2009/44 - 8. 9. 2009

elektrorevue
ISSN 1213 - 1539

METODA PRO SNÍŽENÍ PŘÍKONU MIKROKONTROLÉRU PRACUJÍCÍHO POD RTOS

Dr. Ing. Karel Dudáček.
Západočeská univerzita v Plzni
Univerzitní 8, Plzeň
Email: dudacek@kiv.zcu.cz

V článku je popisována metoda pro snížení příkonu mikrokontroléru procujícího pod RTOS. Principem metody je odstranění nutnosti obsluhy periodických hodinových přerušení v době mezi spuštěním naplánovaných periodických úloh. Před spuštěním naplánované úlohy je provedena resynchronizace systémového času a obnovena normální činnost procesorového jádra.

1. ÚVOD

Celá řada v současné době konstruovaných elektronických zařízení je řízena mikrokontroléry. Návrhář tak získá možnost pružného přizpůsobení parametrů zařízení aktuálním potřebám bez nutnosti zásahu do hardwarového návrhu. Nedílnou součástí návrhu je potom i návrh programového vybavení pro mikrokontrolér. Při vytváření programového vybavení pro mikrokontroléry lze s výhodou využít operačních systémů reálného času (RTOS).

Častým požadavkem na takto konstruovaná zařízení je kromě jejich základní funkčnosti též provoz s bateriovým napájením. Často je požadován i několikaletý provoz zařízení bez nutnosti výměny baterií. Omezená kapacita energetického zdroje, ať už na bázi primárních článků nebo akumulátorů, klade potom zvýšené nároky na minimalizaci spotřeby energie celého zařízení. Tu je nutné řešit na všech úrovních návrhu, technologií výroby čipů počínaje a návrhem programového vybavení konče. Výrobci vybavují mikrokontroléry řadou různých nízkopříkonových režimů. Jejich účelné využití vyžaduje odpovídající řešení programového vybavení, které může nízkopříkonové režimy mikrokontroléru ve větší nebo menší míře využívat.

V tomto článku je popsána metoda pro snížení spotřeby energie mikrokontroléru, který pracuje pod operačním systémem reálného času. Popsaná metoda je založena na maximalizaci času, po který může být mikrokontrolér v přepnut do nízkopříkonového režimu tak, že se minimalizuje počet časových přerušení pro synchronizaci časové základny RTOS. Využívá hardwarové časovací obvody, které představují běžné vybavení většiny mikrokontrolérů.

2. PŘÍKON MIKROKONTROLÉRU

Příkon mikrokontroléru může být s určitým zjednodušením popsán vztahem:

$$P_m = C_e \cdot V_{CC}^2 \cdot f_{CLK} + I_L \cdot V_{CC} \ , \qquad (1)$$

kde P_m ... příkon mikrokontroléru,

C_e ... ekvivalentní kapacita, daná technologií

V_{CC} ... napájecí napětí mikrokontroléru,

f_{CLK} ... taktovací (hodinová) frekvence a

I_L ... svodový proud obvodu.

První (dynamický) člen uvedeného vztahu je v korespondenci s nábojem, který se přenáší přes ekvivalentní kapacity při přechodu mezi logickými úrovněmi a závisí proto na frekvenci hodinového signálu procesoru. Druhý (statický) člen je dán svodovým proudem obvodu I_L a je na frekvenci hodinového signálu nezávislý. Pokud mikrokontrolér pracuje s frekvencí hodinového signálu řádově srovnatelnou s maximální přípustnou pracovní frekvencí, je hodnota dynamického členu mnohonásobně větší než hodnota statického členu a příkon mikrokontroléru je ovlivňován především pracovní frekvencí a velikostí napájecího napětí. Při přepnutí do nízkopříkonového režimu, ve kterém je frekvence hodinového signálu snížena na hodnotu typicky řádu desítek kHz nebo je hodinový signál procesorové jednotky úplně odpojen, nabývá na významu statický člen a příkon mikrokontroléru je dán především velikostí napájecího napětí.

Ze vztahu (1) tedy vyplývá, že příkon mikrokontroléru roste lineárně s hodinovou frekvencí. Snížením hodinové frekvence by proto bylo možné příkon mikrokontroléru přiměřeně snížit.

Uvažujme nyní určitou danou úlohu τ, která je periodicky spouštěná s periodou T_τ. Předpokládejme dále, že při každém spuštění je k provedení úlohy zapotřebí n_τ taktů hodinového signálu procesoru. Energie, spotřebovaná procesorem v jedné spouštěcí periodě je potom

$$E_\tau = C_e \cdot V_{CC}^2 \cdot n_\tau + I_L \cdot V_{CC} \cdot T_\tau \ , \qquad (2)$$

kde E_τ ... energie potřebná k výpočtu úlohy τ.

Je zřejmé, že při zachování konstantního napájecího napětí V_{CC} energetické nároky procesoru nezávisí na frekvenci, nýbrž jen na počtu taktů hodinového signálu. Samotné snížení hodinové frekvence potom nepřináší snížení energetické náročnosti při výpočtu dané úlohy.

