CanSat

poznamky

2.díl – programování palubního počítače s STM32

Praha 2011

Vladimír Váňa

Úvod

V současné době jsou k dispozici výkonné 32 bitové počítače s jádrem ARM. Cenově jsou srovnatelné s ATMega. Vzhledem k značnému rozšíření ARM se jejich použití stává levnější, než využití ATMega. Např. startkit **STM32VL Discovery** firmy **STMicroelectronics** je levnější, než obdobné startkity *Arduina*. Za méně peněz tak dostaneme výkonnější 32bitový počítač. Vyšší výkon využijeme např. při zpracování obrazu apod. Proto jsme pro palubní počítač zvolili právě STM32 firmy **STMicroelectronics** mající v Praze 8 vývojové laboratoře .



ST Market Position



Rank 2010	Rank 2009	Company	Country of origin	Revenue (million \$ USD)	2010/2009 changes	Market share
1	1	Intel Corporation	USA	40 394	+24.3%	13.2%
2	2	Samsung Electronics	South Korea	27 834	+60.8%	9.3%
3	3	Toshiba Semiconductors	 Japan 	13 010	+26.8%	4.3%
4	4	Texas Instruments	USA	12 944	+34.1%	4.3%
5	9	Renesas Electronics (1)	• Japan	11 840	+129.8%	3.9%
6	7	Hynix	South Korea	10 577	+69.3%	3.5%
7	5	STMicroelectronics	France Italy	10 290	+20.9%	3.4%
8	13	Micron Technology (2)	USA	8 853	+106.2%	2.9%
9	6	Qualcomm	USA	7 200	+12.3%	2.4%
10	14	Broadcom	USA	6 506	+52.1%	2.1%
11	15	Elpida Memory	• Japan	6 678	+74.2%	2.3%
12	8	Advanced Micro Devices	USA	6 355	+22.0%	2.1%
13	11	Infineon Technologies	Germany	6 226	+39.7%	2.0%

1.1 Procesory s jádrem ARM

Procesory ARM lze v současně době nalézt v mnoha elektronických zařízeních. Své uplatněni nalézají všude tam, kde je potřeba velký výpočetní výkon při malé spotřebě např. při zpracováni obrazu, vyhodnocování GPS signálů, při mobilní komunikaci, atd.

Historie vzniku procesorů ARM sahá až ke konci 80-tých let 20. Století, kdy vznikl první RISC procesor **ARM** ve firmě *Acorn RISC Machine*. Následující úspěchy s procesory ARM1, ARM2 a ARM3 vedly k osamostatnění divize procesorů ARM pod novým názvem *Advanced RISC Machine*. Následovaly další procesory ARM6 - ARM 11 a také přejmenování firmy na ARM Ltd.

1.1.1 ARM1

Historicky prvním typem byl ARM1. Tento procesor byl použit v několika vývojových systémech pro BBC a PC ale především to byl prototyp, který byl rychle nahrazen typem ARM2. Procesoru ARM1 bylo vyrobeno řádově 100 ks, neobsahovaly instrukce násobení ani koprocesor a jeho instrukce.

1.1.2 ARM2

Nedlouho po ARM1 byl vyvinut typ ARM2, který již obsahoval 27 registrů z toho zároveň přístupných 16. Měl jíž čtyři pracovní módy (USR, IRQ, FIQ a SVC), umožňoval adresování až 64MB, měl třístavové zřetězení při zpracovávání instrukcí a jeho hodinový kmitočet byl 8MHz. Tomu odpovídá výpočetní výkon 4 -4,7 MIPS.

1.2.3 ARM3

Tento typ byl jen malým vylepšením předcházejícího typu. Oproti typu ARM2 byla přidána paměť cache (4k), koprocesor a zvýšena rychlost hodinového kmitočtu na 12 – 33 MHz. Výpočetní výkon při 12 MHz byl 7MIPS, při 24MHz 13,26 MIPS a při 33MHz 17,96 MIPS.

1.2.4 ARM4 a ARM5

Tyto procesory nebyly nikdy vyrobeny.

1.2.5 ARM6

První komerční procesor s plnou 32-bitovou adresovou sběrnicí (adresovat lze až 4 GB), 32 32bitových registrů, 6 pracovních módů (USR, IRQ, FIQ, SVC, Abort a Undefine). Procesoru ARM6 bylo vyráběno několik verzí.

1.2.6 ARM7

Identický s ARM6 ale s větší rychlostí. Některé varianty obsahují hardwarovou násobičku. Většina změn nastala v časování různých signálů. Typ ARM700 má zvětšenou paměť cache (8k) a větší účinnost s ohledem na spotřebu. Na 40 MHz má typ ARM710 výkon 36MIPS. Dostupné verze jsou:

ARM7	– holý chip
ARM7D	– jádro s podporou ladění
ARM7DM	– ARM7D s hardwarovou násobičkou
ARM7DMI	 – ARM7DM s podporou rychlého zpracování přerušení
ARM70DM	– ARM7DMI jako chip
ARM700	- ARM7 + MMU + cache + WriteBack Buffer
ARM7500	- ARM7 + MMU + cache + Writeback Buffer + IOMD + VIDC20

Uvedené typy mohou být i ve verzi Thumb (možnost zpracování 16-ti bitových instrukcí)

1.2.7 ARM8

Přímo kompatibilní s ARM6 a 7. Obsahuje 5-ti stavové zřetězení zpracovávání instrukcí. Cache paměť zůstává stejně veliká ale slouží pro WriteBack, dále je přidána 64-bitová násobička. Chipy jsou vyráběny technologií 0,5 um, která umožňuje dosažení výkonu až 80MIPS při 3,3V a 80MHz.

1.2.8 StrongARM

Rychlá varianta rodiny ARM. Architektura je podobná jádru ARM8, také využívá 5-ti stavové zřetězení. Rozdíl je v rozdělení paměti cache na dvě části. Na cache pro instrukce a pro data (Harvardská architektura). Každá cache má velikost 16kB. Při výrobě je použita technologie 0,35 um a výkony jsou od 115 MIPS při 100MHz až po 230 MIPS při 200MHz.

1.2.9 ARM9

Opět vychází z předchozího typu ARM8, ze kterého si bere 5-ti stavové zřetězení ale vnitřní architektura je Harvardská, jako u StrongARM. Velikosti pamětí cache může být 0kB – ARM9TDMI, 4kB/4kB – ARM940T, 8kB/8kB – ARM922T a 16kB/16kB – ARM920T. Použitá technologie je opět 0,35 um a výkon je až 200MIPS při 180 MHz.

1.2.10 ARM10E

Cesta za dalším zrychlení vedla k procesoru se šestistupňovým zřetězení, rozdělenou pamětí cache (32k/32k) a technologii 0,13 um

1.2.11 ARM11

Tento typ má již osmi-stavové zřetězení, podporu cache paměti (4 – 64 kB) a technologii výroby opět 0,13 um. Hodinová frekvence je přes 1GHz a výkony jsou 740 Dhrystone 2.1 MIPS. Spotřeba je 0.6mW/MHz (0.13µm, 1.2V).



1.2.12 Cortex M3

Dalším vylepšením procesorů ARM7 jsou procesory s novým jádrem ARM Cortex M3. Oproti ARM7 nabízejí zejména vyšší výpočetní výkon při stejné frekvenci hodin, rychlejší odezvu na přerušení, až 240 úrovní přerušení, výkonnější a paměťově úspornější instrukční soubor atd. Jádro založeno na tří-stupňovém zřetězení s Harvardskou architekturou a s inovovanou instrukční sadou Thumb (Thumb-2). Technologie výroby je 0,18 um. Při testu Dhrystone dosahuje výkonu 1,25 DMIPS/MHz a spotřeba je 0,19mW/MHz. Rodina ARM Cortex zahrnuje typy ARM Cortex-A8, ARM Cortex-A9 MPCore,ARM Cortex-A9 Single Core Processor, ARM Cortex-M1, ARM Cortex-M3 a ARMCortex-R4(F). Tyto procesory se vyrábí ve třech sériích, u každé z nich je imple-mentován Thumb-2 instrukční soubor. Série ARM Cortex-A aplikační procesory určeny pro komplexní operační systémy auživatelské aplikace, podporuje 32-bit ARM, Thumb a Thumb-2 instrukční soubory. Série ARM Cortex-R Jsou určeny do segmentu embedded systémů pro práci v reálném čase, podporují 32-bit ARM, Thumb a Thumb-2 instrukční soubory. Série Jsou určeny zejména do segmentu, kde je kladen důraz především na nízkou cenu, podporuje pouze Thumb-2 instrukčním souborem.

1.3 Procesory STMicroelectronics

STMicroelectronics patří k firmám nabízejícím 32bitové mikrokontroléry s jádrem od firmy ARM. Novinkou v sortimentu jsou mikrokontroléry STM32_založené na nejnovějším jádru ARM[®] Cortex™ -M3. Toto jádro je navrženo specielně pro embeded aplikace vyžadující vysoký výkon, běh v reálném času a nízký příkon (pozn.: určeno písmenem M za pomlčkou). Další předností tohoto jádra jsou jeho nízké nároky na plochu čipu. Díky tomuto faktu je i výsledná cena mikrokontroléru nízká, takže může konkurovat i některým výkonným 8bitovým mikrokontrolérům. To, že je tento mikrokontrolér směrován do oblasti jednodušších, nebo chcete-li lacinějších, aplikací je vidět i podle použitých LQFP pouzder s relativně nízkým počet pinů (48pinů, 64pinů a 100pinů). To však neznamená, že jde o "ošizený" mikrokontrolér, spíše naopak. Jádro pracuje s tříúrovňovým překrýváním fází vykonávání instrukcí se spekulativním vykonáváním skokových instrukcí. Oboje techniky minimalizují nutný počet taktů k vykonání instrukcí, takže i instrukce dělení a násobení jsou vykonávány během jednoho taktu. Pro lepší porovnání je vhodné uvést dosahovaný počet instrukcí za sekundu. Jádro se může "pyšnit" 1,25 DMIPS/MHz (Dhrystone 2.1 Benchmark). To však není vše co nové jádro přináší. Další novinkou je nový instrukční soubor Thumbe-2. V podstatě jde o pokračování úspěšné myšlenky s redukcí 32bitové délky instrukcí tak, aby se délka programu zmenšila a výkon jádra zůstal co nejvíce zachován. Používáním kratších instrukcí se zmenšuje velikost výsledného kódu programu a protože může být použito menší, a tedy lacinější, paměti programu, cena řešení se snižuje. Že investice do "vývoje" Thumb -2 se vyplatila, je možné posoudit i ze "suchých" čísel: jádro Cortex™-M3 je při vykonávání Thumb[®]-2 o 70% efektivnější než ARM7TDMI-S vykonávající instrukce Thumb[™] přičemž kód využívající Thumb[™]-2 je o cca 10% kratší než Thumb. Zde se sluší připomenout, že nové jádro dokáže vykonávat i původní sadu Thumb. Další novinkou je možnost vnořování přerušení (nesting interrupts), přičemž počet úrovní přerušení je maximálně 256 a tyto priority se navíc dají dynamicky měnit.





V sortimentu firmy STMicroelectronics najdeme tři řady mikrokontrolérů-základní, výkonnou a nízkopříkonovou. Rozdíly spočívají v maximální frekvenci jádra (36 MHz oproti 72 MHz), menší paměti SRAM (max. 16 KB oproti 20 KB) a menšímu počtu periferií. V základní řadě mikrokontrolrů není u žádného mikrokontroléru implementováno rozhraní USB 2.0FS ani řadič sběrnice CAN ani PWM. Ostatní parametry a funkce jsou u obou řad shodné. I u nejmenšího představitele základní řady, typu STM32F101C6, najdete velké množství periferií, neboť i o zdroji hodinového kmitočtu se dá, díky širokým možnostem, hovořit jako o periferii. Vyjma standardního krystalového oscilátoru pro externí krystal 4 MHz až 16 MHz, najdeme u této periferie taktéž interní, kalibrovaný, RC oscilátor s kmitočtem 8 MHz, interní RC oscilátor 32 kHz (pro watchdog i obvod RTC) i oscilátor pro 32 kHz krystal (pro obvod RTC). Oba rychlejší oscilátory jsou napojeny na jednotku PLL, která dokáže "vynásobit" kmitočet 2× až 16×. Tento vynásobený kmitočet je pak použit pro taktovní mikrokontroléru. Kmitočet z "pomaloběžných" oscilátorů je pak použit pro obvod RTC. Obvod je dělán velmi jednoduše. Sestává se z programovatelné předěličky na jejímž výstupu jsou sekundové impulsy. Tyto impulsy jsou čítány 32 bitovým čítačem, jehož stav může být komparován s hodnotou uloženou v registru RTC_ALR. Už podle názvu jde poznat, že se tzv. "alarm", tj. možnost vzbudit mikrokontrolér po uplynutí nadefinovaného počtu sekund. Interním 32 kHz oscilátorem pak jsou obvody watchdog, přičemž jeden je klasický, druhý je typu "windowed", tj. řízeny dva



znovunastavení obvodu MUSÍ proběhnout v určitém časovém období, tzv. "oknu", jinak, i když dojde k pokusu o restart (reload) tohoto obvodu, je generován reset.

I když jsou obvody zajišťující základní chod mikrokontroléru zajímavé, z hlediska uživatele budou obvody časovačů podstatně zajímavější. U zmiňovaného nejjednodušího mikrokontroléru nové řady jsou implementovány celkem dva. Pokud je někdo překvapen nízkým počtem, měl by vědět, že obvody časovačů jsou u 32bitových mikrokontrolérů podstatně komplexnější než běžně používané časovače u 8 bitových mikrokontrolérů. Nejinak je tomu v tomto případě a tak v **jeden kanál** časovače se skládá z časové základny kterou tvoří 16 bitový up/down (nastavitelný) čítač, kterému je předřazen 16 bitový programovatelný předdělič. S tímto čítačem spolupracují celkem 4 jednotky "input capture/output compare/PWM". Možnost provozovat čítač v režimu "up" tak i "down" dává tušit, že při generování PWM signálů budeme mít možnost si zvolit mezi PWM signály "zarovnanými" na hranu nebo na střed ("edge or center aligned PWM"). První z nich je vhodná pro výkonnové aplikace ve spínaných zdrojích, zatímco druhá jmenovaná je vhodná pro řízení elektromotorů. Po tomto, byť krátkém, výčtu možností jednoho kanálu časovače, je jasné, že možnosti časovašů, i u tohoto nejjednoduššího mikrokontroléru v řadě, jsou více jak postačující. Větší typy pak diponují celkem 3 kanály časovačů.

Další částí, která je u 32bitových mikrokontrolérů často používána, jsou řadiče přímého přístupu do paměti. Tyto řadiče mají za úkol odlehčit jádru mikrokontroléru od rutinní činnosti, jakými přenosy dat jsou. Druhým důvodem jsou možné vysoké komunikační rychlosti mikrokontroléru s okolím. Každý ze 7 kanálů dokáže přenášet data mezi dvěma místy v paměti, mezi pamětí a periferií a mezi dvěma periferiemi. Zajímavou vlastností je možnost volby rozdílné velikosti dat u zdroje a cíle přenosu (bytově orientovaná periferie a slovně orientovaná paměť). Samozřejmostí je pak podpora kruhového bufferu.

Další periferie jsou již dobře známé z 8 bitových mikrokontrolérů i když některé jejich vlastnosti byly přizpůsobeny novým podmínkám. Na čipu najdeme jedno rozhraní I²C, jedno rozhraní SPI a celkem dva kanály velmi populárního rozhraní USART. Dále je na čipu implementován 12 bitový (!) analogově-digitální převodník (ADC) s předřazeným analogovým multiplexerem 10:1.

Přestože výkon těchto mikrokontrolérů značně překonává možnosti běžných 8 bitových mikrokontrolérů, zůstává spotřeba mikrokontroléru na přijatelné úrovni. Při maximální frekvenci 36MHz, běžícím programu (z interní paměti Flash) a povolených všech periferiích je typický odběr přibližně 22mA. Příznivý vliv na spotřebu měla jednak moderní technologie a též nízké pracovní napětí, neboť mikrokontrolér pracuje s napájecím napětím 3,3V.

1.4 Startkit STM32VL Discovery

STMicroelectronics je výrobce startkitu STM32VL Discovery <u>http://www.st.com/stm32-discovery</u> (<u>http://www.st.com/internet/evalboard/product/250863.jsp</u>)





Kit se skládá ze dvou částí. Část na obrázku nahoře je nazvána ST-Link (obsahuje ARM čip STM32F103C8T6) a část ve spodní polovině je vlastní **Value Line** mikrořadič typu STM32F100RBT6. Tyto části tvoří spolu integrální součást. Po obou delších stranách má kit 2 x 28 pinů a na příčné straně ještě 6 dalších pinů, tyto piny můžeme použít např. pro připojení LCD s HD44780. Pokud použijeme obslužný sw pro LCD z *Katedry měření ČVUT*, bude LCD připojen takto:

- PB10 DB4
- PB11 DB5
- PB12 DB6
- PB13 DB7
- PB14 E
- PB15 RS



R/W je spojen s GND

STM32F100RBT6 obsahuje 128 kB Flash, 8 kB RAM a je zapouzdřen v LQFP 64, tedy poměrně velkém pouzdře 10 x 10 mm.

Kit obsahuje 4 LED, z toho 2 LED slouží pro ST-Link část, kde jedna signalizuje napájení (z USB konektoru) a druhá komunikaci prostřednictvím SWD. Další dvě LED jsou připojeny k Value Line mcu (barva zelená=PC9 a modrá=PC8).



1.5 Palubní počítač Cansatu SPŠE Ječná



STM32F100RBT6







Tento počítač obsahuje STM32F100RBT6, stejně jako startkit STM32 VL Discovery. Můžeme proto programy odladit na STM32VL Discovery a poté beze změn nahrát do palubního počítače.

Dále je uveden stručný návod jak nahrát vytvořený program do procesoru pro zde zmíněný palubní počítač.

1. Propojíme palubní počítač s PC pomocí sériové linky RS232 (nemáme-li na PC RS232, lze použít i převodník USB -> RS232)

2. Připojíme napájení + 3,3 V palubního počítače

V.Váňa: CanSAT Book for Students – part 2 12



3. Stiskneme a držíme tlačítko BOOT připojené k palubnímu počítači a stiskneme a uvolníme tlačítko RESET, rovněž vně připojené k palubnímu počítači. Poté uvolníme i tlačítko BOOT. Takto je mikroprocesor připraven k nahrání programu.

4. Na PC spustíme aplikaci STMicroelectronics Flash Loader Demonstrator V2.1.0 a na úvodní obrazovce vybereme komunikační port (COM0, COM1, ..), ostatní nastavení ponecháme a pokračujeme dále.

5. Další dvě okna stačí pouze potvrdit a zastavíme se až u okna s nabídkou *Download to device*, kde vyber náš přeložený a zkompilovaný soubor s příponou *.hex. Po potvrzení se již nahraje program do procesoru.



Výše uvedená vnější tlačítka Boot a Reset můžeme umístit např. na "základní desku"

Kromě již zmíněných tlačítek **Boot** a **Reset** je na desce i tlačítko **User** a dvě LED diody (modrá a zelená). Tlačítko user i obě LED jsou připojeny ke stejným pinům STM32F100, jako u startkitu **STM32VL Discovery**.







Podrobnější návod na obsluhu Flash Loader Demonstrator naleznete ve firemní literatuře STM.

Program můžeme nahrát také pomocí protokolu **SWD** z startkitu **STM32VL Discovery**. Propojit desky stačí třemi vodiči za předpokladu, že připojená deska má své vlastní napájení.

