Sériový bootloader pro mikrokontrolér S32K144

Bc. Diana Bátrlová

Diplomová práce
2018

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2017/2018

ZADÁNÍ DIPLOMOVÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Diana Bátrlová**
Osobní číslo:	**A17733**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Forma studia:	**kombinovaná**
Téma práce:	**Sériový bootloader pro mikrokontrolér S32K144**
Téma anglicky:	**A Serial Boot-loader for an S32K144 Micro-controller**

Zásady pro vypracování:

1. **Zpracujte literární rešerši na dané téma.**
2. **Prostudujte hardwarové vlastnosti mikropočítače NXP S32K144.**
3. **Navrhněte způsob implementace sériového bootloaderu.**
4. **Vytvořte programové vybavení sériového bootloaderu pro daný mikropočítač.**
5. **Ověřte správnou funkci bootloaderu.**

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **MARTIN, Trevor. The designer's guide to the Cortex-M processor family: a tutorial approach. Kidlington, Oxford: Newnes is an imprint of Elsevier, 2013. ISBN 9780080982960.**
2. **NXP SEMICONDUCTORS. S32K1xx Series Reference Manual, Rev.4, [online]. 2017 [cit. 2017-11-20]. Dostupné z: http://www.nxp.com**
3. **NXP SEMICONDUCTORS. S32K1xx Data Sheet, Rev.4, [online]. 2017 [cit. 2017-11-20]. Dostupné z: http://www.nxp.com**
4. **PINKER, Jiří. Mikroprocesory a mikropočítače. 1. vyd. Praha: BEN - technická literatura, 2004, 159 s. ISBN 80-7300-110-1.**
5. **VÁŇA, Vladimír. ARM pro začátečníky. 1. vyd. Praha: BEN - technická literatura, 2009, 195 s. ISBN 978-80-7300-246-6.**

Vedoucí diplomové práce: **Ing. Petr Dostálek, Ph.D.**
Ústav automatizace a řídicí techniky

Datum zadání diplomové práce: **1. prosince 2017**

Termín odevzdání diplomové práce: **16. května 2018**

Ve Zlíně dne 11. prosince 2017

L.S.

doc. Mgr. Milan Adámek, Ph.D.
děkan

prof. Mgr. Roman Jašek, Ph.D.
garant oboru

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne …….

podpis diplomanta

ABSTRAKT

Cílem diplomové práce je navrhnout a vytvořit programové vybavení pro sériový bootloader k mikrokontroléru S32K144. Mikrokontrolér se nachází na vývojové desce S32K144EVB, která se připojuje k počítači pomocí microUSB kabelu. K přenosu dat bude využíván univerzální asynchronní přijímač-vysílač a data budou posílána přes terminál. Výsledná funkčnost bootloaderu je schopnost naprogramovat jinou aplikaci do flash paměti mikrokontroléru a následně tuto aplikaci spustit. Tento proces bude možné provádět opakovaně.

Klíčová slova: S32K144, LPUART, Sériový Bootloader, S32 Design Studio, Srecord

ABSTRACT

The aim of this diploma thesis is to design and create software for serial bootloader for S32K144 microcontroller. The microcontroller is located on the S32K144EVB evaluation board, which connects to the computer using a microUSB cable. A universal asynchronous receiver-transmitter will be used for data transmission and the data will be sent through the terminal. The ultimate bootloader functionality is that it will be able to program another application into the flash memory of the microcontroller and then run the application. This process can be repeated.

Keywords: S32K144, LPUART, Serial Bootloader, S32 Design Studio, Srecord

Děkuji vedoucímu diplomové práce panu Ing. Petru Dostálkovi, Ph.D. a konzultantu Ing. Lukáši Zádrapovi za ochotu a odborné rady. Dále bych chtěla poděkovat firmě NXP Semiconductor, která mi poskytla zázemí pro vývoj programu. Největší díky nejen za trpělivost, ale i za velkou oporu patří mému příteli Markovi. Děkuji tedy za možnost rozšířit si znalosti z různých oblastí mikropočítačů.

Prohlašuji, že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Diplomová práce si klade za cíl vytvoření sériového bootloaderu pro mikrokontrolér S32K144 od firmy NXP. Pro vývoj programu byla použita vývojová deska S32K144EVB a vývojové prostředí S32K Design Studio.

Jak z názvu práce vypovídá, aplikace bootloaderu je nakonfigurována pro komunikaci s počítačem prostřednictvím rozhraní sériového portu UART (univerzální asynchronní přijímače a vysílače), který je integrován v mikrokontroléru. Aby bylo možné komunikovat s počítačem, je propojen s rozhraním sériového COM (komunikačního) portu. V tomto případě se jedná o virtuální COM port.

Bootloader je program, pro který je v paměti vyhrazeno omezené místo. Konkrétně ve flash paměti, která je schopna uchovávat data i po odpojení napájení. Jeho úkolem je naprogramovat do této paměti uživatelskou aplikaci neboli jiný program, aniž by k tomu potřeboval vývojové prostředí a programátor. Z důvodu rizika nebo velké pravděpodobnosti, že bootloader přeprogramuje sám sebe, by měla být paměť rozdělena na dvě části. Malá část paměti bude vyhrazena pro program bootloaderu a zbytek pro uživatelskou aplikaci. K naprogramování uživatelské aplikace slouží tzv. image (obraz), který je vygenerován po dokončení finální verze aplikace. Představuje kompletní program, který je zapsán v jednom souboru. Typicky je generován v programovacím prostředí S32 Design Studiu s příponou .s19.

Z těchto důvodů se práce zaměřuje i na předzpracování dat ještě před tím, než je bude program bootloaderu číst. Dále se zaměřuje na vytvoření jednoduché uživatelské aplikace, která bude sloužit jako ukázka spolu s vygenerovaným souborem typu .s19.

V teoretické části se nachází základní popis bootloaderu, mikrokontroléru a vývojové desky. Dále možnosti programování mikrokontroléru, systémový generátor hodin, detailní popis souborového formátu Srec., sériová komunikace a přerušení.

V praktické části se nachází popis vývojového prostředí, použitého terminálu, programová část bootloaderu, aplikace a aplikace pro zarovnání. V závěru diplomové práce se nachází návod k použití a ověření funkčnosti.

I. TEORETICKÁ ČÁST

1 BOOTLOADER

Při programování mikrokontrolerů se běžně používají programátory, které využívají USB (univerzální sériovou sběrnici) k propojení MCU (mikrokontroléru) s počítačem. Jedná se o rozhraní USB-to-BDM nebo v případě MCU S32K144 se používá OSBDM rozhraní. Díky tomuto rozhraní lze kód programu ladit (debuggovat) a programovat ho do paměti MCU. A umožňuje tak čtení nebo zápis registrů. Navíc použití OSBDM neboli OpenSDA rozhraní je z hlediska praktičnosti výhodné, protože není potřeba vývojovou desku napájet přes externí napájení 12V. Stačí jeden microUSB kabel. Bootloader nepotřebuje vývojové prostředí na to, aby aplikaci naprogramoval.

1.1 Jak a k čemu se bootloader používá

Jelikož se mikrokontrolér S32K144 používá pro automobilové aplikace, je žádoucí možnost programování či přeprogramování aplikace bez nutnosti použití externího programátoru. Programování a případné spuštění uživatelské aplikace provádí programový kód bootloaderu, který se nachází v paměti MCU. Jeho spuštění lze provádět automaticky po připojení napájení nebo na příslušnou vnější událost, kterou může být např. stisk tlačítka.

Aby bylo možné spouštět aplikaci uživatele nebo aplikaci bootloaderu, je potřeba rozdělit flash paměť na dvě sekce. To se provádí pomocí změny adres v souboru nacházejícím se v adresáři s názvem *Linker Files*. U MCU S32K144 se jako první spouští sektor na začátku flash paměti od adresy 0x0000_0000. Z toho důvodu je bootloader umístěn v horní části, aby se zajistilo jeho spuštění i v případě, kdy se uživatelská aplikace nahraje např. poškozená.

Jakmile je vše připraveno, může bootloader načítat uživatelskou aplikaci postupně přes sériové rozhraní UART. S použitím terminálu, který nabízí jednoduché uživatelské rozhraní pro ovládání bootloaderu, jako je např. mazání, programování flash, nastavení přenosové rychlosti a spuštění uživatelské aplikace. Lze také odeslat soubor obsahující adresy a data uživatelské aplikace. Ty jsou následně přijímány do zásobníku (bufferu) po jednom bajtu.

Pro tento účel mohou být použita i jiná rozhraní jako je CAN (Controller Area Network), I2C (Inter-Integrated Circuit), SPI (Serial Peripheral Interface) a další, která jsou dostupná na daném MCU.

2 SPECIFIKACE MCU A DESKY S32K144

V této kapitole se nachází základní popis a periferie mikrokontroléru S32K144 a vývojové desky S32K144EVB. Tato zařízení jsou od firmy NXP a jsou využity pro vývoj aplikace bootloaderu.

2.1 MCU S32K144

Jedná se o 32bitový mikrokontrolér, zaměřený na univerzální automobilové a vysoce spolehlivé průmyslové aplikace. Je založen na jádře ARM Cortex-M4F/M0+. Umožnuje běh procesoru ve vysoko rychlostním režimu až 112 MHz s podporou DSP (Digital Signal Processor) a FPU (Floating point unit). Velikost Flash paměti je 512 KB [1].

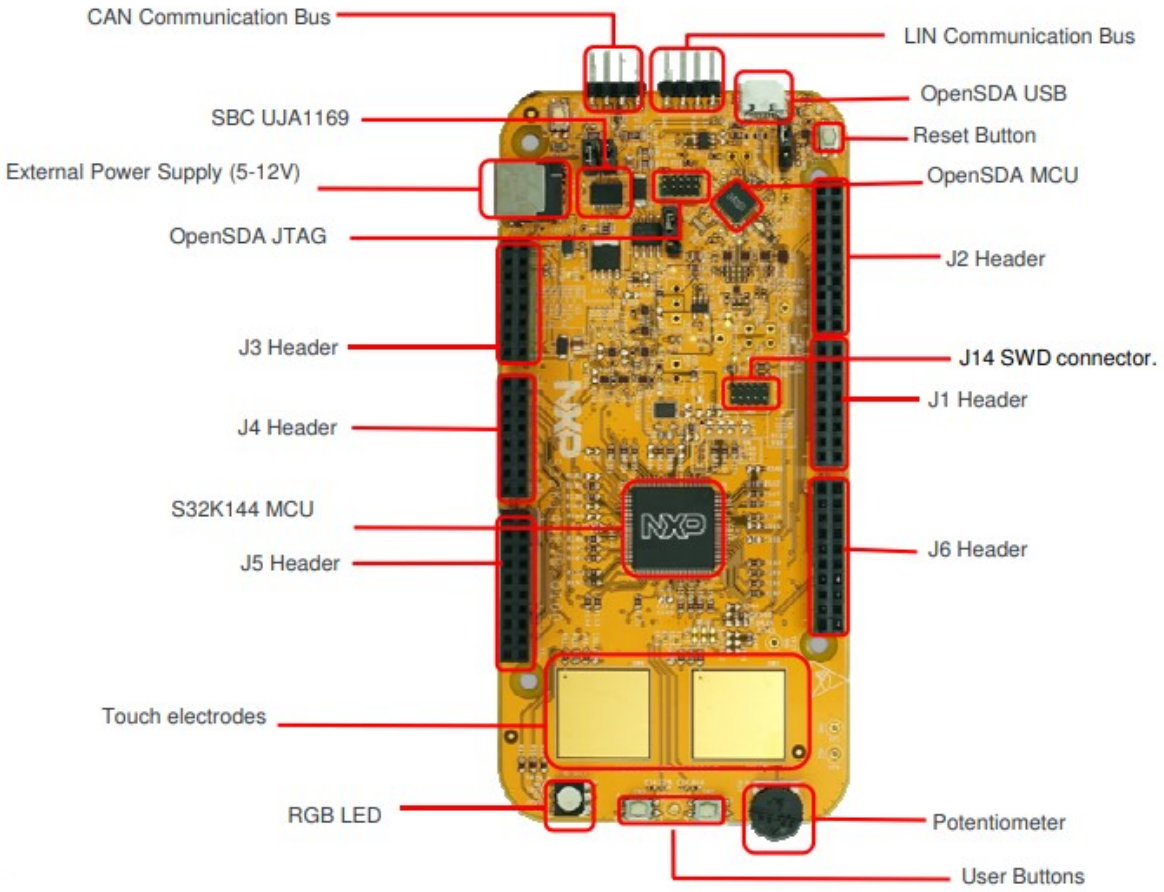
V normálním režimu umožňuje běh procesoru až 80 MHz. Dále MCU podporuje režimy STOP, VLPR (Very Low Power RUN) a VLPS (Very Low Power Stop) režimu. [1]

- Analogové moduly
  - Obsahují až dva 12. bitové analogově-digitální převodníky (ADC) s až 32 kanálovými analogovými vstupy na modul
  - Jeden analogový převodník (High-speed comparator - CMP) s interním 8bitovým digitálně analogovým převodníkem (DAC)
- Komunikační rozhraní
  - Až 3 univerzální asynchronní přijímače a vysílače s nízkou spotřebou energie (LPUART) moduly s podporou přímého přístupu do paměti.
  - Až 3 sériové periferní rozhraní s nízkou spotřebou energie (Low-Power Serial Peripheral Interface - LPSPI)
  - Až 2 LPI2C (Low-Power Inter-Integrated Circuit) s podporou přímého přístupu do paměti
  - Až 3 FlaxCan moduly
  - Modul FlexIO pro flexibilní a vysoce výkonné sériové rozhraní
- I/O napájení podporuje 2,7 V až 5,5 V
- Okolní provozní teplota se pohybuje od -40°C do 125°C v normálním běhu programu. Ve vysokorychlostním se teplota pohybuje od -40°C do 105°C
- Pouzdro 100 LQFP (Low Profile Quad Flat Pack). [8]

2.2 Zkušební deska S32K144EVB-Q100

Firma NXP nabízí k některým mikrokontrolérům vývojové desky. Výhodou je, že pro S32K144 je právě taková vývojová deska dostupná. Zatím pouze ve variantě pro 100 pinový mikrokontrolér S32K144. Její základní vlastnosti jsou popsány níže.