Označme nyní výpočetní čas úlohy τ jako X_τ, kde

$$X_\tau = \frac{n_\tau}{f_{CLK}} \ . \qquad (3)$$

Při uvážení výše uvedených skutečností může v závislosti na konkrétním řešení může nastat několik základních situací:

elektrorevue
ISSN 1213 - 1539

a) Výpočetní čas úlohy $X_\tau < T_\tau$. V čase, zbývajícím po dokončení úlohy k jejímu dalšímu spuštění, je procesorová jednotka ve stavu dynamického čekání, tj. pracovní frekvence procesoru není snížena a procesor tráví čas v čekací smyčce. Této situaci odpovídá obr. 1a. Příkon mikrokontroléru je dán vztahem (1). Z hlediska energetické náročnosti se jedná o nejméně vhodné řešení.

b) Výpočetní čas úlohy $X_\tau < T_\tau$. V čase, zbývajícím po dokončení úlohy k jejímu dalšímu spuštění, je procesorová jednotka přepnuta do nízkopříkonového režimu, kdy je odpojena od zdroje hodinového signálu (viz obr. 1b). Střední příkon mikrokontroléru

$$\overline{P}_m = C_e \cdot V_{CC}^2 \cdot \frac{n_\tau}{T_\tau} + I_L \cdot V_{CC} \qquad (4)$$

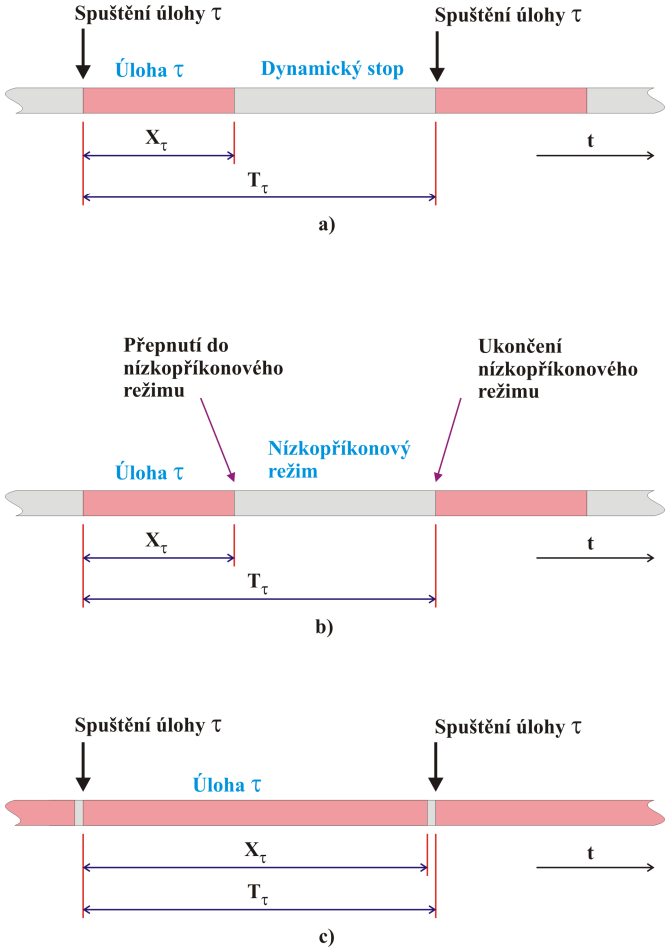
nezávisí na frekvenci hodinového signálu procesoru.

c) Frekvence hodinového signálu f_{CLK} je vhodně upravena (snížena) na hodnotu $f_{CLK\,MIN}$ tak, aby platilo*) $X_\tau = T_\tau$ (viz obr. 1c). Procesorová jednotka mikrokontroléru proto nemusí přecházet do čekacího stavu. V této situaci lze využít skutečnosti, že maximální pracovní frekvence mikrokontroléru závisí na napájecím napětí (větší frekvence vyžaduje větší napájecí napětí). Při snížení frekvence na hodnotu lze snížit též hodnotu napájecího napětí na hodnotu $V_{CC\,MIN}$ ($V_{CC\,MIN} < V_{CC}$) a v souladu se vztahem (1) snížit příkon mikrokontroléru. Tato technika, v literatuře označovaná jako DVS (Dynamic Voltage Scaling) je jednou ze výchozích technik pro návrh metod na minimalizaci energetických nároků mikrokontroléru.

*) Poznámka: V uvedených zjednodušených vztazích nepočítáme s časem, který je spojen s administrativou operačního systému (plánování úloh), obsluhou přerušení hardwarových časovačů, přepínáním do/z nízkopříkonového režimu a podobně. Jako výpočetní čas X_τ úlohy musí být uvažován nejhorší případ, tj. nejdelší možný výpočetní čas úlohy τ.

3. Nízkopříkonové režimy mikrokontroléru

Ve snaze vyhovět požadavkům na minimalizaci příkonu elektronických zařízení zavádí celá řada výrobců mikrokontrolérů speciální nízkopříkonové řady. Jejich architektura je navržena tak, aby umožnila v závislosti na řešené úloze a okamžité situaci nastavení takového režimu, který je z hlediska spotřeby energie optimální. Základní technikou je možnost volby zdroje hodinového signálu a jeho frekvence nezávisle pro procesorovou jednotku a pro jednotlivé periferní obvody na čipu mikrokontroléru. Nevyužité periferní jednotky mohou mít hodinový signál úplně odpojen. Na obr. 2 je znázorněno řešení obvodů pro generování hodinového signálu, které je v různých modifikacích obvyklou součástí nízkopříkonových mikrokontrolérů.