14

```
Připojení SWD:
  SWD 2 <-> SWCLK (PA14),
  SWD 3 <-> GND,
  SWD 4 <-> SWDIO (PA13).
```





Podrobnější popis je na <u>http://neuron.feld.cvut.cz/publicwiki/STM32/STM32-VL-DISCOVERY</u> a na <u>http://neuron.feld.cvut.cz/publicwiki/Ulohy/STM32</u>

2. Práce v prostředí uVision4 Keil

2.1 Instalace prostředí Keil

Spustíme instalační soubor MDK420.exe. Objeví se



Setup MDK-ARM V4.20	
Welcome to Keil μVision Release 3/2011	
This SETUP program installs: MDK-ARM V4.20 This SETUP program may be used to update a previous p However, you should make a backup copy before process It is recommended that you exit all Windows programs be	product installation. eding. fore continuing with SETUP.
Follow the instructions to complete the product installation	n.
— Keil μVision4 Setup	<< Back Next S Cancel

Klikneme na tlačítko Next, poté se objeví licenční ujednání

Setup MDK-ARM V4.20	
License Agreement Please read the following license agreement carefully.	
To continue with SETUP, you must accept the terms of the License Agreement. To accept the agreement, click the check box below.	
END USER LICENCE AGREEMENT FOR MDK-ARM THIS END USER LICENCE AGREEMENT ("LICENCE") IS A LEGAL AGREEMENT BETWEEN YOU (EITHER A SINGLE INDIVIDUAL, OR SINGLE LEGAL ENTITY) AND ARM LIMITED ("ARM") FOR THE USE OF THE SOFTWARE ACCOMPANYING THIS LICENCE. ARM IS ONLY WILLING TO LICENSE THE SOFTWARE TO YOU ON CONDITION THAT YOU ACCEPT ALL OF THE TERMS IN THIS LICENCE. BY CLICKING "I AGREE" OR BY INSTALLING OR OTHERWISE USING OR COPYING T	
I agree to all the terms of the preceding License Agreement	
— Keil μVision4 Setup Cancel	



Zaškrtneme souhlas s licenčním ujednáním a klikneme na tlačítko Next, tím postoupíme k výběru adresáře

Setup MDK-ARM V4.20	×
Folder Selection Select the folder where SETUP will install files.	
SETUP will install $\mu Vision4$ in the following folder.	
To install to this folder, press 'Next'. To install to a different folder, press 'E folder. — Destination Folder	frowse' and select another
C:\Keil	Browse
Update Installation: Create backup tool folder	
Backup old files to C:\Keil\Backup.001	
— Keil μVision4 Setup	
<< Back	Next

Ponecháme navržené umístění a klikneme na tlačítko Next, dostaneme



Setup MDK-ARM V4.20							
Customer Informati Please enter your i	ion nformation.						
Please enter your n	Please enter your name, the name of the company for whom you work and your E-mail address.						
First Name:	Y						
Last Name:	Microsoft						
Company Name:	Microsoft						
E-mail:	Y						
— Keil μVision4 Setup							
	<< Back Next >> Cancel						

Vyplníme výše uvedené informace a opět klikneme na Next. Potom začne instalace

Setup MDK-ARM V4.20	
Setup Status	
μVision Setup is performing the requested operations. Install Files Installing Clock.ini.	
— Keil μVision4 Setup	<< Back Nerty> Cancel



Po ukončení instalace

Setup MDK-ARM V4.20	— X —
File installation completed	
μVision Setup has installed all files successfully.	
 Retain current µVision configuration. Add example projects to the recently used project list. 	
Preselect Example Projects for	
Simulated Hardware	_
— KeiluVision4 Setun	2
	<< Back Next >> Cancel

Změníme předpřipravené příklady projektů z Simulace hardware na STM32 Discovery Boards:



Setup MDK-ARM V4.20	×
File installation completed	
μVision Setup has installed all files successfully.	
 Retain current µVision configuration. Add example projects to the recently used project list. 	
ST STM32Fxxx \ STM321xx \ STM32-Discovery Boards	-
— Keil μVision4 Setup << Back	Cancel

Potvrdíme tlačítkem Next, dostaneme ještě

Setup MDK-ARM V4.20	
Keil μVision4 Setup completed MDK-ARM V4.20	
μVision Setup has performed all requested operations successfully. ✓ Launch Driver Installation: "ULINK Pro Driver V1.0"	
✓ Show Release Notes.	
— Keil μVision4 Setup << Back	Finish Cancel



Potvrdíme tlačítkem Finish. Poté se tento driver nainstaluje



Nakonec se objeví informační stránka s poznámkami



A Keil Development Suite for ARM - Release Notes - Window	s Internet Explorer						
C:\Keil\ARM\Hlp\Release_Notes.htm							
🖕 Oblíbené položky 🛛 🚖 🏉 Navrhované weby 🕶							
6 Keil Development Suite for ARM - Release Notes	🛅 🔻 🖾 👻 🖃 😽 Vástroje 🕶 🕢 🗸						
Release Notes for RealView Microcontroller Development Kit This file contains release notes and last minute changes. Information in this file, the accompany manuals, and software is							
Contents	All rights reserved.						
contents							
 <u>What's New in the RealView Microcontro</u> <u>Example Programs</u> 	ller Development Kit						
3. <u>Device Database™</u>							
4. <u>Peripheral Simulation</u>							
6. <u>Contact Details</u>							
The RealView Microcontroller Development Kit supports software development and debugging for ARM7/9, Cortex-M, and Cortex-R based microcontroller devices. A detailed description of the RealView Microcontroller Development Kit may be found in Complete User's Guide Selection which may be accessed on the µVision Project Workspace - Books page. The Getting Started User's Guide gives you a good starting point, since it introduces you to the RealView Microcontroller Development Kit and gives guidelines for programming ARM devices.							
What's New in the RealView Microcontroller Development Kit							
The following sections list the changes instituted in each release of the ARM toolset.							
RealView Microcontroller Development Kit - Version 4.20							
[ARM Processor Support]							
Hotovo	Počítač Chráněný režim: Vypnuto 🎻 👻 🔍 100% 💌						

Stránku uzavřeme, uVision 4 Keil máme nainstalovaný.

2.2.1 Vytvoření našeho prvního programu – blikač LED 1

Nejprve si vytvoříme adresářovou strukturu pro snadnější orientaci v projektu. Tuto strukturu můžeme vytvořit z několika podadresářů. Jejich počet, pojmenování, stejně jako i v nich vnořených podadresářů si volí programátor sám. Nicméně se ustálilo několik doporučených struktur. Zvolíme např. následující strukturu.





Náš první projekt nazveme **program01** a stejně pojmenujeme i adresář, který ho bude obsahovat. Složku **libraries**, včetně všech souborů zkopírujeme z **STM32VLDISCOVERY firmware package** (získáme z **AN3268** – aplikační poznámka /app. note/ 3268 stažením z firemních stránek STMicroelectronics). Do podadresáře **source** ještě zkopírujeme soubor **stm32f10x_conf.h**

Soubor Označit Příkazy Síť Zobrazit Konfigurace	Start				Nápověda
2 *** 🗧 🖬 👫 \$* 4 🔶	🗯 🙃 🗱 🗱 🏘	📧 🗱 🍙 📃			
[-c-] ▼ [] 32 496 160 kB z 147 353 596 kB v \	[-c-] • [] 32 496 160 k	cB z 147 353 596 kB vo	lných		N
c:\ARM_priklady\program01\source*.* * ▼	c:\zaloha\arm\STM32\ST	M32MCU\an3268\stm3	2vldiscovery_pack	age\Project\Demo\inc*.	* *
1Název Přípc Velikost	1Název	Přípc Ve	likost Datum	Atributy	
▲[]	金[]	<0	IR> 10.04.2011	14:26 —	
	stm32f10x_conf	h	3 188 14.09.2010	15:49 -a-	
	stm32f10x_it	h	1 901 13.09.2010	14:36 -a-	

Nyní již spustíme vývojové prostředí Keil uVision4

W µVision4		
File Edit View Project Flash Debug	Peripherals Tools SVCS Window Help	
🗋 🗋 🚰 📓 🛃 🛔 🖓 🗠		- 🔍 🥙 🔍 🔹 🔿 🖓 🗐 🔸
🔗 🖾 🕮 🧼 🔐 🔛		
Project 🦊 🖸		
Build Output		
<		
n		4

V menu vybereme **Project** \rightarrow **New** μ **Vision Project** ...



Wision4		
File Edit View	Project Flash Debug Peripherals Tools SVCS Window H	elp
i 🗅 🗃 🖌 🥔	New µVision Project	
	New Multi-Project Workspace	
Designet	Open Project	
rojec	Close Project	
	Export	•
	Manage	•
	Select Device for Target	
	Remove Item	
	A Options	Alt+F7
	Clean target	
	Build target	F7
	Rebuild all target files	
	Batch Build	
	Ø Translate	Ctrl+F7
	Stop build	

Dostaneme okno

Jspořádat 🔻 Nová složka			= • 0
 Plocha Stažené soubory Knihovny Dokumenty Hudba Obrázky Videa Počítač 	Název položky	Datum změny 22.5.2011 20:58 23.5.2011 13:00 23.5.2011 13:22	Typ Složka soubo Složka soubo
Mistní disk (C:)		m	
Název souboru:	ίστανμ		2



Zvolíme název souboru

🖌 🖉 🕌 🕨 Počítač 🕨 Místní	disk (C:) ARM_priklady program01	Prohledat: program01
Uspořádat 🔻 Nová složka		III - 🔞
 Plocha Stažené soubory Knihovny Dokumenty Hudba Obrázky Videa Počítač 	Název položky	Datum změny Typ 22.5.2011 20:58 Složka soubor 23.5.2011 13:00 Složka soubor 23.5.2011 13:22 Složka soubor
Místní disk (C:) Místní disk (E:)	+ (
Název souboru: program01 Uložit jako typ: Project Files (*.	uvproj)	
Skott claška		Ulořit Storno

A potvrdíme tlačítkem **Uložit**. Dostaneme



PU			
Vendor: Actel			
Device:			
Toolset:			
Data base	Descripti	on:	
E Actel			*
🕀 💊 Analog Devices			
🖻 💊 ARM			
E Atmel			
Austria Micro Systems	H		
E Cirrus Logic			
Ember			
Energy Micro			
Freescale Semiconductor			
E Fujitsu Semiconductors			
Generic			
		I	*
It in the gounded by Fnillps)	- 4		•

Tj. jsme dotázáni na konkrétní typ CPU. V Data base nejprve vybereme výrobce **STMicroelectronics** A kliknutím na křížek vlevo od **ikonky STMicroelectronics** dostaneme již seznam jednotlivých čipů



PU			
Vendor: STMicroelectronics			
Device:			
Toolset			
Toolact.			
Data base		cription:	
STMicroelectronics		223	
STA2051			
CI STM32F100C4			
STM32F100C6			
STM32F100C8			
STM32F100CB			
STM32F100R4			
STM32F100R6			
STM32F100R8			
STM32F100RB			
STM32F100RC			
STM32F100RD			
STM32F100RE			÷
STM32F100V8	+		

Vybereme STM32F100RB



PU Vendor: STMicroelectronics Device: STM32F100RB Toolset: ARM	
Data base	Description:
STMicroelectronics	ARM 32-bit Cortex-M3 Microcontroller, 24MHz, 128kB Flash, 8kB SRAM, PLL, Embedded Internal RC 8MHz and 40kHz, Real-Time Clock, Power Saving Modes, JTAG and SWD, 7-channel DMA, Nested Interrupt Controller, 12-bit 16-ch A/D Converter, 2 12-bit 2-ch D/A converters, 51 Fast I/O Port 2 SPI, 2 I2C, 3 USART, CRC calculation unit, SysTick Timer, 2 Watchdog Timers, 16-bit 6-ch advanced timer with PWM, 6 16-bit Timers with Input Capture, Output Compare and PWM
- CI STM32F100V8 -	- III

A výběr potvrdíme tlačítkem **OK**. Dostaneme dotaz



Potvrdíme tlačítkem **Ano**, dostaneme



C:\ARM_priklady\program01\program01.uvproj - µVision4	
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help	
🗋 🗃 🛃 🕹 🔤 🖉 クク クク クク クク ク ク ク ク ク ク ク ク ク	🕞 🗟 🥙 🍭 💿 📀 🗞 💼 🗣 🔦
📨 🕮 🥔 🚉 🎬 Target 1 💽 🔊 📥 🗟	
Project 7 🖸	
E Source Group 1	
startup_stm32f10x_md_vl.s	
	\searrow
Project Books {} Func 0, Temp	
Build Output	џ 🖬
	^
κ.	*
	Simulation

V podokně **project** se nám zobrazila struktura projektu. Nyní tuto strukturu upravíme, tj přejmenujeme název projektu **Target 1** na **program01**, složku **Source Group 1** přejmenujeme na **startup**. Dále vytvoříme složky **source**, **output** a **libraries**. Klikneme proto na ikonku **File Extensions**, **Books and Enviroment**

C:\ARM_priklady\program01\program01.uv	proj - µVision4	
File Edit View Project Flash Debug	Peripherals Tools SVCS Window Help	
🗋 🎯 🖼 👹 🕹 🖧 🗠 😭	← → 巻 巻 巻 巻 澤 澤 /// //(- 🔍 🐡 🔍 🖕 🔹 🔿 🚷 🔳 - 🌂
🔗 🕮 🕮 🧼 🗮 🛛 🎇 🕅 Target 1		
Project 4 23	File Extensions, Books and Environment Configure file extensions, books and environment	

Dostaneme



Project Components	Folders/E	tensio	ons Bo	ooks					
Project Targets: Target 1	<u>n</u> 🗙	†	€ Gr	oups: ource Group	1	╳│	4	Files: startup_stm32f10x_md_v	<mark>× ↑ ↓</mark> .s
Set as C	urrent Targe	t						Add Files	

Kurzor umístíme (popř. ponecháme) v políčku **Project Targets** a zmáčkneme klávesu **F2**. Nyní již můžeme název projektu přejmenovat



oject Components Folders/Extensions	Books	
Project Targets:	Groups: 🖄 🗙	Image: Startup_stm32f10k_md_vl.s
Set as Current Target		Add Files

Obdobně přemístíme kurzor do políčka Groups a stiskneme klávesu F2



Project Targets: 🖄 🗙 🗲	Groups: Market Y ↓	Files:	↑ ↓
program01	Source Group 1	startup_stm32f10x_md_vl.s	

A adresář přejmenujeme na startup



oject Components	olders/Extension	s Books		
Project Targets: program01	[™] X ↑ •	Groups: E	✓ ★ ✓ Files: startup_stm32f10;	× + ↓ c_md_vl.s
Set as Curre	ent Target	. 1	Add	l Files

A potvrdíme vše tlačítkem OK. Nyní máme strukturu

	1992.92.22	Flash	Debug
0	3 H		9 6
10		prog	ram01
			4 🖂
	i 🥔	i 🧼 🚉 🙀	i 🥔 🔜 🙀 prog

Nyní vytvoříme další adresáře struktury projektu. Proto znovu klikneme na ikonku File Extensions, Books and Enviroment

V.Váňa: CanSAT Book for Students – part 2 33



Project Tamete:	X X	Groupe	↓ ↓ ×	Filee X	1+1
program01		startup		startup_stm32f10x_md_vl.s	
Set as Current	Target			Add Files	1

V části **Groups** (i **Project Targets**) si všimněme ikonek



Umožňují přidávat, mazat a měnit pořadí složek. Klikneme na nejlevější ikonku



roject Components Folders/Extensions	Books	
Project Targets: È X ★ ↓	Groups: 🕅 🗙 🗲 🗲	Files: ★ ★
	ß	
Set as Current Target		Add Files

Nyní napíšeme jméno vytvářené složky



A stiskneme **Enter**. Tím máme tuto položku vytvořenou a opět klikneme na ikonku vytvořit a vytvoříme položku **output** a poté ještě **source** a **CMSIS**. Potvrdíme vše tlačítkem **OK**. Dostaneme





Ještě potřebujeme do adresářů projektu soubory. Takže opět spustíme File Extensions, Books and Enviroment

oject Components Folders/Extensions	Books	
Project Targets: Mrs ★ ★ ↓	Groups: X +	Files: ★ ★ ★ startup_stm32f10x_md_vl.s
		ß
Set as Current Target		Add Files

36

Označíme složku libraries a poté klikneme na tlačítko Add Files


nponents, Environr roject Components	Folders/Ex	Books tensions	Books				
Project Targets: program01		↑ ↓	Groups: startup librartes output source CMSIS	<u>₩</u> × +	Files:		× + +
Set as Cur	rent Target		I			Add Files	
			OK	Cancel			Help

Dostaneme



Project Targ	Add Files to Group 'libraries'	<u> </u>
program01	Oblast hledán í: 🌗 program01	• E 🕂 📰 •
	Název položky	Datum změny Ty
	ibraries	22.5.2011 20:58 SI
	output	23.5.2011 13:00 SI
	u source	23.5.2011 13:22 5
	•	
	Název	Add
	Souboru: Soubory typu: C Source file (*.c)	Close
	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	

Dále v okně Add Files to Group libraries klikneme na položku libraries

Oblast hledán í: 🛛 🏭 libraries	1	- 🗢 🖻 💣 💷	•
Název položky		Datum změny	Т
🔒 CMSIS		23.5.2011 13:15	SI
STM32F10x_StdPeriph_Driver	2	23.5.2011 13:15	SI
	Veliko	SC: 975 KD	
<	Složky	y: inc, src ory: Release_Notes_for_	STM32
∢ Název souboru:	Složky Soubo	/: Inc, src ory: Release_Notes_for Ad	STM32



Nyní vybereme STM32F10x_StdPeriph_Driver

Add Files to Group 'libraries'		x
Oblast hledání: 🚺 STM32F10x_StdPeriph_Driv 💌	← 🗈 💣 📰 ▾	
Název položky	Datum změny	T
🍌 inc	23.5.2011 13:15	SI
SIC .	23.5.2011 13:15	SI
۰ [Þ
Název souboru:	Add	
Soubory typu: C Source file (*.c)	Close	
	C2	

A konečně vybereme položku **src**

Oblast hledání: 🌗 src	🝷 🗢 🖻 🔻
Název položky	Datum změny
misc 📉	7.9.2010 10:12
stm32f10x_adc	7.9.2010 10:12
stm32f10x_bkp	7.9.2010 10:12
stm32f10x_can	7.9.2010 10:12
stm32f10x_cec	7.9.2010 10:12
۲. [] III	
Název souboru:	Add
Soubory typu: C Source file (*.c)	✓ Close

Tak jsme se konečně dopracovali k cíli a budeme vybírat potřebné knihovní soubory. Vybereme vždy soubor a potvrdíme tlačítkem Add. Takto vybereme následující čtyři soubory misc.c, stm32f10x_flash.c , stm32f10x_gpio.c a stm32f10x_rcc.c a nakonec stiskneme tlačítko Close. Dostaneme

V.Váňa: CanSAT Book for Students – part 2 39



Project Components Fold	lers/Extensions Books	1		
Project Targets:) 🗙 🗲 🗲 Groups	e 🛅 🗙 •	†	Files: X 🕈 🗲
program01	startup libraries output source CMSIS			misc.c stm32f10x_flash.c stm32f10x_gpio.c stm32f10x_rcc.c
				\searrow

Dále přejdeme ke složce **source**



Project Components	Folders/Ex	tensions	Books			
Project Targets:)	Groups:	Files:		× + +
program01			startup libraries output			
			source			
Set as Cu	urrent Target			Ľ	ACZFI	es

Klikneme na Add Files



Project Targe program01	Add Files to Group 'source'		X ↑ ↓
	Oblast hiedan I: Source		-
	Nazev polozky	Datum zmeny	<u></u>
	stm32f10x_conf	14.9.2010 15:49	
	• [•
	Název stm32f10x_conf	Add	
	Soubory typu: Text file (*.txt; *.h; *.inc)	→ Close	
			_

A vybereme soubor **stm32f10x_conf.h** a přidáme ho tlačítkem **Add** a poté klikneme na **Close**



roject Components	Folders/Ex	tensions	Books			
Project Targets:		† f	Groups:	<u>™</u> ×++	Files:	× + +
program01			startup libraries		stm32f10x_conf.h	
			source			
Set as Cu	urrent Target				Add Files	

Nakonec přidáme soubory do složky CMSIS



Project Components Folders/Extensions	Books	
Project Targets: Mr ★ ★ ↓	Groups: X + + startup libraries output source CMSIS	Files: X + 4
Set as Current Target		Add Files

Klikneme na Add Files a v podadresáři libraries najdeme adresáře CMISS a ATM32F10x_StdPeriph_Driver



Oblast hledán í: 🚺 libraries	-	.
Název položky	Datum změr	у Т
CMSIS	23.5.2011 13:	15 S
JSTM32F10x_StdPeriph_Driver	23.5.2011 13:	15 S
• m		,
Název souboru:		Add
	-	Close

V CMSIS vybereme

Oblast hledán í	í: 🔒 Core Support	- 🗢 🖻 🖻	* 📰 🕇
Název položi	ky ^	Datum zn	něny 1
core_cm	3	7.9.2010 1	0:12 (
∢ [Název	III core cm3		Add
∢ Název souboru:	III core_cm3		- <u>A</u> dd

A dále



🖗 Add Files t	o Group 'CMSIS'		×
Oblast hledán	í: 🔒 STM32F10x	- 🔁 🖆 📰	•
Název polož	ky ^	Datum změny	Тյ
📕 startup		23.5.2011 13:16	SI
system_s	tm32f10x	7.9.2010 10:12	C
•	Ш		•
Název souboru:	system_stm32f10x	Ad	d
Soubory typu:	C Source file (*.c)		e
	-5x1 (YR (WY		

Dostaneme

Project Components Folders/Extensions	Books	
Project Targets: 📉 🗙 🗲 🗲	Groups: 📉 🗙 🗲	Files: X 🗲 🗲
program01	startup libraries output source CMSIS	system_stm32f10x.c core_cm3.c
Set as Current Target		Add Files
	ONAZ Canad	Hala



Potvrdíme **OK**

Project	4
program01 startup startup_stm startup_stm libraries misc.c stm32f10x_ stm32f10x_ stm32f10x_ stm32f10x_ stm32f10x_ stm32f10x_ cotput core	n32f10x_md_vl.s flash.c gpio.c rcc.c conf.h n32f10x.c
Build Output	Func 0, Temp

Nyní nám schází poslední soubor, který bude obsahovat zdrojový kód aplikace. V menu vybereme





