čo, ako, prečo .....

Zabudnite na to, že naprogramujete aplikáciu bez jedinej chyby, alebo bez nutnosti neskoršej úpravy,
či už vplyvom zmeny časti hardwéru, vylepšenia funkčnosti, alebo frflaním investora, že on to chce tak - a - tak.
Jedine teda ak používate OTP, EPROM, alebo inú elektricky neprepisovateľnú pamäť pre svoju aplikáciu (dierny štítok...).
V tom prípade ďalej ani nečítajte.

My ostatní si hlavne s príchodom FLASH technológie do sveta MCU a pamätí, nechávame možnosť ďalšej aktualizácie našich aplikácii.
A čím komfortnejšie a jednoduchšie to bude, tým lepšie.

Chceme, aby aplikácia bola upgradovateľná čo najjednoduchšie: bez nutnosti otvárať, rozoberať, alebo iným pracným spôsobom sa dostávať k ISP/JTAG/.. konektoru patriacemu procesoru, alebo k pamäti.
Nehľadiac na to, že nie všade sa dá dostať s pécečkom či notebookom a s pripojeným programátorom.

Ak súhlasíte, nastal čas aplikovať "bootloader".

Čo je to bootloader?

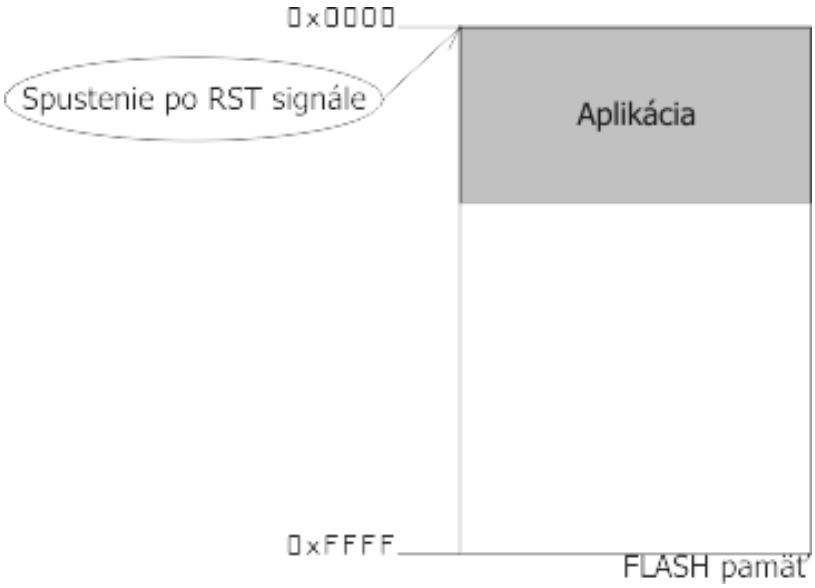
Už z názvu vyplýva, že je to niečo čo zabezpečí nahratie (Loading) niečoho pri spustení (at boot).

Je to program, ktorý spúšťame ešte pred spustením samotnej aplikácie. Označuje sa tiež aj ako "BootStrap"
Tento program nám môže zabezpečovať rôzne funckie od nastavení registrov HW až po testovanie HW, závisí to od použitej platformy.
Proces spúšťania a vykonávania bootloadera, sa označuje aj ako "Bootstrap sekvencia, bootstrapping".
Pri komerčných PC, bootloader napríklad zabezpečuje výber Operačného systému (LILO, GRUB).
Pri embedded systémoch, je hlavne zaujímavá možnosť aktualizovať aplikáciu vo Flash pamäti - "preflešovanie" (U-Boot).

Vo finále sa s bootloaderom dajú robiť aj ďalšie veci ako napríklad čítať/modifikovať obsah EEPROM, konfigurovať systémové parametre zariadenia, alebo testovať zariadnie.
Tieto ďalšie funkcie už ale závisia od možností procesora, rozložením kódovej pamäte a jej veľkosti.

Ako sa bootloader aktivuje?

Pozrime sa, na rozloženie v pamäti kódu (Flash) "štandardnej" aplikácie (adresy 0x0000 a 0xFFFF sú len pre označenie začiatku a konca pamäte):



Pri aktivovaní signálu reset mikroprocesora, sú vynulované (alebo nastavené na default hodnoty) niektoré jeho registre.
Po uvoľnení signálu reset, sa začne vykonávať program a to inštrukciou z kódovej pamäte z adresy 0x0000.
No a práve na adrese 0x0000 sa nachádza main() našej aplikácie (veľmi zjednodušene povedané!), ktorá sa spustí.
Toto je štandartný postup spustenia aplikácie.
Ale predstavte si, že pred samotným spustením našej aplikácie, spustíme ešte niečo iné - bootloader.

Bootloader teda "vsunieme" na adresu 0x0000, hneď pred našu aplikáciu.
Po uvoľnení reset signálu, sa začne vykonávať program "bootloader".
Iným spôsobom ako aktivovať bootloader je, zabezpečenie spustenia programu z adresy bootloadera a nie z adresy 0x0000.
Takýto spôsob používa napríklad Atmel pri svojich AVR procesoroch. Jednoducho správnym nastavením poistiek (fuses) procesora, je pozmenená štart adresa z 0x0000 na adresu vyhradenú pre bootloader.
Toto je postup pri použití jednej pamäte pre aplikáciu a aj pre bootloader (napr. interná Flash pamäť MCU).