Obr. 1: Periodicky spouštěná úloha

Obecné blokové schéma obvodů pro generování hodin je na obr. 2. Hodinový signál může být generován některým ze 4 oscilátorů. Každý oscilátor lze podle potřeby programově zapínat nebo vypínat. Vypínání oscilátorů je kontrolováno pomocnými obvody, které brání vypnutí oscilátoru, který je aktuálně používán k synchronizaci procesoru nebo periferií. Základní bloky generátoru hodinových signálů jsou následující:

- Rychlý krystalový oscilátor s vnějším krystalem slouží obvykle jako hlavní zdroj hodinového signálu pro procesor. Jeho nevýhodou je poměrně pomalý náběh oscilací (řádově milisekundy). Pokud je v době čekání procesorové jednotky z důvodu úspory energie odpojen a zapíná se například pro obsluhu přerušení, může být pomalý náběh oscilátoru na závadu.
- Pomalý krystalový oscilátor (tzv. „subclock oscillator“) se kromě synchronizace procesoru v pomalém módu používá obvykle pro synchronizaci obvodů hodin reálného času (je-li jimi mikrokontrolér vybaven) nebo jiných časovačů, které slouží jako zdroj hodinových tiků pro softwarově realizovanou časovou zá-

elektrorevue
ISSN 1213 - 1539

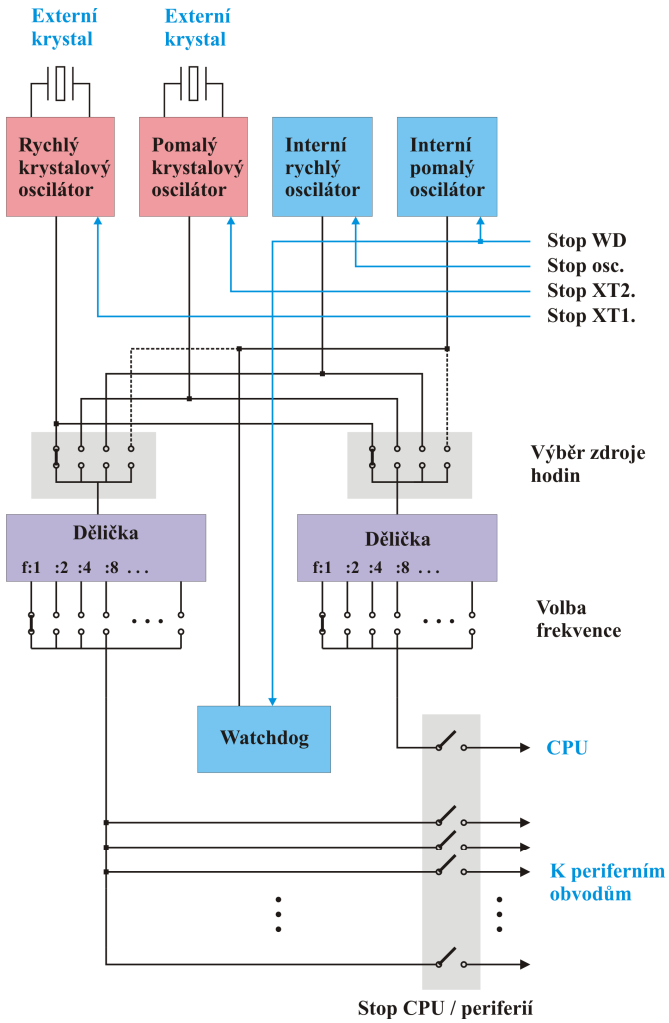
kladnu operačního systému. Jejich frekvence se proto často volí 32,768 kHz.

- Interní rychlý oscilátor může být alternativně použit jako hlavní zdroj hodinového signálu pro procesor. Mívá obvykle poněkud horší stabilitu frekvence ve srovnání s krystalovým oscilátorem. Jeho frekvenci lze v určitých mezích programově nastavovat až do maximální hodnoty, která je přípustná pro procesor. Doba náběhu oscilací je relativně velmi krátká (jednotky mikrosekund), takže v situaci, kdy je oscilátor zapínán pro obsluhu přerušení, může být latence systému udržena v přijatelných mezích.
- Interní pomalý oscilátor pracuje obvykle na frekvencích řádově 10 – 100 kHz. Je důležitý především pro aplikace, které vyžadují nezávislý zdroj hodinového signálu pro watchdog. Oscilátor se potom zapíná nebo vypíná současně s tímto obvodem. Alternativně může být u některých mikrokontrolérů použit i jako zdroj hodin pro procesor a pro periferní obvody.
- Obvody pro výběr zdroje hodin umožňují programově volit oscilátor, který bude použit jako zdroj hodin pro procesor nebo pro periferní obvody. Přepínání je možné kdykoliv za chodu. Po zapnutí krystalového oscilátoru je ovšem nutné nejprve počkat na ustálení oscilací a teprve poté jej lze připojit jako zdroj hodin.
- Dělička frekvence a za ní následující výběrové obvody dovolují použít pro synchronizaci procesoru a periferních obvodů buď základní frekvenci zvoleného oscilátoru, nebo některou z nižších frekvencí odebíranou ze zvoleného stupně děličky. Výběrové obvody mohou být samostatné pro nezávislou volbu frekvence každé z periferií nebo mohou být pro všechny periferie společné.

Popsaný generátor hodinových signálů je velmi flexibilní. Programově řízené vypínání oscilátorů a volba frekvencí pro procesor a pro jednotlivé periferie umožňuje za chodu řídit výkon mikrokontroléru tak, aby odpovídal okamžité situaci. V případě, kdy programové vybavení pracuje pod RTOS, je obvykle generátor hodin řízen příslušnými moduly operačního systému.