Dostaneme



V.Váňa: CanSAT Book for Students – part 2

ARM STM32 programming



Tento soubor uložíme jako main.c



Dostaneme a vyplníme

Uspořádat 👻 🛛 N	ová složka				
 Plocha Stažené soubc Knihovny Dokumenty Hudba Obrázky Videa Počítač Místní disk (C: 	nry)	E	Název položky	Datum změny 14.9.2010 15:49 13.9.2010 14:36	Typ C compiler hea C compiler hea
👝 Místní disk (E:)	-	« [
Název souboru:	main.c				
Uložit jako tvo:	All Files (* *)				-

Klikneme na Uložit. Tak máme



C:\ARM_priklady\program01\program01.uvproj - µVision4	
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help	
□ 📽 🖬 🗿 🖇 毎週 タ P ↔ ↦ 🖗 祭 祭 津 津 非 振 🙆 💿 💽 🕵 🌮	🍳 🕘 🔿 🔗 🍓 🔲 🔍
😥 🕮 🚳 🥪 拱 🛛 🙀 🛛 program01 💽 🔊 👗 📥	
Project 4 🖬 🗋 main.c	▼ ×
<pre>>> program01 1 1 >>> startup stm32f10x_md_vl.s >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>></pre>	I
E Project Books {} Func 0, Temp	
Build Output	4 🖬
	~
<	H1 C1

Ještě znovu spustíme File Extensions, Books and Enviroment



Project Components Folders/Extensions	Books	
Project Targets: 🖄 🗙 🗲 🗲	Groups: 📉 🗙 🗲 🗲	Files: X 🗲 🗲
program01	startup libraries output	stm32f10x_conf.h
	source	
Set as Current Target		Add Files
	OK Cancel	Help

Klikneme na Add Files



Project Targe	Add Files to Group 'source'	
programer	Oblast hledání: 🔐 source 💌	← • • • •
	Název položky	Datum změny Tj
	· · · · · · · · · · · · · · · · · · ·	•
	Název main souboru:	Add
	Soubory typu: C Source file (*.c)	▼ Close

A přidáme main.c



roject Components	Folders/Extens	sions	Books			
Project Targets:	10 × 🗲	F	Groups:	🛅 🗙 🗲 🖣	Files:	× + +
program01			startup libraries output		stm32f10x_conf.h main.c	
			source CMSIS		-	
Set as Cu	rrent Taroet	1			Add F	Files
	ion raigat	9				

Potvrdíme **OK**. Máme nyní





Do zatím prázdného souboru main.c umístíme a uložíme zdrojový kód

```
#include "stm32f10x.h"
//#define HSE
                  //Pouziti externiho oscilatoru
#define HSI //Pouziti interniho oscilatoru
/* funkcni prototypy */
void RCC Configuration(void);
void GPIO Configuration(void);
void Delay(vu32 nCount);
/* vstupni funkce programu */
int main(void) {
     /* konfigurace zdroju hodinoveho signalu */
     RCC Configuration();
      /* konfigurace I/O portu */
     GPIO_Configuration();
      /* blikaci smycka */
      while (1) {
            /* prepneme stav pinu PC9 */
            GPIO WriteBit(GPIOC, GPIO Pin 9, (BitAction)(1 -
GPIO ReadOutputDataBit(GPIOC, GPIO Pin 9));
            /* cekame */
            Delay(0x5FFFF);
            Delay(0x5FFFF);
            Delay(0x5FFFF);
            Delay(0x5FFFF);
```

V.Váňa: CanSAT Book for Students – part 2

ARM STM32 programming



```
Delay(0x5FFFF);
            Delay(0x5FFFF);
      }
}
 /* nastaveni zdroju hodinoveho signalu (HSE) */
void RCC Configuration(void) {
#ifdef HSE
     ErrorStatus HSEStartUpStatus;
      /* reset nastaveni */
     RCC DeInit();
    /* aktivace signalu HSE (External High Speed oscillator) */
     RCC HSEConfig(RCC HSE ON);
      /* kontrola stability oscilatoru */
      HSEStartUpStatus = RCC WaitForHSEStartUp();
      if (HSEStartUpStatus == SUCCESS) {
            /*Nastaveni delicek HS*/
            RCC HCLKConfig(RCC SYSCLK Div1); //HCLK = 24 MHz, AHB
            RCC_PCLK2Config(RCC_HCLK_Div1); //APB1 = 24 MHz
            RCC PCLK1Config(RCC HCLK Div1); //APB2 = 24 MHz
            /* nastaveni PLL */
            RCC_PLLConfig(RCC_PLLSource PREDIV1, RCC PLLMul 4); //PLLCLK =
24 MHz
      }
#endif
#ifdef HSI
     /* reset nastaveni */
     RCC DeInit();
      /* aktivace signalu HSI (Internal High Speed oscillator) */
     RCC HSICmd(ENABLE);
     while (RCC GetFlagStatus(RCC FLAG HSIRDY) == RESET);
      /* nastaveni delicek HS */
     RCC HCLKConfig(RCC SYSCLK Div1); //HCLK = 24 MHz, AHB
      RCC PCLK1Config(RCC HCLK Div1); //APB1 = 24 MHz
      RCC PCLK2Config(RCC HCLK Div1); //APB2 = 24 MHz
      /* nastaveni PLL */
     RCC PLLConfig(RCC PLLSource HSI Div2, RCC PLLMul 8); //PLLCLK = 24
MHz
#endif
      /* nastaveni FLASH */
      FLASH SetLatency (FLASH Latency 2);
      FLASH PrefetchBufferCmd(FLASH PrefetchBuffer Enable);
     RCC PLLCmd (ENABLE);
      while (RCC GetFlagStatus(RCC FLAG PLLRDY) == RESET);
      /*Zhroj hodin pro AHB*/
      RCC SYSCLKConfig(RCC SYSCLKSource PLLCLK);
      while (RCC GetSYSCLKSource() != 0x08);
      /* aktivace zdroje HS pro periferie */
```

```
V.Váňa: CanSAT Book for Students – part 2
```



```
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC , ENABLE);
}
/* konfigurace I/O portu */
void GPIO Configuration(void) {
     /* struktura s informacemi o konfiguraci pinu */
     GPIO InitTypeDef GPIO InitStructure;
      /* konfigurujeme pin PC7 jako output push-pull */
     GPIO InitStructure.GPIO Pin = GPIO Pin 9;
      GPIO_InitStructure.GPIO_Speed = GPIO_Speed_10MHz;
      GPIO InitStructure.GPIO Mode = GPIO Mode Out PP;
      GPIO Init(GPIOC, &GPIO InitStructure);
}
/* cekani */
void Delay(vu32 nCount) {
     while (nCount--);
}
```

Takže máme



C:\ARM_priklady\program01\program01.ur	/proj - μVision4	
File Edit View Project Flash Debug	Peripherals Tools SVCS Window Help	
🗋 🗁 属 😹 🕹 🖧 🧐 😁		🗖 🗟 🌾 🔍 🖕 🔹 🔗 🍓
😂 🍱 🖾 🧼 🚉 🛛 🙀 🛛 program01	🖸 🎊 🛔 🗟	
Project 🛛 🕈 💌	main.c*	▼ ×
	<pre>01 #include "stm32f10x.h" 02 //#define HSE //Pouziti externiho 03 #define HSI //Pouziti interniho osc 04 /* funkcni prototypy */ 05 void GPIo_Configuration(void); 06 void GPIO_Configuration(void); 07 void Delay(vu32 nCount); 08 09 /* vstupni funkce programu */ 10 int main(void) { 11 /* konfigurace zdroju hodinoveho s 12 RCC_Configuration(); 13 /* konfigurace I/O portu */ 14 GPIO_Configuration(); 15 /* blikaci smycka */ 16 while (1) { 17 /* prepneme stav pinu PC9 */ 18 GPIO_WriteBit(GPIOC, GPIO_Pin_ 19 /* cekame */ 20 Delay(0x5FFFF); 21 Delay(0x5FFFF); 22 Delay(0x5FFFF); 23 Delay(0x5FFFF); 24 Delay(0x5FFFF); 25 Delay(0x5FFFF); 26 , *</pre>	oscilatoru cilatoru signalu */ _9, (BitAction)(1 - GPIO_ReadOutputDataBit(GPI
Rulid Output		
		Simulation L:1 C:1

Ještě nastavíme "Target Options" projektu, klikneme na ikonku

File	Edit	View	Project	Flash	Debug	Peripherals	Tools	SVCS	Window	Help
	3	0	3. ED	8	90	~ ~ P	1			//= // _R
٢	*	0		prog	gram01		<u>s</u>	6 6		
Project					4 🖂	🖉 📩 main	A Tar	get Optio	ons	
B	progra	m01				01 #inc	Co	nfigure t	arget optic	ons
-	esta	rtup startı	up stm32	f10x m	d vl.s	02 //#d 03 #def	lefine Tine H	e HS. ISI	E //P //Pouz	ouzit iti i

A dostaneme



Microe	lectronics	STM32F100R	B	10	Code C	Generation	1		
Onersti		INega				se Cross-I	Module Optimiza	tion	
operatil	ig system.	Twone		<u> </u>	ΓU	lse MicroL	ів Г	Big Endian	
						lse Link-Ti	me Code Gener	ation	
Read/(Only Memo	ory Areas			Read/	Write Men	nory Areas		
default	off-chip	Start	Size	Startup	de fault	off-chip	Start	Size	NoInit
Г	ROM1:			C	Г	RAM1:			Γ
Г	ROM2:			- c	Г	RAM2:			Ξ Ξ
Г	ROM3:	Í	1	- c	Г	RAM3:			
	on-chip					on-chip		,	
$\overline{\mathbf{v}}$	IROM1:	0x8000000	0x20000	œ	•	IRAM1:	0x20000000	0x2000	
	IROM2:			C		IRAM2:			

Vybereme záložku **Output**



evice Target Output Listing User C	+++ Asm Linker Debug Utilities	
Select Folder for Objects	Name of Executable: program01	
 Create Executable: .\program01 ✓ Debug Information ☐ Create HEX File ✓ Browse Information C Create Library: .\program01.LIB 	Create Batch File	

Zaškrtneme Create HEX File



cutable: program01
☐ Create Batch File

Dále klikneme na tlačítko Select Folder for Object a máme



Select Fo	Browse for Folder		
Create E Fo	lder: 🔒 program01		
Debu N	ázev položky	Datum změny 🔺 Bat	ch File
🔽 Creat	libraries	22.5.2011 20:58	
Brown	output	23.5.2011 13:00	
	source	23.5.2011 16:00	
Create L	program01.plg	23.5.2011 16:29	
	program01.uvgui.vladimir	23.5.2011 16:03 👻	
₹	III	- F	
Pa	th:	Ĭ	
		ок	

Vybereme složku output

Folder:	🎉 output	- 🔁 🖆 📼	
Název položi	sy ^	Datum změny	т
	Hiedani neodpovid	aji zadne polozky.	
•	Ш		•
Path:	C:\ARM_priklady\program0	1\output\	
		OK	2



A potvrdíme OK. Poté v Options for Target vybereme záložku Debug

Use Simulato Limit Speed 1	or Settings	Use: ULINK Cortex Debugger	ettings
I Load Applica Initialization File:	ation at Startup 🔽 Run to main()	✓ Load Application at Startup ✓ Run to mail Initialization File:	in()
Restore Debug	g Session Settings Ints I Toolbox Vindows & Performance Analyzer Display	Restore Debug Session Settings	Edit
CPU DLL:	Parameter:	Driver DLL: Parameter:	
SARMCM3.DLL		SARMCM3.DLL	
Dialog DLL:	Parameter:	Dialog DLL: Parameter:	
DCM.DLL	pCM3	TCM.DLL pCM3	

V pravé části nahoře rozvineme **listbox**



evice Target	Output Listing User C/C++ Asm		
 Use Simulato Limit Speed to 	r Settings o Real-Time		K Cortex Debugger Sl∡ting K Cortex Debugger
Load Applica	tion at Startup 🔽 Run to main()	Load RDI I Altera Stella Initializatio Cortes	nterface Driver a Blaster Cortex Debugger inis ICDI im Systems JTAGjet x-M/R J-LINK/J-Trace IR Debugger Edit.
Restore Debug	session Settings nts I I Toolbox Iindows & Performance Analyzer Display	Restore ULIN NULin Watch V Memory	K Pro Cortex Debugger nk Debugger ints
CPU DLL:	Parameter:	Driver DLL:	Parameter:
SARMCM3.DLL		SARMCM3.DL	L
Dialog DLL:	Parameter:	Dialog DLL:	Parameter:
IDCM.DLL	-pullio	TCM.DLL	-pcivis

A z této nabídky vybereme ST-Link Debuger



Image: Settings C Use: ST-Link Debugger ✓ Set Image: Limit Speed to Real-Time ✓ Settings ✓ Image: Limit Speed to Real-Time ✓ Note that the set of t	Settings	C Debugger ▼ Setting			
Imit Speed to Real-Time			Use: ST-Lin	tor Settings	 Use Simulat
Image: Construction at Startup			B	to Real-Time	Limit Speed
Initialization File: Initialization File:	ain()	ion at Startup 🛛 🕅 Run to main()	V Load Applicat	ation at Startup 🔽 Run to main()	Coad Applic
Restore Debug Session Settings Edit I E Image: Breakpoints Image: Collabor			Initialization File:	c	Initialization File
Restore Debug Session Settings Restore Debug Session Settings Image: Breakpoints Image: Toolbox Image: Watch Windows & Performance Analyzer Image: Watch Windows Image: Memory Display Image: Watch Windows	Edit	Edit	ļ	Edit	
Image: Breakpoints Image: Toolbox Image: Breakpoints Image: Toolbox Image: Watch Windows & Performance Analyzer Image: Watch Windows Image: Watch Windows Image: Memory Display Image: Watch Windows Image: Watch Windows		Session Settings	Restore Debug	ug Session Settings	Restore Debu
Watch Windows & Performance Analyzer Watch Windows Memory Display Memory Display		ts 🔽 Toolbox	Breakpoin	oints 🔽 Toolbox	✓ Breakpo
I ✓ Memory Display I ✓ Memory Display		ndows	₩ Watch Wi	Windows & Performance Analyzer	Watch V
		isplay	Memory D	Display	Memory
CPU DLL: Parameter: Driver DLL: Parameter:		Parameter:	Driver DLL:	Parameter:	CPU DLL:
SARMCM3.DLL SARMCM3.DLL			SARMCM3.DLL	I	SARMCM3.DL
Dialog DLL: Parameter: Dialog DLL: Parameter:			Dialog DLL	Parameter:	Dialog DLL:
		Parameter:	Didiog DEL.		

A dále klikneme na radiobutton use nalevo od právě použité nabídky



C Han Considerate	California de	I G they let up	
Limit Speed t	to Real-Time	J 🖲 Use: ST-Lin	ik Debugger <u>Settings</u>
Load Application File:	ation at Startup 🔽 Run to main()	Load Applica	ition at Startup TRun to main()
	Edit	I	Edit
-Restore Debug	g Session Settings	Restore Debug	g Session Settings
🔽 Breakpoi	nts 🔽 Toolbox	Breakpoir	nts 🔽 Toolbox
🔽 Watch W	/indows & Performance Analyzer	I Watch W	lindows
Memory I	Display	Memory D	Display
CPU DLL:	Parameter:	Driver DLL:	Parameter:
SARMCM3.DLL		SARMCM3.DLL	
Dialog DLL	Parameter:	Dialog DLL:	Parameter:
		-	Louis

Ještě klikneme na tlačítko Settings napravo od této nabídky. Dostaneme



C 11 C 1		0.000		
Use Simula	tor I to Real-Time	Settings	(• Use: ST-Link	k Debugger <u>Settings</u>
Load Applic	cation at Startup	Run to main()	Load Applicat	ion at Startup TRun to main()
		STLink Setup	×	Edit
-Restore Deb	ug Session Settings —	Protocol		ssion Settings
Breakpo	oints 🔽 Toolbi	🔹 🕡 JTAG		Toolbox
Watch	Windows & Performance			ows
Memory	Display	C SWD		ау
CPU DLL:	Parameter:	ОК	Cancel	rameter:
SARMCM3.DL	L			
				-
Dialog DLL:	Parameter:		Dialog DLL:	Parameter:
DCM DUI	DCM3		TCM DU	-pCM3

Zvolíme radiobutton **SWD** a potvrdíme **OK**

Protocol-	
C JTAG	2
	Cancel

Máme tak



evice Target	Output Listing User C/C++ Asm	Linker Debug Utilities	
C Use Simulato	r Settings	🕫 Use: ST-Link Debugger 🗨 Sett	tings
Load Applica	ition at Startup 🔽 Run to main()	Load Application at Startup TRun to main Initialization File:	0
- Restore Debug	a Session Settings Ints I Toolbox Andows & Performance Analyzer Display	Restore Debug Session Settings	
CPU DLL:	Parameter:	Driver DLL: Parameter:	
SARMCM3.DLL		SARMCM3.DLL	
Dialog DLL:	Parameter:	Dialog DLL: Parameter:	

Ještě vybereme záložku Utilities



ptions for Target 'program01'	x
Device Target Output Listing User C/C++ Asm Linker Debug Utilities	
Configure Rash Menu Command	
Use Target Driver for Flash Programming	
ULINK Cortex Debugger 💽 Settings 🔽 Update Target before Debugging	
Init File:Edit	
C Use External Tool for Flash Programming	
Command:	
Arguments:	
I Run Independent	
OK Cancel Defaults Help	

A nastavíme Target Driver for Flash Programming



Device Target Configure Flash	'program01' Output Listing User C/C++ Asm Linker Debug Utilities Menu Command	
(• Use Target Init File: C Use Extern Command:	ULINK Cortex Debugger Settings Vupdate Target before Debugging ULINK Cortex Debugger Stellaris ICDI Signum Systems JTAGjet Cortex-M/R J-LINK/J-Trace ST-Link Debugger ULINK Pro Cortex Debugger ULINK Pro Cortex Debugger	
	Run Independent	
2	OK Cancel Defaults	Help

Na ST-Link Debuger



Options for Target 'p	program01'	
Device Target O	utput Listing User C/C++ Asm Linker Debug Utilities	
Configure Flash N	Menu Command	
Use Target [Driver for Flash Programming	
S	T-Link Debugger Settings Vpdate Target before Debugging	
Init File:	Edit	
Command:	Tool for Flash Programming	
Arguments:		
Г	Run Independent	
	Cancel Defaults Help	

Dále klikneme na záložku C/C+



Preprocessor Sym	bols		
Define:			
Undefine:			
Language / Code	Generation		
		Strict ANSI C	Warnings:
Optimization: Lev	el 0 (-00) 🔻	Enum Container always int	<unspecified> -</unspecified>
Optimize for Time Solit Load and Store Multiple		 Plain Char is Signed Read-Only Position Independent 	
			I Thumb Mode
One ELF Sect	ion per Function	Read-Write Position Independent	
Include			
Paths			
Misc			
Compiler			
control VARM	IVINC/ST/STM32F10	aL g 000 -apcs≕nterwork 1 °C:\Keil\ARM\(<"-o ".\output*.o" -omf_browse ".\output*	.crf" -depend ".\output\".d"

Do texboxu **Define** vložíme řetězec STM32F10X_MD_VL, USE_STDPERIPH_DRIVER

A do texboxu Include Path řetězec

.\Libraries\CMSIS\CM3\CoreSupport;.\Libraries\CMSIS\CM3\DeviceSupport\ST\STM32F10x;.\Librari es\STM32F10x_StdPeriph_Driver\inc;.\source Potvrdíme **OK**.

Nyní již provedeme překlad



File Edit View P	oject Flash Debug Peripherals Tools SVCS Window Help	
D C C C C C C C C C C C C C C C C C C C	New µVision Project New Multi-Project Workspace Open Project Close Project	
startu	Export Manage	
initiands initiands	Select Device for Target 'program01' Remove Item	
stm3	Clean target	Alt+F7
	Build target	F7
📄 stm3	Rebuild all target files	
	Batch Build	
syste di sorte	Translate C:\ARM_priklady\program01\source\main.c	Ctrl+F7

Po úspěšném překladu máme




Nyní naprogramuje vytvořený binární kód do procesoru. Přes USB připojíme **STM32VL DISCOVERY** kit. Zmáčkneme **Ctrl + F5**, abychom spustili debugging. Objeví se





Potvrdíme **OK** a po chvíli dostaneme





A nakonec pomocí **F5** naprogramuje STM32. Na startkitu bliká zelená LEDka. Vývojové prostředí můžeme zavřít.

Při vytváření zdrojového kódu v **main.c** jsme převzali kód z DVD přílohy k tomuto výukovému materiálu. Zbývá tedy vysvětlit si jeho jednotlivé části a také s kódem experimentovat. Předtím ale učiníme několik poznámek k práci se startkitem i vývojovým prostředím.

Pro nahrání programu do paměti flash STM32 bych očekával použití volby



V.Váňa: CanSAT Book for Students – part 2

ARM STM32 programming



Bohužel toto nefunguje, což ostatně konstatuje i materiál z katedry měření ČVUT a proto je programování STM32 provádět oklikou – nejdřív pomocí **Ctrl F5** spustit debugging a potom pomocí **F5** již přenést program do flash STM32.

