH) {

	static uint8_t avg_counter = 0;
	static uint64_t cnt_sum = 0;

	cnt_sum += idleHookCounter;
	idleHookCounter = 0;

	if (avg_counter++ == 9) {
		//printf("Prosecni IdleHook counter je: %lld\n", cnt_sum / 10);

		// Idle Hook bez dodatnih taskova je: ~2500000 - Kalibrisati spram računara

		uint64_t average = cnt_sum / 10;

		float odnos = (float)average / 2500000;
		if (odnos > 1) {
			odnos = 1;
		}
		int procenat = odnos * 100;

		printf("Zauzetost procesora u je: %d\n", 100-procenat);

		cnt_sum = 0;
		avg_counter = 0;
	}

}
```

- Jednostavna Idle Hook funkcija koja samo povećava brojač za jedan

```
void vApplicationIdleHook(void) {

	idleHookCounter++;
}
```

2. Run Time Statistics

https://www.freertos.org/rtos-run-time-stats.html

Nažalost ovu temu nismo uspeli da pokrenemo na PC-ju. Biće samo napravljen pregled kako bi izgledalo.

3. Assert u FreeRTOS-u

FreeRTOS ima mogućnost korišćenja Assert-a na taj način da kaže koja tačno linija u kom fajlu je stvorila problem zbog kog je pao assert.

Definisana je funkcija *configASSERT(int)* kojoj se pri pozivu prosleđuje Int parametar. Ako je taj parametar 0, assert je pao odnosno došlo je do nekog problema. Ako je parametar različit od 0, samo se prolazi kroz tu funkciju i nastavlja dalje.

- Funkcija koja se dalje poziva ako je Assert pao izgleda ovako:

```
void vAssertCalled( unsigned long ulLine, const char * const pcFileName )
{
static BaseType_t xPrinted = pdFALSE;
volatile uint32_t ulSetToNonZeroInDebuggerToContinue = 0;

	/* Called if an assertion passed to configASSERT() fails. */

	/* Parameters are not used. */
	( void ) ulLine;
	( void ) pcFileName;

	printf( "ASSERT! Line %ld, file %s, GetLastError() %ld\r\n", ulLine,
pcFileName, GetLastError() );

	taskENTER_CRITICAL();
	{
		/* You can step out of this function to debug the assertion by
using
		the debugger to set ulSetToNonZeroInDebuggerToContinue to a
non-zero
		value. */
		while( ulSetToNonZeroInDebuggerToContinue == 0 )
		{
			__asm{ NOP };
			__asm{ NOP };
		}
	}
	taskEXIT_CRITICAL();
}
```