4. RTOS A PLÁNOVÁNÍ ÚLOH

Použití operačního systému reálného času přináší programátorovi řadu výhod. Dostává k dispozice fungující a ověřené prostředky pro plánování jednotlivých úloh, jejich synchronizaci a předávání zpráv mezi úlohami. Pro použití v mikrokontrolérech s velmi omezenými zdroji (velikost kódové a datové paměti, výkon procesoru) existují RTOS, jejichž binární kód vyžaduje pro uložení do paměti méně než 10 kB a vyžadují 2 – 4 kB datové paměti (při nasazení na „běžném“ 16bitovém mikrokontroléru). Jsou tedy snadno použitelné i v nízkopříkonových aplikacích. Při praktickém použití se zdrojový kód RTOS překládá společně se zdrojovým kódem aplikace a výsledný binární kód se zavádí do paměti mikrokontroléru. Dostupnost zdrojového kódu operačního systému dává programátorovi možnost v případě potřeby upravit kód a implementovat tak různé speciální funkce. V tomto článku popsaná úprava předpokládá možnost tohoto způsobu práce.



Obr. 2: Typické uspořádání zdroje hodinového signálu mikrokontroléru

Plánování a spouštění úloh pod RTOS probíhá na základě priorit. Každá z úloh má přidělenou pevnou nebo proměnnou prioritu. Použije-li běžící úloha volání systémové služby, která může ovlivnit její stav nebo stav některé jiné úlohy (práce se semaforem, vložení zprávy do schránky, ...), je spuštěn plánovač který vyhodnotí situaci a spustí připravenou úlohu, která má v daném okamžiku nejvyšší prioritu. Pro podrobnější vysvětlení funkcí RTOS odkazujeme čtenáře například na literaturu [1].

Důležitou funkcí RTOS je údržba systémového času a práce s časovači. Jeho koncepce je následující:

Některý z hardwarových časovačů mikrokontroléru je při startu systému inicializován tak, aby generoval periodická přerušení (hodinové tiky) s pevně nastavenou periodou. Ta se obvykle volí v rozsahu 1 – 100 ms podle celkové koncepce systému. V obslužném programu přerušení

elektrorevue
ISSN 1213 - 1539

je potom aktualizována hodnota systémového času, časovačů jednotlivých úloh a programově realizovaných časovačů.

V řídicím bloku každé úlohy je proměnná, která reprezentuje její časovač. Je-li úloha ve stavu pozastavení, určuje hodnota časovače dobu, po kterou má úloha v tomto stavu setrvat. Časovač je dekrementován obslužnou procedurou hardwarového časovače. Po dosažení nulové hodnoty je daná úloha zařazena mezi připravené úlohy.

Kromě časovačů jednotlivých úloh existují v RTOS další programově ošetřované časovače, které mohou jednotlivé úlohy podle potřebu využívat například k měření časových intervalů, volání určité funkce („callback“) ve stanovené době apod.

Protože aktualizace stavu časovačů může způsobit změny ve stavu jedné nebo více úloh (například tím, že úloha je po uplynutí stanované doby pozastavení zařazena mezi připravené úlohy), spouští se na konci obslužného programu časového přerušení plánovač úloh, který spustí úlohu s nejvyšší prioritou.

Při plánování úloh může nastat situace, kdy není žádná z úloh připravena ke spuštění. Příkladem může být velmi jednoduchý případ, kdy je v systému jediná úloha, která je periodicky spouštěna časovačem. Po dokončení čeká úloha na další spuštění po uplynutí stanovené doby a není proto do té doby připravena ke spuštění. Pro řešení podobné situace je v RTOS zavedena speciální úloha („Idle Task“), která má ze všech úloh nejnižší prioritu a je vždy připravena ke spuštění. Plánovač potom spouští tuto úlohu, kdykoliv je třeba využít čas procesoru, který nemůže být využit žádnou aplikační úlohou.

Součástí aplikačního programového vybavení mohou být procedury pro obsluhu asynchronních přerušení, generovaných například řadiči periferií. Při psaní obslužných procedur přerušení pod RTOS musí být dodržena určitá pravidla. Obvykle se v obslužné proceduře provede jen nejnutnější ošetření zdroje přerušení (např. periferního řadiče). Další kroky jsou potom provedeny v samostatné úloze, která je z obslužné procedury spuštěna vhodným synchronizačním mechanismem, například zasláním zprávy nebo nastavením semaforu. Na konci obslužné procedury se potom spouští plánovač úloh, který podle situace buď vrátí výpočet do přerušené úlohy nebo spustí jinou připravenou úlohu s vyšší prioritou.

5. Navrhovaná úprava RTOS pro snížení příkonu

Metody snížení spotřeby energie aplikací pracujících pod RTOS jsou v současné době předmětem intenzivního výzkumu. Zmíníme zde některé základní techniky, které jsou popsány v literatuře. Detailní studii energetických nároků několika různých RTOS lze najít například v literatuře [2].

Jak vyplývá z popisu funkcí RTOS uvedených v předchozí kapitole, je „nadbytečný“ výpočetní výkon procesoru utrácen tak, že plánovač spustí úlohu Idle. Z hlediska spotřeby energie je proto této úloze třeba věnovat pozornost. Jednoduchým řešením je přepnutí procesoru do vhodného nízkopříkonového režimu. V úloze Idle může k novému spuštění plánovače a následnému přepnutí úloh dojít z některé z následujících příčin:

- přerušením od systémového časovače (hodinovým tikem),
- asynchronním přerušením.

Procesor proto musí být přepnut do nízkopříkonového režimu, který lze ukončit přerušením.

Pro další minimalizaci energetických nároků byla navrženo množství metod. Některé z prezentovaných algoritmů (např. [3]) pracují na základě předběžného (off-line) výpočtu minimální přípustné hodinové frekvence pro procesor, při které je zaručeno provedení všech naplánovaných úloh v požadovaném čase. Metody přitom vycházejí ze znalosti maximální výpočetní doby jednotlivých úloh. Řada dalších algoritmů ([4], [5], [6] a další) je orientována na řízení výkonu procesoru podle okamžité potřeby změnou frekvence hodinového signálu a následnou úpravu napájecího napětí na minimální možnou hodnotu. V práci [7] jsou analyzována kritéria optimality plánování a prezentovány algoritmy pro optimální plánování úloh v takzvaných „soft real-time“ systémech s limitovanou dostupnou energií, která není dostatečná k provedení všech naplánovaných úloh.