Toto ovšem funguje bez problémů u uVision4 nainstalovaném pod 32bitovými Windows7. Při použití 64 bitových se sice podaří projekt přeložit a získat soubory **axf** a **hex**, ale při pokusu o debugging dojde ke zkolabování uVision4.

K problémům může dojít i na 32 bitových windows, kdy se nepodaří navázat komunikaci s STM32VL Discovery a dostáváme chybová hlášení, např.



Chyba může být způsobená tím, že máme nižší verzi souboru **STLinkUSBDriver.dll** než 4.0.1.2. Zde je na místě si říci, co je to **ST Link**. Na startkitu jsou totiž dva MCU – STM32F100RBT6, který programujeme a s kterým pracujeme a dále menší čip STM32F103C8T6, který je hlavní částí programátoru **ST Link** realizujícího protokol **SWD**. Tímto programátorem můžeme náš program přenést do STM32F100RBT6 na startkitu STM32VL Discovery, ale (po překonfigurování propojek) pomocí kablíku i chip umístěný na samostatné destičce, což bude náš případ při použití STM32F100RBT6 jako řídícího počítače letového kusu CANSATu – je zbytečné v CANSATU mít umístěn celý STM32VL Discovery kit – je zbytečně rozměrný, těžký a obsahuje i programátor, který v CANSATu je zbytečný. Proto startkit budeme používat jen pro vývoj software.

Programový kód můžeme do startkitu přenášet nejen z prostředí **uVision 4**, ale lze použít **STM32 ST-Link Utility**



STM32 ST-Link Utility	
File View Target Help	
Memory display	Device Information
Address: 0x08000000 Size: 0x6000 Data Width: 32 bit _	Device Device ID Flash size
Disconnected Device ID :	

Tento sw nainstalujeme z instalačního souboru **STM32 ST-Link Utility_V1.2.exe**. Pokud máme tento sw již nainstalovaný, připojíme pomocí usb startkit k počítači. Nejprve se přesvědčíme, že máme nastavený protokol **SWD**.



Memory disp	Connect		Device Information
Address:	Disconnect	Data Width: 32 bit 💌	Device Device ID
	Erase Chip Erase Bank1 Erase Bank2 Erase Sectors		Flash size
	Program Program & Verify		
	MCU Core		
	Settings		

V menu proto vybereme Target \rightarrow Settings... a dostaneme okno Settings. Radiobuton na SWD

File View Target Help	
Address: 0x08000000 Size: 0x1554 Data Width: 32 bit 💌	Device Information Device ID Flash size
Settings Protocol C JTAG SWD Cancel OK	
Disconnected Device ID :	

V.Váňa: CanSAT Book for Students – part 2

ARM STM32 programming



Poté vybereme v menu **Target → Connect**

Address: Disconnect Data Width: 32 bit Device Device ID Erase Chip Flash size Erase Bank1 Erase Bank2 Erase Sectors	Memory disp	Connect		Device Information
Program Program & Verify Option Bytes MCU Core Settings	Address:	Disconnect Erase Chip Erase Bank1 Erase Bank2 Erase Sectors Program Program & Verify Option Bytes MCU Core Settings	Data Width: 32 bit	Device ID Flash size

79

Dostaneme



Memory displ	ay	- a - a -					Device Information
Address:	0x08000000	Size:	0x6000	Data Width:	32 bit 👤	Device Device ID	Value Line device 0x420
Device Memor	y @ 0x080000	: 000				Flash size	128 Kbyte
Address	0	4	8	С	ASCII		
0x08000000	20000718	08002A41	08000691	08000693	A*'		
0x08000010	08000697	0800069B	0800069F	00000000	—ź	na seren en el composition de la composit	
0x08000020	00000000	00000000	00000000	080006A3	Ł	111	
0x08000030	080006A5	00000000	080006A7	080006A9	Ą§	©	
0x08000040	08002A5B	08002A5B	08002A5B	08002A5B	[*[*[*	[*.,	
0x08000050	08002A5B	08002A5B	08002A5B	08002A5B	[*[*[*	[*.,	
0x08000060	08002A5B	08002A5B	08002A5B	08002A5B	[*[*[*	[*	
0x08000070	08002A5B	08002A5B	08002A5B	08002A5B	[*[*[*	[*.,	
0x08000080	08002A5B	08002A5B	08002A5B	00000000	[*[*[*		
0x08000090	00000000	00000000	00000000	08002A5B	·····[*	
0x080000A0	08002A5B	08002A5B	08002A5B	08002A5B	[*[*[*	[*	
0.00000000	08002A5B	0800245B	0800245B	08002A5B	*]*]*]	[*	

Tím jsme si ověřili, že u našeho startkitu je **ST-Link** funkční. Doporučuji toto ověření provést jako první akci, kterou se startkitem budeme provádět.

STM32 ST-Link Utility můžeme použít i k naprogramování MCU . V menu vybereme File – Open File



Open fil	e					De	evice Information
Save file Close Fi	Save file as Close File		0x6000	Data Width:	32 bit 👤	Device Device ID Flash size	Value Line device 0x420 128 Kbyte
Exit		4	8	с	ASCII		
0x08000000	20000718	08002A41	08000691	08000693	A*'	"	
0x08000010	08000697	0800069B	0800069F	00000000	—ź		
0x08000020	00000000	00000000	00000000	080006A3	ł		
0x08000030	080006A5	00000000	080006A7	080006A9	Ą§	©	
0x08000040	08002A5B	08002A5B	08002A5B	08002A5B	[*[*[*]*		
0x08000050	08002A5B	08002A5B	08002A5B	08002A5B	[*[*[*.	·[*	
0x08000060	08002A5B	08002A5B	08002A5B	08002A5B	[*[*[*.	.[*	
0x08000070	08002A5B	08002A5B	08002A5B	08002A5B	[*[*[*.	.[*	
0x08000080	08002A5B	08002A5B	08002A5B	00000000	[*[*[*.		
0x08000090	00000000	00000000	00000000	08002A5B	[*	
0x080000A0	08002A5B	08002A5B	08002A5B	08002A5B	[*[*[*.	.[*	
0x080000B0	08002A5B	08002A5B	08002A5B	08002A5B	[*[*[*.	.[*	

dostaneme



File View	Target He	lp								
- Meniory dispr	dy 		0.000		00 ka 🛛	-		Device I	nformation	
Address:	0x080000	U Size:	UX3L39	Data Width:	32 DIC 🔻		Device	Value X x42	e Line devid :0	e,
Device Memo	y @ 0x0800	ULEVIIL						28	Kbyte	
Address	0	Oblast hledán	í: 🚺 output		-	← 🖻 (* 📰 🔻			
0x08000000	20000688	Název polož	ky ^			Datum zr	měny	Ту		
0x08000010	080001CD	program	01			24.5.2011	22:25	Bi		
0x08000020	00000000									
0x08000030	080001D5									
0x08000040	080001DB									
0x08000050	080001DB	•	1	11]		•		
0x08000060	080001DB	Název	program01				Otevří			
0x08000070	080001DB	souboru:								
0x08000080	080001DB	Soubory typu:	BIN Files (*.bir	1)		-	Stomo			
0x08000090	00000000		C Otevřít jen	pro čtení						
0x080000A0	080001DB	080001DB	080001DB	080001DB	00		J			
0x080000B0	080001DB	080001DB	080001DB	080001DB	ŰŰ	Űí	ï			
•		III								•

Vybereme **program01.bin** a klikneme na tlačítko **Otevřít**. Soubor program01.bin jsme ovšem předem získali pomocí utility fromelf.exe , kterou najdeme v C:\Keil\ARM\Bin40\ a spustíme s parametry: **fromelf --bin program01.axf --output program01.bin**



memory disp	ay					1	Device Information
Address:	0x08000000) Size:	0x3C39	Data Width:	8 bit 💽	Device Device ID	Value Line device 0x420
Binary file : pro	ogram01.bin					Flash size	128 Kbyte
Address	0	4	8	С	ASCII		
0x00000000	20000688	0 STM32 S	TLink Utility				
0x00000010	080001CD	08					
0x00000020	00000000	0	[program0	1.bin] opened	successfully.		
0x0000030	080001D5	00	NOTE:The	file is NOT dou	wnloaded to the d	evice.	
0x00000040	080001DB	08	Do you wa	ne to downloa	and now:		
0x00000050	080001DB	08		<u>.</u>			
0x00000060	080001DB	08			OK	Storno	
0x00000070	080001DB	02		-	~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~		
0x00000080	080001DB	080001DB	080001DB	00000000	ŨŪŨ		
0x00000090	00000000	00000000	00000000	080001DB	Ű	¥11.7	
0x000000x0	080001DB	080001DB	080001DB	080001DB	<mark>000</mark>	.Ű	
0.0000000	080001 DB	080001DB	080001DB	080001DB	ŰŰŰ	.Ű	
0X000000000		present strength and			ALL MARKED AND AND AND AND AND AND AND AND AND AN		

Potvrdíme , dostaneme



memory dispr	lay] [Device Information
Address:	0x08000000	Size:	0x3C39	Data Width:	B bit 💌	Device Device ID	Value Line device 0x420
Binary file : pro	ogram01.bin					Flash size	128 Kbyte
Address	0	4	8	С	ASCII		
0x00000000	20000688	080001Ç1	080001 C9	080001 CB	ÉĔ		
0x00000010	080001CD	080001 Dov	wnload [prog	am01.bin]		×	
0x00000020	00000000	000000			Progra		
0x0000030	080001D5	000000 S	tart address : 0	×08000000			
0x00000040	080001DB	080001			Canc		
	00000100	080001					
0x00000050	080001DB	000001					
0x00000050 0x00000060	080001DB	080001					
0x00000050 0x00000060 0x00000070	080001DB 080001DB 080001DB	080001 080001DB	080001DB	080001DB	<u>00</u>	.Ű	
0x00000050 0x00000060 0x00000070 0x00000080	080001DB 080001DB 080001DB 080001DB	080001 080001DB 080001DB	080001DB 080001DB	080001DB 00000000	000	Ű	
0x0000050 0x0000000 0x0000070 0x0000080 0x0000090	080001DB 080001DB 080001DB 080001DB 00000000	080001 080001DB 080001DB 00000000	080001DB 080001DB 00000000	080001DB 00000000 080001DB	000. 000.	Ū	
0x0000050 0x0000000 0x0000000 0x0000000 0x000000	080001DB 080001DB 080001DB 080001DB 00000000 080001DB	080001 080001DB 080001DB 080001DB 00000000 080001DB	080001DB 080001DB 00000000 080001DB	080001DB 00000000 080001DB 080001DB	000 000 0	0 0	

Po kliknutí na tlačítko Program již začne vlastní programování

	. Program
tart address : 0x08000000	-13- <u>-</u>
lash Programming	Cancel
ish Programming	Lancei

Po dokončení programování ještě ukončíme spojení tj v menu vybereme Target ->- Disconet



memory disp	Connec	:t					Device Information	
Address:	Discon	nect 🔓		Data Width	32 bit 💌	Device Device ID	Value Line device 0x420	
Device Mem	Erase C	hip				Flash size	128 Kbyte	
Address	Erase Bi	ank1		С	ASCII			
0x08000000	Erase B	ank2	10	9 080001CB	ÁÉ	Ë		
0x08000010	Erase Se	ectors	1D:	L 0000000	ÍĎŃ	ÍĎŃ		
0x08000020	Program	n	000	080001D3		ó		
0x08000030	Program	n & Verify	1D	7 080001D9	Ő×	.Ů		
0x08000040	Ontion	Option Bytes 1DB 080001DB 000						
0x08000050	option by	oyces	1D	B 080001DB	ŰŰŰ.	Ű		
0x08000060	MCU C	ore	1D	B 080001DB	ŪŪŪ.	ŨŨŨ		
0x08000070	Setting	s	1D	B 080001DB	ŪŪŪ.	Ű		
0x0800080	080001DB	080001DB	080001D	B 0000000	<mark>000</mark> .			
0x08000090	00000000	00000000	0000000	080001DB		Ű		
0x080000A0	080001DB	080001DB	080001D	B 080001DB	<u>0</u> 00.	Ű		
0x080000B0	080001DB	080001DB	080001D	B 080001DB	<u> </u>	Ű		
•		III	- 1.				۲	

Po resetování MCU již běží nový program.

V zbývající části této podkapitoly si konečně vysvětlíme jednotlivé části zdrojového kódu v main.c

Nakonec uděláme pokus – zrychlíme čekací smyčku

```
/* blikaci smycka */
while (1) {
    /* prepneme stav pinu PC9 */
    GPIO_WriteBit(GPIOC, GPIO_Pin_9, (BitAction)(1 -
GPIO_ReadOutputDataBit(GPIOC, GPIO_Pin_9)));
    /* cekame */
    Delay(0x000FF);
    //Delay(0x5FFFF);
    //Delay(0x5FFFF);
    //Delay(0x5FFFF);
    //Delay(0x5FFFF);
    //Delay(0x5FFFF);
    //Delay(0x5FFFF);
    //Delay(0x5FFFF);
    }
}
```

A osciloskopem budeme pozorovat výstupní průběh na pinu PC9





Při použití Delay (0x000FF); jsme docílili kmitočet obdélníků cca 12kHz. Ještě o něco změníme zpoždění Delay (0x0000F) a nyní se průběh změní na





Výstupní kmitočet je cca 70kHz

2.2.2 Program blikač LEDky podruhé

V složce program02 vytvoříme podadresáře



Spustíme Keil uVision4 a V menu vybereme Project \rightarrow New µVision Project ...



🖞 µVision4		а÷.,		Sec.		· an
Create New Project	čítač → Místní disk (C:) →	ARM	_priklady 🕨 program02 🕨	•	↓ Prohledat: progr	am02 🔎
Uspořádat 🔻 N	ová složka					:≕ • 🔞
Plocha		*	Název položky		Datum změny	Тур
📕 Stažené soubo	ory	-	JA CMSIS		25.5.2011 20:09	Složka souborů
Math a saw			退 libraries		25.5.2011 20:08	Složka souborů
Dokumontu			鷆 output		25.5.2011 20:08	Složka souborů
- Hudba			🍌 source		25,5,2011 20:08	Složka souborů
S Obrázky		III	鷆 startup		25.5.2011 20:08	Složka souborů
Videa						
🖳 Počítač						
🏭 Místní disk (C:	:)					
💼 Místní disk (E:)	-	1	ш)
Název souboru:	program02					•
Uložit jako typ:	Project Files (*.uvproj)					•
Skrýt složky					Uložit	Storno

Napíšeme program02, klikneme na Uložit atd. (viz postup u program01)



PU		
Vendor: STMicroelectronics		
Device: STM32F100RB		
Toolset: ARM		
Database	Desertation	
	ARM 32-bit Cortex-M3 Microcontroller 24MHz 128kB Bash 8kB SBAM	
STA2051	PLL, Embedded Internal RC 8MHz and 40kHz, Real-Time Clock,	
STM32F100C4	Power Saving Modes, JTAG and SWD, 7-channel DMA. Nested Interrupt Controller.	
STM32F100C6	12-bit 16-ch A/D Converter, 2 12-bit 2-ch D/A converters, 51 Fast I/O Port	
STM32F100C8	SysTick Timer, 2 Watchdog Timers,	
STM32F100R4	16-bit 6-ch advanced timer with PWM, 6 16-bit Timers with logit Canture, Output Compare and PWM	
STM32F100R6	to tobic times with input capture, output compare and t with	
STM32F100R8		
STM32F100RC		
STM32F100RD		
STM32F100RE		-
	J • [

Vybereme CPU STM32F100RB a stiskneme **OK**. Další nastavení stejné jako u **program01**.





Zdrojový kód v souboru main.c bude

```
/* blikac LED varianta 2 - hodne jednoducha
 *
 * ver 1.0
 * Vd for SPSE Jecna 2011 lessons
 */
#include "stm32f10x.h"
void Delay( IO uint32 t nTick);
void InitAll(void);
int main(void)
{
  InitAll();
  while (1)
  {
    /* prepneme stav pinu PB5 */
            GPIOB->BSRR = GPIO BSRR BS5;
            /* cekame */
            Delay(0x5FFFF);
            Delay(0x5FFFF);
            Delay(0x5FFFF);
            Delay(0x5FFFF);
            Delay(0x5FFFF);
V.Váňa: CanSAT Book for Students – part 2
                                        90
```

ARM STM32 programming



```
Delay(0x5FFFF);
            GPIOB->BSRR = GPIO BSRR BR5;
            Delay(0x5FFFF);
            Delay(0x5FFFF);
            Delay(0x5FFFF);
            Delay(0x5FFFF);
            Delay(0x5FFFF);
            Delay(0x5FFFF);
  }
}
void InitAll(void)
                   {
     RCC->APB2ENR |= RCC APB2ENR IOPBEN;
      GPIOB->CRL &= ~(GPIO CRL MODE5 | GPIO CRL CNF5);
      GPIOB->CRL |= GPIO CRL MODE5 0;
}
```

LEDku máme připojenou k **PB5** – v ukázce jsem nepoužil LEDky na startkitu, ale připojil jsem LEDku externí.





2.2.3 Blikač s LED potřetí, navíc s tlačítkem

Postup vytváření projektu a odpovídajících souborů bude stejný jako u předchozích programů, proto nám budou stačit následující informace – seznam souborů projektu a zdrojový kód souboru **main.c**. Stejně tomu bude i v dalších programech a tak vždy uvedeme jen obsah **main.c** a strukturu projektu.



A zdrojový kód hlavní funkce main

```
/* STM32 VL Discovery Library
 *
 * ver 1.0
 * Vd for SPSE Jecna 2011 lessons
 */
/* Includes -----*/
#include "stm32f10x.h"
#include "STM32vldiscovery.h"
void GPI0_Inicialization(void);
void Delay(__I0 uint32_t nTick);
int main(void)
{
 GPI0_Inicialization();
```

V.Váňa: CanSAT Book for Students – part 2



```
STM32vldiscovery LEDOn(LED4); // LED4 - modra dioda
  STM32vldiscovery_LEDOn(LED3); // LED3 - zelena dioda pridal jsem
  Delay(0xAFFFFF);
  while (1)
  {
    if (STM32vldiscovery PBGetState (BUTTON USER) != 0) //kdyz se stiskne
tlacitko, tak se provede nasledujici:
    {
        STM32vldiscovery_LEDOn(LED3); // LED3 - zelena dioda
        STM32vldiscovery LEDOff(LED4); // LED4 - modra dioda
        Delay(0xAFFFFF);
        STM32vldiscovery LEDOff(LED3); // LED3 - zelena
                                                          dioda
        STM32vldiscovery_LEDOn(LED4); // LED4 - modra dioda
    }
  }
}
void GPIO Inicialization(void)
{
  STM32vldiscovery LEDInit(LED3);
  STM32vldiscovery_LEDInit(LED4);
 STM32vldiscovery_PBInit(BUTTON_USER, BUTTON MODE GPIO);
 STM32vldiscovery_LEDOff(LED3);
 STM32vldiscovery_LEDOff(LED4);
}
void Delay( IO uint32 t nTick)
{
  for(; nTick != 0; nTick--);
}
```

2.2.4 Blikač s LED počtvrté, s tlačítkem





```
/* STM32 VL Discovery Library
 * ver 1.0
 * Vd for SPSE Jecna 2011 lessons
 */
/* Includes -----
#include "stm32f10x.h"
#include "STM32_Discovery.h" //mirne jina knihovna, nez v program03
void GPIO Inicialization(void);
void Delay( IO uint32 t nTick);
int main(void)
{
 GPIO Inicialization();
  STM32 Discovery LEDOn(LED4); // LED4 - modra dioda
  STM32 Discovery LEDOn(LED3); // LED3 - zelena dioda pridal jsem
  Delay(0xAFFFFF);
  while (1)
  {
    if (STM32 Discovery PBGetState (BUTTON USER) != 0) //kdyz se stiskne
    //tlacitko, tak se provede nasledujici:
    {
```



```
STM32_Discovery_LEDOn(LED3); // LED3 - zelena dioda
        STM32 Discovery LEDOff(LED4); // LED4 - modra dioda
        Delay(0xAFFFFF);
        STM32 Discovery LEDOff(LED3); // LED3 - zelena
                                                          dioda
        STM32_Discovery_LEDOn(LED4); // LED4 - modra dioda
    }
  }
}
void GPIO Inicialization(void)
{
  STM32_Discovery_LEDInit(LED3);
  STM32_Discovery_LEDInit(LED4);
 STM32_Discovery_PBInit(BUTTON_USER, BUTTON_MODE_GPIO);
 STM32_Discovery_LEDOff(LED3);
 STM32_Discovery_LEDOff(LED4);
}
void Delay( IO uint32 t nTick)
{
 for(; nTick != 0; nTick--);
}
```

2.2.5 SysTick



V main.c

V.Váňa: CanSAT Book for Students – part 2



/*

Inovace ve vzdělávání na území hl.m. Prahy 2011, proj.č.2046

```
* SysTick main.c
* Blikani LED pomocí SysTick
* podle MCU
 cviceni SPSE
*/
#include <stddef.h>
#include "stm32f10x.h"
#include "STM32 Discovery.h"
#include "main.h"
 IO uint32 t Count;
static IO uint32 t CasProDelay;
void GPIO Inicializace(void);
void Delay(__IO uint32_t nCount);
void UbyvaniCasu Delay(void);
int main(void)
{
  GPIO Inicializace();
  /* Nastaveni casovace SysTick na 1 msec preruseni */
  if (SysTick Config(SystemCoreClock / 1000))
  {
   /* Capture error */
   while (1);
  }
  while (1)
    {
      // kontrola stisku USER tlacitka
     if(0 != STM32 Discovery PBGetState(BUTTON USER))
        {
         // tj. když je stisknuto tlacitko
        Count = 350; // delsi cas
        }
        else
        {
         // tj. když není stisknuto tlacitko
        Count = 100; // kratsi cas
      }
      STM32_Discovery_LEDOn(LED3); // rozsvitime LED3 - zelena
     Delay(Count);
     STM32 Discovery LEDOff(LED3); // zhasnem LED3 - zelena
     Delay(Count);
    }
}
```

void GPIO_Inicializace(void)

V.Váňa: CanSAT Book for Students – part 2

ARM STM32 programming



```
{
 STM32 Discovery LEDInit(LED3);
 STM32 Discovery LEDInit(LED4);
 STM32 Discovery PBInit (BUTTON_USER, BUTTON_MODE_GPIO);
 STM32 Discovery_LEDOff(LED3);
  STM32 Discovery_LEDOff(LED4);
}
/**
  * Obrief Inserts a delay time.
  * @param nTime: specifies the delay time length, in milliseconds.
  * @retval : None
  *,
void Delay( IO uint32 t nTick)
{
 CasProDelay = nTick;
 while(CasProDelay != 0); // cas se snizuje pomoci preruseni
}
void UbyvaniCasu Delay(void)
{
 if (CasProDelay != 0x00)
 {
   CasProDelay--;
  }
}
```

V stm32f10x_it.c

```
#include "stm32f10x_it.h"
#include "main.h"
void SysTick_Handler(void)
{
     UbyvaniCasu_Delay();
```

2.3 LCD displej

}

Velkého rozšíření mezi konstruktéry jako zobrazovací alfanumerické zařízení dosáhl inteligentní LCD displej s řadičem **HD44780** nebo jeho klon , takže všechny mají stejnou instrukční sadu a stejné ovládaní. Tohoto rozšíření dosáhli díky výhodným vlastnostem. Je to především relativně snadné ovládání, možnost definice až osmi vlastních znaků (např. čeština) , různorodost ve výběru velikosti (1 x 16 až 4 x 40 znaků) při zachování jednotného ovládání a v neposlední řadě se jejich specifikace stává defacto průmyslovým standardem . To usnadňuje ladění na HW i SW úrovni a případnou výměnu displeje (např. za větší, podsvícený apod.). Velkou výhodou LCD je malý odběr zobrazovací

V.Váňa: CanSAT Book for Students – part 2 97



matice, malé rozměry a nízká hmotnost ve srovnání s klasickou obrazovkou, lepší geometrie a ostrost zobrazení, delší životnost, stálost obrazu. Nevýhodou, která je však již u některých LCD odstraněna (na úkor ceny), je teplotní závislost, kdy kapalné krystaly při záporných a vysokých kladných teplotách ztrácejí své fyzikální vlastnosti a displej přestává dočasně pracovat.