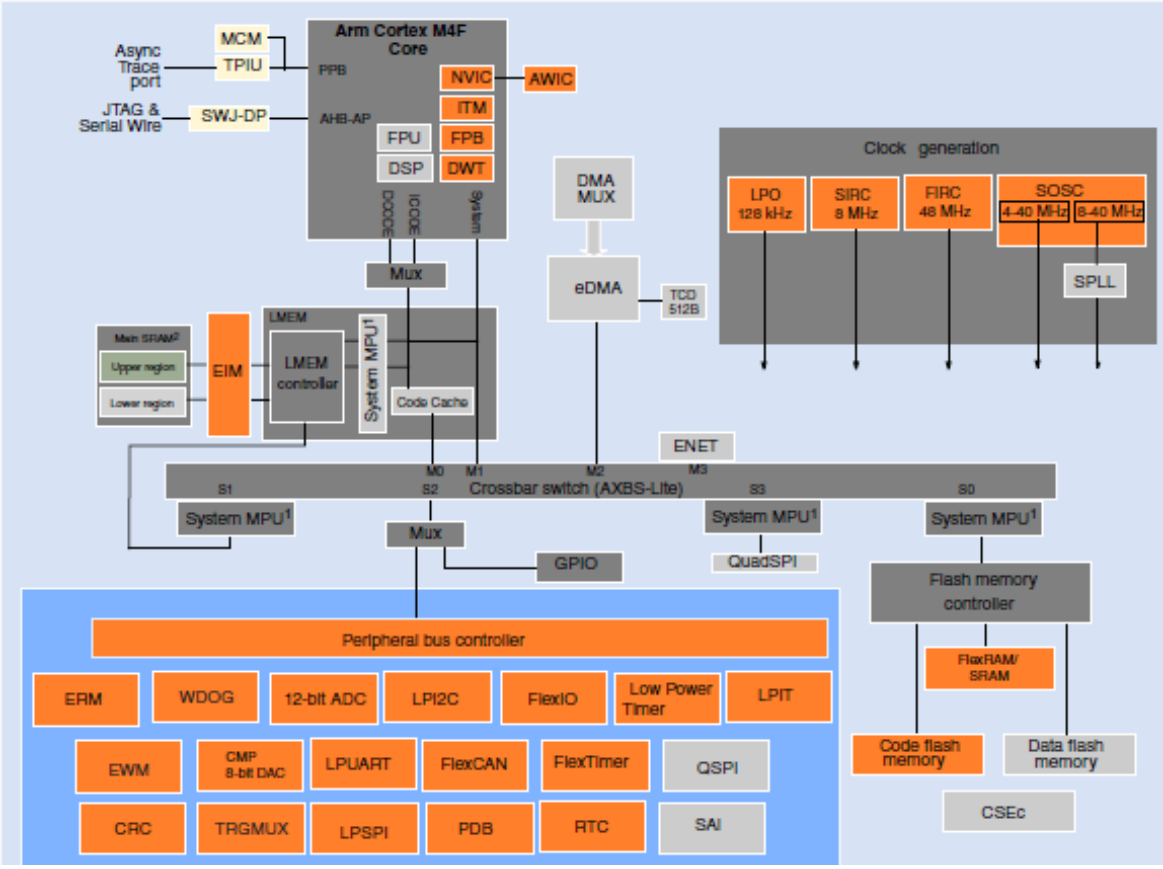
- Vestavěné připojení CAN (Controller Area Network), LIN (Local Interconnect Network), UART/SCI (Serial Communication Interface)
- Integrace SBC (System Basis Chip) a LIN fyzické vrstvy (TJA1027)
- Potenciometr, RGB LED (Light-Emitting Diode), dvě tlačítka a dvě dotykové plochy
- Integrovaný sériový a ladící adaptér s otevřeným standardem (OpenSDA) s podporou několika standardních ladících rozhraní
- Možnost flexibilního napájení přes microUSB nebo přes externí napájení 12 V [2]



Obr. 1 : Zkušební deska S32K144 [2]

2.3 Blokový diagram MCU

Portože je blokový diagram vytvořen pro více druhů MCU, můžou se některé periferie u jednotlivých MCU lišit. Tyto periferie jsou zvýrazněny světle šedou barvou. Oranžovou barvou jsou na druhou stranu zobrazeny ty periferie, které se nachází na všech zařízeních reprezentovaných rodinou S32K. V diagramu lze najít poznámky u některých preriferií. Poznámka číslo 1 znamená, že na tomto NXP zařízení systém MPU (Memory Protection Unit) implementuje bezpečnostní mechanizmy, které zabraňují tomu, aby master přistupupoval k omezeným oblastem paměti. Tento systém MPU poskytuje ochranu paměti na úrovni tzv. Crossbar Switch (kolekce přepínačů uspořádaných v maticové konfiguraci). Takže například Core, DMA nebo Ethernetu můžou být přiřazena různá přístupová práva ke každé chráněné oblasti paměti. Jádro verze M4 Arm v této rodině nezahrnuje technologii Arm Core MPU, která by současně sledovala pouze přístupy vyvolané jádrem. Poznámka s čílem 2 znamená, že je blog určen pro zařízení se specifickou velikostí SRAM. Poznámka číslo 3 znamená, že pokud zařízení běží ve vysoko rychlostním módu jsou povoleny příkazy EEPROM (Electrically Erasable Programmable Read-Only Memory) a Security (CSEc). [2]



Obr. 2: Blokové schéma MCU [1]

Ke každému MCU se nachází referenční manuál nebo datasheet. MCU S32K144 má po revizi oba dokumenty. Na stránkách NXP jsou k nalezení i další užitečné dokumenty. Je důležité podotknout, že nejsou určeny jen pro jeden typ MCU, ale pro celou rodinu S32K. Datasheet není tak rozsáhlý, ale obsahuje například informace o funkcích MCU, porovnání mezi jednotlivými rodinami MCU, pro které je vytvořen. Dále se zde nachází různé elektrické charakteristiky nebo specifikace, zobrazení pinů u pouzder pro MCU a další. Za to v referenčním manuálu najdeme detailní popis registrů, modulů a dalšího nastavení, které MCU umožňuje. K vývojové desce je poskytnuto schéma. Schéma může být dostupné v několika revizích. Každá deska má ze spodní strany nalepený štítek s označením revize. Dále zpravidla i stručný průvodce pro použití vývojové desky. Můžeme tam najít například popis jednotlivých komponent nacházejících se na desce, možnosti napájení nebo nastavení jumperů.

3 DOSTUPNÉ STANDARDY PRO DEBUGOVÁNÍ NXP ZAŘÍZENÍ

Pro zavádění programu do paměti MCU se používají programovací adaptéry. Níže jsou popsány programovací rozhraní, které jsou používány u různých zařízení od firmy NXP.

Pro standardní rozhraní **JTAG** (Joint Test Action Group) / **SWD** (Serial Wire Debugging) může být například použit OpenSDA debugger a programátor s microUSB rozhraním. Jedná se o ARM Debugger a je to debugger a programátor, který podporuje nejpopulárnější MCU s ARM jádrem včetně: ARM7 / 9/11, Cortex-M0 / M3 / M4 atd.

Pro rozhraní **BDM** (Background Debug Mode) můžou být například použity níže popsané programátory. Jsou především používány pro MCU S12Z a na starších MPC5xx.

USB Multilink Universal je kompatibilní se zařízeními ARM od mnoha výrobců, stejně jako modely NXP Kinetis, LPC, S32, ColdFire V1 / + V1, ColdFire V2-4, MPC55xx / 56xx / 57xx, HCS08, RS08, HC (S) 12 a SPC5 společnosti STMicroelectronics. [5]

CYCLONE UNIVERSAL podporuje širokou škálu zařízení ARM Cortex od mnoha výrobců, stejně jako mnoho zařízení, která nejsou vybavena ARM, a to od NXP (S32, Qorivva (MPC5xxx), MPC5xx / 8xx, DSC, S12Z, RS08, S08, HC08, HC (S) 12 (X), Coldfire, Kinetis, LPC) a STMicroelectronics STM32, SPC5, STM8. [5]

USBDM je ladicí hardwarové rozhraní pro řadu mikroprocesorů NXP. Je navržen tak, aby pracoval se softwarem společnosti NXP Codewarrior pod operačními systémy Windows a Linux. K dispozici je také sada samostatných programátorů. [6]

4 SYSTÉMOVÝ GENERÁTOR HODIN

V MCU S32K144 je integrován systémový generátor hodin. Jedná se o modul, který poskytuje výběr z několika zdrojů hodin pro MCU.

Jedním z nich je externí oscilátor, který může nabývat hodnoty od 4 MHz do 40 MHz. Na vývojové desce využité pro bootloader se nachází 8 MHz krystal. Existuje také možnost interního RC oscilátoru. Pro SIRC (Slow Internal Reference Clock) má hodnotu 8 MHz a pro FIRC (Fast Internal Reference Clock) 48 MHz. [1]

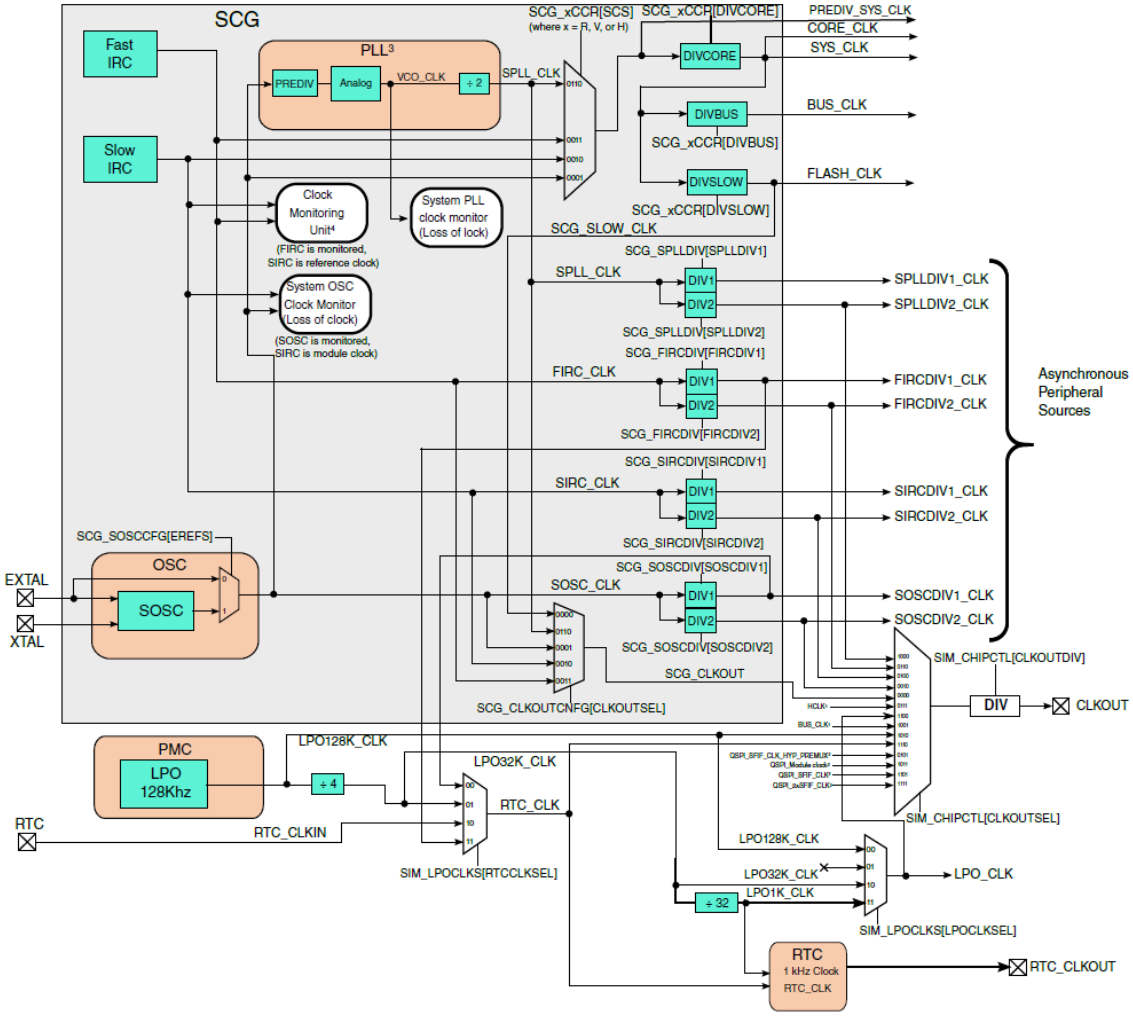
Pro generování vyšších frekvencí se používá fázový závěs PLL (Phrase Lock Loop). Využitím fázového závěsu je možno dosáhnout frekvence až 160 MHz.

Maximální frekvence v *normal run* módu (normálním provozním módu) může být 80 MHz a minimální frekvencí je *BUS clock* frekvence. V *normal run* módu může být frekvence *BUS clock* maximálně 40 MHz. [1]

Tento mód, je vybrán po jakémkoli resetu MCU.

V *high speed run* módu (vysoko rychlostním provozním módu) lze konfigurovat až 112 MHz, při použití fázového závěsu jako zdroje systémových hodin MCU. V *high speed run* módu může být *BUS clock* maximálně 56 MHz. [1]

Na obrázku níže je diagram, který zobrazuje nastavení hodin. CKLOUT je pin, na kterém lze měřit nastavenou frekvenci hodin například pomocí osciloskopu. V dolní části obrázku lze vidět všechny možnosti nastavení registru, ve kterém se nastavuje zdroj systémového generátoru hodin. Jedná se o registr SCG_CLKOUTCNF. Nastavení pro QSPI (Quad Serial Peripheral Interface) je pro jiný typ MCU. [1]



Obr. 3: Blokové schéma hodinového generátoru [1]

5 SOUBOROVÝ FORMÁT SREC

Pro ukládání uživatelské aplikace do flash paměti bootloaderu, byl použit souborový formát SREC, který byl vytvořen společností Motorola. V případě bootloaderu slouží k programování flash paměti mikrokontroléru a můžeme ho najít nejen pod názvy jako je SREC, ale také pod názvy S19, SRECORD, S28 a S37.

V případě uživatelské aplikace, která byla naprogramována v programovacím prostředí S32 Design Studiu, byl vygenerován souborový formát s příponou .s19.

