Fakulta Informačních technologií
Vysoké učení technické v Brně

IMP - Mikroprocesorové a vestavěné systémy

Sada demo aplikací nad FreeRTOS

Tomáš Dvořáček (xdvora3d)

1 Úvod

Zadáním tohoto projektu bylo seznámit se s principem vytváření vestavěných aplikací a nastudovat principy programování řízení periferií z úrovně úloh FreeRTOS.

Základním požadavkem bylo vytvořit bare-metal aplikaci a následně druhou aplikaci s operačním systémem FreeRTOS, obě pracující s alespoň jednou vstupní a výstupní periferií.

1.1 Použité technologie

Při řešení tohoto projektu jsem používal Kinetis Design Studio a utilitu ProcessorExpert s přidaným modulem pro FreeRTOS.

2 Bare-metal aplikace

Princip fungování této aplikace spočívá v blikající LED diodě a tlačítkem, které toto blikání zastavuje / zapíná.

2.1 Implementace

Blikání je implementováno pomocí časovače se zdrojem periodického přerušení *FTM0_MOD* a periodou 1000 ms, přičemž se při vygenerovaném přerušení volá funkce *NegVal(),* která zneguje bitovou hodnotu na vývodu s připojenou LED. Zastavování a spouštění blikání je řešeno pomocí externího přerušení z pinu na kterém je připojeno tlačítko. Pokud je tlačítko stisknuto, objeví se na pinu sestupná hrana a generuje se přerušení, přičemž se zjišťuje stav proměnné *state.* V této proměnné je uchováván aktuální stav časovače (BUSY nebo IDLE), kde se přepínáním mezi těmito stavy přerušení od časovače pozastavuje nebo spouští, což má za následek právě zastavování a spouštění blikání LED.

```
void ExtInt1_OnInterrupt(void)
{
    LDD_TDriverState state = TimerIntLdd1_GetDriverState(TimerIntLdd1_DeviceData);

    if(state == PE_LDD_DRIVER_BUSY)
    {
        TimerIntLdd1_Disable(TimerIntLdd1_DeviceData);
    }
    else
    {
        TimerIntLdd1_Enable(TimerIntLdd1_DeviceData);
    }
}
```

Obrázek 1: Externí přerušení s přepínáním stavů časovače

3 Aplikace s FreeRTOS

Tato aplikace funguje na podobném principu jako výše popsaná Bare – metal aplikace s tím, že blikání LED je zapouzdřeno v úloze operačního systému FreeRTOS. Samotná úloha je vytvořena tímto způsobem:

```
if (FRTOS1_xTaskCreate(
      LED0_blink,  /* task function */
      "ledblink", /* task name for kernel awareness */
      configMINIMAL_STACK_SIZE, /* task stack size */
      (void*)NULL, /* optional task startup argument */
      tskIDLE_PRIORITY,  /* initial priority */
      NULL
    ) != pdPASS) {
    for(;;){}
  }
```

Obrázek 2: Vytvoření úlohy ve FreeRTOS

Samotná implementace blikání LED v úloze *LED0_blink()* vypadá takto:

```
static void LED0_blink(void * param)
{
    (void) param;

    for(;;)
    {
        LED0_NegVal();
        FRTOS1_vTaskDelay(1000/portTICK_PERIOD_MS);
    }
}
```

Obrázek 3: Blikání LED a zpoždění úlohy

3.1 Vytvoření zpoždění

Podobně jako v předchozí aplikaci je blikání realizováno negací aktuální hodnoty na daném pinu, ale zpoždění je vytvořeno pomocí funkce *vTaskDelay()*. Konstanta *portTICK_PERIOD_MS* udává periodu Ticků. To, s jakou periodou se Ticky generují, závisí na zvolené frekvenci, v tomto případě 100 Hz. Z tohoto vyplývá, že perioda jednoho ticku bude 10 ms, celková doba zpoždění bude 100 ticků (1000 / 10), každý po 10 ms, takže výsledné zpoždění bude jedna sekunda.

3.2 Přepínání stavu úlohy

Úloha ve FreeRTOS může mít několik stavů:

- Ready
- Running
- Blocked
- Suspended
- Deleted

Pro tuto část projektu jsou důležité stavy Running a Suspended, tedy aktuálně běžící úloha (LED dioda bliká) a pozastavená úloha (LED dioda nebliká). Přepínání stavů se děje ve funkci, která se zavolá v případě externího přerušení od stisknutého tlačítka takto:

```
void EInt1_OnInterrupt(void)
{
    TaskHandle_t task = FRTOS1_xTaskGetHandle("ledblink");
    eTaskState status = FRTOS1_eTaskGetState(task);

    if(status == eSuspended)
    {
        FRTOS1_xTaskResumeFromISR(task);
    }
    else
    {
        FRTOS1_vTaskSuspend(task);
    }

}
```

Obrázek 4: Přepínání stavu úlohy