S displejem lze komunikovat buď osmi nebo čtyřbitově. Náš program předpokládá čtyřbitovou komunikaci a propojení displeje s některým z portů STM32 MCU podle obr (např. k PORTB):



#define	LCD_E	GPIO_Pin_14
#define	LCD_RS	GPIO_Pin_15
#define	LCD_D4	GPIO_Pin_10
#define	LCD_D5	GPIO_Pin_11
#define	LCD_D6	GPIO_Pin_12
#define	LCD_D7	GPIO_Pin_13
#define	LCD_PORT	GPIOB

Při této čtyřbitové komunikaci slouží DB4, DB5,DB6 a DB7 pro přenos instrukcí do řadiče LCD a pro zápis či čtení dat do/z paměti kódů znaků (DDRAM) nebo paměti uživatelských znaků (CGRAM). To,zda se zapisují data či instrukce je dáno signálem na RS (0 ... vstup instrukce, 1 Data)., signál RD určuje zápis/čtení do LCD a EN znamená **platná data**. Pomocí HD44780 můžeme na diplej zobrazovat pouze znakovou sadou uloženou v paměti ROM na řadiči, která je pro řadič vlastní viz.obrázek



	000	001	0	0 1 0	0 1 0	0 1 1	0 1 1	1 0 1	1 0 1	1 1 0	1 1 0	1 1 1	1 1 1
↓↓↓ ×××××00000	.o [1 0	o a	\mathbf{P}^{1}	0 N	1 P	0	1	0 7	1	0 α	1 p
xxxx0001		ļ	1	Ā	Q	а	۹	0	7	Ŧ	4	ä	q
xxxx0010		"	2	В	R	b	r	Г	1	7	×	β	θ
××××0011		#	3	С	S	С	S	L	ウ	Ŧ	Ð	ε	60
××××0100		\$	4	D	Τ	d	t	~	I	ŀ	Þ	μ	Ω
××××0101		Ζ	5	Ε	U	e	u	•	ł	,	l	Q	ü
xxxx0110		8,	6	F	Ų	f	V	₹	ħ	-	Ξ	ρ	Σ
××××0111		,	7	G	W	9	ω	7	ŧ	Z	Ē	q	π
xxxx1000		(8	Η	Х	h	Х	4	2	ネ	'n	ŗ	$\overline{\times}$
xxxx1001)	9	Ι	Υ	i	Э	Ċ	ን	J	Ib	-1	ч
xxxx1010		*	:	J	Ζ	j	z	т		ù	V	i	Ŧ
××××1011		+	;	К	Γ	k	{	Ħ	ħ	F		×	Я
××××1100		,	<	L	¥	1	Ι	Þ	Ð	7	7	¢	Ħ
××××1101		-	=	Μ]	Μ)	ב	Z	Υ	2	ŧ	÷
××××1110		•	>	Ν	^	n	÷	Ξ	t	т і	Ŷ	ñ	
××××1111		/	?	0	_	0	÷		У	2		ö	

Stejně jako v předchozích příkladech, i nyní uvedeme jen nezbytné informace





Soubor LCD.c

```
/* STM32 VL Discovery Library
 * LCD driver LCD.c
 * ver 1.0
   SPSE Jecna 2011 Vladimir Vana
 *
 */
/* Includes -----
---*/
#include "LCD.h"
/** @defgroup Private_TypesDefinitions */
TIM TimeBaseInitTypeDef TIM TimeBaseStructureMili =
{36000,TIM_CounterMode_Up,0,0,0};
void delay ms(uint16 t time);
void LCD Inicialization(void);
void SendCommandLCD(uint8 t cmd);
void SendCharLCD(uint8 t character);
void LCD Clear(void);
void LCD CursorHome(void);
void printLCD(uint8 t* message);
void SendBitsLCD(uint8 t bits);
void Send4bitLCD(uint8 t cmd);
void FirstRow(void);
void SecondRow(void);
/**
 * @brief Nastaveni LCD a pinu.
  * @retval : None
  */
void LCD_Inicialization(void)
{
  GPIO InitTypeDef GPIO InitStructure;
  RCC APB2PeriphClockCmd(LCD PORT CLK, ENABLE);
  GPIO InitStructure.GPIO Pin = LCD E | LCD RS | LCD D4 | LCD D5 | LCD D6 |
LCD D7;
  GPIO InitStructure.GPIO Mode = GPIO Mode Out PP;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO Init(LCD PORT, &GPIO_InitStructure);
  GPIO ResetBits (LCD PORT, LCD RS);
     //RS=0:Instruction
  GPIO ResetBits(LCD PORT, LCD E);
      //E=0:Disable
  //delay us(200000);
  delay ms(200);
  Send4bitLCD(0x30);
                      // dle datasheetu cekat vice nez 4.1 ms
  //delay us(8000);
 delay ms(8);
```

100

ARM STM32 programming



```
GPIO ResetBits(LCD PORT, LCD E);
                                                           //E=0:Disable
 Send4bitLCD(0x30);
  //delay us(1000);
 delay ms(1);
 GPIO ResetBits(LCD PORT, LCD E);
                                                            //E=0:Disable
 Send4bitLCD(0x30);
  //delay us(1000);
 delay ms(1);
 GPIO ResetBits(LCD PORT, LCD E);
                                                            //E=0:Disable
 // delay us(1000);
 delay ms(1);
  // ted by mel byt display v 8-bitovem modu a definitivne ho prepneme do
4-bitoveho modu!
 Send4bitLCD(0x20);
  //delay us(1000);
 delay ms(1);
 GPIO ResetBits (LCD PORT, LCD E);
                                                           //E=0:Disable
  //delay us(1000);
 delay ms(1);
  // nyni jsme ve 4-bitovem rezimu
  // Prikaz 0 0 1 DL N F - -
  // N = pocet radek 1=2 radky, 0=1 radka
 // DL = bitu komunikace 1=8bitu, 0=4bity
  // F = Font 1=5x10bodu, 0=5x8 bodu
 SendCommandLCD(0x28);
  //delay_us(1000);
 delay ms(1);
  // zvedej sam adresu, posunuj kurzor
 SendCommandLCD(0x0E); //nyni OK
  //delay us(1000);
 delay ms(1);
 //PosliPrikazLCD(0b00000110);
 LCD Clear(); // vymazani LCD a navrat kurzoru na prvni pozici prvniho
radku
}
/**
 * @brief Smaze LCD a nastavi LCD kurzor na home pozici
 * @retval : None
 */
void LCD Clear(void)
{
   SendCommandLCD(0x01);
}
/**
 * @brief Nastavi LCD kurzor na home pozici
  * @retval : None
  */
void LCD CursorHome(void)
{
   SendCommandLCD(0x02);
}
```

101

ARM STM32 programming



```
/**
  * @brief Posle prikaz do LCD
 * @param cmd prikaz do LCD
 * @retval : None
 */
void SendCommandLCD(uint8 t cmd) //v promenne cmd mame zaslany prikaz
{
     GPIO ResetBits (LCD PORT, LCD RS);
      //RS=0:Instruction
     SendBitsLCD(cmd); // posle bity jako instrukci
}
/**
 * @brief Posle znak do LCD
 * @param character znak
 * @retval : None
 */
void SendCharLCD(uint8 t character) //v promenne cmd je znak poslany na LCD
{
     GPIO SetBits(LCD PORT, LCD RS);
                                              //RS=1:Data
     SendBitsLCD(character); // posle bity jako data
}
/**
 * @brief Posle bity do LCD
 * @param bits znak nebo prikaz
  * @retval : None
  */
void SendBitsLCD(uint8 t bits)
{
     Send4bitLCD(bits); // zaslani bitu 4-7
     bits = bits<<4; // bitovy posun na zaslani bitu 0 az 3</pre>
     Send4bitLCD(bits); // zaslani bitu 0-3
}
/**
 * @brief Posle retezec znaku do LCD
 * @param message pointer na retezec
  * @retval : None
  */
void printLCD(uint8 t* message)
{
     uint8 t i=0;
     if (!message) return;
     while (message[i]!='\0')
      {
           if(i%20==0){
                 if(i%40==0){ // pokud je na konci druheho radku prejde na
zacatek prvniho
                       FirstRow();
                       delay ms(1);
                 }else{
                                         // pokud je na konci prvniho radku
prejde na zacatek druheho
                       SecondRow();
V.Váňa: CanSAT Book for Students – part 2 102
                                                        ARM STM32 programming
```



```
}
            }
            SendCharLCD(message[i]);
            i++;
      }
}
/**
 * @brief Prejde na prvni radek
 * @retval : None
 */
void FirstRow(void)
{
     SendCommandLCD(0x80); // prejde na zacatek prvniho radku - na adresu
prvniho znaku
    delay ms(1);
}
/**
 * @brief Prejde na druhy radek
 * @retval : None
  */
void SecondRow(void)
{
     SendCommandLCD(0xC0); // prejde na zacatek druheho radku - na
adresu prvniho znaku
   delay ms(1);
}
/**
 * @brief Posle po 4 bitech prikaz nebo znak do LCD
 * @param command
  * @retval : None
  */
void Send4bitLCD(uint8 t cmd)
{
    GPIO ResetBits(LCD PORT, LCD E);
     delay ms(1);
     if (cmd & 0x10) // vymaskovani 4 bitu
      {
           GPIO SetBits (LCD PORT, LCD D4);
      } else {
           GPIO ResetBits (LCD PORT, LCD D4);
      }
     if (cmd & 0x20) // vymaskovani 5 bitu
      {
           GPIO SetBits(LCD PORT, LCD D5);
      } else {
           GPIO ResetBits(LCD PORT, LCD D5);
      }
      if (cmd & 0x40) // vymaskovani 6 bitu
      {
           GPIO SetBits(LCD PORT, LCD D6);
      } else {
            GPIO ResetBits(LCD PORT, LCD D6);
      }
```

V.Váňa: CanSAT Book for Students – part 2



```
if (cmd & 0x80) // vymaskovani 7 bitu
      {
           GPIO SetBits(LCD PORT, LCD D7);
      } else {
           GPIO_ResetBits(LCD_PORT, LCD D7);
      }
     GPIO SetBits(LCD PORT, LCD E);
     //E=1:Enable
     delay ms(1);
     GPIO ResetBits(LCD PORT, LCD E);
                                                               //E=0:Disable
}
/**
 * @brief Cekat n milisekund
 * @param time kolik ms se ma cekat
 * @retval None
 */
void delay ms(uint16 t time)
{
 RCC APB1PeriphClockCmd(RCC APB1Periph TIM4, ENABLE);
 TIM TimeBaseStructureMili.TIM Period = ((time+1) * 2)-1;
 TIM TimeBaseInit(TIM4, &TIM TimeBaseStructureMili);
 TIM SelectOnePulseMode(TIM4, TIM OPMode Single);
 TIM SetCounter(TIM4,2);
 TIM Cmd(TIM4, ENABLE);
 while (TIM GetCounter(TIM4)){};
 TIM Cmd(TIM4, DISABLE);
}
```

Soubor LCD.h

V.Váňa: CanSAT Book for Students – part 2

```
/* STM32 VL Discovery Library
* LCD driver LCD.h
* ver 1.0
 * Vd for SPSE Jecna 2011 lessons
*/
#ifndef LCD H
#define LCD H
                             -----
/* Includes -----
---*/
#include "stm32f10x.h"
/** @defgroup Exported Types
                            */
/** @defgroup Exported Functions */
void delay ms(uint16 t time);
void LCD Inicialization(void);
void SendCommandLCD(uint8 t cmd);
                                                   ARM STM32 programming
```

104



```
void SendCharLCD(uint8_t cmd);
void LCD_Clear(void);
void LCD_CursorHome(void);
void printLCD(uint8_t* message);
void Send4bitLCD(uint8_t cmd);
void FirstRow(void);
void SecondRow(void);
/** @defgroup Defines */
// piny pro ovladani displeje - zapojeni stejne jako na Katedre mereni CVUT
#define LCD_E GPIO_Pin_14
#define LCD_RS GPIO_Pin_15
#define LCD_D4 GPIO_Pin_15
#define LCD_D5 GPIO_Pin_11
#define LCD_D5 GPIO_Pin_12
#define LCD_D6 GPIO_Pin_13
#define LCD_PORT GPIOB
#define LCD_PORT_CLK RCC_APB2Periph_GPIOB
```

```
Soubor main.c
```

#endif /* LCD_H_ */

```
/* STM32 VL Discovery Library
 * LCD driver LCD.c
 * ver 1.0 for SPSE Jecna lectures
 * Created Vd on: Apr 20, 2011
*/
/* Includes -----
---*/
#include <stddef.h>
#include "stm32f10x.h"
#include "STM32vldiscovery.h"
#include "LCD.h"
uint8 t LCD Message1[] = "Test displeje.";
uint8 t LCD Message2[] = "Stisk tlacitka! ";
/* Private function prototypes -----
---*/
void GPIO Inicialization(void);
void Delay( IO uint32 t nTick);
int main(void)
{
 LCD Inicialization();
 GPIO Inicialization();
STM32vldiscovery LEDOn(LED4); // LED4 - modra dioda
V.Váňa: CanSAT Book for Students – part 2 105
                                                      ARM STM32 programming
```



```
STM32vldiscovery LEDOn(LED3); // LED3 - zelena dioda pridal jsem
  printLCD(LCD Message1);
  while (1)
  {
    if (STM32vldiscovery PBGetState (BUTTON USER) != 0)//kdyz se stiskne
tlacitko, tak se provede nasledujici:
    {
        STM32vldiscovery LEDOn(LED3); // LED3 - zelena dioda
            STM32vldiscovery LEDOff(LED4); // LED4 - modra dioda
       LCD Clear(); // vymazani LCD a navrat kurzoru na prvni pozici
prvniho radku
       printLCD(LCD Message2);
                                   // zprava je delsi nez oba radky
displeje - demonstruje prechod mezi radky v metode printLCD v knihovne
LCD.c
        Delay(0xAFFFFF);
        STM32vldiscovery_LEDOff(LED3); // LED3 - zelena
                                                          dioda
           STM32vldiscovery_LEDOn(LED4); // LED4 - modra dioda
       LCD Clear(); // vymazani LCD a navrat kurzoru na prvni pozici
prvniho radku
       printLCD(LCD Message1);
    }
  }
}
void GPIO Inicialization(void)
{
  STM32vldiscovery LEDInit(LED3);
  STM32vldiscovery_LEDInit(LED4);
  STM32vldiscovery PBInit (BUTTON USER, BUTTON MODE GPIO);
 STM32vldiscovery_LEDOff(LED3);
  STM32vldiscovery LEDOff(LED4);
}
void Delay( IO uint32 t nTick)
{
  for(; nTick != 0; nTick--);
}
```

2.4 UART TX – vysílání znaků (program06)







Soubor main.c

```
/* STM32 VL Discovery Library
 * vysilan1 pres USART
```

V.Váňa: CanSAT Book for Students – part 2

ARM STM32 programming



```
* ver 1.0 Program06
 * Vf for SPSE Jecna lessons - vysilani 4800Bd znaku - napr. do GPS
 */
#include <stddef.h>
#include <stdio.h>
#include "stm32f10x.h"
#include "STM32vldiscovery.h"
#define BUFF SIZE 40
uint8 t RX prijato = 0;
                         // priznak ukonceni prijmu z PC
// buffery na prijimane a odesilane znaky
uint8 t R Buff[BUFF SIZE], T Buff[BUFF SIZE];
void USART Inicializace(void);
void GPIO Inicialization(void);
int fputc(int ch, FILE * f);
void Delay(__IO uint32_t nTick);
uint8 t Strcmp (const uint8 t * s1, const uint8 t * s2); // porovnani str
void Strcpy(uint8 t *d1, const uint8 t *s1); // kopirovani str
int main(void)
{
  GPIO Inicialization();
   // aktivujeme USART
  USART Inicializace();
  STM32vldiscovery LEDOn(LED4); // LED4 - modra dioda
  STM32vldiscovery LEDOn(LED3); // LED3 - zelena dioda pridal jsem
  while (1)
  {
    if (STM32vldiscovery PBGetState (BUTTON USER) != 0) //kdyz se stiskne
tlacitko, tak se provede nasledujici:
    {
        STM32vldiscovery LEDOn(LED3); // LED3 - zelena dioda
           STM32vldiscovery LEDOff(LED4); // LED4 - modra dioda
           Strcpy (T Buff, "> Tlacitko stisknuto - ahoj\n\r"); //
nastavime co se odesle po USART
           USART ITConfig(USART1, USART IT TXE, ENABLE); // povolime
preruseni
         Delay(0xAFFFFF);
        STM32vldiscovery LEDOff(LED3); // LED3 - zelena dioda
           STM32vldiscovery_LEDOn(LED4); // LED4 - modra dioda
           Strcpy (T Buff, "> stiskni tlacitko - nazdar\n\r"); //
nastavime co se odesle po USART
           USART ITConfig(USART1, USART IT TXE, ENABLE); // povolime
preruseni
V.Váňa: CanSAT Book for Students – part 2
                                     108
                                                        ARM STM32 programming
```


```
}
  }
}
void GPIO Inicialization(void)
  STM32vldiscovery LEDInit(LED3);
  STM32vldiscovery LEDInit(LED4);
  STM32vldiscovery_PBInit(BUTTON_USER, BUTTON MODE GPIO);
  STM32vldiscovery LEDOff(LED3);
  STM32vldiscovery LEDOff(LED4);
}
void USART Inicializace(void)
{
      GPIO InitTypeDef GPIO InitStructure;
      USART InitTypeDef USART InitStructure;
      NVIC InitTypeDef NVIC InitStructure;
      /* pustime hodiny do periferie (protoze jde o alternativni funkci,
musi
      * jit hodiny i do AFIO periferie! */
     RCC APB2PeriphClockCmd(RCC APB2Periph GPIOA | RCC APB2Periph USART1 |
RCC APB2Periph AFIO, ENABLE);
      // PA9 je Tx
      GPIO InitStructure.GPIO Pin = GPIO Pin 9;
      GPIO InitStructure.GPIO Mode = GPIO Mode AF PP; // alternate
function!
      GPIO InitStructure.GPIO Speed = GPIO Speed 50MHz;
      GPIO Init(GPIOA, &GPIO InitStructure);
      // PA10 je Rx
      GPIO InitStructure.GPIO Pin = GPIO Pin 10;
      GPIO InitStructure.GPIO Mode = GPIO Mode IN FLOATING;
     GPIO Init(GPIOA, &GPIO InitStructure);
     USART InitStructure.USART BaudRate = 4800;
      /* Pokud mate dobry prevodnik USB-ttl, tak muzete pochopitelne
      * zvysit rychlost. Hodnota 4800 je pouzivana GPS */
      USART InitStructure.USART WordLength = USART WordLength 8b;
      USART InitStructure.USART StopBits = USART StopBits 1;
      USART InitStructure.USART Parity = USART Parity No;
     USART InitStructure.USART HardwareFlowControl =
USART HardwareFlowControl None;
      USART InitStructure.USART Mode = USART Mode Rx | USART Mode Tx;
      USART_Init(USART1, &USART_InitStructure);
      USART ITConfig(USART1, USART IT RXNE, ENABLE);
      USART Cmd (USART1, ENABLE);
      // jeste zbyva konfigurace preruseni
```

