Mikroprocesorové a vestavěné systémy
Sada aplikací nad FreeRTOS

Michal Rein (xreinm00)

19.12.2020

Obsah

1 Úvod do problematiky

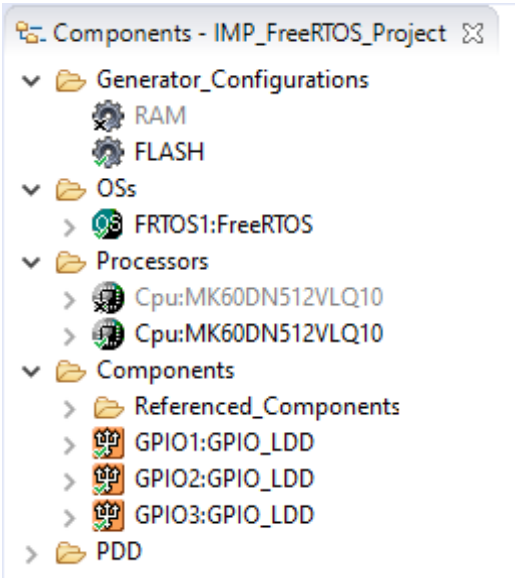
Smyslem a zadáním projektu je vytvoření sady demoaplikací za použití vstupních a výstupních periferií nad real-time operačním systémem FreeRTOS na platformě FITkit3, která disponuje mikrokontrolérem Kinetis K-60 s jádrem ARM Cortex-M4. Na FITkit3 najdeme sadu LED diod, tlačítka a piezzo, které lze využít pro demonstraci a ovládání výsledné aplikace.

Jak již bylo zmíněno, aplikace poběží nad operačním systémem FreeRTOS. Jedná se o real-time operační systém, který se zpravidla využívá v real-time systémech, ve kterých běží současně více úloh a je nutné těmto procesům přidělovat procesorový čas a prostředky se závislostí na prioritách.

2 Popis řešení

Celá aplikace vznikla v IDE Kinetis Design Studio s použitím integrovaného nástroje Processor Expert (PEx), který umožňuje snadnou konfiguraci komponent pro zvolený mikrokontrolér. Pro import komponenty FreeRTOS do nástroje PEx jsem využil balíček dostupný na stránce mcuoneclipse.com[1].

Pomocí PEx jsem si následně nakonfiguroval operační systém FreeRTOS, komponenty GPIO1, GPIO2 a GPIO3, které obsahují inicializaci modulu GPIO a namapování jednotlivých pinů vstupně-výstupních periferií LED diod, tlačítek a piezza. Konkrétní porty a piny, na kterých jsou fyzicky periferie dostupné jsem našel v technické dokumentaci a demoaplikacích pro zařízení FITkit3. Výsledné schéma komponent popisuje Obrázek 1.



Obrázek 1Komponenty PEx

[1] https://mcuoneclipse.com/2014/10/21/mcuoneclipse-releases-on-sourceforge/

Zdrojové a hlavičkové soubory pro tyto komponenty byly následně vygenerovány rovněž nástrojem PEx, které lze nalézt ve složce Generated_Code nacházející se v adresáři s projektem.

Hlavní zdrojový soubor se vstupním bodem programu *main.c* se nachází ve složce Sources. V něm lze nalézt definice procesů běžících nad FreeRTOS, které jsou následně vytvořeny a spojeny s OS metodou *xTaskCreate()*. Tyto hlavní procesy jsou celkem 3, přičemž se jedná o proces obsluhující blikání LED diod (*blinkingLEDTask*), proces detekující a obsluhující stisk tlačítek pro změnu módu blikání LED diod (*btnPressCheckTask*) a následně proces pro obsluhu tlačítka spínající piezzo (*buzzerHandlerTask*).

Pomocí tlačítek *SW2, SW3, SW4* a *SW5* lze libovolně měnit jeden z 5 módů blikání diod, přičemž mód označený jako *MODE0* slouží pouze k zastavení blikání diod. V módu 0 se aplikace nachází při jejím startu a následně po opětovném stisknutí stejného tlačítka, které aktivovalo některý z ostatních módů. Tlačítko *SW6* poté slouží k obsluze piezza, po jehož stisknutí se ozve krátké pípnutí/zabzučení.

Veškerou funkčnost metod ve zdrojovém souboru jsem se snažil patřičně zakomentovat a popsat jejich funkcionalitu. Pro podrobnější popis programu tedy prosím využijte i samotný kód.

3 Zhodnocení

Výsledné řešení bylo úspěšně otestováno na platformě FITkit3. Při změnách tick rate operačního systému se mění rychlost blikání LED diod i intenzita hluku bzučáku, z čehož plyne, že jednotlivé procesy jsou opravdu spravovány daným operačním systémem FreeRTOS. Pro alespoň patrnou zvukovou odezvu piezza jsem musel zvýšit tick rate operačního systému z výchozích 100Hz na 1000Hz. Při vyšších frekvencích je sice zvuk výraznější, nechtěl jsem však mikrokontrolér zbytečně zatěžovat.

4 Odkazy

Odkaz na video prezentující řešení: https://youtu.be/TbD3DqAiIa4