Lightweight bootloader pro mikrokontroléry NXP s jádrem S08

Jiří Žahour, Jindřich Křivka, Kamil Kosturik
Katedra aplikované elektroniky a telekomunikací
Fakulta elektrotechnická
Západočeská univerzita v Plzni
zahourj@kae.zcu.cz, jkrivka@kae.zcu.cz, kosturik@kae.zcu.cz

Lightweight bootloader for NXP microcontrolers with S08 core

Abstract – **This paper describes a simple bootloader for the NXP microcontrollers. The first part deals with basic principles, memory layout and system interrupts. In the second part the implementation of the bootloader via CAN bus is described. Implementation also includes a transport protocol optimized for a S19 files.**

Keywords – NPX, Freescale, S08, bootloader, CAN, transport protocol, flash

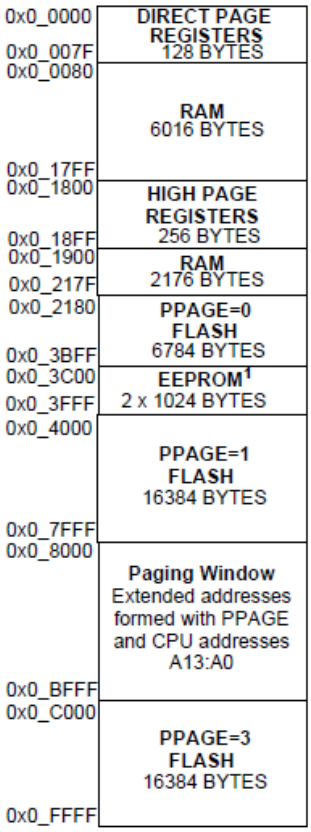
I. ÚVOD

Bootloader je velmi vhodným doplňkem pro embedded aplikace, který umožňuje přeprogramování firmwaru přímo v cílové aplikaci bez nutnosti rozebírání a připojování externího programátoru. Cílem tohoto článku je popsat princip funkce bootloaderu vyvinutého pro mikrokontroléry firmy NXP (dříve Freescale) s jádrem S08. Tento bootloader byl následně použit v praktické aplikaci řídících jednotek komunikujících po sběrnici CAN.

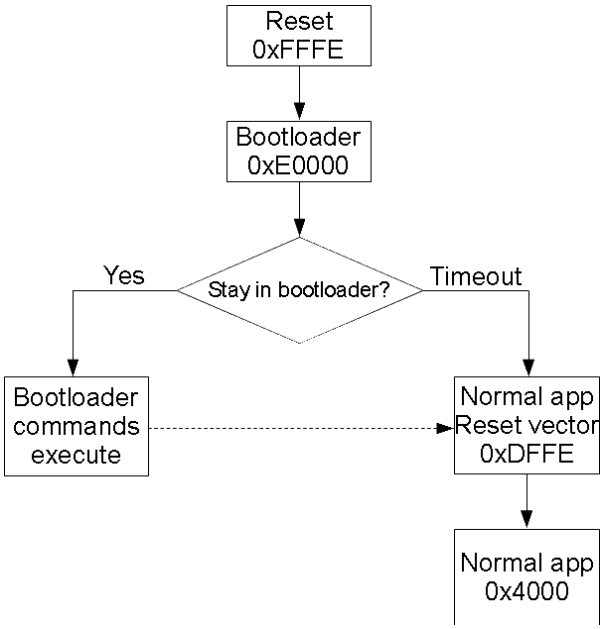
II. ZÁKLADNÍ PRINCIP

Bootloader je vlastně samostatný program, který má v případě potřeby za úkol přepsat data v programové paměti. Je nutné, aby byl od vlastní aplikace oddělen, protože jinak by případná chyba při nahrávání nového firmware mohla způsobit nefunkčnost celého programu včetně bootloaderu. Tato chyba by tedy znemožnila další aktualizaci firmwaru jinak, než pomocí programátoru.