V typické aplikaci kompilátor nebo assembler převede zdrojový kód programu do strojového kódu a vloží jej do HEX souboru. Soubor HEX je pak importován programátorem, aby přenesl strojový kód do cílového systému pro načítání a provádění. [3]

Souborový formát se skládá ze série ASCII znaků, přičemž jeden bajt se skládá ze dvou ASCII znaků. Programy, které vytvářejí HEX záznamy, používají znaky pro ukončení řádku v závislosti na operačním systému. Oba souborové formáty si jsou velice podobné svou strukturou, proto soubor HEX lze překonvertovat do formátu S19, ale není to potřeba, protože S32 Design Studio umožňuje generování souboru typu S19. [3]

5.1 Struktura záznamu

Srecord je tvořen záznamy, díky kterým lze zjistit informace o programu. Například kde končí a na jakou adresu se mají zapsat programová data. Jednotlivé záznamy jsou zapsány v následující struktuře:

- **Typ záznamu** se skládá ze znaků z nichž jeden je S a další je číslice od 0 do 9, která definuje typ záznamu, podle počtu bitů adresy, na které začínají datové záznamy
- **Počet bajtů** se skládá ze dvou hexadecimálních číslic. Určují počet bajtů, které zabírají data včetně adresy a kontrolního součtu. Pole může nabývat maximální hodnoty 255 a minimální hodnoty 3 pro 16bitové adresní pole plus 1 bajt kontrolního součtu
- **Adresa** je určená typem záznamu a její velikost představují hexadecimální číslice. Jejich počet může být 4, 6, nebo 8.
- **Data** představuje posloupnost číslic 2n hex, pro n bajtů dat. U záznamů S1 / S2 / S3 je typický maximální počet 32 bajtů na záznam
- **Kontrolní součet** zabírá poslední dva bajty reprezentované dvěma hexadecimálními čísly. Výpočet viz obrázek 4. [3]

5.2 Typy záznamů

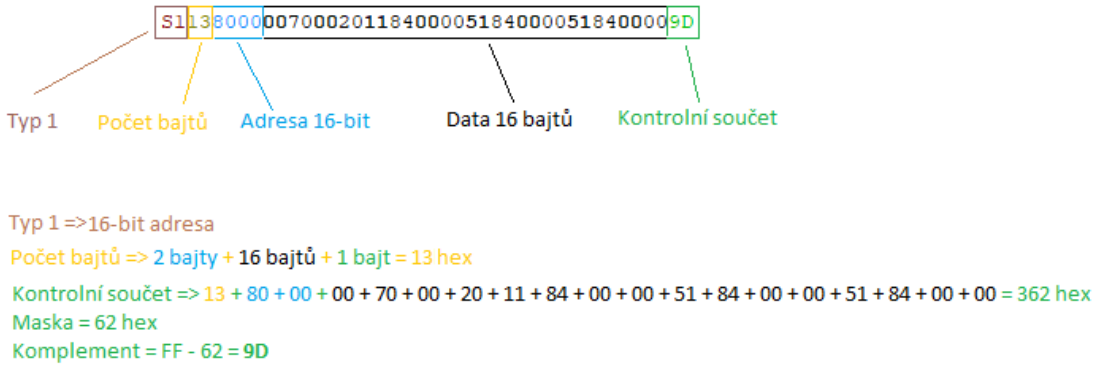
Každý záznam bez ohledu na adresu začíná typicky hlavičkovým záznamem, který se značí typem S0. Dále následují datové záznamy, které představují typy S1, S2 a S3. Za nimi se mohou vyskytovat volitelné záznamy S5 nebo S6. Jako poslední jsou ukončovací záznamy S7, S8 nebo S9. Hlavičkový záznam S0 a ukončovací záznamy bývají zpravidla po jednom. Datových záznamů může být i více.

Níže jsou popsány záznamy, které obsahují i datové pole.

- **S0 -** hlavičkový záznam s 16bitovou adresou nesoucí informace o souboru.
- **S1 -** jedná se o datový záznam určující 16bitovou adresu. To znamená, že data začínají na 16bitovém adresním poli
- **S2 -** jedná se o datový záznam určující 24bitovou adresu. Data začínají na 24bitovém adresním poli.
- **S3 -** jedná se o datový záznam určující 32bitovou adresu. Data začínají na 32bitovém adresním poli [3]

Následují záznamy, které neobsahují datové pole.

- S5 – záznam, který se nemusí vyskytovat v souboru S19. Obsahuje 16bitový počet S1/S2/S3 záznamů
- S6 - záznam, který se také nemusí vyskytovat v souboru S19. Obsahuje 24bitový počet S1/S2/S3 záznamů
- S7 - používá se k ukončení záznamů typu S3
- S8 - používá se k ukončení záznamů typu S2
- S9 – používá se k ukončení záznamů typu S1 [3]

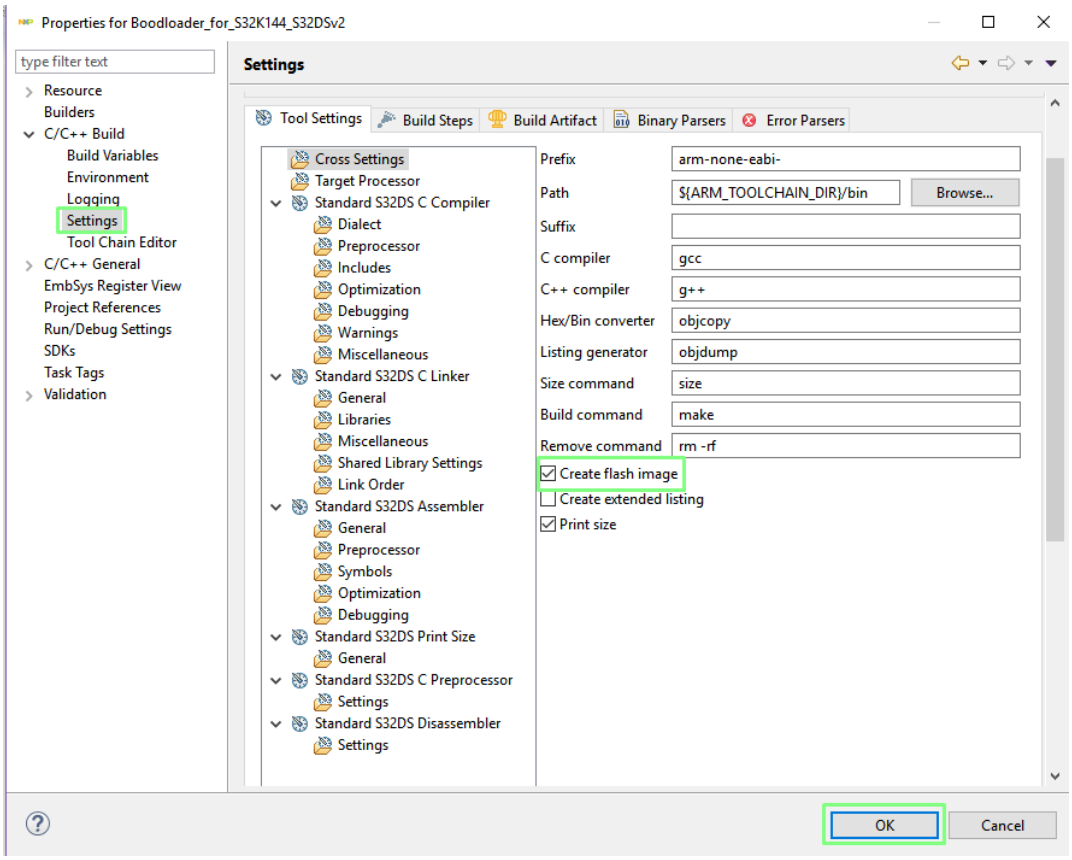


Obr. 4: Příklad čtení záznamu

5.3 Generování souborového záznamu

Pro tvorbu programu bylo použito programovací prostředí S32 design studio, ve kterém byl následně vygenerován Srecord. Aby se vygeneroval, je potřeba tuto možnost nastavit. Kliknutím pravým tlačítkem na *název projektu -> Properties -> C/C++ Build -> Settings -> Create flash image.*

Následně stačí zkompilovat program a záznam se vygeneruje.



Obr. 5: Generování souboru Srecord v S32DS

6 SÉRIOVÁ KOMUNIKACE

Bootloader projekt využívá k přenesení dat sériovou komunikaci, při které se sekvenčně přenáší data po jednom bitu přes komunikační kanál nebo počítačovou sběrnici. Tento způsob přenosu je v tomto případě použití ideální. Z důvodu krátké vzdálenosti přenosu je výhodnější, jelikož odstraňuje některé problémy, které vznikají při paralelním přenosu dat.

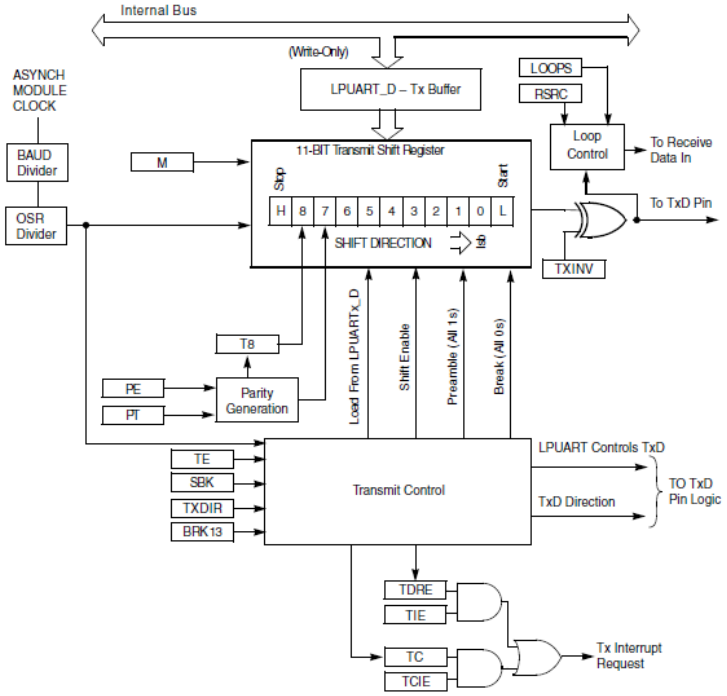
Jedná se o 8-bitový asynchronní přenos, kdy jsou jednotlivé bity přenášeny pomocí rámců. Velikost rámců se může lišit v souvislosti s různými druhy mikrokontrolérů. Každý rámec obsahuje jeden nebo dva stop bity a jeden start bit. Jako výchozí je úroveň signálu v logické jedničce, tedy když neprobíhá přenos. Při změně úrovně signálu na logickou nulu je zahájen přenos dat, přičemž se jako první posílá LSB (nejméně významný bit) a jako poslední MSB (nejvýznamnější bit), po kterém následuje stop bit. [10]

MCU S32K144 umožňuje využití LPUART (nízkospotřebový) modul, který podporuje základní UART (univerzální asynchronní přijímač a vysílač) s rozhraním pro přímý přístup do paměti. Asynchronní mód umožňuje zároveň přijímat i vysílat, tedy full-duplex. Má programovatelnou přenosovou rychlost. [1]

Piny pro vysílání se označují jako TxD a pro příjem jako RxD.

6.1 Vysílací část

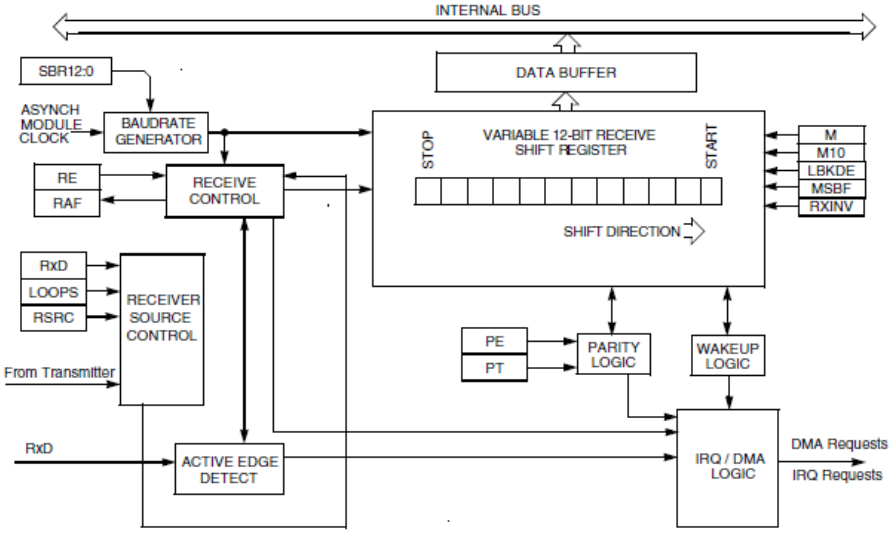
Data jsou posílána přes interní sběrnici a následně zapsána do zásobníku *LPUART_D-TX Buffer*. Z tohoto zásobníku jsou data přepsána do registru *Transmit Shift Registrer*. Díky tomuto přesunu dat se komunikace zefektivní, protože zatímco se data posílají, do zásobníku se načítají data následující.



Obr. 6: LPUART vysílání dat [1]

6.2 Přijímací část

Příjem dat začíná ve chvíli, kdy se na RXD pinu objeví logická nula, tedy start bit, po kterém následují datové bity. Jejich logické úrovně jsou čteny v pravidelných intervalech a jsou ukládány do registru s názvem *Receive Shifh Register*. Jakmile je přijat stop bit, tak jsou data přesunuta do zásobníku určeného pro data (data buffer).



Obr. 7: LPUART příjem dat [1]

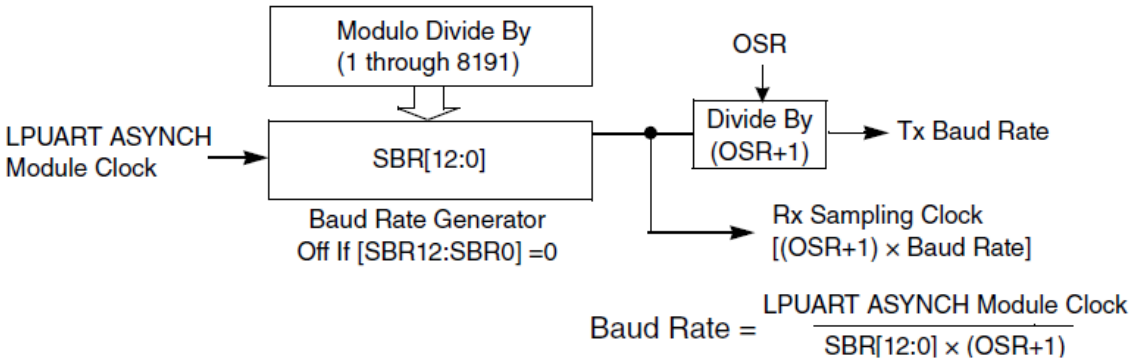
6.3 Přenosová rychlost

Přenosová rychlost neboli *baud rate* udává počet přenesených bitů za sekundu, tedy komunikační rychlost jakou se přenáší informace. Pokud je nastavena přenosová rychlost na 9600, bude přenášeno přes sériový port maximálně 9600 bitů za sekundu. Tato hodnota je konfigurována při inicializaci periferie UART. Přenosová rychlost musí být konfigurována také na straně terminálu na stejný počet přenášených bitů za sekundu. Jednotkou je Kb/s.

Na straně MCU je v generátoru přenosové rychlosti implementován 13bitový modulový čítač. Odvozuje přenosovou rychlost pro přijímač a vysílač. Podporuje plně duplexní, asynchronní, sériovou NRZ (Non Return To Zero) komunikaci. I když přijímač a vysílač používají jeden generátor přenosové rychlosti, tak pracují nezávisle na sobě. Je možné jej konfigurovat nezávisle na *bus clock* frekvenci. [1]

Přenosová rychlost se konfiguruje v registru s názvem BAUD. Důležitým bitovým polem je OSR. Zde je možné nastavit převzorkování s poměrem 4x až 32x. Díky tomu lze dosáhnout lepší spolehlivosti přenosu. Pro poměr 4, 5, 6 a 7 je třeba povolit vzorkování pro obě hrany hodin přenosové rychlosti. Výše popsaná konfigurace se týká přijímače.

Dalším důležitým 13bitovým polem je SBR, jehož hodnota se dá nastavit od 1 do 8191. Tato hodnota se používá při výpočtu přenosové rychlosti v rovnici níže. Source clock znamená zdroj hodin, pro generátor přenosové rychlosti. Zdrojem hodin je frekvence oscilátoru. Nastavení BAUD registru se může provádět pouze v případě, že je přijímač a vysílač zakázán.



Obr. 8: Generování přenosové rychlosti [1]

Vzhledem k tomu, že je přenosová rychlost generována, je přenos vystaven dvěma chybám.