Řada v literatuře popsaných metod využívá poměrně složité plánovací algoritmy. Tím se zvyšují administrativní nároky RTOS a zhoršuje se celková energetická bilance.

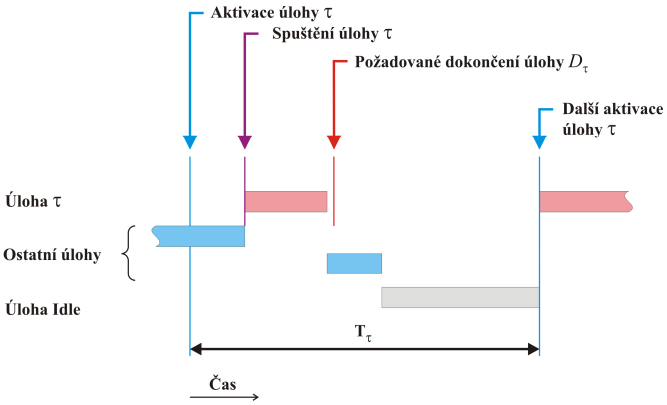
Uvedené metody vycházejí z předpokladu, že požadovaná doba dokončení úlohy je přibližně srovnatelná s periodou spouštění úlohy T_τ. V řadě případů se přímo předpokládá $D_\tau = T_\tau$. Výpočetní čas úlohy je upraven snížením frekvence hodinového signálu tak, aby úloha byla provedena v čase co možná nejbližším D_τ. Ve zbývajícím čase $T_\tau - X_\tau$ se spouští některá z dalších úloh nebo úloha Idle (viz obr. 3). Kritérium dokončení úlohy v požadovaném čase musí být pochopitelně dodrženo pro všechny úlohy, které jsou v systému zavedené.

V praxi se vyskytuje řada případů, kdy je $D_\tau << T_\tau$. Perioda spouštění úloh přitom může být velmi velká, řádově sekundy, minuty nebo i větší. V činnosti procesoru se potom střídají fáze relativně intenzivní výpočetní zátěže s fázemi, kdy není plánováno spuštění žádné aplikační úlohy a plánovač proto spouští úlohu Idle. Celková energetická bilance je v této situaci z velké části dána právě energetickou náročností dlouhé fáze, kdy se provádí pouze úloha Idle.

Jak vyplývá z výše uvedeného popisu, je úloha Idle pravidelně přerušována přerušením od systémového časovače. Procesor proto musí být převeden z nízkopříkonového do aktivního režimu. Po dokončení obslužné procedury, není-li plánováno spuštění jiné úlohy, se procesor vrací zpět do nízkopříkonového režimu (viz obr. 4). Tento re-

elektrorevue
ISSN 1213 - 1539

žim budeme dále označovat jako režim IDLE1. Je-li v aktivním režimu generován signál pro procesor krystalovým oscilátorem, není při poměrně krátké periodě přerušovacích signálů účelné jeho vypínání. Rozběh krystalového oscilátoru vyžaduje poměrně dlouhou dobu a je poměrně energeticky náročný.



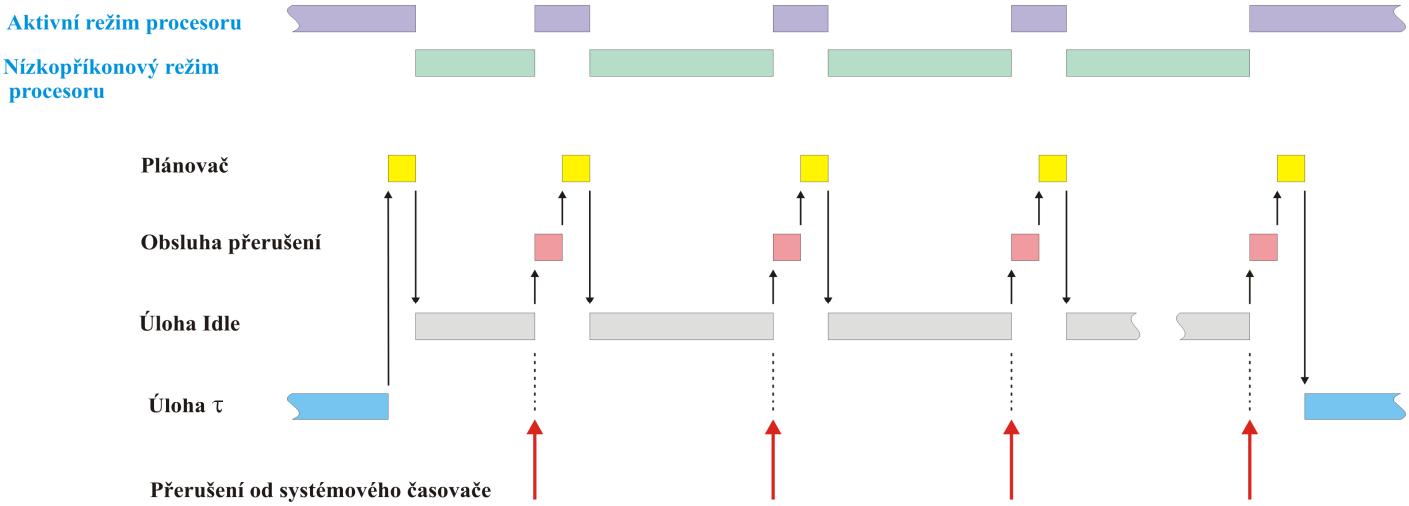
Obr. 3: Periodicky spouštěná úloha a úloha Idle

Navrhovaná úprava RTOS vychází ze situace popsané v předchozím odstavci. Umožňuje minimalizaci energetických nároků mikrokontroléru v případě, kdy jsou spouštěny úlohy s velmi dlouhou periodou, přičemž úlohy samotné vyžadují poměrně velký výpočetní výkon.