Příklad uspořádání paměti mikrokontroléru s jádrem S08 je zobrazena na obrázku I. Na samém konci paměti se nachází „reset vector“ (na obrázku adresa 0xFFFE). Mikrokontrolér si po resetu z této adresy vezme adresu začátku samotného programu, který následně začne vykonávat. Ve standartním nastavení, se hned před „reset vektorem“ nachází tabulka vektorů přerušení, která má délku 0x80. Na rozdíl od reset vektoru, není její pozice pevná a lze ji přesunout. Mikrokontrolér umožňuje před reset vektor vklínit oblast, kterou lze uzamknout proti přepisu. Velikost této oblasti je nastavitelná po blocích o velikosti 8 kB. Nastavení oblasti lze za běhu změnit jen 1x, a to ihned po resetu mikrokontroléru. Jakékoliv další změny při běhu programu již možné nejsou. Takto uzamčená oblast se s výhodou použije pro nahrání bootloaderu, který tak bude chráněn proti jakémukoliv pokusu o jeho změnu z běžícího programu.



Obrázek I. **Příklad programové paměti [3]**



Obrázek II. **Blokové schéma průběhu bootování**

Celý systém je vhodné koncipovat tak, aby po resetu mikrokontroléru byl puštěn nejprve bootloader, ze kterého se následně pouští hlavní aplikace. Toto uspořádání zajistí možnost přeprogramování i v případě, že již byla nahrána nefunkční aplikace. Vývojový diagram na obrázku II zobrazuje postup programu po resetu mikrokontroléru.

Jistou komplikací může být systém přerušení. Lokalizace tabulky vektorů pro přerušení lze nastavit pouze po resetu mikrokontroléru. Proto nelze mít 2 tabulky pro přerušení v bootloaderu a pro normální aplikaci a při přeskoku mezi aplikacemi mezi nimi „přepínat“. Musíme tedy nadefinovat jednu pevnou tabulku. Existují 2 možnosti jak se s tímto problémem vypořádat.

První možnost je se obejít v bootloaderu bez přerušení a umístit tabulku vektorů do oblasti těsně před „chráněnou oblastí“. Tato možnost je pochopitelně velmi snadná na implementaci, ovšem přináší určité omezení pro aplikaci bootloaderu.

Druhou možností je nadefinovat tabulku vektorů v chráněné oblasti a napsat obsluhy přerušení pro každý vektor. V každé této funkci, by se na základě jedné proměnné rozhodlo, zda je mikrokontrolér v režimu bootloaderu, nebo normální aplikace. V prvém případě by zde byla implementována funkce dle potřeb bootloaderu. Pokud by mikrokontrolér byl v režimu hlavní aplikace, následoval by odskok programu do oblasti tabulky vektorů „před chráněnou oblast“, kde by byli umístěny adresy funkcí přerušení hlavní aplikace.

III. UPDATE FIRMWARU POMOCÍ SBĚRNICE CAN

Hlavní motivací vývoje tohoto bootloaderu bylo umožnění aktualizace firmwaru v řídicích jednotkách komunikujících po sběrnici CAN. Vzhledem k jednoduchosti zpracování CAN zpráv, není nutné v bootloaderu používat přerušení. Proto byla tabulka vektorů přerušení umístěna před „chráněnou oblast paměti“, tedy dle popisu výše byl zvolen jednodušší způsob.

Pokud po napsání zdrojového kódu hlavní aplikace program přeložíme, výstupem překladače bude soubor S19, který obsahuje ASCII záznamy. Každý tento záznam udává svůj typ, délku, adresu kam mají být data nahrána, samotná data a kontrolní součet. Obecně může být záznam dlouhý až 39 bytů, z toho 32 datových. [1] [2] Mikrokontrolér umožňuje programovat paměť flash dvěma způsoby – po bytech, nebo v blocích po 32 bytech. „Blokový“ způsob programování je značně rychlejší, kdy zápis 32 bytů trvá kratší dobu, než zápis 1 bytu prvním způsobem, proto je vhodné využívat v bootloaderu „blokový mód“. [3]

Než jsou data do programové paměti bootloaderem zapsána, je nutné nejprve dodat data pro instrukci blokového zápisu. Je obecně známo, že přenosový paket CAN může přenášet maximálně 8 datových bytů. Není tedy možné blok dat přenést v jedné CAN zprávě. Na odesílací straně musí být data rozdělena do příslušného počtu zpráv a v bootloaderu poté opět správně spojena. Pro tento účel zde byl implementován následující „transport protocol“.