V.Váňa: CanSAT Book for Students – part 2 109



```
NVIC PriorityGroupConfig(NVIC PriorityGroup 1);
      NVIC InitStructure.NVIC IRQChannel = USART1 IRQn;
      NVIC InitStructure.NVIC IRQChannelPreemptionPriority = 0;
      NVIC InitStructure.NVIC IRQChannelSubPriority = 0;
      NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
      NVIC Init(&NVIC InitStructure);
}
/**
  * @brief Porovnani dvou retezcu
  * @param retezec1, retezec2
  * @retval : 0 - nerovna se, 1 - rovna se, typ uint8 t
  */
uint8 t Strcmp (const uint8 t * s1, const uint8 t * s2)
{
    for(; *s1 == *s2; ++s1, ++s2)
       if (*s1 == 0 || *s1 == 0x0d)
            return 0;
    return *(unsigned char *)s1 < *(unsigned char *)s2 ? -1 : 1;
}
/**
  * @brief Prekopirovani jednoho retezce do druheho
  * @param pointer na cilovy retezec, pointer na zdrojovy retezec
  * @retval : none
  */
void Strcpy(uint8 t *d1, const uint8 t *s1)
 uint8 t i;
 for (i=0; s1[i] != '\0'; ++i)
   d1[i] = s1[i];
     d1[i] = ' \setminus 0';
}
/**
 * @brief Cekani
  * @param nTick: delka cekaci smycky
  * @retval : None
  */
void Delay( IO uint32 t nTick)
{
  for(; nTick != 0; nTick--);
}
/* Zapis znaku 'ch' do souboru 'f' */
int fputc(int ch, FILE * f)
 {
  /* Predani znaku pro odeslani UARTem */
  USART SendData(USART1, (u8) ch);
    /* Cekame, dokud neni prenos dokoncen */
       while (USART GetFlagStatus(USART1, USART FLAG TXE) == RESET);
       return ch;
}
```

110

V.Váňa: CanSAT Book for Students – part 2



PA9 je připojen na pin 11 obvodu MAX3232 a po převedení na úroveň RS232 na pin 2 konektoru CANON9 (poz. určitě znáte dnes již klasický obvod MAX232 pro převod TTL $\leftarrow \rightarrow$ RS232 jenž je napájen 5V, obvod MAX3232 je s ním pinově kompatibliní a může být napájen 5V či 3.3V přičemž při napájení 5V se chová jako obvod MAX232, při nápájení 3.3V slouží k převodu mezi RS232 a 3V logikou s nímž mj pracuje i startkit **STM32 VL DISCOVERY**)

oubor Akce	Zobrazit Nápověda
• 🔿 📰 🔽	
VLAD	
Adaptér	y hostitele SD
Adaptér	y PCMCIA
Baterie	
Diskové	jednotky
Grafické	adaptéry
👂 🟺 Hostitel	ské řadiče sběrnice IEEE 1394
> 🔮 Jednotk	y DVD/CD-ROM
Klávesni	ce
Modem	у
Monitor	У
🕞 📲 Myši a ji	ná polohovací zařízení
Počítač	
A Porty (C	OM a LPT)
Proli	fic USB-to-Serial Comm Port (COM6)
Proceso	ry
Přenosn	á zařízení
👂 🥁 Řadiče II	DE ATA/ATAPI
👂 🖣 🕅 Řadiče L	JSB (Universal Serial Bus)
🔈 🐗 🕺 Řadiče z	wuku, videa a her
Síťové a	daptéry
b 📲 Systéme	ová zařízení
D - T Zařízení	pro infračervený přenos
> Tařízení	pro zpracování obrázků
D Zařízení	standardu HID



becné Nastavení portu	Ovladač F	odrobnosti	
E	Bity za sekundi	J: 4800	•
	Datové bit	y: 8	•]
	Parita	a: Žádná	•
	Stop-bit	y: 1	•
	Řízení tok	u: Žádné	-
		Upřesnit	Obnovit výchozi



RealTerm: Serial Capture Program 2.0.0.57	- 10 C	
> Tlacitko stisknuto - ahojur > stiskni tlacitko - nazdaru > Tlacitko stisknuto - ahojur		
Display Port Capture Pins Send Echo Port 12	C 12C-2 12CMisc Misc	<u>In</u> <u>Clear</u> Freeze
Baud 4800 ▼ Port 6 Parity Data Bits Stop Bits • None • 8 bits • 1 bit 2 bits • Odd • 7 bits • Hardware Flow Control Hardware Flow Control • Mark • 5 bits • DTR/DSR • RS485-rts	Open Spy ✓ Change ✓ Software Flow Control ✓ Receive Xon Char: 17 Transmit Xoff Char: 19 Winsock is: C C Raw ✓ Telnet ✓	Status Disconnect RXD (2) TXD (3) CTS (8) DCD (1) DSR (6) Ring (9) BREAK Error
clear terminal screen, return to start of page	Char Count:180	CPS:0 Port: 6 4800 8N1 None









2.5 UART RX – příjem znaků (program07)









Soubor main.c

```
/* STM32 VL Discovery Library
* Vd pro SPSE Jecna ARM Lectures
* prijem znaku z USARTu, zobrazeni na LCD
* Created on: Apr 20, 2011 program07
*/
/* Includes -----
                                _____
___*/
#include <stddef.h>
#include <stdio.h>
#include <string.h>
#include "stm32f10x.h"
#include "STM32vldiscovery.h"
#include "LCD.h"
/* Private typedef ------
---*/
/* Private define -----
                        _____
---*/
#define BUFF SIZE 40
```



```
// priznak ukonceni prijmu z PC
uint8 t RX prijato = 0;
// buffery na prijimane a odesilane znaky
uint8 t R Buff[BUFF SIZE], T Buff[BUFF SIZE];
uint8 t LCD Message1[] = "Prijem GPS 4800Bd";
uint8 t LCD Message2[] = "Stisk tlacitka! ";
/* Private function prototypes -----
---*/
void USART Inicializace(void);
void GPIO Inicialization(void);
int fputc(int ch, FILE * f);
void Delay(__IO uint32_t nTick);
uint8_t Strcmp (const uint8_t * s1, const uint8_t * s2); // porovnani str
void Strcpy(uint8_t *d1, const uint8_t *s1); // kopirovani str
void vynulovaniRXregistru(void);
int main(void)
{
 LCD Inicialization();
 GPIO Inicialization();
  // aktivujeme USART
 USART Inicializace();
 STM32vldiscovery LEDOn(LED4); // LED4 - modra dioda
 STM32vldiscovery LEDOn(LED3); // LED3 - zelena dioda pridal jsem
 printLCD(LCD Message1);
 while (1)
  {
   if (RX prijato == 1) // neco jsme prijali - zakoncene CR
      {
     LCD Clear(); // vymazani LCD a navrat kurzoru na prvni pozici prvniho
radku
     Strcpy (LCD Message1, R Buff );
     printLCD(LCD Message1);
     vynulovaniRXregistru();
     RX prijato = 0; // vymazeme flag ukonceni prijmu
      }
  }
}
void GPIO Inicialization(void)
{
 STM32vldiscovery LEDInit(LED3);
 STM32vldiscovery LEDInit(LED4);
 STM32vldiscovery PBInit(BUTTON USER, BUTTON MODE GPIO);
 STM32vldiscovery LEDOff(LED3);
 STM32vldiscovery_LEDOff(LED4);
```

V.Váňa: CanSAT Book for Students – part 2 117



```
}
void USART Inicializace(void)
{
     GPIO InitTypeDef GPIO InitStructure;
     USART InitTypeDef USART InitStructure;
     NVIC InitTypeDef NVIC InitStructure;
      /* pustime hodiny do periferie (protoze jde o alternativni funkci,
musi
      * jit hodiny i do AFIO periferie! */
     RCC APB2PeriphClockCmd(RCC APB2Periph GPIOA | RCC APB2Periph USART1 |
RCC APB2Periph AFIO, ENABLE);
      // PA9 je Tx
      GPIO InitStructure.GPIO Pin = GPIO Pin 9;
     GPIO_InitStructure.GPIO Mode = GPIO Mode AF PP; // alternate
function!
     GPIO InitStructure.GPIO Speed = GPIO Speed 50MHz;
     GPIO Init(GPIOA, &GPIO InitStructure);
      // PA10 je Rx
     GPIO InitStructure.GPIO Pin = GPIO Pin 10;
     GPIO InitStructure.GPIO Mode = GPIO Mode IN FLOATING;
     GPIO Init(GPIOA, &GPIO InitStructure);
     USART InitStructure.USART BaudRate = 4800;
      /* Pokud mate dobry prevodnik USB-ttl, tak muzete pochopitelne
      * zvysit rychlost. Hodnota 4800 je pro prijem z GPS */
     USART InitStructure.USART_WordLength = USART_WordLength_8b;
     USART InitStructure.USART StopBits = USART StopBits 1;
     USART InitStructure.USART Parity = USART Parity No;
     USART InitStructure.USART HardwareFlowControl =
USART HardwareFlowControl None;
     USART InitStructure.USART Mode = USART Mode Rx | USART Mode Tx;
     USART Init(USART1, &USART InitStructure);
     USART ITConfig(USART1, USART IT RXNE, ENABLE);
     USART Cmd(USART1, ENABLE);
      // jeste zbyva konfigurace preruseni
     NVIC PriorityGroupConfig(NVIC PriorityGroup 1);
     NVIC InitStructure.NVIC IRQChannel = USART1 IRQn;
     NVIC InitStructure.NVIC IRQChannelPreemptionPriority = 0;
     NVIC InitStructure.NVIC IRQChannelSubPriority = 0;
     NVIC InitStructure.NVIC IRQChannelCmd = ENABLE;
     NVIC Init(&NVIC InitStructure);
}
  * @brief Porovnani dvou retezcu
  * @param retezec1, retezec2
  * @retval : 0 - nerovna se, 1 - rovna se, typ uint8 t
  */
```

118

```
V.Váňa: CanSAT Book for Students – part 2
```



```
Inovace ve vzdělávání na území hl.m. Prahy 2011, proj.č.2046
```

```
uint8_t Strcmp (const uint8_t * s1, const uint8_t * s2)
{
    for(; *s1 == *s2; ++s1, ++s2)
       if(*s1 == 0 || *s1 == 0x0d)
            return 0;
    return *(unsigned char *)s1 < *(unsigned char *)s2 ? -1 : 1;</pre>
}
/**
 * @brief Prekopirovani jednoho retezce do druheho
 * @param pointer na cilovy retezec, pointer na zdrojovy retezec
  * @retval : none
  */
void Strcpy(uint8 t *d1, const uint8 t *s1)
{
 uint8_t i;
 for (i=0; s1[i] != '\0'; ++i)
   d1[i] = s1[i];
     d1[i] = ' \setminus 0';
}
void Delay( IO uint32 t nTick)
{
 for(; nTick != 0; nTick--);
}
/* Zapis znaku 'ch' do souboru 'f' */
int fputc(int ch, FILE * f)
 {
 /* Predani znaku pro odeslani UARTem */
  USART SendData(USART1, (u8) ch);
   /* Cekame, dokud neni prenos dokoncen */
      while (USART GetFlagStatus(USART1, USART FLAG TXE) == RESET);
       return ch;
 }
void vynulovaniRXregistru(void)
{
 uint8 t i;
 for(i=0; i<40; i++)</pre>
  R Buff[i]= '\0';
 ;
}
```

2.6 ADC (program08)

Mezi GND a +3,3V připojíme potenciometr cca 22k, jeho běžec spojíme s **PC4** mající význam **ADC1_INT14**

V.Váňa: CanSAT Book for Students – part 2 119













Soubor main.c

```
/* ADC v STM32 VL Discovery
** main.c
** program08
** Vd for SPSE Jecna course
*/
#include <stddef.h>
#include "string.h"
#include "stm32fl0x.h"
#include "STM32 Discovery.h"
#include "LCD.h"
GPIO_InitTypeDef GPIO_InitStructure;
uint8_t LCD_text[] = "Mereni SPSE";
uint16_t Hodnota=0;
uint16_t Hodnota0ld=1;
void GPIO Inicializace(void); // nastaveni vstupne/vystupnich pinu na kitu
```

V.Váňa: CanSAT Book for Students – part 2 122



V.Váňa: CanSAT Book for Students – part 2

Inovace ve vzdělávání na území hl.m. Prahy 2011, proj.č.2046

```
void ADC Inicializace(void);
void Delay(__IO uint32_t nTick);
uint16 t ADC1 Read(void);
void itoa(uint16 t n, int8 t s[]);
void reverse(int8 t s[]);
void KopirujLCDtext(int8 t s[]);
void barGraph(uint16 t Hodnota);
int main(void)
{
  //GPIO LCD Inicializace();
  LCD Inicialization();
  STM32 Discovery LEDOn (LED4); // LED4 - modra
  //Delay(0xAAAA);
  GPIO Inicializace();
  //LCD Init();
  LCD Clear();
  ADC Inicializace();
  printLCD(LCD text);
  Delay(0xAAAAA);
  STM32 Discovery LEDOff(LED4); // LED4 - modra
  while (1)
  {
        Delay(OxAAAAA);
        Delay(OxAAAAA);
        Delay(0xAAAAA);
        LCD Clear();
        //Hodnota = ADC1 Read(); //Hodnota je cele cislo mezi 0 a 4095
        Hodnota = (ADC1 Read() * 3000)/4095;
        itoa(Hodnota,LCD text);
        *strcat(LCD text, " mV"); //k s1 prida s2
        printLCD(LCD text);
  }
}
void GPIO Inicializace(void)
{
 STM32 Discovery PBInit (BUTTON USER, BUTTON MODE GPIO);
 STM32 Discovery LEDOff(LED3);
 //STM32 Discovery LEDOff(LED4);
}
void ADC Inicializace(void)
{
 ADC InitTypeDef ADC InitStructure;
 GPIO InitTypeDef GPIO InitStructure;
  /* ADCCLK = PCLK2/1 = 24/2 = 12MHz*/
  RCC ADCCLKConfig(RCC PCLK2 Div2);
  /* povolime hodiny do portu */
 RCC APB2PeriphClockCmd(RCC APB2Periph GPIOC, ENABLE);
 GPIO InitStructure.GPIO Pin = GPIO Pin 4; // tady pripojime merene
napeti
GPIO InitStructure.GPIO Speed = GPIO Speed 50MHz;
```

123



```
GPIO InitStructure.GPIO Mode = GPIO Mode AIN;
  GPIO_Init(GPIOC, &GPIO_InitStructure);
  /* povolime hodiny do A/D prevodniku */
  RCC APB2PeriphClockCmd(RCC APB2Periph ADC1, ENABLE);
  //ADC DeInit(ADC1);
  /* ADC1 Configuration --
 --*/
  ADC InitStructure.ADC Mode
                                             = ADC Mode Independent;
  ADC
      InitStructure.ADC_ScanConvMode
                                             = DISABLE;
  ADC
      InitStructure.ADC ContinuousConvMode = DISABLE;
      InitStructure.ADC_ExternalTrigConv
  ADC
                                             = ADC ExternalTrigConv None;
  ADC
      InitStructure.ADC DataAlign
                                             = ADC DataAlign Right;
  ADC
      InitStructure.ADC NbrOfChannel
                                             = 1;
  ADC
      Init( ADC1, &ADC InitStructure );
   /* ADC1 Regular Channel 14 */
  ADC RegularChannelConfig( ADC1, ADC Channel 14, 1,
ADC SampleTime 239Cycles5);
  //ADC RegularChannelConfig(ADC1, ADC Channel 14, 1,
ADC SampleTime 13Cycles5);
  /* Enable ADC1 external trigger conversion */
  ADC ExternalTrigConvCmd( ADC1, ENABLE );
  ADC Cmd(ADC1, ENABLE);
  /* Provedeme kalibraci prevodniku */
  /* Enable ADC1 reset calibration register */
  ADC ResetCalibration(ADC1);
  /* Check the end of ADC1 reset calibration register */
  while(ADC GetResetCalibrationStatus(ADC1));
  /* Start ADC1 calibration */
  ADC StartCalibration(ADC1);
  /* Check the end of ADC1 calibration */
  while(ADC GetCalibrationStatus(ADC1));
}
void barGraph(uint16 t Hodnota)
if ((Hodnota > 100) && (Hodnota <= 200)) printLCD("X");</pre>
if ((Hodnota > 200) && (Hodnota<=300)) printLCD("XX");</pre>
if ((Hodnota > 300)&&(Hodnota<=400)) printLCD("XXX");</pre>
if ((Hodnota > 400)&&(Hodnota<=500)) printLCD("XXXX");</pre>
if ((Hodnota > 500) && (Hodnota <= 600)) printLCD("XXXXX");</pre>
if ((Hodnota > 600) && (Hodnota<=700)) printLCD("XXXXXX");</pre>
if ((Hodnota > 700) && (Hodnota <= 800)) printLCD("XXXXXXX");</pre>
if ((Hodnota > 800)&&(Hodnota<=900)) printLCD("XXXXXXXX");</pre>
if ((Hodnota > 900)&&(Hodnota<=1000)) printLCD("XXXXXXXX");</pre>
if ((Hodnota > 1000) && (Hodnota<=1100)) printLCD("XXXXXXXXX");</pre>
if ((Hodnota > 1100) && (Hodnota<=1200)) printLCD("XXXXXXXXXX");</pre>
if ((Hodnota > 1200)&&(Hodnota<=1300)) printLCD("XXXXXXXXXXXX");</pre>
if ((Hodnota > 1300) && (Hodnota<=1400)) printLCD("XXXXXXXXXXXX");</pre>
};
uint16 t ADC1 Read(void)
// Start the conversion
                                                         ARM STM32 programming
V.Váňa: CanSAT Book for Students – part 2
                                      124
```



```
ADC SoftwareStartConvCmd(ADC1, ENABLE);
 // Wait until conversion completion
 while(ADC GetFlagStatus(ADC1, ADC FLAG EOC) == RESET);
 // Get the conversion value
 return ADC GetConversionValue(ADC1);
}
void Delay( IO uint32 t nTick)
{
 for(; nTick != 0; nTick--);
}
void itoa(uint16 t n, int8 t s[])
{
   int i, sign;
   if ((sign = n) < 0) /* record sign */
      n = -n;
                       /* make n positive */
   i = 0;
   do {
            /* generate digits in reverse order */
      s[i++] = n % 10 + '0';  /* get next digit */
    if (sign < 0)</pre>
       s[i++] = '-';
   s[i] = ' \setminus 0';
   reverse(s);
}
void reverse(int8 t s[])
{
   int i, j, k;
   char c;
    k = strlen(s) - 1;
    for (i = 0, j = k; i<j; i++, j--) {</pre>
       c = s[i];
       s[i] = s[j];
       s[j] = c;
   }
 }
void assert func (const char *file, int line, const char *func, const char
*failedexpr)
{
 while(1)
{ }
}
void assert(const char *file, int line, const char *failedexpr)
{
  __assert_func (file, line, NULL, failedexpr);
}
```





2.7 ADC s výstupem na LCD jako BarGraph – program09



Project	д
🖃 🚵 program09	
🗄 📇 source	
👝 🚠 main.c	
⊞ 🔛 LCD.c	
🗄 🔛 STM32_Discovery.c	
🗈 🔝 stm32f10x_it.c	
🖨 📇 libraries	
🕀 🔛 misc.c	
🗄 🔝 stm32f10x_gpio.c	
in itm32f10x_rcc.c	
i stm32f10x_tim.c	
💼 🔝 stm32f10x_usart.c	
🔄 🔝 stm32f10x_adc.c	
🛱 😋 CMSIS	
🔁 🔛 core_cm3.c	
🔄 🔝 system_stm32f10x.c	
🗄 🛅 startup	
Project Books () Europians (), 1	Cemplates