První chyba vzniká kvůli dělení datového typu integer, protože nemusíme získat přesnou hodnotu cílové frekvence. Další chyba může být způsobena synchronizací s asynchronními LPUART hodinami, kdy dojde k fázovému posunu. [1]

6.4 Softwarové řízení toku dat

Jedná se o metodu, která používá speciální kódy pro řízení toku dat. Tyto kódy jsou vysílány přes primární komunikační kanál a nazývají se XON pro obnovení přenosu a XOFF pro pozastavení přenosu. V tomto případě se jedná o systém používající ASCII znaky. Tyto řídící znaky nejsou přímo specifikované, ale obecně je XOFF reprezentován decimální hodnotou 19 a XON je reprezentován hodnotou 17. [4]

Výhodou je, že v porovnání s řízení toku pomocí hardwaru není vyžadována speciální implementace hardwaru a ani mnoho vodičů, kde stačí jen dva pro příjem a vysílání. Nevýhodou je, že odesílání XOFF znaku může mít zpoždění, kvůli čekání ve frontě zásobníku.

Jakmile jedna strana už nemůže přijímat data, tak vyšle XOFF. Ten příjme druhá strana a pozastaví přenos. V případě, že první strana už může vysílat, druhá strana přenos obnoví.

U bootloaderu je to využíváno v případě, že MCU nestíhá zapisovat data do paměti flash a tím odebírat znaky ze zásobníku, do kterého se ukládají přijatá data. Zásobník se tedy bude plnit. Jakmile je zásobník skoro plný, vyšle vysílač XOFF znak, aby pozastavil přicházející tok dat. Pokud by to neudělal, došlo by ke stavu, že už nemůže přijímat další data, v tom případě by se začal zásobník přijatých znaků přepisovat. Jakmile se v zásobníku uvolní místo, vyšle vysílač XON a terminál posílání dat zase obnoví.

7 PŘERUŠENÍ

Například pro přijímání a odesílání dat přes sériové rozhraní je využito přerušení. Tabulka vektorů přerušení se nachází v souboru startup_S32K144.S.

Tabulka obsahuje adresy/ vektory umístění příslušné obsluhy přerušení v paměti. Tyto vektory jsou reprezentovány identifikačním číslem. Každé přesušení má svoji prioritu. Každý vektor má velikost 4B.

V některých případech je potřeba zakázat nebo povolit přerušení. K tomuto účelu jsou použity instrukce assembleru.

7.1 NVIC modul

Pro přesušení u S32K144 MCU se používá NVIC (Nested-Vectored Interrupt Controller) modul, který zjednodušuje programovatelnost. NVIC modul podporuje až 32 externích přerušení a umožňuje nastavit až 16 úrovní priority.

Zdroje přerušení lze najít v souboru *S32K1xx_DMA_ interrupt_mapping.xlsx* v příloze k referenčnímu manuálu. [1]

7.2 Resetovací vektor

Je to přerušení, které se vykoná po resetu. Ten je vyvolán resetovacím signálem.

Když procesor vystupuje z resetu, tak se nastavuje ukazatel zásobníku (stack pointer), který se nachází ve vektorové tabulce na offsetu 0x00 a čítač instrukcí (program counter), jehož začátek se nachází ve vektorové tabulce na offsetu 0x04. Na adrese 0x00 a 0x04 se nachází resetovací vektory.

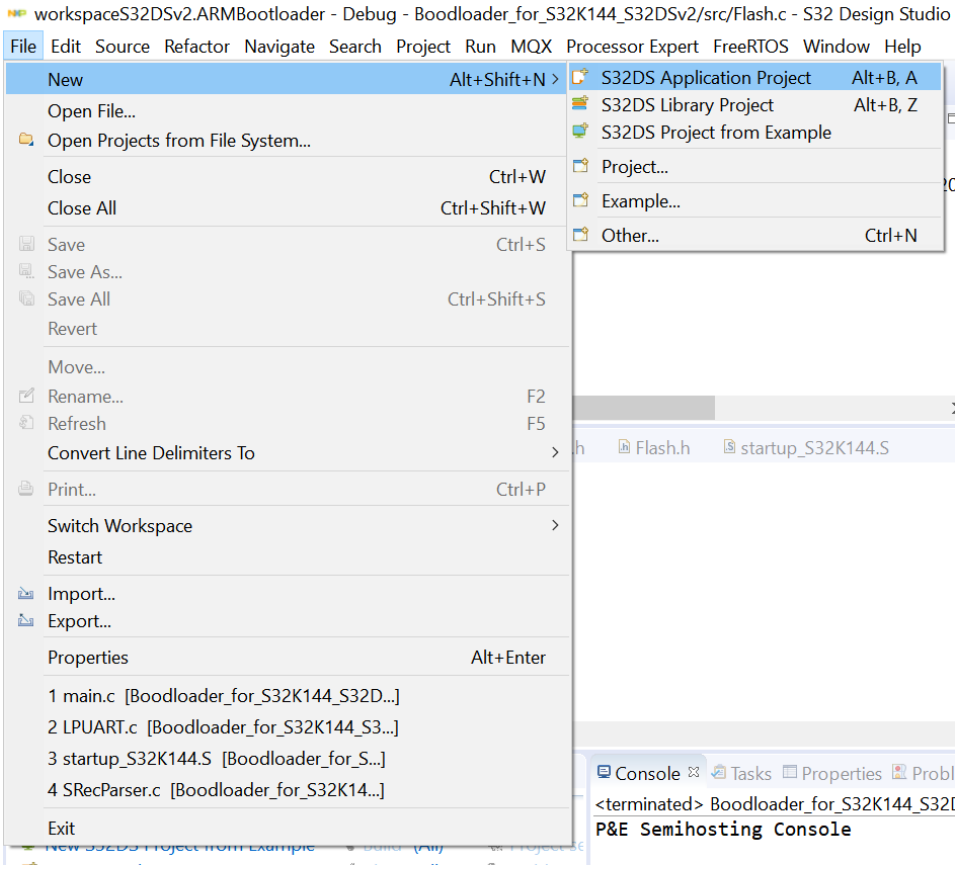
Jakmile je CPU (centrální procesorová jednotka) schopna vykonávat instrukce, měla by vždy začínat právě na této adrese. Jedná se o výchozí pozici v adresovém prostoru, kde by se měla nacházet první instrukce, kterou CPU vykoná po resetu.

II. PRAKTICKÁ ČÁST

8 VÝVOJOVÉ PROSTŘEDÍ S32 DESIGN STUDIO

K programování vývojového kitu bylo použito bezplatné vývojové prostředí S32 Design Studio verze 2.0. To je určeno pro MCU na bázi ARM užívané v automobilovém průmyslu. Instalační soubor programu je dostupný na NXP webových stránkách. Program aplikace sériového bootloaderu byl napsán v jazyce C s využitím několika příkazů z assembleru. Je založeno na open-source softwaru Eclipse. Mezi jeho funkce patří kompilace, ladění programu a samozřejmě editace.

S32DS nabízí pro vybrané typy MCU programové příklady pro použití periferií. Je užitečné zejména v případě, když je potřeba si nastudovat například novou periferii. V příkladech bývá popis základního použití a je to skvělý startovní bod, protože čtení referenčního manuálu zabere poněkud více času. Avšak první volbou tohoto menu je vytvoření prázdného projektu. Každý projekt obsahuje knihovnu vytvořenou speciálně pro daný typ MCU. V této knihovně se nachází možné způsoby nastavení registrů, které jsou pro daný typ MCU nastavitelné. Nicméně se může stát, že nedošlo k poslední revizi, a z toho důvodu je důležité se řídit referenčním manuálem.

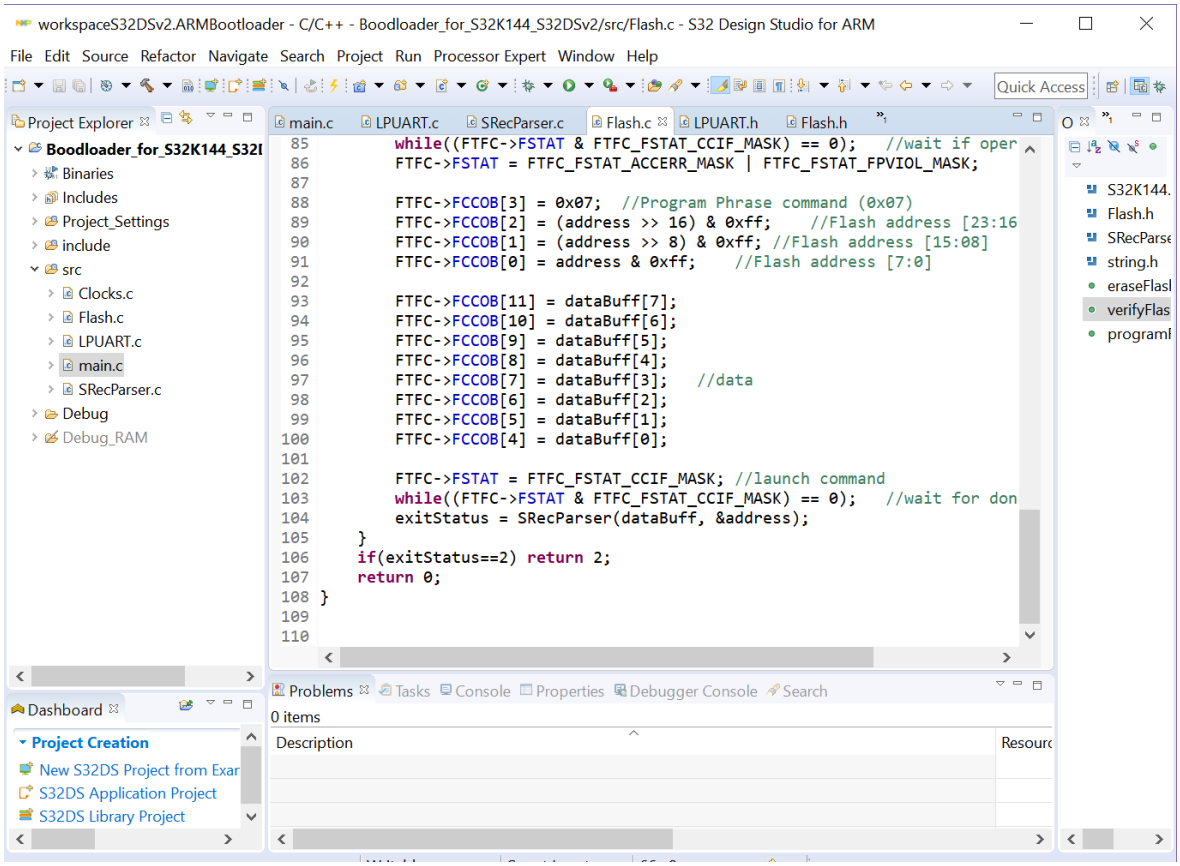


Obr. 9: Založení projektu v prostředí S32DS

S32DS umožňuje dva základní typy zobrazení: v normálním módu a ladícím módu. V normálním módu je zpravidla na levé straně průzkumník projektů. Na spodní straně se nachází okno s několika záložkami, obsahující například přehled chyb programu, konzoli a vlastnosti. Zobrazení jednotlivých oken si lze přenastavit podle potřeb.

Tento mód je výhodný z hlediska programování, jelikož je programovací okno větší, ale při ladění postačí i zmenšené okno s programem. Existuje i možnost ladění programu v normálním módu.

Důležitým tlačítkem je kompilace projektu a nachází se na horní liště pod hlavním menu. Představuje ji tlačítko s kladívkem. Všechny úkony, které jsou zobrazeny obrázky, lze provádět z hlavního menu.

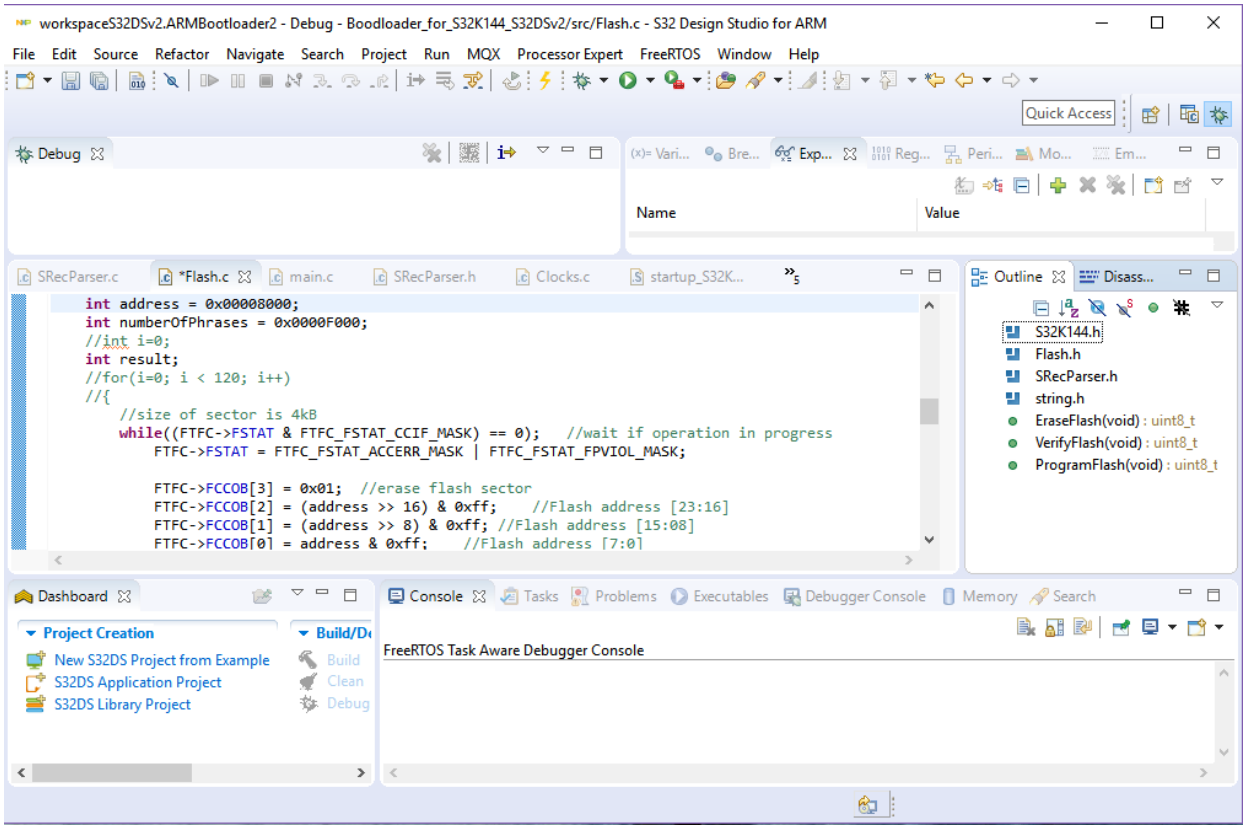


Obr. 10: S32DS normální mód

Před laděním programu je potřeba nastavit konfiguraci ladění. Konfigurace pro ladění nabízí několik způsobů programování. Možnost Debug_RAM slouží primárně pro ladící účely, kdy se celý program, tedy data i instrukce, zavede do paměti RAM (Random-Access memory). Výhoda tohoto způsobu ladění spočívá v tom, že neubírá flash paměti počet zapisovacích cyklů a je při nahrávání rychlejší než při nahrávání dat do flash paměti.