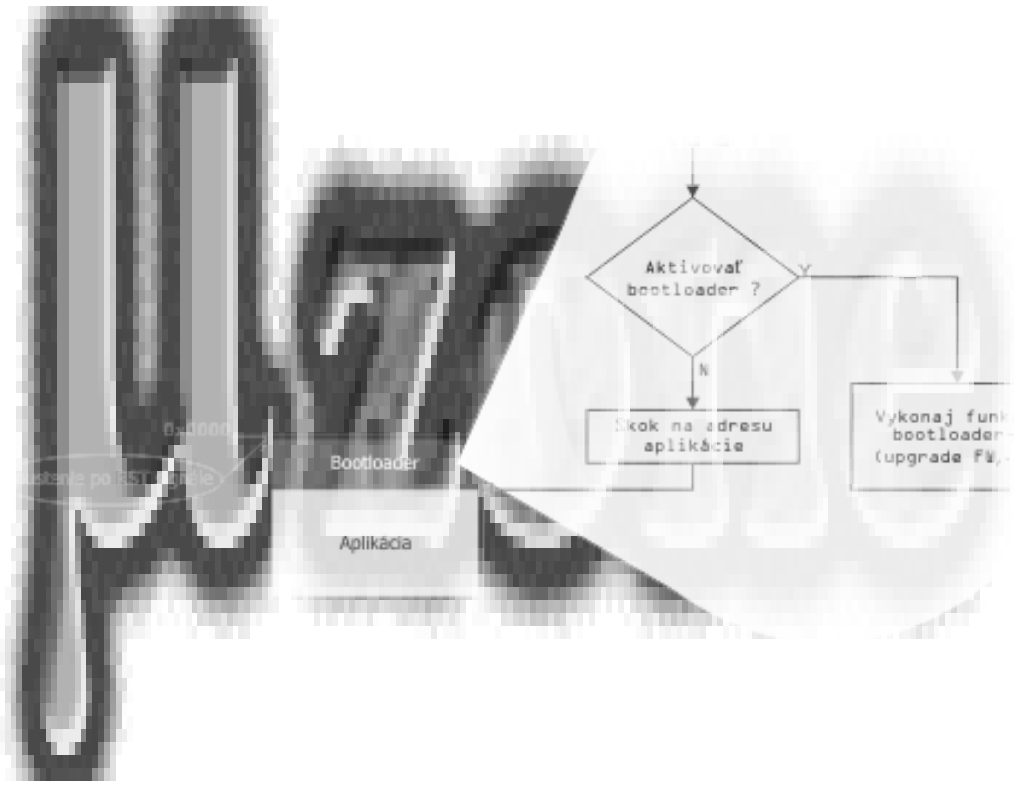
Existujú aj ďaľšie riešenia a to použitím fyzicky dvoch Flash pamätí, a ich prepínanie pomocou externej logiky, ale to momentálne vynechám.

Čo sa tam deje?

Bootloader teda po spustení skontroluje podmienku:

- aktivovať funkcie bootloadera? (napríklad aktualizáciu aplikácie - firmware upgrade) ?

- spustiť samotnú aplikáciu (Run) ?



Definovanie tejto podmienky je plne v našej kompetencii.
Podmienku môžeme definovať buď ako hardwérové testovanie pinu, celého portu, aktivácie UARTU, alebo prečítaním konfiguračného slova z EEPROM.
Po otestovaní tejto podmienky, následne bootloader pokračuje.
- spustením funkcií bootloadera (aktualizáciu aplikácie, kde definovaným rozhraním príjme nový kód aplikácie, a nahradí ním pôvodný kód).
- spustením aplikácie, t.j. vykonaním nepodmieneného skoku na adresu v kódovej pamäti, kde sa nachádza main() našej aplikácie.

Ďalšie podrobnejšie špecifikácie, ako aj možnosti bootloadera, závisia ako od použitého procesora a veľkosti dostupnej FLASH pamäte, tak aj od návrhu hardwéru aplikácie.

Aby to nebolo také jednoduché

Pri aplikovaní bootloadera treba ale dávať pozor na niektoré zásadné požiadavky.

Bootloader musí byť rýchly.
Nezabúdajme, že vykonávaním bootloadera, sa odkladá spustenie aplikácie samotnej. Takže náhodne vygenerovaný reset, či už watchdog-časovačom, alebo externým zdrojom v aplikácii, by užívateľ nemal ani postrehnúť.

Bootloader by mal byť malý (veľkosť kódu)
Toto je už ale viac závislé od hardwéru a použitého MCU.
Napríklad procesor ATMega8, má vyhradenú časť vo Flash pamäti pre bootloader o veľkosti max. 1024 bytov.
Preto je mnoho bootloaderov písaných priamo v asembleri.

Prečo použiť bootloader?

Ako som už v úvode písal, hlavne pre možnosť aktualizovať kód aplikácie v pamäti.
Tým, že si dokážeme naprogramovať aplikáciu do pamäte spôsobom, ktorý je pre dané zariadenie a prostredie najlepší, máme k dispozícii pomerne široké možnosti ako to zrealizovať.
Tým, že môžeme použit vlastné programovacie rozhranie a protokol (USB, CAN, RS485, RS232, SPI...), máme možnosť použitia aj vlastného programovacieho hardwéru (notebook, PLC, riadiaci PC, čierna skrinka,...).

Prvotné naprogramovanie bootloadera samotného je síce treba vykonať "bežným" programátorom, ale táto činnosť sa v drvivej väčšine deje v čase kompletizácie zariadenia (je tu možnosť dodávania predprogramovaných čipov priamo výrobcom, ale nie pre malé série....), teda mimo prostredia cieľového nasadenia, kde je aktualizácia kódu zväčša obtiažna a cenovo náročnejšia.

V ďalšom článku sa zamerám na použitie bootloadera, ako nástroja pre preflashovanie aplikácie, už pre konkrétny typ procesora.