Souboru main.c

```
/* ADC s STM32 VL Discovery
** main.c
** vystup na LCD jako BarGraph
** Vd pro SPSE Jecna ARM Courses
*/
#include <stddef.h>
#include <string.h>
#include "stm32f10x.h"
#include "STM32_Discovery.h"
#include "LCD.h"
/* Private typedef -----
---*/
GPIO InitTypeDef GPIO InitStructure;
uint8_t LCD_text[] = "Mereni SPSE";
uint16_t Hodnota=0;
uint16_t HodnotaOld=1;
/* Private function prototypes ------
---*/
void GPIO Inicializace(void); // nastavení vstupne/vystupnich pinu na kitu
V.Váňa: CanSAT Book for Students – part 2 127
                                                     ARM STM32 programming
```



```
void ADC Inicializace(void);
void Delay(__IO uint32_t nTick);
uint16 t ADC1 Read(void);
void itoa(uint16 t n, int8 t s[]);
void reverse(int8_t s[]);
void barGraph(uint16 t Hodnota);
int main(void)
{
  LCD Inicialization();
  STM32 Discovery LEDOn(LED4); // LED4 - modra
  GPIO Inicializace();
  LCD Clear();
  ADC Inicializace();
  printLCD(LCD text);
  Delay(0xAAAAA);
  STM32 Discovery LEDOff(LED4); // LED4 - modra
  while (1)
  {
        Delay(0xAAAAA);
        Hodnota = ADC1 Read()/3 ;
        if (HodnotaOld!= Hodnota)
        {
           LCD Clear();
         barGraph(Hodnota);
          HodnotaOld = Hodnota ; //aby se porad neprekreslovalo
        }
  }
}
void GPIO Inicializace(void)
{
  STM32 Discovery PBInit (BUTTON USER, BUTTON MODE GPIO);
  STM32 Discovery LEDOff(LED3);
}
void ADC Inicializace(void)
{
 ADC InitTypeDef ADC InitStructure;
 GPIO InitTypeDef GPIO InitStructure;
  /* ADCCLK = PCLK2/1 = 24/2 = 12MHz*/
  RCC ADCCLKConfig(RCC PCLK2 Div2);
  /* povolime hodiny do portu */
  RCC APB2PeriphClockCmd(RCC APB2Periph GPIOC, ENABLE);
  GPIO InitStructure.GPIO Pin = GPIO Pin 4; // tady pripojime merene
napeti
  GPIO InitStructure.GPIO Speed = GPIO Speed 50MHz;
  GPIO InitStructure.GPIO Mode = GPIO Mode AIN;
  GPIO Init(GPIOC, &GPIO InitStructure);
  /* povolime hodiny do A/D prevodniku */
  RCC APB2PeriphClockCmd(RCC APB2Periph ADC1, ENABLE);
```

128



```
/* ADC1 Configuration -----
___*/
  ADC InitStructure.ADC Mode
                                            = ADC Mode Independent;
  ADC InitStructure.ADC ScanConvMode
                                            = DISABLE;
     InitStructure.ADC ContinuousConvMode = DISABLE;
  ADC
     InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
InitStructure.ADC DataAlign = ADC DataAlign Right;
  ADC
  ADC InitStructure.ADC DataAlign
  ADC
     InitStructure.ADC NbrOfChannel
                                            = 1;
  ADC Init( ADC1, &ADC InitStructure );
  /* ADC1 Regular Channel 14 */
  ADC RegularChannelConfig( ADC1, ADC Channel 14, 1,
ADC SampleTime 239Cycles5);
  /* Enable ADC1 external trigger conversion */
  ADC ExternalTrigConvCmd ( ADC1, ENABLE );
  ADC Cmd(ADC1, ENABLE);
  /* Provedeme kalibraci prevodniku */
  /* Enable ADC1 reset calibration register */
  ADC ResetCalibration(ADC1);
  /* Check the end of ADC1 reset calibration register */
  while(ADC GetResetCalibrationStatus(ADC1));
  /* Start ADC1 calibration */
 ADC StartCalibration (ADC1);
  /* Check the end of ADC1 calibration */
  while(ADC GetCalibrationStatus(ADC1));
}
void barGraph(uint16 t Hodnota)
{
if ((Hodnota > 100) && (Hodnota<=200)) printLCD("X");</pre>
if ((Hodnota > 200) && (Hodnota<=300)) printLCD("XX");</pre>
if ((Hodnota > 300)&&(Hodnota<=400)) printLCD("XXX");</pre>
if ((Hodnota > 400)&&(Hodnota<=500)) printLCD("XXXX");</pre>
if ((Hodnota > 500)&&(Hodnota<=600)) printLCD("XXXXX");</pre>
if ((Hodnota > 600)&&(Hodnota<=700)) printLCD("XXXXXX");</pre>
if ((Hodnota > 700)&&(Hodnota<=800)) printLCD("XXXXXXX");</pre>
if ((Hodnota > 800)&&(Hodnota<=900)) printLCD("XXXXXXXX");</pre>
if ((Hodnota > 900)&&(Hodnota<=1000)) printLCD("XXXXXXXXX");</pre>
if ((Hodnota > 1000)&&(Hodnota<=1100)) printLCD("XXXXXXXXX");</pre>
if ((Hodnota > 1100)&&(Hodnota<=1200)) printLCD("XXXXXXXXXXX");</pre>
if ((Hodnota > 1200)&&(Hodnota<=1300)) printLCD("XXXXXXXXXXXX");</pre>
if ((Hodnota > 1300)&&(Hodnota<=1400)) printLCD("XXXXXXXXXXXXX");</pre>
};
uint16 t ADC1 Read(void)
  // Start the conversion
 ADC SoftwareStartConvCmd(ADC1, ENABLE);
  // Wait until conversion completion
  while(ADC GetFlagStatus(ADC1, ADC FLAG EOC) == RESET);
 // Get the conversion value
```

129

V.Váňa: CanSAT Book for Students – part 2



```
return ADC GetConversionValue(ADC1);
}
void Delay( IO uint32 t nTick)
{
 for(; nTick != 0; nTick--);
}
void itoa(uint16 t n, int8 t s[])
{
    int i, sign;
    if ((sign = n) < 0) /* record sign */
        n = -n;
                          /* make n positive */
    i = 0;
    do {
               /* generate digits in reverse order */
    s[i++] = n % 10 + '0';  /* get next digit */
} while ((n /= 10) > 0);  /* delete it */
    if (sign < 0)
       s[i++] = '-';
    s[i] = '\0';
   reverse(s);
}
void reverse(int8 t s[])
{
   int i, j;
   char c;
    for (i = 0, j = strlen(s)-1; i<j; i++, j--) {</pre>
       c = s[i];
        s[i] = s[j];
       s[j] = c;
   }
}
void assert func(const char *file, int line, const char *func, const char
*failedexpr)
{
 while(1)
{ }
}
void assert(const char *file, int line, const char *failedexpr)
{
    assert func (file, line, NULL, failedexpr);
}
```





2.8 PWM (program10)







V.Váňa: CanSAT Book for Students – part 2

132



```
/*
* PWM main.c
* Intenzita svetla LED dana PWM - program10
* Vd pro SPSE Jecna ARM lectures
* podle MCU serialu
*/
/* Includes ------
---*/
#include "stm32f10x.h"
#include "STM32 Discovery.h"
/* Private typedef ------
___*/
/* Private define ------
---*/
#define DELKA 0xAAAAA
// Periodu volime pro cca 35 kHz: 24 MHz / 666 = 36 KHz
#define PERIODA 665
//#define NOREMAP
/* pro vyzkouseni PWM na vystupu PBO a PB1 odkomentujete definovani NOREMAP
*/
/* Private macro ------
___*/
/* Private variables ------
___*/
TIM_TimeBaseInitTypeDef TIM TimeBaseStructure;
TIM OCInitTypeDef TIM OCInitStructure;
GPIO InitTypeDef GPIO InitStructure;
int ZmenaIntenzity = 1;
uint16 t Intenzita = 0;
uint16 t PrescalerValue = 0;
/* Private function prototypes ------
---*/
void GPIO Inicializace(void); // nastavení vstupne/vystupnich pinu na kitu
void TIM3 Inicializace(void); // nastaveni Timeru3
void Delay(__IO uint32_t nTick); // cekaci smycka
void My_Led3_Off(void); // zhasnuti zelene LED
void My Led3 On(void);
                            // rozsviceni zelene LED
/* Private functions -----
---*/
/**
 * @brief Main program.
 * @param None
 * @retval : None
 */
```

V.Váňa: CanSAT Book for Students – part 2 133



```
int main(void)
{
  /* inicializace timeru3 - povolime mu hodinovy takt */
  RCC APB1PeriphClockCmd(RCC APB1Periph TIM3 , ENABLE);
  /* GPIOA a GPIOB spuštění hodin */
  RCC APB2PeriphClockCmd(RCC APB2Periph GPIOA | RCC APB2Periph GPIOB |
              RCC APB2Periph GPIOC |RCC APB2Periph AFIO, ENABLE);
  GPIO Inicializace();
#ifdef NOREMAP
  /* pro nepremapovany vystup TIM3 chan 3 a 4 */
  GPIO InitStructure.GPIO Pin = GPIO Pin 0 | GPIO Pin 1;
  GPIO InitStructure.GPIO Mode = GPIO Mode AF PP;
  GPIO InitStructure.GPIO Speed = GPIO Speed 50MHz;
  GPIO Init(GPIOB, & GPIO InitStructure);
#else
  /* pro premapovany vystup TIM3 chan 3 a 4 */
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8 | GPIO Pin 9;
  GPIO InitStructure.GPIO Mode = GPIO Mode AF PP;
  GPIO InitStructure.GPIO Speed = GPIO Speed 50MHz;
  GPIO Init(GPIOC, &GPIO InitStructure);
  /* musime remapovat funkci PC8,PC9 na timer3 chanel 3 a 4 output */
  GPIO PinRemapConfig(GPIO FullRemap TIM3, ENABLE);
#endif
  /* musime nastaveit Timer3 */
  TIM3 Inicializace();
  while (1)
  {
     TIM OCInitStructure.TIM Pulse = Intenzita;
     TIM OC3Init(TIM3, &TIM OCInitStructure);
     TIM OC3PreloadConfig(TIM3, TIM OCPreload Enable);
     TIM ARRPreloadConfig(TIM3, ENABLE);
     if ((ZmenaIntenzity>0 && Intenzita<PERIODA) || (ZmenaIntenzity<0 &&
Intenzita>1) )
      {
            Intenzita = Intenzita + ZmenaIntenzity;
      }
     else
      {
#ifdef NOREMAP
      STM32 Discovery LEDOn(LED4);
                                              // zapnem LED4 - modra
#endif
      ZmenaIntenzity = - ZmenaIntenzity; // obratime smer zmeny svetla
      Delay(0xFFFFF);
                                                     // chvili pockame -
osetreni kmitani kontaktu
#ifdef NOREMAP
```

134

V.Váňa: CanSAT Book for Students – part 2



```
STM32 Discovery LEDOff(LED4); // zapnem LED4 - modra
     /* tedy pri zmene typu zhasinani/rozsvecovani klikneme modrou led */
#endif
     }
    Delay(0xFFF); Delay(0xFFF);
                                                   // kratka pauza aby
      Delay(0xFFF);
zmena intenzity nebyla moc rychla
   if(0 != STM32 Discovery PBGetState(BUTTON USER))
    {
     // tj. když je stisknuto tlacitko
#ifdef NOREMAP
    STM32 Discovery LEDOn(LED3);
                                              // zapnem LED3 - zelena
#endif
     // nastavime 100% PWM pro CH4, tj. remapovana LED3 zelena bude svitit
naplno
     My Led3 On();
     Delay(0xFFFFF);
                                                    // chvili pockame -
osetreni kmitani kontaktu
#ifdef NOREMAP
    STM32 Discovery LEDOff(LED3);
                                             // vypnem LED3 - zelena
#endif
     // nastavime 0% PWM pro CH4, tj. LED3 zelena nebude svitit
     My Led3 Off();
   }
 }
}
/**
 * @brief Zhasnuti zelene LED.
 * @param None
 * @retval : None
 */
void My Led3 Off(void)
{
     /* PWM1 Mode configuration: Channel4 */
     TIM OCInitStructure.TIM Pulse = 0;
     TIM OC4Init(TIM3, &TIM OCInitStructure);
     TIM OC4PreloadConfig(TIM3, TIM OCPreload Enable);
     TIM ARRPreloadConfig(TIM3, ENABLE);
}
/**
 * @brief Rozsviceni zelene LED.
 * @param None
 * @retval : None
 */
void My Led3 On(void)
{
     /* PWM1 Mode configuration: Channel4 */
     TIM OCInitStructure.TIM Pulse = PERIODA;
     TIM OC4Init(TIM3, &TIM OCInitStructure);
     TIM OC4PreloadConfig(TIM3, TIM OCPreload Enable);
     TIM ARRPreloadConfig(TIM3, ENABLE);
```

```
V.Váňa: CanSAT Book for Students – part 2 135
```



}

Inovace ve vzdělávání na území hl.m. Prahy 2011, proj.č.2046

```
/**
 * @brief Configure the GPIO Pins.
 * @param None
 * @retval : None
 */
void GPIO Inicializace(void)
 STM32 Discovery LEDInit(LED3);
 STM32 Discovery LEDInit(LED4);
 STM32_Discovery_PBInit(BUTTON_USER, BUTTON MODE GPIO);
 STM32 Discovery LEDOff(LED3);
 STM32 Discovery LEDOff(LED4);
}
/**
 * @brief Configure the Tim3.
 * @param None
 * @retval : None
 */
void TIM3 Inicializace(void)
{
 /* Compute the prescaler value */
 PrescalerValue = (uint16 t) (SystemCoreClock / 24000000) - 1;
 /* Uvedenych 24000000 plati pro nezmeneny System Clock!! */
  /* Time base configuration */
 TIM TimeBaseStructure.TIM Period = PERIODA;
  TIM TimeBaseStructure.TIM Prescaler = PrescalerValue;
 TIM TimeBaseStructure.TIM ClockDivision = 0;
 TIM TimeBaseStructure.TIM CounterMode = TIM CounterMode Up;
 TIM TimeBaseInit(TIM3, &TIM TimeBaseStructure);
  /* PWM1 Mode configuration: Channel3 */
 TIM OCInitStructure.TIM OCMode = TIM OCMode PWM1;
  TIM OCInitStructure.TIM OCPolarity = TIM OCPolarity High;
  TIM OCInitStructure.TIM OutputState = TIM OutputState Enable;
  TIM OCInitStructure.TIM Pulse = 100;
  TIM OC3Init(TIM3, &TIM OCInitStructure);
  TIM OC3PreloadConfig(TIM3, TIM OCPreload Enable);
  /* PWM1 Mode configuration: Channel4 */
  TIM OCInitStructure.TIM OutputState = TIM OutputState Enable;
  TIM OCInitStructure.TIM Pulse = 1;
 TIM OC4Init(TIM3, &TIM OCInitStructure);
  TIM OC4PreloadConfig(TIM3, TIM OCPreload Enable);
  TIM ARRPreloadConfig(TIM3, ENABLE);
/* TIM3 enable counter */
```

V.Váňa: CanSAT Book for Students – part 2 136



```
TIM_Cmd(TIM3, ENABLE);
}
void Delay( IO uint32 t nTick)
{
  for(; nTick != 0; nTick--);
}
```

Oscilloscope - NI ELVISmx	
Sample Rate: 200,00 kS/s	Basic Settings Advanced Settings Channel 1 Settings Source Source
	AI 0
	Image: Scale vertical volts/Div Coupling DC v DC v Scale vertical volts/Div Vertical volts/Div Volts/Div Position (Div)
	1 V 0 ★ 2 V 0 ★ Timebase Trigger Type Slope Type Slope ✓ Source Level (v) Chan 0 Source 1,2 ★
Timeout CH 0 Meas: RMS: 2,642 V Freq: 36,055 kHz Vp-p: 2,972 V Cursors Settings Cursors On C1 CH 0 C2 CH 0 C1 CH 0 C1 CH 1	S0 us Horizontal Position (%) S0 us 22 Im Instrument Control Acquisition Mode Device Acquisition Mode Dev1 (NI myDAQ) Run Continuously Run Stop Autoscale Image: Stop

Všimněme si, že platí naše poznámka na začátku zdrojového kódu

// Periodu volime pro cca 35 kHz: 24 MHz / 666 = 36 KHz
#define PERIODA 665

2.9 DAC (program11)

Ve většině programů ukazujících převod DAC se obvykle generuje nějaký signál mající pilový, sinusový apod. průběh. Programujeme v nich čítače, plníme paměť hodnotami funkce sinus apod. a mj. i převod DAC. Takže kód pro převod je ztracen v mnoha dalším kódu.

V.Váňa: CanSAT Book for Students – part 2 137



Aby vynikla jednoduchost naprogramování DAC, budeme postupovat trochu jinak. O převod DAC doplníme náš **program08** ukazující převod ADC, kdy jsme mezi GND a +3V připojili potenciometr 22k a jeho běžec na **PC4**. Po převodu ADC jsme získanou hodnotu po vynásobení konstantou zobrazili na LCD, který tak ukazoval napětí na běžci potenciometru v mV.

Nyní tuto digitální hodnotu ještě převody DAC převedeme na analogovou. Použijeme přitom dva DAC kanály a budeme měřit napětí na výstupech převodníků tj**. PA4** a **PA5**.







Soubor main.c

```
/* ADC a DAC v STM32 VL Discovery
** main.c
** program11
** Vd for SPSE Jecna course
*/
#include <stddef.h>
#include "string.h"
#include "stm32f10x.h"
#include "STM32_Discovery.h"
#include "LCD.h"
GPIO_InitTypeDef GPIO_InitStructure;
uint8_t LCD_text[] = "Testovani ADC a DAC Mereni SPSE";
uint16_t Hodnota=0;
uint16_t HodnotaOld=1;
```



```
/* Private function prototypes -----
---*/
void GPIO Inicializace(void); // nastavení vstupne/vystupnich pinu na kitu
void ADC Inicializace(void);
void Delay( IO uint32 t nTick);
uint16 t ADC1 Read(void);
void itoa(uint16 t n, int8 t s[]);
void reverse(int8 t s[]);
void KopirujLCDtext(int8 t s[]);
void DAC Inicializace(void);
int main(void)
{
  //GPIO LCD Inicializace();
 LCD Inicialization();
 STM32_Discovery_LEDOn(LED4); // LED4 - modra
  //Delay(0xAAAA);
 GPIO Inicializace();
 DAC Inicializace();
  //LCD Init();
 LCD Clear();
 ADC Inicializace();
 printLCD(LCD text);
 Delay(0xAAAAA);
 STM32 Discovery LEDOff(LED4); // LED4 - modra
 while (1)
  {
       Delay(0xAAAAA);
       Delay(0xAAAAA);
       Delay(OxAAAAA);
       LCD Clear();
       Hodnota = ADC1 Read(); //Hodnota je cele cislo mezi 0 a 4095
      /* Set DAC Channel1 DAC DHR12R1 register */
      //DAC SetChannel1Data(DAC Align 12b R, (uint32 t)Hodnota DAC1);
      DAC SetChannel1Data(DAC Align 12b R, (uint32 t)Hodnota);
      /* Set DAC Channel2 DAC DHR12R1 register */
     //DAC SetChannel2Data(DAC Align 12b R, (uint32 t)Hodnota DAC2);
      DAC SetChannel2Data(DAC Align 12b R, (uint32 t)Hodnota);
       Hodnota = (Hodnota * 3000)/4095;
       itoa(Hodnota,LCD text);
       *strcat(LCD text, " mV"); //k s1 prida s2
       printLCD(LCD text);
  }
}
void GPIO Inicializace(void)
{
 STM32 Discovery PBInit (BUTTON USER, BUTTON MODE GPIO);
 STM32 Discovery_LEDOff(LED3);
}
```



```
void ADC Inicializace(void)
{
 ADC InitTypeDef ADC InitStructure;
 GPIO InitTypeDef GPIO InitStructure;
  /* ADCCLK = PCLK2/1 = 24/2 = 12MHz*/
 RCC ADCCLKConfig(RCC PCLK2 Div2);
  /* povolime hodiny do portu */
 RCC APB2PeriphClockCmd(RCC APB2Periph GPIOC, ENABLE);
 GPIO InitStructure.GPIO Pin = GPIO Pin 4;
                                             // tady pripojime merene
napeti
 GPIO InitStructure.GPIO Speed = GPIO Speed 50MHz;
  GPIO InitStructure.GPIO Mode = GPIO Mode AIN;
  GPIO Init(GPIOC, &GPIO InitStructure);
  /* povolime hodiny do A/D prevodniku */
 RCC APB2PeriphClockCmd(RCC APB2Periph ADC1, ENABLE);
  //ADC DeInit(ADC1);
  /* ADC1 Configuration -----
 __*/
 ADC_InitStructure.ADC Mode
                                           = ADC Mode Independent;
 ADC_InitStructure.ADC_ScanConvMode
                                           = DISABLE;
 ADC InitStructure.ADC ContinuousConvMode = DISABLE;
 ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
                                          = ADC DataAlign Right;
 ADC InitStructure.ADC DataAlign
 ADC InitStructure.ADC NbrOfChannel
                                          = 1;
 ADC Init( ADC1, &ADC InitStructure );
  /* ADC1 Regular Channel 14 */
 ADC RegularChannelConfig( ADC1, ADC Channel 14, 1,
ADC SampleTime 239Cycles5);
 //ADC RegularChannelConfig(ADC1, ADC Channel 14, 1,
ADC SampleTime 13Cycles5);
 /* Enable ADC1 external trigger conversion */
 ADC ExternalTrigConvCmd( ADC1, ENABLE );
 ADC Cmd(ADC1, ENABLE);
  /* Provedeme kalibraci prevodniku */
 /* Enable ADC1 reset calibration register */
 ADC ResetCalibration(ADC1);
  /* Check the end of ADC1 reset calibration register */
 while(ADC GetResetCalibrationStatus(ADC1));
  /* Start ADC1 calibration */
 ADC StartCalibration(ADC1);
 /* Check the end of ADC1 calibration */
 while(ADC GetCalibrationStatus(ADC1));
}
void DAC Inicializace(void)
  GPIO InitTypeDef GPIO InitStructure;
 DAC InitTypeDef
                            DAC InitStructure;
 /* GPIOA Periph clock enable */
 RCC APB2PeriphClockCmd (RCC APB2Periph GPIOA, ENABLE);
  /* DAC Periph clock enable */
 RCC APB1PeriphClockCmd(RCC APB1Periph DAC, ENABLE);
```