Dále je potřeba zvolit programátor. V případě programování S32K144 MCU byl zvolen OpenSDA – sériový a ladící adaptér.

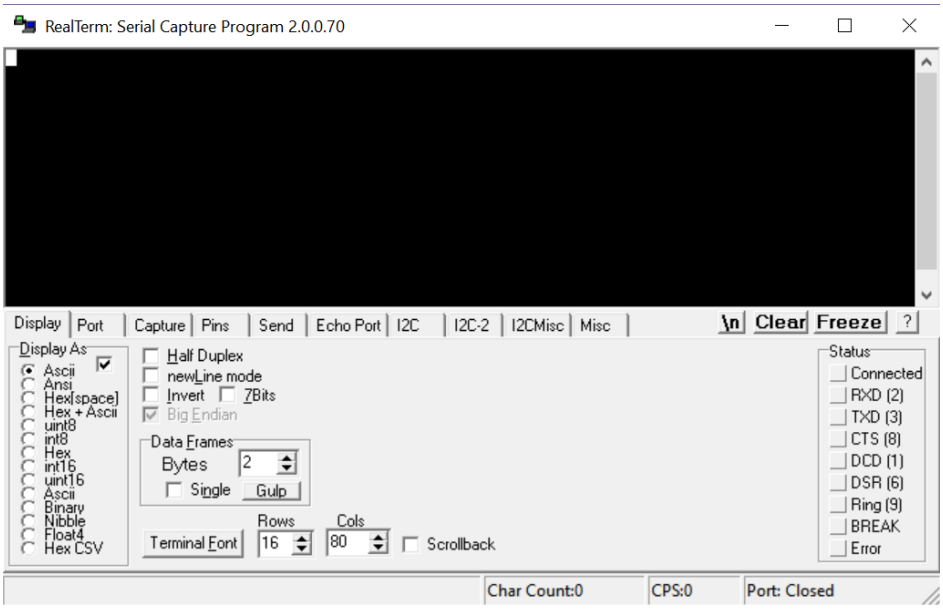
Po spuštění ladění se na dotázání, vývojové prostředí přepne do ladícího módu, který lze vidět na obrázku níže. Pod základní prvky pro ladění patří jeho spuštění pomocí zelené šipky, pauza, stop a velmi užitečné krokování. Všechny tyto nástroje se nachází v horní části vývojového prostředí. Velice užitečnou funkcí je možnost sledování hodnoty v proměnné. V záložce *Memory* lze nahlédnout do paměťového prostoru na požadovanou adresu.



Obr. 11: S32DS ladící mód

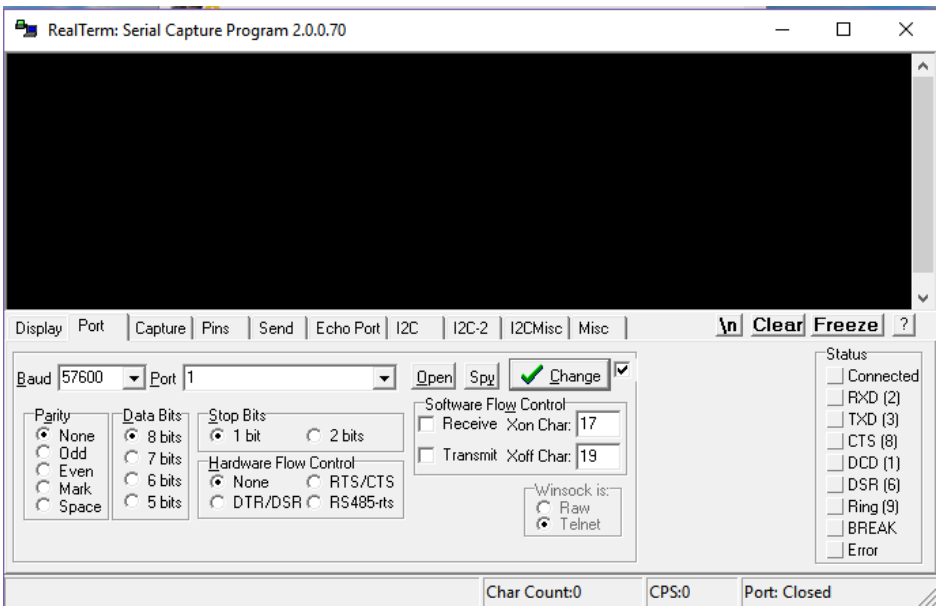
9 TERMINÁL REALTERM

K vysílání a zachytávání dat na sériovém portu je používán bezplatný terminál RealTerm verze 2.0.0.70. V záložce *Display* lze nastavit, v jakém zobrazení budou data zobrazovány na displeji terminálu. Pro zobrazování vstupních znaků z klávesnice je třeba zaškrtnout *HalfDuplex* a *newLinemode*.



Obr. 12: Zobrazení vstupních znaků

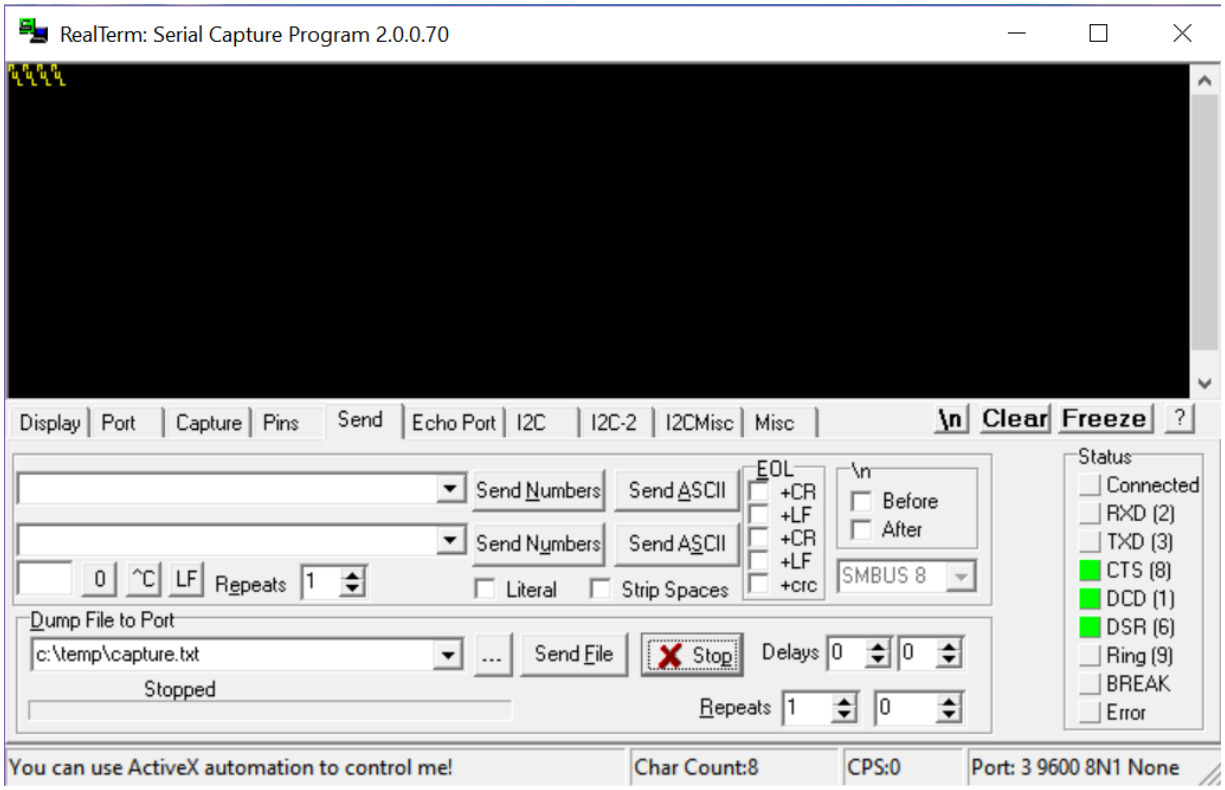
V záložce *Port* lze nastavit přenosovou rychlost a komunikační port. Terminál nabízí možnost povolit Xon, Xoff, počet datových bitů a počet stop bitů. Po stisknutí tlačítka *Open* se terminál připojí.



Obr. 13: Připojení terminálu

Po připojení je možné v sekci status pozorovat stavy jednotlivých komunikačních linek.

Pro odesílání souboru je potřeba přejít do záložky s názvem *Send*. V sekci *Dump File to Port* se po stisknutí tlačítka se třemi tečkami vybere požadovaný soubor z adresáře. Cesta k souboru se zobrazí v textovém boxu. Pod ním je zobrazen ukazatel průběhu odesílání dat neboli *progress bar*, na kterém je možné po stisku tlačítka *Send File* pozorovat, kolik dat je odesláno.

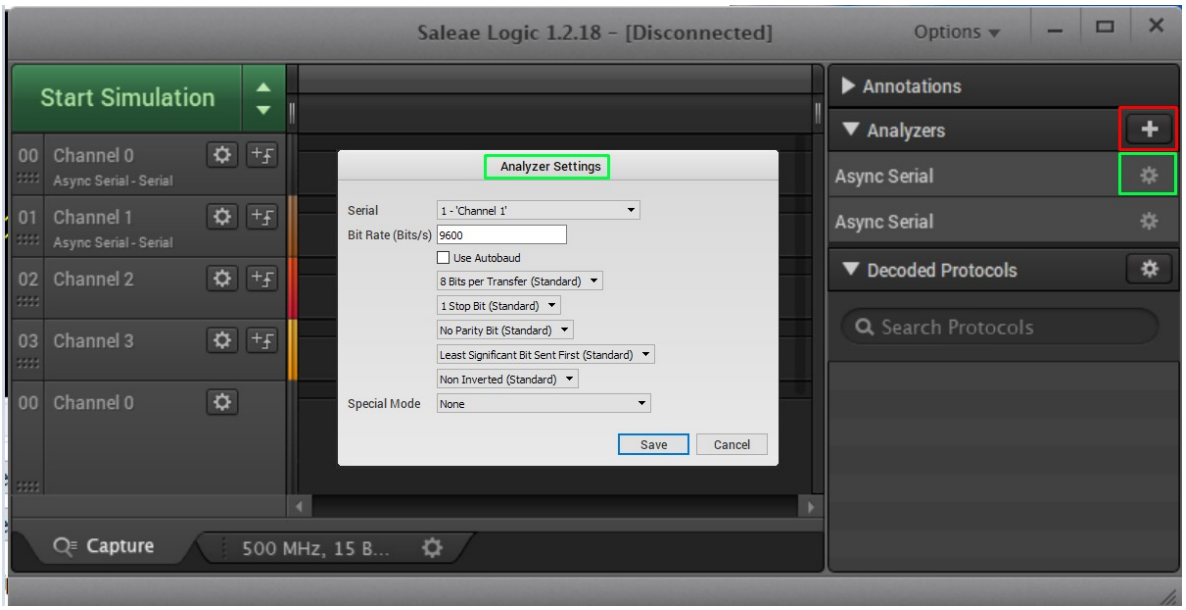


Obr. 14: Odesílání dat přes terminál

10 LOGICKÝ ANALYZÁTOR

V této práci je využíván logický analyzátor SALEAE Logic Pro 8, díky kterému lze sledovat tok dat na pinech určených pro sériovou komunikaci. Obecně se jedná o 8 kanálový logický analyzátor. Připojuje se k počítači pomocí USB 3.0 a používá software Logic pro záznam a zobrazení analogových i digitálních signálů. Logické analyzátory se používají k ladění embedded aplikací. Slouží k ověření funkcionality a chyb například při komunikaci mikrokontroléru s jinou komponentou přes sériové, IIC (Inter-Integrated Circuit) nebo SPI rozhraní. Logický analyzátor se připojí na vstupně výstupní digitální piny, použité pro komunikaci a zaznamenává činnost během testování. Zaznamenaná komunikace se poté zobrazí v okně softwaru Logic.

Pro spuštění zaznamenávání slouží tlačítko *Start Simulation* a pro ukončení *Stop*. V softwaru Logic, v sekci *Analyzers*, se po stisku tlačítka + zvolí komunikace *Async Serial* (červeně zvýrazněná oblast na obrázku číslo 15). Následně je potřeba nastavit jednotlivým kanálům základní specifikaci, jako je přenosová rychlost, počet datových bitů, atd. Na obrázku číslo 15 tomuto nastavení odpovídá zeleně zvýrazněná oblast.



Obr. 15: Nastavení analyzátoru pro kanál 1

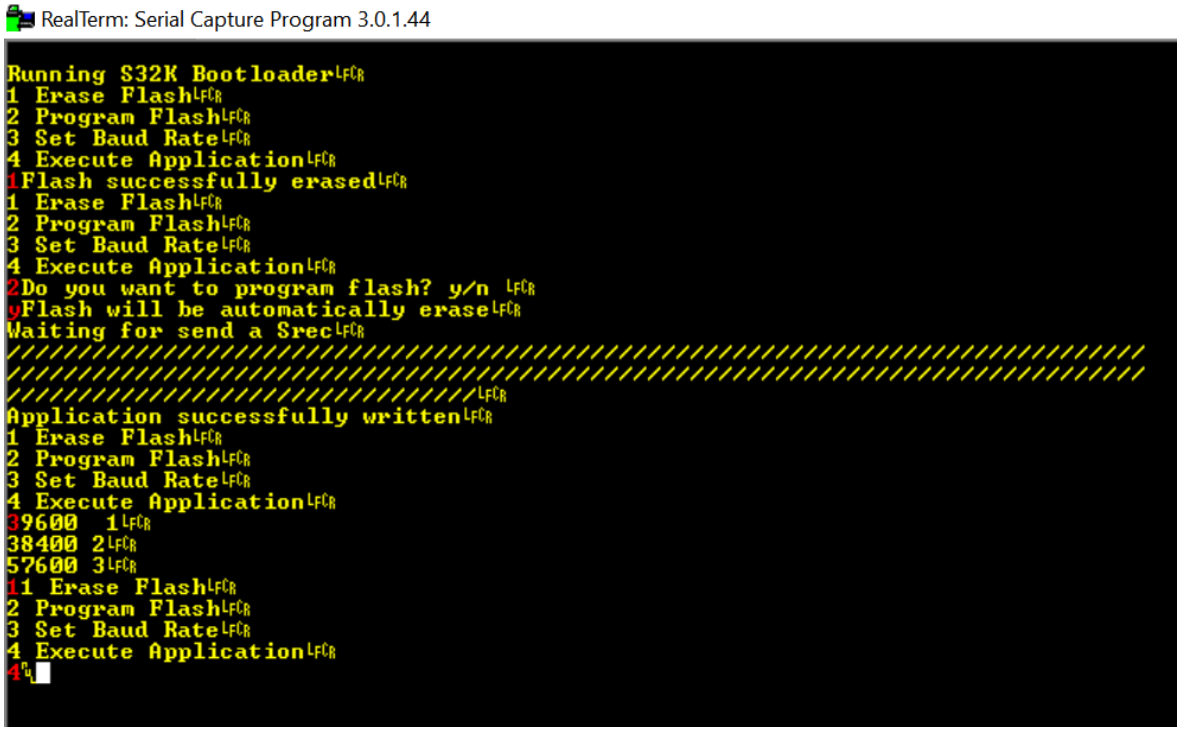
11 PROGRAMOVÁ ČÁST BOOTLOADERU

Jak už bylo výše zmíněno, aplikace bootloaderu slouží k nahrávání uživatelské aplikace do flash paměti MCU. K interakci uživatele s aplikací bootloaderu slouží terminál, přes který jsou odesílána a přijímána data pomocí sériového rozhraní. V případě, že aplikace není naprogramována, se automaticky spustí bootloader. Jinak se bootloader spouští jen v případě, že je stisknuto tlačítko SW3.