Předpokladem pro použití navržené metody je možnost kaskádního zapojení hardwarových časovačů/čítačů mikrokontroléru tak, jak je znázorněno na obr. 5. Výstupní signál ze systémového časovače A, který je běžně využíván pro generování periodických přerušení, je zaveden na vstup čítače B. Ten je nakonfigurován tak, že generuje přerušení po určitém předem nastaveném počtu pulsů na vstupu. Vstupní hodinový signál pro časovač A je generován pomalým krystalovým oscilátorem (tzv. subclock oscillator). Většina mikrokontrolérů je vybavena vícefunkčními čítači/časovači, které podobné propojení umožňují. Jako alternativní řešení lze využít propojení vně mikrokontroléru: časovač A musí být potom konfigurován tak, aby kromě přerušení generoval pulsy na některém vývodu mikrokontroléru. Ten je vně mikrokontroléru propojen se vstupním pinem čítače B, nakonfigurovaného do funkce čítače vnějších pulsů. Princip metody je následující:

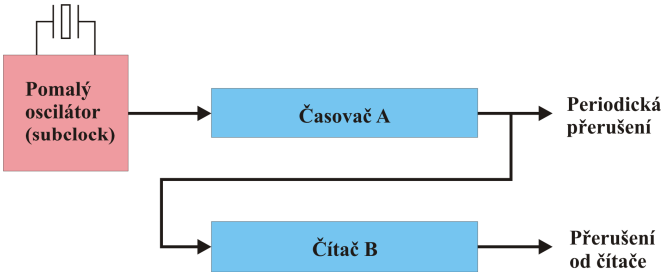
Není-li v systému žádná připravená úloha, spustí se úloha Idle. Přitom se provedou (kromě běžných operací, které souvisí s přepínání kontextu) následující kroky:

- Přečte se stav jednotlivých softwarových časovačů. Z hodnot ve všech aktivních časovačích se určí minimální hodnota N_{MIN}, která udává počet tiků systémového časovače, po jejichž uplynutí bude provedena plánovaná akce. Je-li N_{MIN} menší než určitá stanovená hodnota N_E, spustí se úloha Idle a je jí předán parametr, který zajistí její provádění běžným způsobem IDLE1 (tj. bez navržené úpravy). Je-li $N_{MIN} \geq N_E$, pokračuje se podle následujícího postupu:
- Hodnota N_{MIN} se zapíše do čítače B. Povolí se generování přerušení čítačem B a čítač B se odblokuje. Přerušení od systémového časovače A jsou zakázána.
- Spustí se úloha Idle, přičemž hodnota předaného parametru zajistí, že se procesor převede do nízkopříkonového režimu, ve kterém je vypnut rychlý oscilátor. Systémový časovač a případně další periferie mikrokontroléru využívají pomalý oscilátor. Tento režim budeme označovat jako IDLE2 (viz obr. 6).



Obr. 4: Režim IDLE1

elektrorevue
ISSN 1213 - 1539



Obr. 5: Kaskádní řazení časovačů/čítačů

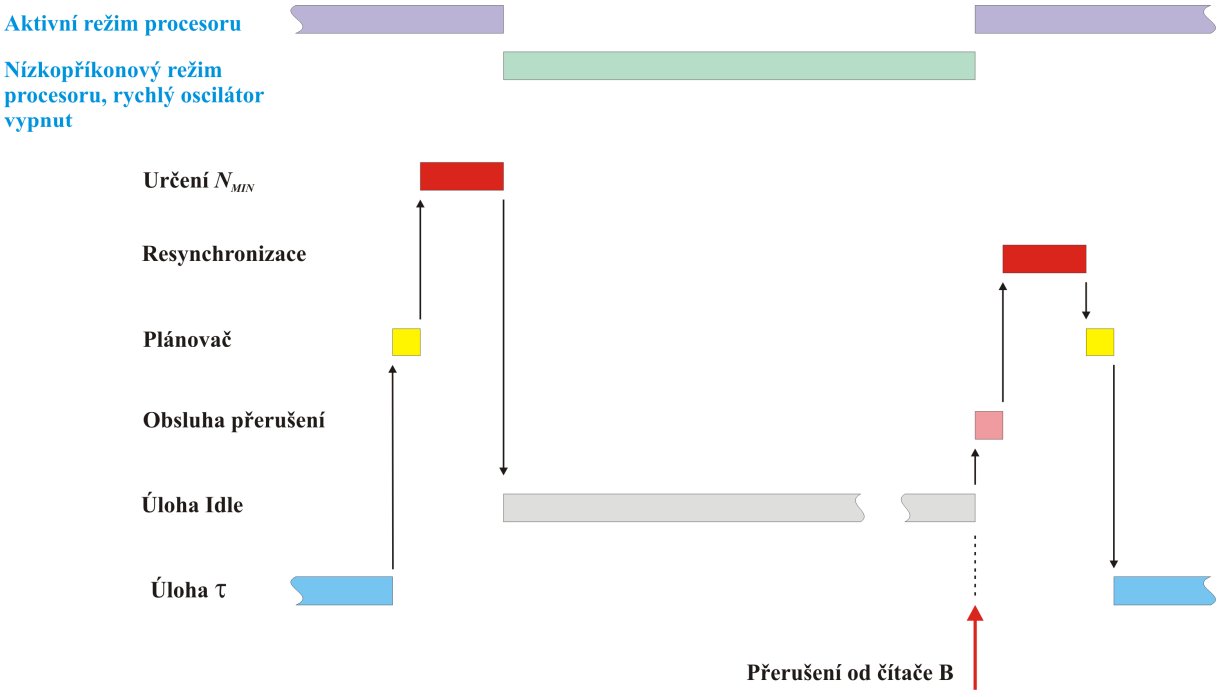
Po uplynutí N_{MIN} taktů systémového časovače A vygeneruje čítač B přerušení. Přerušení převede mikrokontrolér do aktivního režimu a v obsluze přerušení se provedou následující kroky:

- Restartuje se rychlý oscilátor. Podle typu mikrokontroléru se restart provede buď automaticky nebo na základě programového zákroku.
- Po rozběhu oscilátoru se resynchronizují softwarové časovače. Údaj časovačů se upraví odečtením hodnoty N_{MIN}.
- Povolí se generování přerušení pro systémový časovač A a zakáže se generování přerušení čítačem B.
- Spustí se plánovač. Další činnost RTOS probíhá normálním způsobem.

Nastane-li v době, kdy je spuštěna úloha Idle a mikrokontrolér pracuje v režimu IDLE2, asynchronní přerušení, je sled operací následující:

- Přerušením se procesor přepne do aktivního režimu. Poté se restartuje se rychlý oscilátor.
- Po rozběhu oscilátoru se resynchronizují softwarové časovače: Programově se přečte obsah čítače B. Byla-li z čítače B přečtena hodnota N_{RES}, upraví se údaj softwarových časovačů odečtením hodnoty N_{MIN} - N_{RES} .
- Obnoví se normální chod časovačů: povolí se generování přerušení pro systémový časovač A a zakáže se generování přerušení pro čítač B.
- Spustí se plánovač. Další činnost RTOS probíhá normálním způsobem.

Z uvedeného popisu vyplývá, že v časovém intervalu mezi spuštěním úlohy Idle a akceptováním přerušení od čítače B nebo akceptováním asynchronního přerušení je procesor v nízkopříkonovém režimu s odpojeným hodinovým signálem a je zastaven rychlý oscilátor. V této době neobsluhuje procesor periodická přerušení a jeho energetické nároky jsou proto minimální. Zjištění stavu časovačů před spuštěním úlohy Idle, změna konfigurace časovačů/čítačů A a B, rozběh rychlého oscilátoru a resynchronizace softwarových časovačů přináší dodatečné energetické nároky. Důležité je proto určení hodnoty N_E, která určuje minimální dobu setrvání systému v nízkopříkonovém režimu tak, aby bylo dosaženo energetické úspory.



Obr. 6: Režim IDLE2

elektrorevue
ISSN 1213 - 1539

Označíme-li

P_e ... příkon mikrokontroléru v normálním režimu, hodinový signál je odvozen od rychlého oscilátoru,

P_{lp1} ... příkon mikrokontroléru v režimu s odpojeným hodinovým signálem pro procesor, avšak s běžícím rychlým oscilátorem,

T_{int} ... čas potřebný k obsluze přerušení systémového časovače,

T_{tick} ... perioda přerušení od systémového časovače,

N_{idle} ... čas daný počtem tiků systémového časovače, které stráví procesor prováděním úlohy Idle,

E_{idle1} ... energie, spotřebovaná mikrokontrolérem při provádění úlohy Idle v režimu IDLE1,

dostaneme pro celkovou energii spotřebovanou mikrokontrolérem při provádění úlohu Idle v režimu IDLE1 následující vztah:

$$E_{idle1} = \left[T_{\text{int}} \cdot P_e + \left(T_{tick} - T_{\text{int}}\right) \cdot P_{lp1}\right] \cdot N_{idle} \quad . \qquad (5)$$

Vzhledem k tomu, že $T_{int} << T_{tick}$, lze výraz dále zjednodušit:

$$E_{idle1} = \left(T_{\text{int}} \cdot P_e + T_{tick} \cdot P_{lp1}\right) \cdot N_{idle} \quad . \qquad (6)$$

Je-li dále

P_{lp2} ... příkon mikrokontroléru v režimu s odpojeným hodinovým signálem pro procesor a se zastaveným rychlým oscilátorem,

T_{sync} ... doba, potřebná k analýze softwarových časovačů, rekonfiguraci čítačů/časovačů a rozběhu oscilátoru,

E_{idle2} ... energie, spotřebovaná mikrokontrolérem při provádění úlohy Idle v režimu IDLE2,

dostaneme za předpokladu stejného zjednodušení jako ve výrazu (6) celkovou spotřebu energie pro úlohu Idle v režimu IDLE2:

$$E_{idle2} = T_{sync} \cdot P_e + T_{\text{int}} \cdot P_e + T_{tick} \cdot P_{lp2} \cdot N_{idle} \quad . \qquad (7)$$

Popsaná metoda je efektivní, pokud platí $E_{idle1} > E_{idle2}$. Pro hraniční podmínku dostaneme

$$N_E = \left\lceil \frac{\left(T_{sync} + T_{\text{int}}\right) \cdot P_e}{T_{\text{int}} \cdot P_e + T_{tick} \cdot \left(P_{lp1} - P_{lp2}\right)} \right\rceil \quad , \qquad (8)$$