Ve vlastních datech CAN zprávy je vytvořen jakýsi „paket“, který vždy obsahuje hlavičku a užitečné data. Jsou zde implementovány 3 typy zpráv:

- Jednoduchý paket – slouží pro přenos zpráv dlouhých 7 bytů nebo méně
- První paket – slouží pro první paket při přenosu zpráv delších jak 7 bytů
- Následný paket – slouží pro každý další paket ve zprávách delších jak 7 bytů

Typ paketu	**Byte 0**		**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
Jedno-duchý paket	Typ 00	Dél-ka	Data	Data	Data	Data	Data	Data	Data
První paket	Typ 01	Dél-ka	Délka	Data	Data	Data	Data	Data	Data
Následný paket	Typ 02	SN	Data	Data	Data	Data	Data	Data	Data

Obrázek III. **Struktura jednotlivých paketů transport protokolu**

Vysílací stanice vždy odesílá pomocí tohoto protokolu vždy celý S19 záznam. V bootloaderu je záznam postupně skládán v paměti RAM. Přenos funguje v režimu „ping pong“. Vysílací stanice pošle paket, bootloader odešle zprávu potvrzující přijetí a

tím si vyžádá další data. Tento postup se stále opakuje, dokud není přijat celý S19 záznam. Poté je u tohoto záznamu zkontrolován kontrolní součet. Pokud vyjde správně, tak jsou data zapsány do paměti flash. Následuje odeslání zprávy potvrzující zápis, a postup se může opakovat, dokud nedojde k nahrání všech S19 záznamů.

Je nutné podotknout, že před programováním je třeba vymazat stará data uložená v mikrokontroléru. Díky „zamčené oblasti“ kde se nachází bootloader se s výhodou použije instrukce kompletního smazání flash. Tato instrukce provede vymazání na celé nechráněné oblasti paměti, tedy nedotkne se bootloaderu. Výhodou této metody oproti mazání po blocích je rychlost.

IV. ZÁVĚR

Popisovaný bootloader byl vyvinut především pro aplikaci řídících jednotek komunikujících po sběrnici CAN. Tyto jednotky jsou použity v prototypových systémech pro redukci škodlivých emisí výfukových plynů spalovacích motorů. Popsaný princip lze nicméně využít pro různé aplikace využívající mikrokontrolér s jádrem S08.

PODĚKOVÁNÍ

Tento článek vznikl za podpory interního projektu na podporu studentských vědeckých konferencí SVK1-2016-006 a projektu SGS-2015-002: Moderní metody řešení, návrh a aplikace elektronických a komunikačních systémů. Dále bych chtěl poděkovat celému organizačnímu týmu konference v Nečtinech za příkladně odvedenou práci, která může být předkládána jako vzor nastávající generaci vědeckých pracovníků.

LITERATURA

[1] Motorola S-records [online]. [cit. 2016-09-20]. Dostupné z: http://www.amelek.gda.pl/avr/uisp/srecord.htm

[2] Wikipedia contributors. SREC (file format). Wikipedia, The Free Encyclopedia [online]. May 12, 2016, 22:04 UTC. Dostupné z: https://en.wikipedia.org/w/index.php?title=SREC_(file_format)&oldid=719967012. cit. 2016-09-20.

[3] Freescale Semiconductor, MC9S08DZ60 Rev. 4[online]. 6/2008 (2013-08-20). Dostupné z: http://cache.freescale.com/files/microcontrollers/doc/data_sheet/MC9S08DZ60.pdf?fpsp=1&WT_TYPE=Data%20Sheets&WT_VENDOR=FREESCALE&WT_FILE_FORMAT=pdf&WT_ASSET=Documentation