Při připojení vývojové desky k počítači, se terminál RealTherm automaticky připojí k virtuálnímu komunikačnímu portu. Pokud se tak nestane, lze se podívat do správce zařízení na konkrétní číslo COM portu.

Uživatelské menu aplikace bootloaderu nabízí následující možnosti:

- Smazání celé flash od počáteční adresy aplikace. Po úspěšném i neúspěšném smazání je vypisováno na terminál příslušné hlášení.
- Programování paměti flash, při kterém se automaticky smaže flash paměť od počáteční adresy aplikace. Z toho důvodu je tato možnost doplněna otázkou, jestli chce uživatel opravdu programovat flash. V případě stisknutí klávesy *y* nebo *Y* program pokračuje v programování, jinak se vrátí zpět do hlavního menu. Po naprogramování každé fráze do flash paměti se vypíše na terminál znak /.
- Změna přenosové rychlosti. Menu nabízí 9600 Kb/s, 38400 Kb/s a 57600 Kb/s. Po změně přenosové rychlosti je potřeba změnit přenosovou rychlost v terminálu. V RealTherm terminálu stačí vybrat požadovanou přenosovou rychlost a stisknout tlačítko *Change*. Po resetu je opět nastavena a výchozí přenosová rychlost 9600 Kb/s.
- Skok do uživatelské aplikace pomocí resetu MCU



Obr. 16: Ukázka aplikace bootloaderu

11.1 Nastavení PLL

Frekvence PLL je nastavena na maximální hodnotu 160 MHz. Je to maximální hodnota frekvence, která může být generována pomocí PLL v normálním pracovním módu.

Tuto frekvenci nastavíme v registru SPLLCFG, podle následující rovnice:

$$\mathrm{SPLL}_{\mathrm{hodiny}} = \frac{\frac{SPLL_{zdroj}}{(PREDIV\ +\ 1)}\ MULT}{2} \qquad (1)$$

Kde:

SPLLL$_{zdroj}$ je frekvence 8 MHz generovaná pomocí oscilátoru.

PREVDIV znamená dělič referenčních hodin PLL a je nastaven na 0

MULT znamená násobitel a je nastaven na hodnotu 40.

U zdroje systémových hodin PLL se nastavují dva děliče SPLLDIV1_CLK a SPLLDIV2_CLK. Oba jsou nastaveny na maximální hodnotu. První z nich dělí 160 / 2 a druhý 160 / 4. Důležité je, aby byly hodiny generované pomocí PLL povoleny až po výše uvedeném nastavení registrů.

Poznámka: v referenčním manuálu se k hodnotě MULT přičítá ještě hodnota 16, pravděpodobně se jedná o chybu. Nastavení hodin bylo testováno na osciloskopu a bez přičítání hodnoty 16 výpočet vycházel správně.

11.2 Nastavení portů

Při nastavování portů se nejprve povolí hodiny pro daný port. Jednotlivé porty u MCU mohou mít různé alternativy pro jejich použití. Tyto alternativy jsou zobrazeny v souboru *S32K144_IO_Signal_Description_Input_Multiplexing.xlsx*.

Porty jsou nastavovány ve dvou případech. Prvním z nich je komunikace přes sériové rozhraní. Aby mohl terminál komunikovat přes sériové rozhraní s MCU, bylo potřeba nakonfigurovat port, který umožňuje nastavení pinu s alternativní funkcí pro UART k přijímání a odesílání dat. Těchto portů se nachází na MCU více, ale bylo potřeba zvolit ten, který je interně propojen s OpenSDA rozhraním. Konkrétně se jedná o POTRC (pin 6 pro příjem a pin 7 pro odesílání). Nastavení je pro LPUART1 a odpovídá mu alternativní funkce číslo 2.

V programu zápis může vypadat následovně:

```
PORTC->PCR[6]|=PORT_PCR_MUX(2);
```

Dalším případem, kdy byl nastaven port, je využití tlačítka pro vstup do programu bootloaderu. Pro tento účel bylo vybráno tlačítko SW3. To je propojeno s pinem číslo 13 na portu C. Tomuto pinu byla nastavena alternativní funkce 1 pro GPIO (General Purpose Input / Output) a port C13 byl konfigurován jako výstup pomocí PDDR (Port Data Direction) registru.

11.3 Watchdog a reset MCU

Je to jeden z modulů MCU s označením WDOG. Jedná se o nezávislý časovač, který je třeba stále obnovovat. Pokud nedojde k jeho obnovení, nastane resetu MCU. Slouží k ochraně proti zacyklení MCU a neplánovanému chování softwaru. Nicméně, byl v programu bootloaderu zakázán.

Mohl by být použit pro reset MCU při skoku do uživatelské aplikace. Namísto toho byla využita možnost registru AIRCR (Application Interrupt and Reset Control Register). Použití tohoto registru nelze najít v referenčním manuálu, ale v obecném uživatelském manuálu pro Cortex –M4 zařízení.

Aby bylo možné zapisovat do tohoto registru, musí být zapsána hodnota 0x5FA do VECTKEY bitového pole. Nastavení SYSRESETREQ do jedničky se spustí systémový požadavek na reset.

11.4 Ochrana paměti Bootloaderu

Aby nemohlo dojít k tomu, že si uživatel přepíše vlastní aplikaci bootloaderu, byla tato část paměti ochráněna. Jakýkoli FTFC (Flash Memory Module) příkaz, který představuje programování nebo mazání paměti nemá vliv na chráněné regiony paměti.

Pro tento účel slouží registr FPROT (Program Flash Protection Register). Funguje na takovém principu, že si rozdělí paměť na jednotlivé regiony. Dostupných regionů je 32. Pro určení konkrétního regionu je potřeba celkovou velikost paměti vydělit hodnotou 32. V aktuálním případě je velikost flash paměti 512 KB. Každý region má tedy 16 KB. Konkrétní zápis se provádí zapsáním 0 do nejnižších dvou bitů registru FPROT[0]. Tím je chráněných 32 KB začínajících na adrese 0x0. Obsah těchto regionů nemůže být změněn.

11.5 Flash paměť

Paměť je používaná pro ukládání programu. Jedná se o nevolatilní paměť, což znamená, že i po odpojení napájení zůstanou data uložena v paměti a neztrácí se.

Flash paměť S32K144 MCU je tvořena jedním blokem o velikosti 512 KB. Flash paměť je rozdělena na sektory, jeden sektor má velikost 4 KB. Jednotlivé sektory lze programovat nebo mazat samostatně.

Aby se nemohlo stát, že uživatelská aplikace přepíše kód bootloaderu, tak je flash rozdělena na dvě části. Nastavení velikosti bootloaderu se provádí v souboru *S32K1xx_flash.ld*.

Stejně tak musí uživatel nastavit velikost a počáteční adresu jeho aplikace. Nastavení pro uživatelskou aplikaci v kódu je ukázáno níže.

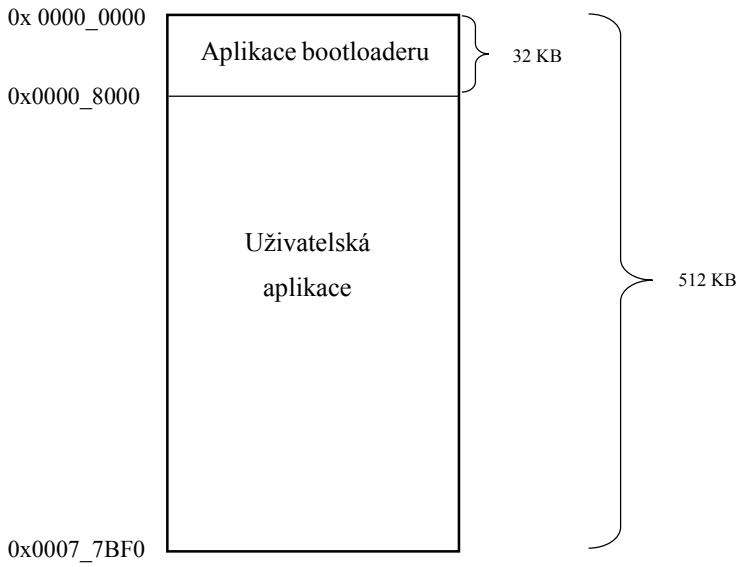
/ Flash */*

m_interrupts (RX) : ORIGIN = 0x00008000, LENGTH = 0x00000400

m_flash_config (RX) : ORIGIN = 0x00008400, LENGTH = 0x00000010

m_text (RX) : ORIGIN = 0x00008410, LENGTH = 0x00077BF0

Paměťová mapa rozdělení flash paměti je zobrazena na obrázku.



Obr. 17: Paměťová mapa

Příkazy pro modifikování obsahu flash paměti vykonává řadič paměti, modul FTFC. Programovací operace je jednosměrná, k obnovení jednotlivých bitů slouží operace mazání. Důležité je podotknout, že flash paměť musí být před programováním ve smazaném stavu. Provádění programových operací bez předchozího smazání je způsob programování, který se nazývá kumulativní a není povolen.

V programu jsou použity tři typy příkazů: programování, mazání a ověřování flash paměti. Tyto příkazy jsou vykonávány z paměti RAM, protože zápis během čtení z flash paměti není povolen. Aby se funkce vykonala z paměti RAM, je použit následující kód, který umístí funkci do sekce *.code_ram*:

__attribute__ ((section(".code_ram"))) uint8_t ProgramFlash(void);

Pro předávání příkazů, adres, dat a dalších parametrů do řadiče FTFC se používá registr FCCOB (Flash Common Command Object).

Předpřipravení záznamu

Provádí se před uložením záznamu do paměti. Je nastaveno tak, že se do paměti uloží jen 24bitová adresa, která odpovídá záznamu S2. V případě čehokoliv jiného se vypíše chybová hláška a paměť je pro jistotu smazána. Také se testuje rozsah paměti. Data se načítají do pole po bajtech. Počet datových a adresových bajtů se zjišťuje ze záznamu.

Programování flash paměti

Příkaz *Program Phrase* je použit pro programování flash paměti. Programuje 8 bajtů a z toho důvodu jsou data v souborovém záznamu zarovnány na 8 datových bajtů. K programování je použit registr FCCOB. Na index 0 až 3 se zapisuje adresa a index 4 až 7 slouží pro data. Takže se dá najednou uložit fráze, která má 8 bajtů.

Mazání flash paměti

Příkaz pro mazání paměti je *Erase Flash Sector*. Tento příkaz smaže všechny bajty nacházející se v sektoru flash paměti. Po vykonání příkazu pro mazání, FTFC ověřuje, že je sektor skutečně smazán. Pokud ověření smazání selže, tak se nastaví FSTAT[MGSTAT0] bit do jedničky.

11.6 Sériová komunikace

Vysílání a přijímání dat funguje na principu, že se při každém znaku, který se odesílá nebo přijímá, vyvolá přerušení. Níže jsou popsány využité funkce.

Inicializace NVIC

Důležité je také zmínit, jakým způsobem se určí konkrétní poloha bitového pole přerušení na základně získaných informací ze souboru uvedeném v kapitole 7.1.

Tab. 1: LPUART vektor přerušení [1]

Vector Offset Address	Cortex-M4 Vector (0-162)	NVIC Interrupt ID	NVIC non-IPR register number	NVIC IPR register number	Name	Module	Instance	Interrupt Type	Groups
0x0000_00C4	49	33	1	8	LPUART	lpuart	LPUART1	Transmit Interrupt	LPUART
0x0000_00C4	49	33	1	8	LPUART	lpuart	LPUART1	Receive Interrupt	
0x0000_00C4	49	33	1	8	LPUART	lpuart	LPUART1	Error Interrupt	
0x0000_00C4	49	33	1	8	LPUART	lpuart	LPUART1	Overrun Interrupt	

Výpočet počáteční adresy pro registry NVICISER a NVICICPR

$$= IRQ \; mod \; 32 \; = \; 33 \; mod \; 32 \; = \; 1$$

Inicializace LPUART

Při inicializaci se nastavuje zdroj hodin. Po nastavení je potřeba hodiny povolit pro LPUART. Dále se nastavuje přenosová rychlost. V CTRL registru se povolí vysílač a přijímač pro LPUART1. LPUART1 byl vybrán proto, že je interně propojen s OpenSDA.

transmitChar(char send)

V této funkci se čeká tak dlouho, dokud se v zásobníku pro odesílané znaky nějaký neobjeví. Pokud ano, vyvolá se TxIRQ přerušení. V případě, že jsou nastaveny požadavky pro odesílání XOn nebo XOff, posílá tyto znaky terminálu. Jakmile jsou v zásobníku znaky, tak je posílá. Jinak je přerušení TxIRQ zakázáno do té doby, než se objeví v zásobníku další znak. Následně se vrátí do funkce, zakážou se přerušení a vloží se znak do zásobníku. Poté se povolí přerušení.

transmitString(char data_string[])

Při posílání více znaků najednou, například na výpis menu, se tyto znaky uloží do pole, které je procházeno postupně. Znaky jsou po jednom vypisovány na terminál.

receiveChar(int wait)

Po zavolání této funkce je implementovaná smyčka, ve které se čeká na přijetí znaku. V tu chvíli se vyvolá přerušení RxIRQ.

Poté je potřeba zakázat přerušení. Znak se ukládá do kruhového zásobníku, počet dostupných míst se v zásobníku sníží o jedna a počet přijatých znaků se zvýší. V případě dosáhnutí konce zásobníku se počet přijatých vynuluje.

V přerušení se také řeší, jestli byl přijat XOn nebo XOff. Při přijetí XOff se zakáže a při přijetí XOn se zase povolí přesušení vysílače. XOff se posílá, když je počet dostupných míst menší než hodnota XOffCount a zároveň XOff ještě nebyl poslán.

Následně se vrátí do funkce receiveChar(wait), zakáže přerušení a vezme ze zásobníku hodnotu, která je návratovou hodnotou a povolí přerušení.

Tato funkce má jeden parametr *wait*, který je nastaven na hodnotu 1, když při načítání záznamu dojde k chybě. V tom případě je potřeba počkat, než terminál odešle všechny zbývající znaky přijímaného souboru a až poté se vrátit do hlavního menu a začít reagovat na uživatelský vstup.