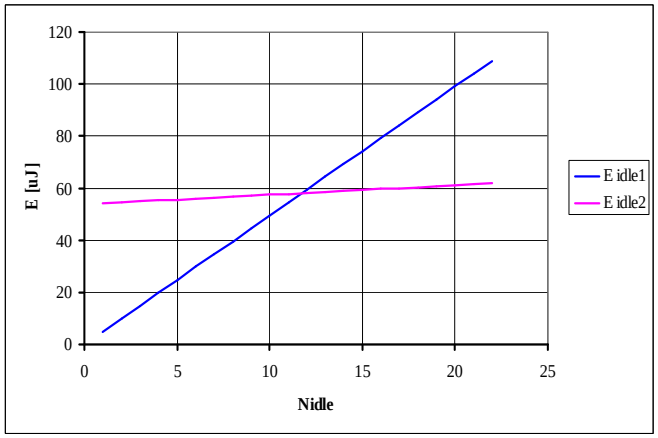
kde N_E je minimální počet tiků systémových hodin které musí procesor strávit v úloze Idle, má-li být režim IDLE2 energeticky výhodnější než režim IDLE1.

V grafu na obr. 7 je znázorněna celková vypočtená energie, kterou spotřebuje určitý konkrétní procesor pro provedení úlohy Idle v režimu IDLE1 a IDLE2. Hodnoty použité pro výpočet jsou uvedené v tab. 1.

Z grafu je zřejmé, že pro úlohu Idle kterou bude procesor vykonávat po dobu kratší než 12 tiků systémového časovače (v použitém příkladu odpovídá 120 milisekundám) je energeticky výhodnější režim IDLE1. Naopak pro provádění úlohy Idle po dobu větší nebo rovnou 12 tikům časovače je energeticky výhodnější režim IDLE2. Pro navrženou úpravu RTOS je proto vhodné použít hodnotu $N_E = 12$.

Parametr	Hodnota	Jednotky
P_{lp1}	0,36	mW
P_{lp2}	0,036	mW
T_{int}	0,18	ms
T_{sync}	7	ms
T_{tick}	10	ms

Tab. 1: Hodnoty pro výpočet spotřebované energie



Obr. 7: Graf energetických nároků v režimech IDLE1 a IDLE2

elektrorevue
ISSN 1213 - 1539

6. ZÁVĚR

Metody snižování energetické náročnosti mikrokontrolérem řízených elektronických zařízení musí vždy vycházet z konkrétních podmínek, za kterých bude zařízení provozováno. Nelze proto navrhnout univerzální koncepci, kterou by bylo možné aplikovat bez detailní analýzy režimu, ve kterém bude mikrokontrolér pracovat. V tomto článku prezentovaná metoda je vhodná především pro systémy, ve kterých se střídají relativně dlouhé periody nečinnosti s krátkými periodami velké výpočetní zátěže. Takové situace nastávají v řadě měřicích zařízení, prostředcích pro kontrolu a úpravu životního prostředí a podobně. Na rozdíl od řady jiných publikovaných metod je prezentovaná metoda velmi nenáročná na výpočetní čas a nezvyšuje proto výrazným způsobem administrativní nároky RTOS. Je proto vhodná i pro systémy s relativně málo výkonnými procesory, které jsou často používané v aplikacích s extrémními nároky na energetickou úspornost. Snížením jejich energetické náročnosti lze potom prodloužit dobu bezobslužného provozu v případě bateriového napájení.

PODĚKOVÁNÍ

Práce vznikla s grantovou podporou Ministerstva školství, mládeže a tělovýchovy České republiky - "University spec. research - 1311".

LITERATURA

[1] LI, Q., YAO, C.: *Real Time Concepts for Embedded Systems*. CMP Books, San Francisco 2003.

[2] BAYNES, K., COLLINS, C., FITERMAN, E., GANESH, B., KOHOUT, P., SMIT, C., ZHANG, T., JACOB, B.: *The Performance and Energy Consumption of Embedded Real-Time Operating Systems*. IEEE TRANSACTIONS ON COMPUTERS, VOL. 52, NO. 11, pp. 1454 – 1469. IEEE NOVEMBER 2003.

[3] PILLARI, P., SHIN, K.G.: *Real-time dynamic voltages scaling for low-power embedded operating systems.* In: Proceedings of the eighteenth ACM symposium on Operating system principles (SOSP), pp. 89 – 102. ACM 2001.

[4] JEJURIKAR, R., GUPTA, R.: *Optimized Slowdown in Real-Time Task Systems*. In: Proceedings of the 16th Euromicro Conference on Real-Time Systems (ECRTS '04), pp. 155 – 164. IEEE 2004.

[5] EJLALI, A., SCHMITZ, M.T., AL-HASHIMI, B.M., MIREMADI S.G., ROSINGER, P.: *Energy efficient SEU-tolerance in DVS-enabled real-time systems through information redundancy*. In: Proceedings of the 2005 international symposium on Low power electronics and design 2005, San Diego, CA, USA August 08 - 10, 2005.

[6] AYDIN, H., MELHEM, R., MOSSE, D., MEJI´A-ALVAREZ, P.: *Power-Aware Scheduling for Periodic Real-Time Tasks*. IEEE TRANSACTIONS ON COMPUTERS, VOL. 53, NO. 5, pp. 584 – 600. IEEE MAY 2004.

[7] ALENAWY, T. A., AYDIN, H.: *Energy-Constrained Scheduling for Weakly-Hard Real-Time Systems*. In: Proceedings of 26th IEEE International Real-Time Systems Symposium (RTSS'05), pp.376 - 385, IEEE 2005.