Požité knihovny

Při založení projektu byly vygenerovány některé knihovny: *system_S32K144.h, startup.h, S32K144_features.h, S32K144.h*. Ve zdrojovém souboru *LPUART.c* jsou použity a upraveny funkce pro implementaci XOn a XOff *char transmitChar(char send), char receiveChar(int wait), void RxISR(void), void TxISR(void)*. A funkce *uint8_t GetHexByte(uint8_t *MemByte)* z existujícího projektu Simple Serial Bootloader for S12Z - AN draft. [9]

11.7 Spuštění aplikace

Aby bylo možné spustit uživatelskou aplikaci ještě před tím, než program spustí aplikaci bootloaderu, byla do souboru *startup_S32K144.S* zavolána funkce *application()*. V této funkci se inicializuje tlačítko *SW3* a poté se porovnává, jestli je stisknuto. Dále se porovnává, jestli jsou první 2 bajty na adrese 0x8000 zapsány. V případě, že tlačítko je stisknuto nebo program není nahrán ve flash paměti, spustí se aplikace bootloaderu. Tato funkce je vytvořena v souboru *startup.c*. Konkrétně je volána ze začátku sekce *Reset_Handler*. V této části programu je volána funkce, která skočí na reset vektor uživatelské aplikace. Nejjednodušším zápisem bylo použití instrukce assembleru.

11.8 Programová část uživatelské aplikace

K výslednému programu je také přiložena jednoduchá uživatelská aplikace, u níž je vygenerován Srecord. Slouží pro ukázku funkčnosti bootloaderu. Výstupem této aplikace je blikání RGB LED. Tato LED má nastavenu červenou barvu.

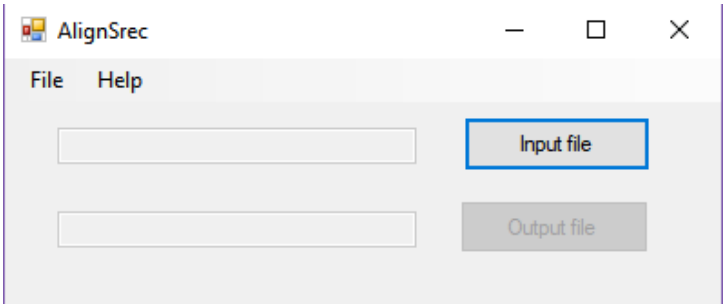
12 PROGRAMOVÁ ČÁST ZAROVNÁNÍ SOUBOROVÉHO FORMÁTU SREC

Pro přípravu dat uživatelské aplikace k nahrávání do flash paměti MCU, byl vytvořen program s názvem AlignSrec. Tento program byl naprogramován v jazyce C# v programovacím prostředí Microsoft Visual Studio Express 2013, který je freeware. Pomocí tohoto programu jsou data zarovnána na 8 bajtů a adresa je upravena na 24bitovou adresu. Na obrázku níže se nachází ukázka nezarovnaného Srecordu, který ještě nebyl upraven po vygenerování z S32 Design Studia. Adresa je 16bitová a je zvýrazněna modrou barvou. Číslo jedna za S určuje typ adresy, a proto je programem také modifikován.

```
S11387401A60044B4FF6FF729A60BD465DF8047BD5
S1138750704700BF0020054020C528D980B489B0E7
S1138760 00AF2A4BFB602A4A2A4B9A4218D00023B6
S1138770FB610AE0274AFB6952F823202449FB697C
S1138780 41F82320FB690133FB61FB689A08FB690C
S1138790 9A42EFD84FF0E0231D4AC3F8082D04E0B5
S11387A04FF0E0231B4AC3F8082D1B4BBB611B4B46
S11387B07B611B4BBB60BA687B69D31AFB6107E022
S11387C0BB695A1CBA617A69511C796112781A70B2
S11387D0FB695A1EFA61002BF2D1124B3B61124B1A
S11387E07B607A683B69D31AFB6104E03B695A1CDD
S11387F03A6100221A70FB695A1EFA61002BF5D106
S1138800 2437BD465DF8047B704700BF00040000B8
S1138810 0080FF1F0080000000084FF1F2C880000E0
S10F8820 2C8800000084FF1F0484FF1F4C
S903841167
```

Obr. 18: Ukázka nezarovnaného Srecordu

Aplikaci představují dvě tlačítka, z nichž tlačítko pro výběr výstupního souboru je povoleno až po vybrání vstupního souboru, který chceme upravit. Po stisknutí *Input file* tlačítka se zobrazí dialogové okno pro výběr souboru. Filtr je nastaven na soubory .s19, ale jde zvolit i filtr zobrazující všechny soubory. Pro *Output file* tlačítko je to podobné s tím rozdílem, že po vybrání cílového souboru se program automaticky vykoná. Dvě textová pole slouží k zobrazení názvu souboru.



Obr. 19: Aplikace pro zarovnání záznamů z Srecordu

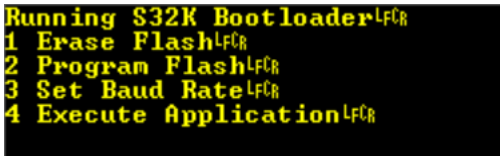
Základní funkčnost programu se nachází v metodě *srecRW()*. Při načtení prvního řádku záznamu se vytvoří pole, které představuje paměť a proto se jeho délka rovná velikosti flash paměti vydělená počtem datových bajtů a jeho šířka se rovná počtu požadovaných datových bajtů, což je v tomto případě 8. Pole je naplněno tak, aby každý bit byl nastaven na hodnotu 1, protože ve smazané flash paměti jsou bity takto nastaveny. V poli flag jsou nastaveny příznaky, které určují změněné řádky v paměti a pouze ty budou zapsány do nového souboru.

Ve funkci *getAddressLine(line)* se selektují jednotlivé záznamy podle jejich adresy do jednotlivých polí pro typ, počet bajtů, adresu a data. Adresa je zarovnána na 24bitovou adresu. Jakmile je záznam takto rozdělen, jsou data ukládána do připraveného pole v případě, že je adresa v rozsahu paměti. Index řádku je určen aktuální adresou podělenou počtem datových bajtů a index na řádku je vypočítán pomocí modula počtu datových bajtů z aktuální adresy. Kontrolní součet je počítán zvlášť ve funkci *checksumCounter(newLine)*.

13 NÁVOD NA POUŽITÍ BOOTLOADERU

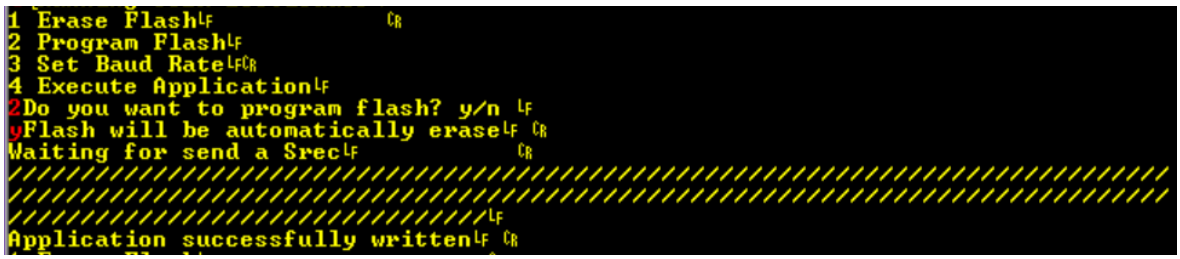
Návod k použití popisuje správné připojení vývojové desky k počítači, a také nastavení jumperů. Dále je popsáno jak nastavit terminál pro komunikaci a postup pro poslání souboru.

1. Z toho důvodu, že bude vývojová deska napájena pomocí USB, je třeba jumper *J107* umístit na pozici 2-3.
2. USB kabelem je nutno propojit počítač s vývojovou deskou (microUSB port *J7*). Po jeho připojení k napájení by měly svítit dvě diody *D2* a *D3*.
3. Po spuštění terminálu RealTherm je třeba nastavit přenosovou rychlost na 9600Kb/s, port, 1start bit, 1 stop bit a 8 datových bitů, poté je potřeba stisknout tlačítko *Open*.
4. Po spuštění vývojového prostředí S32 Design Studia je potřeba zkompilovat projekt a nahrát ho do MCU.
5. Aplikace bootloaderu se spustí automaticky, protože tam uživatelská aplikace ještě nebyla nahrána. V případě, že je aplikace nahraná v paměti MCU, začne se vykonávat. Držením tlačítka *SW3* a vyvoláním resetu po stisku tlačítka reset se spustí aplikace bootloaderu.
6. Bootloader je spuštěn a na terminálu je zobrazeno hlavní menu aplikace.
   První možnost 1 slouží k mazání flash paměti. Možnost 2 k programování flash paměti s automatickým smazáním.



Obr. 20: Hlavní menu

7. Po výběru možnosti 2 je požadováno potvrzení. Pro potvrzení programování flash slouží znak *y* nebo *Y*.



Obr. 21: Úspěšné nahrání programu aplikace do paměti

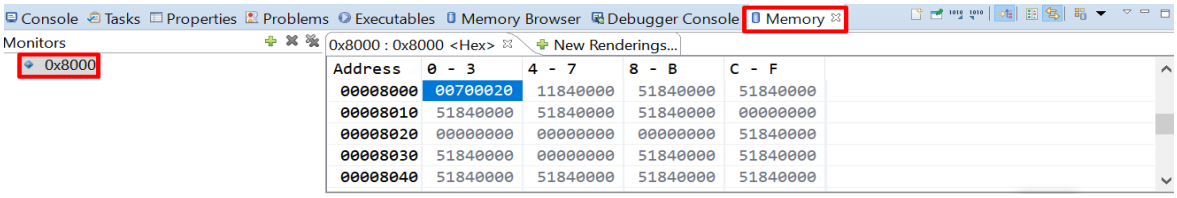
8. Posílání Srecordu pomocí terminálu se nachází v záložce *Send*. Pomocí tlačítka s třemi tečkami lze vybrat požadovaný Srecord. Následně se začne soubor posílat stiskem tlačítka *Send File*. Po úspěšném nahrání se zobrazí hláška *Application succesfully written*.
9. Po úspěšném nahrání lze spustit aplikaci stiskem klávesy 4.
10. Bootloader také nabízí možnost změny přenosové rychlosti za běhu programu. Pod číslem 3 se nachází nastavení pro přenosovou rychlost 9600 Kb/s, 38400 Kb/s a 57600 Kb/s. Po zvolení jiné přenosové rychlosti, než je aktuálně nastavena, je potřeba změnit přenosovou rychlost na terminálu, jak je popsáno výše. Nevýhodou je, že se po změně přenosové rychlosti nevypíše menu. Po stisku klávesy se buď vykoná událost menu, nebo se menu znovu zobrazí. Po resetu se nastaví původní přenosová rychlost.

Návod k použití programu pro zarovnání Srecordu:

1. Program pro zarovnání Srecordu se spouští pomocí souboru s názvem *AlignSrec.exe*
2. Po stisku tlačítka *Input file* se zobrazí dialogové okno pro výběr požadovaného souboru pro zarovnání z adresáře.
3. Pro provedení zarovnání souboru je potřeba stisknout tlačítko *Output file* a vybrat nebo vytvořit soubor, do kterého bude původní soubor zarovnán.
4. Po stisku tlačítka *Ok* se akce vykoná a zobrazí se hláška o úspěšně provedené operaci.

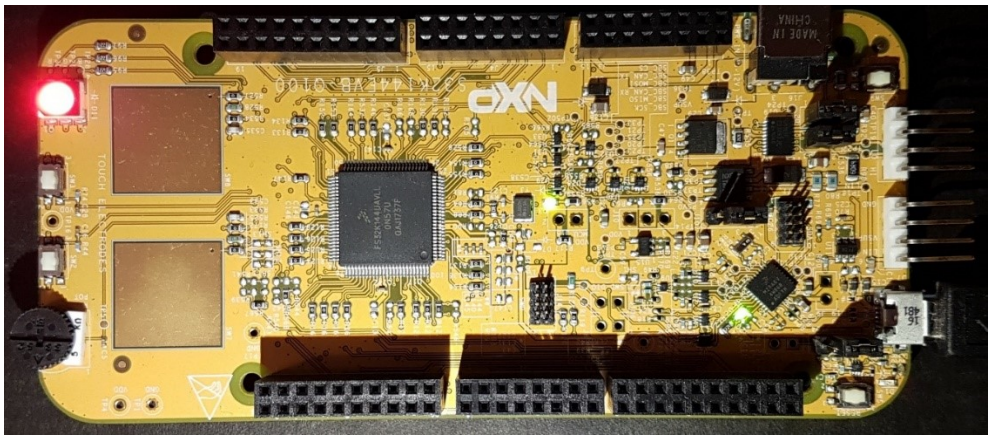
14 OVĚŘENÍ FUNKČNOSTI

Funkčnost aplikace byla ověřena pomocí vývojového prostředí S32 Design Studio a uživatelské aplikace. Pomocí vývojového prostředí lze ověřit, že byla aplikace skutečně zapsána na požadovanou adresu v režimu ladění. Po naprogramování aplikace byl program pozastaven. V okně *Memory* byla zkontrolována flash paměť od počáteční adresy uživatelské aplikace a porovnána se záznamy v souboru Srec.



Obr. 22: Zobrazení adresového prostoru flash

Výše uvedené ověření sloužilo především v případě, kdy ještě nebyl implementován skok do uživatelské aplikace. Po dokončení aplikace bootloaderu ověření probíhalo skokem na počáteční adresu uživatelské aplikace, po kterém se úspěšně spustil její kód.

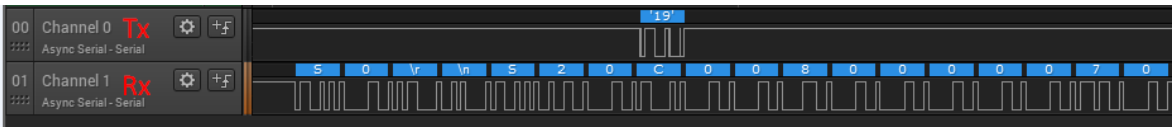


Obr. 23: Uživatelská aplikace

Pro jistotu bylo vyzkoušeno více různých druhů uživatelských aplikací, ale všechny z nich, byly naprogramovány v prostředí S32 Design Studio. I když se podařilo každou z nich úspěšně nahrát, bylo potřeba otestovat jestli XOff a XOn funguje. Z důvodu rychlosti BUS clock totiž nebylo vůbec potřeba posílat XOff. Pro ověření tedy bylo vytvořeno zpoždění, před vložením znaku do zásobníku přijímače. Díky tomu byla vytvořena fronta a XOff byl vygenerován. Nejjistějším způsobem, jak ověřit data, která jsou odesílána a přijímána, bylo použití analyzátoru. Na obrázku 23 je vidět, jak je analyzátor připojen na vyvedené LPUART1 Tx a Rx piny.

Obr. 24: Měření toku dat na Rx a Tx pinech

Výsledkem bylo ověření, že vysílač správně poslal hodnotu XOff, která je nastavena na 0x13 v programu bootloaderu. Na obrázku níže lze vidět výsledek získán pomocí analyzátoru. Kanál 0 je připojen na Tx pin a posílá po částečném zaplnění zásobníku přijímače XOff. Na kanálu 1 jsou zobrazena data, která terminál posílá. Bohužel terminál nepřestal data posílat. Je to pravděpodobně způsobeno tím, že tento způsob komunikace není podporován OpenSDA USB-serial rozhraním na desce S32K144. Další variantou může být použití jiného terminálu, proto byl program testován na hyperterm terminálu. Bohužel se stejným výsledkem.



Obr. 25: Ověření odesílání XOff

ZÁVĚR

Cílem práce bylo prostudování potřebných informací a hardwarových vlastností MCU S32K144 k realizaci sériového bootloaderu. Klíčové informace byly získány z referenčního manuálu v revizi 6 [1] a také ze schématu pro vývojovou desku S32K144EVB. Způsob implementace sériového bootloaderu byl navrhnut vzhledem k získaným informacím tak, aby se bootloader nacházel na začátku adresového prostoru a za ním uživatelská aplikace.

Programové vybavení sériového bootloaderu bylo vytvořeno ve vývojovém prostředí S32 Design Studio. Softwarová architektura obsahuje dvě nezávislé aplikace. Obě jsou naprogramovány v jazyce C. Jedna z nich je aplikace bootloaderu a druhá je uživatelská aplikace.

Uživatelská aplikace byla vytvořená pro ověření správné funkčnosti bootloaderu a je přiložena k diplomové práci. S využitím vývojového prostředí může být uživatelská aplikace nahrána do paměti MCU a muže být laděna, ale pro výrobní účely je vhodné nahrávat uživatelskou aplikaci do paměti pomocí bootloaderu. K tomuto účelu slouží soubor Srecord vygenerovaný ve vývojovém prostředí v projektu uživatelské aplikace. V tomto souboru se mohou vyskytovat záznamy, které nejsou zarovnány na stejnou délku. Z toho důvodu byl vytvořen program ve vývojovém prostředí Visual Studio Express v jazyce C#. Jeho primární funkce je předpřipravení záznamu na jeho následné nahrávání. Zarovnává záznamy, ale také upravuje adresy v záznamu na 24bitovou adresu. Tento program je také přiložen k diplomové práci.

Při ověřování funkčnosti, bylo zjištěno, že terminál nepozastavuje posílání dat po přijetí znaku XOff. Některé USB to Serial převodníky tuto funkčnost podporují, ale bohužel v průběhu psaní diplomové práce nebyl k dispozici pro testování. Existuje možnost, že je problém na straně terminálu, ale testovaní na jiném terminálu, tuto možnost nepotvrdilo. Z tohoto důvodu nebyla tato funkčnost potvrzena. V tomto případě je výhodou vysoká frekvence *BUS clock*, díky níž nebylo potřeba pozastavovat posílání dat z terminálu, protože se zásobník pro příjem dat nestihl naplnit a data byla správně nahrána do flash paměti. Následně byla uživatelská aplikace úspěšně spuštěna.

Práce na sériovém bootloaderu bude i nadále pokračovat. Nejbližším plánem je otestování funkčnosti implementace XOn / XOff a rozšíření aplikace o možnost bootloaderu přeprogramovat sám sebe.

SEZNAM POUŽITÉ LITERATURY

[1] NXP SEMICONDUCTORS. S32K1xx Series Reference Manual, Rev.4, [online]. 2017 [cit. 2017-11-20]. Dostupné z: https://www.nxp.com/docs/en/reference-manual/S32K-RM.pdf

[2] NXP SEMICONDUCTORS. S32K144 EVB: QUICK START GUIDE. In: *S32K144EVB* [online]. REV4.2, s. 45 [cit. 2018-05-13]. Dostupné z: https://www.nxp.com/docs/en/quick-reference-guide/S32K144EVB-QSG.pdf

[3] Wikipedia contributors. SREC (file format) [Internet]. Wikipedia, The Free Encyclopedia; 2018 Feb 26, 11:22 UTC [cited 2018 May 13]. Available from: https://en.wikipedia.org/w/index.php?title=SREC_(file_format)&oldid=827729295.

[4] Wikipedia contributors. Software flow control [Internet]. Wikipedia, The Free Encyclopedia; 2017 Nov 15, 10:35 UTC [cited 2018 May 13]. Available from: https://en.wikipedia.org/w/index.php?title=Software_flow_control&oldid=810456382

[5] USB MULTILINK USB Multilink Debug Probe. *PEmicro* [online]. [cit. 2018-05-13]. Dostupné z: http://www.pemicro.com/products/product_viewDetails.cfm?product_id=15320168&productTab=1

[6] USBDM: USBDM Debugger interface for Freescale RS08,HCS08,HCS12,Coldfire and ARM-Kinetis Devices. *USBDM* [online]. V4.12 [cit. 2018-05-13]. Dostupné z: http://usbdm.sourceforge.net/USBDM_V4.12/html/index.html

[7] NXP SEMICONDUCTORS. ROMERO, Osvaldo. S32K144EVB-Q100. In: *S32K144EVB-SCH* [online]. 2016, s. 6 [cit. 2018-05-13]. Dostupné z: https://www.nxp.com/downloads/en/schematics/S32K144EVB-SCH.pdf

[8] NXP SEMICONDUCTORS. S32K1xx Data Sheet, Rev.4, [online]. 2017 [cit. 2017-11-20]. Dostupné z: http://www.nxp.com

[9] ŠESTÁK, Radek. Simple Serial Bootloader for S12Z - AN draft. *NXP Semiconductors* [online]. 2018 [cit. 2018-05-13]. Dostupné z: https://community.nxp.com/docs/DOC-335384

[10] Wikipedia contributors. Universal asynchronous receiver-transmitter [Internet]. Wikipedia, The Free Encyclopedia; 2018 Apr 19, 04:15 UTC [cited 2018 May 15]. Available from: https://en.wikipedia.org/w/index.php?title=Universal_asynchronous_receiver-transmitter&oldid=837169104.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

UART	Universal Synchronous / Asynchronous Receiver and Transmitter
COM	Communication
MCU	Microcontroller
USB	Universal Serial Bus
CAN	Controller Area Network
IIC	Inter-Integrated Circuit
SPI	Serial Peripheral Interface
DSP	Digital Signal Processor
FPU	Floating Point Unit
ADC	Analog-to-Digital Converter
VLPR	Very Low Power RUN
VLPS	Very Low Power Stop
DAC	Digital-to-Analog Converter
LPUART	Low Power Universal Synchronous / Asynchronous Receiver and Transmitter
LPI2C	Low-power Inter-integrated circuit
LPSPI	Low-power Serial peripheral interface
LQFP	Low Profile Quad Flat Pack
SBC	System Basis Chip
LIN	Local Interconnect Network
SCI	Serial Communication Interface
LED	Light-Emitting Diode
EEPROM	Electrically Erasable Programmable Read-Only Memory
JTAG	Joint Test Action Group
SWD	Serial Wire Debugging

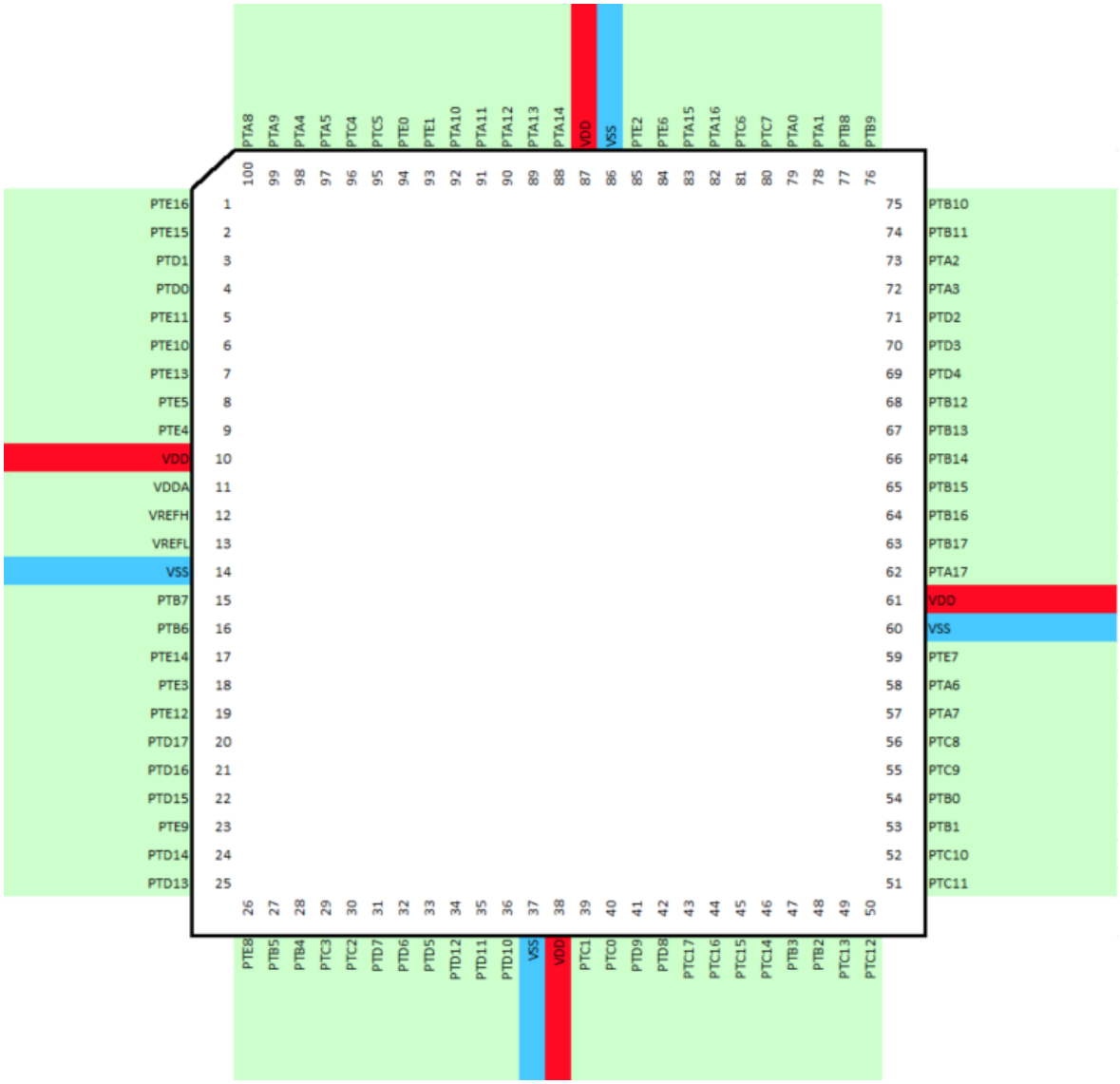
BDM	Background Debug Mode
SIRC	Slow Internal Reference Clock
FIRC	Fast Internal Reference Clock
PLL	Phrase Lock Loop
QSPI	Quad Serial Peripheral Interface
NRZ	Non Return To Zero
NVIC	Nested-Vectored Interrupt Controller
CPU	Central Processin Unit
RAM	Random-access memory
GPIO	General Purpose Input / Output
PDDR	Port Data Direction
AIRCR	Application Interrupt and Reset Control Register
FPROT	Program Flash Protection Register
FTFC	Flash Memory Module
CMP	High-speed Comparator
FCCOB	Flash Common Command Object
MPU	Memory Protection Unit

SEZNAM OBRÁZKŮ

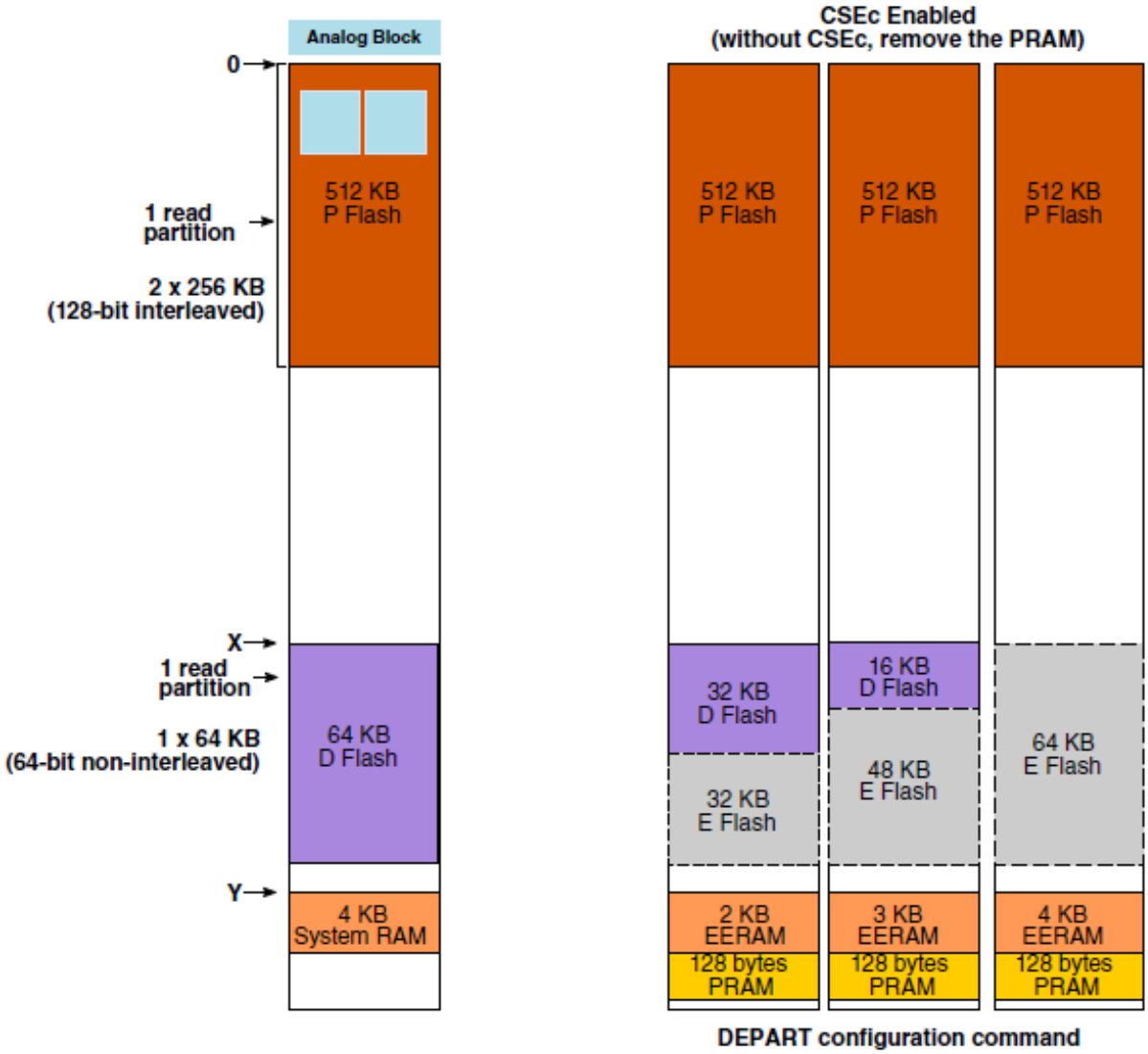
SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: PINOUT S32K144 [1]



PŘÍLOHA P II: PAMĚŤOVÁ MAPA [1]



PŘÍLOHA P III: CD

- Adresář *Dokumentace* - obsahuje diplomovou práci ve formě fulltext.pdf.
- Adresář *Bootloader* - obsahuje vyexportovaný projekt z S32 Design Studia pro sériový bootloader
- Adresář *UserApp* - obsahuje vyexportovaný projekt z S32 Design Studia pro uživatelskou aplikaci.
- Adresář *AlignSrec* – obsahuje vyexportovaný projekt z Visual Studia Express 2013 pro zarovnání souboru typu Srecord