V.Váňa: CanSAT Book for Students – part 2



```
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4 | GPIO_Pin_5;
  GPIO InitStructure.GPIO Mode = GPIO Mode AIN;
  GPIO_Init(GPIOA, &GPIO_InitStructure);
  /* DAC channel1 Configuration */
  DAC InitStructure.DAC Trigger = DAC Trigger None;
             // zadny trigger, sami si budeme poustet signal na vystup
  DAC_InitStructure.DAC_WaveGeneration = DAC_WaveGeneration_None;
             // viz text k tomuto dilu - nepozadujeme sutomatickou
generaci!
 DAC InitStructure.DAC OutputBuffer = DAC OutputBuffer Enable;
             // kdyz nepovol9te output buffer a zatizite vystup, tak
napeti bude klesat!!
  DAC_Init(DAC_Channel_1, &DAC_InitStructure);
  DAC Init(DAC Channel 2, &DAC InitStructure);
  /* Enable DAC Channels */
  DAC_Cmd(DAC_Channel_1, ENABLE);
  DAC Cmd(DAC Channel 2, ENABLE);
}
uint16_t ADC1_Read(void)
{
  // Start the conversion
 ADC SoftwareStartConvCmd(ADC1, ENABLE);
  // Wait until conversion completion
 while(ADC GetFlagStatus(ADC1, ADC FLAG EOC) == RESET);
  // Get the conversion value
 return ADC GetConversionValue(ADC1);
}
void Delay( IO uint32 t nTick)
{
 for(; nTick != 0; nTick--);
}
void itoa(uint16 t n, int8 t s[])
{
    int i, sign;
    if ((sign = n) < 0) / * record sign */
                         /* make n positive */
       n = -n;
    i = 0;
              /* generate digits in reverse order */
    do {
       s[i++] = n % 10 + '0'; /* get next digit */
                                /* delete it */
    } while ((n /= 10) > 0);
    if (sign < 0)
       s[i++] = '-';
    s[i] = ' \setminus 0';
    reverse(s);
}
void reverse(int8 t s[])
  int i, j, k;
```

V.Váňa: CanSAT Book for Students – part 2



```
char c;
     k = strlen(s) -1;
    for (i = 0, j = k; i<j; i++, j--) {</pre>
        c = s[i];
        s[i] = s[j];
        s[j] = c;
    }
 }
void assert func(const char *file, int line, const char *func, const char
*failedexpr)
{
 while(1)
 { }
}
void assert(const char *file, int line, const char *failedexpr)
{
   __assert_func (file, line, NULL, failedexpr);
}
```

2.10.1 DAC (program11B) – generátor trojúhelníkového průběhu

Úpravou programu **program11** dostaneme **program11b** v němž číselné hodnoty, které budeme převádět na analogové získáme výpočtem z hodnoty sw čítače **Counter**, která se mění v rozsahu 0 až 100 tak, že při každém průchodu hlavní nekonečnou smyčkou se zvětší o jedničku, popř. se po dosažení hodnoty 100 vynuluje. Vypočítané hodnoty pro D/A konverzi jsou v rozsahu 0 až 4095.





A zdrojový kód main.c

```
/* DAC v STM32 VL Discovery - výstupní generované signaly na PA4 a PA5
** velikost ciselne hodnoty, kterou prevadime na analogovou hodnotu
** pocitame z hodnoty citace
** main.c
** program11b - upraveno podle MCU serialu - díl 19
** Vd for SPSE Jecna course
*/
#include <stddef.h>
#include "string.h"
#include "stm32f10x.h"
#include "STM32_Discovery.h"
TIM_TimeBaseInitTypeDef TIM TimeBaseStructureMili =
{36000,TIM CounterMode Up,0,0,0}; //musi byt !!!!!
GPIO InitTypeDef GPIO InitStructure;
uint16 t Hodnota=0;
  IO uint8 t c;
  IO uint16 t Perioda = 100; // budeme mit celkem 100 dilku na periodu
```

V.Váňa: CanSAT Book for Students – part 2


```
IO uint16 t Counter = 0;
                                // interni citac generatoru probehu
 IO uint16 t Hodnota DAC1 = 0;
 IO uint16 t Hodnota DAC2 = 0;
/* Private function prototypes -----*/
void GPIO Inicializace(void); // nastavení vstupne/vystupnich pinu na kitu
void Delay( IO uint32 t nTick);
//void delay ms(uint16 t time);
void DAC Inicializace(void);
void Generuj DAC(void);
void BliknutiLEDZelena(void);
int main(void)
{
 //delay ms(2);
 GPIO Inicializace();
 STM32 Discovery LEDOn(LED4); // LED4 - modra
 DAC Inicializace();
 Delay(0xAAAAA);
 STM32 Discovery LEDOff(LED4); // LED4 - modra
 while (1)
 {
       Generuj DAC();
       BliknutiLEDZelena(); //dva krat Delay(0x1FFFF);
 }
}
void GPIO Inicializace(void)
 RCC APB1PeriphClockCmd(RCC APB1Periph TIM4, ENABLE); //musi byt !!!!!!
 STM32 Discovery LEDInit(LED3);
 STM32_Discovery_LEDInit(LED4);
 STM32 Discovery PBInit (BUTTON USER, BUTTON MODE GPIO);
 STM32 Discovery LEDOff(LED3);
 STM32 Discovery LEDOff(LED4);
}
void BliknutiLEDZelena(void)
{
 STM32 Discovery LEDOn(LED3);
 Delay(0x1FFFF);
 STM32 Discovery LEDOff(LED3);
 Delay(0x1FFFF);
}
void DAC Inicializace(void)
  GPIO InitTypeDef GPIO InitStructure;
 DAC InitTypeDef DAC InitStructure;
 /* GPIOA Periph clock enable */
 RCC APB2PeriphClockCmd(RCC APB2Periph GPIOA, ENABLE);
 /* DAC Periph clock enable */
 RCC APB1PeriphClockCmd(RCC APB1Periph DAC, ENABLE);
```

V.Váňa: CanSAT Book for Students – part 2



```
GPIO InitStructure.GPIO Pin = GPIO Pin 4 | GPIO Pin 5;
  GPIO InitStructure.GPIO Mode = GPIO Mode AIN;
  GPIO Init(GPIOA, &GPIO InitStructure);
  /* DAC channel1 Configuration */
  DAC InitStructure.DAC Trigger = DAC Trigger None;
             // zadny trigger, sami si budeme poustet signal na vystup
  DAC InitStructure.DAC WaveGeneration = DAC WaveGeneration None;
  // viz text v MCU serialu - nepozadujeme sutomatickou generaci!
  DAC InitStructure.DAC OutputBuffer = DAC OutputBuffer Enable;
//kdyz nepovolite output buffer a zatizite vystup, tak napeti bude klesat!!
  DAC Init(DAC Channel 1, &DAC InitStructure);
  DAC Init(DAC Channel 2, &DAC InitStructure);
  /* Enable DAC Channels */
 DAC_Cmd(DAC_Channel_1, ENABLE);
 DAC_Cmd(DAC_Channel_2, ENABLE);
void Generuj DAC (void)
{
      STM32 Discovery LEDOff(LED4); // Vypneme LED4 - modrou
      if (Counter <= (Perioda /2))</pre>
        {
            Hodnota DAC1 = ((uint32 t)Counter)*2 * 0xFFF / Perioda;
//vysledek je v rozmezi 0 az 4095
           Hodnota DAC2 = 0xFFF - (((uint32 t)Counter)* 2 * 0xFFF /
Perioda);
        }
      else
        {
            Hodnota DAC1 = 0xFFF - (((uint32 t)Counter - (Perioda /2))*2 *
OxFFF / Perioda); //vysledek je v rozmezi 0 az 4095
            Hodnota DAC2 = (((uint32 t)Counter) - (Perioda /2))*2 * 0xFFF /
Perioda;
        }
      /* Set DAC Channel1 DAC DHR12R1 register */
      DAC SetChannel1Data(DAC Align 12b R, (uint32 t)Hodnota DAC1);
      /* Set DAC Channel2 DAC DHR12R1 register */
     DAC SetChannel2Data(DAC Align 12b R, (uint32 t)Hodnota DAC2);
     Counter++;
      if (Counter >= Perioda)
        {
            Counter = 0;
           STM32 Discovery LEDOn(LED4);
// zapneme LED4 - modra - signal zacatku periody
        }
```

void Delay(__IO uint32_t nTick)



```
{
  for(; nTick != 0; nTick--);
}
/*
void delay ms(uint16 t time)
{
 RCC APB1PeriphClockCmd(RCC APB1Periph TIM4, ENABLE);
 TIM TimeBaseStructureMili.TIM Period = ((time+1) * 2)-1;
 TIM TimeBaseInit(TIM4, &TIM_TimeBaseStructureMili);
 TIM SelectOnePulseMode(TIM4, TIM OPMode Single);
 TIM SetCounter(TIM4,2);
TIM Cmd(TIM4, ENABLE);
while (TIM GetCounter(TIM4)){};
TIM Cmd(TIM4, DISABLE);
}
*/
void __assert_func(const char *file, int line, const char *func, const char
*failedexpr)
{
 while(1)
 { }
}
void assert(const char *file, int line, const char *failedexpr)
{
    _assert_func (file, line, NULL, failedexpr);
}
```

Průběhy výstupních napětí na PA4 a PA5





2.10.2 DAC (program11c) – generátor trojúhelníkového průběhu 5Hz s využitím přerušení Program byl upraven podle 20.dílu MCU seriálu.



Project	ф 💌
🖃 🛅 program	11c
🗄 🔄 sour	ce
🗄 🖃 n	nain.c
±	TM32_Discovery.c
+ * s	tm32f10x_it.c
🗄 🚗 librar	ries
🛓 🗄 n	nisc.c
	tm32f10x_adc.c
 ⊕£] si	tm32f10x_dac.c
± s	tm32f10x_dma.c
	tm32f10x exti.c
	tm32f10x flash.c
 ⊕ s	tm32f10x gpio.c
	tm32f10x rcc.c
 □ =	tm32f10x tim.c
startu	ar
I I I s	tartup_stm32f10x_mc
	IS
the H c	ore cm3.c
	vstem_stm32f10x.c
•	II
Б Рг 🧑 Во.	{} Fu 0, Te
Build Output	

Dále main.c

```
/* DAC v STM32 VL Discovery – výstupní generované signaly na PA4 a PA5
** velikost ciselne hodnoty, kterou prevadime na analogovou hodnotu
** vyuzivane preruseni, jeho obsluhu jsme dali do stm32f10x.c !!!!!!!
** main.c
** program11c - upraveno podle MCU serialu - díl 20
** Vd for SPSE Jecna course
*/
#include <stddef.h>
#include "string.h"
#include "stm32f10x.h"
#include "STM32 Discovery.h"
GPIO_InitTypeDef GPIO_InitStructure;
IO uint8 t c;
/* Private function prototypes -----
                                                    _____
---*/
```



```
void GPIO Inicializace(void); // nastavení vstupne/vystupnich pinu na kitu
void Delay( IO uint32 t nTick);
void DAC Inicializace(void);
void Generuj DAC(void);
void BliknutiLEDZelena(void);
void NVIC Inicializace(void);
void TIM3 Inicializace(void);
void TIM3 IRQHandler(void);
int main(void)
{
  GPIO Inicializace();
  DAC Inicializace();
  NVIC_Inicializace();
  TIM3 Inicializace();
  STM32 Discovery LEDOn(LED4); // LED4 - modra
  Delay(0xAAAAA);
  STM32 Discovery LEDOff(LED4); // LED4 - modra
  while (1)
  {
        BliknutiLEDZelena(); //dva krat Delay(0x1FFFF);
  }
}
void GPIO Inicializace(void)
{
  //RCC APB1PeriphClockCmd(RCC APB1Periph TIM4, ENABLE); //musi byt !!!!!!
  STM32 Discovery LEDInit(LED3);
  STM32_Discovery_LEDInit(LED4);
  STM32 Discovery PBInit (BUTTON USER, BUTTON MODE GPIO);
  STM32 Discovery LEDOff(LED3);
  STM32 Discovery LEDOff(LED4);
}
void BliknutiLEDZelena(void)
{
 STM32 Discovery LEDOn(LED3);
 Delay(0x1FFFF);
 STM32 Discovery LEDOff(LED3);
 Delay(0x1FFFF);
}
void DAC Inicializace(void)
  GPIO InitTypeDef GPIO InitStructure;
                            DAC InitStructure;
 DAC InitTypeDef
  /* GPIOA Periph clock enable */
  RCC APB2PeriphClockCmd (RCC APB2Periph GPIOA, ENABLE);
  /* DAC Periph clock enable */
  RCC APB1PeriphClockCmd(RCC APB1Periph DAC, ENABLE);
```

V.Váňa: CanSAT Book for Students – part 2



```
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4 | GPIO_Pin_5;
 GPIO InitStructure.GPIO Mode = GPIO Mode AIN;
 GPIO Init(GPIOA, &GPIO InitStructure);
  /* DAC channel1 Configuration */
 DAC InitStructure.DAC Trigger = DAC Trigger None;
            // zadny trigger, sami si budeme poustet signal na vystup
 DAC_InitStructure.DAC_WaveGeneration = DAC_WaveGeneration_None;
             // viz text k tomuto dilu - nepozadujeme sutomatickou
generaci!
 DAC InitStructure.DAC OutputBuffer = DAC OutputBuffer Enable;
             // kdyz nepovol9te output buffer a zatizite vystup, tak
napeti bude klesat!!
 DAC Init(DAC Channel 1, &DAC InitStructure);
 DAC_Init(DAC_Channel 2, &DAC InitStructure);
  /* Enable DAC Channels */
 DAC_Cmd(DAC_Channel_1, ENABLE);
 DAC_Cmd(DAC_Channel_2, ENABLE);
}
void NVIC Inicializace(void)
 NVIC_InitTypeDef NVIC_InitStructure;
 /* Configure the NVIC Preemption Priority Bits */
 NVIC PriorityGroupConfig(NVIC PriorityGroup 1);
  /* Clear the SC EXTI IRQ Pendinpbg Bit */
 NVIC ClearPendingIRQ(TIM3 IRQn);
 NVIC InitStructure.NVIC IRQChannel = TIM3 IRQn;
 NVIC InitStructure.NVIC IRQChannelPreemptionPriority = 0;
 NVIC InitStructure.NVIC IRQChannelSubPriority = 0;
 NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
 NVIC Init(&NVIC InitStructure);
void TIM3 Inicializace(void)
 TIM TimeBaseInitTypeDef TIM TimeBaseStructure;
 RCC APB1PeriphClockCmd(RCC APB1Periph TIM3 , ENABLE);
                                                              // pustíme
hodiny
 /* -
                        _____
   TIM3 Configuration: Output Compare Toggle Mode:
   TIM3CLK = 24 MHz, Prescaler = 24000, TIM3 counter clock = 1kHz
 __*/
  /* Time base configuration */
 TIM TimeBaseStructure.TIM Period = 1; //nas pilovy prubeh by mel mit
f=25Hz
 TIM TimeBaseStructure.TIM Prescaler = 24000;
 TIM TimeBaseStructure.TIM ClockDivision = 0;
 TIM TimeBaseStructure.TIM CounterMode = TIM CounterMode Down;
TIM TimeBaseStructure.TIM RepetitionCounter = 0;
```

151

V.Váňa: CanSAT Book for Students – part 2



```
TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);
  /* TIM IT enable */
 TIM ITConfig(TIM3, TIM IT Update, ENABLE);
 /* TIM enable counter */
 TIM_Cmd(TIM3, ENABLE);
}
void Delay( IO uint32 t nTick)
{
 for(; nTick != 0; nTick--);
void assert func(const char *file, int line, const char *func, const char
*failedexpr)
{
 while(1)
 { }
}
void assert(const char *file, int line, const char *failedexpr)
{
   assert func (file, line, NULL, failedexpr);
```

A ještě **stm32f10x_it.c** který obsahuje obsluhu přerušení provádějící D/A převod.

```
/**
 * @file
           Project/STM32F10x StdPeriph Template/stm32f10x it.c
 * @version V3.3.0 doplnene o DA prevod s prerusenim
 */
#include "stm32f10x it.h"
void NMI Handler(void)
{
void HardFault Handler(void)
 /* Go to infinite loop when Hard Fault exception occurs */
 while (1)
  {
  }
}
void MemManage Handler(void)
  /* Go to infinite loop when Memory Manage exception occurs */
 while (1)
```



}

Inovace ve vzdělávání na území hl.m. Prahy 2011, proj.č.2046

```
void BusFault Handler(void)
 /* Go to infinite loop when Bus Fault exception occurs */
 while (1)
 {
 }
}
void UsageFault Handler(void)
{
 /* Go to infinite loop when Usage Fault exception occurs */
 while (1)
 {
 }
}
void SVC Handler(void)
{
}
void DebugMon Handler(void)
{
}
void PendSV Handler(void)
{
}
void SysTick Handler(void)
{
}
****/
/*
               STM32F10x Peripherals Interrupt Handlers
*/
/* Add here the Interrupt Handler for the used peripheral(s) (PPP), for
the */
/* available peripheral interrupt handler's name please refer to the
startup */
/* file (startup stm32f10x xx.s).
*/
/****
            ****/
void TIM3 IRQHandler(void)
// původní funkce void Generuj_DAC(void)
```

V.Váňa: CanSAT Book for Students – part 2 153



```
{
                                        // interni citac generatoru
 static uint16 t Counter = 0;
probehu
 __IO uint16_t Hodnota_DAC1 = 0;
  __IO uint16_t Hodnota DAC2 = 0;
 if (TIM GetITStatus(TIM3, TIM IT Update) != RESET)
  {
    /* Clear TIM3 update interrupt pending bit*/
   TIM ClearITPendingBit(TIM3, TIM IT Update);
   // ----- zde zacina puvodni fce Generuj DAC
     STM32 Discovery LEDOff(LED4); // Vypneme LED4 - modrou
     if (Counter <= (Perioda /2))
       {
           Hodnota DAC1 = ((uint32 t)Counter)*2 * 0xFFF / Perioda;
           Hodnota DAC2 = 0xFFF - (((uint32 t)Counter)* 2 * 0xFFF /
Perioda);
       }
     else
       {
           Hodnota_DAC1 = 0xFFF - (((uint32_t)Counter - (Perioda /2))*2 *
0xFFF / Perioda);
          Hodnota DAC2 = (((uint32 t)Counter) - (Perioda /2))*2 * 0xFFF /
Perioda;
       }
     /* Set DAC Channel1 DAC DHR12R1 register */
     DAC SetChannel1Data(DAC Align 12b R, (uint32 t)Hodnota DAC1);
     /* Set DAC Channel2 DAC DHR12R1 register */
     DAC SetChannel2Data(DAC Align 12b R, (uint32 t)Hodnota DAC2);
     Counter++;
     if (Counter >= Perioda)
       {
           Counter = 0;
           STM32 Discovery LEDOn(LED4); // zapneme LED4 - modra - signal
zacatku periody
       }
     // ----- zde konci puvodni fce Generuj DAC
  }
}
```





2.10.3 DAC (program11d) – generátor trojúhelníkového a sinusového signálu s využitím DMA Program byl upraven podle 22.dílu MCU seriálu.

```
/* DAC v STM32 VL Discovery – výstupní generované signaly na PA4 a PA5
** velikost ciselne hodnoty, kterou prevadime na analogovou hodnotu
** vyuzivame DMA
** main.c
** program11d - upraveno podle MCU serialu - dí1 22
** Vd for SPSE Jecna course
*/
#include <stddef.h>
#include "string.h"
#include "stm32f10x.h"
#include "STM32 Discovery.h"
GPIO_InitTypeDef GPIO_InitStructure;
DMA InitTypeDef
                            DMA InitStructure;
TIM TimeBaseInitTypeDef
                            TIM TimeBaseStructure;
DAC InitTypeDef
                            DAC InitStructure;
#define DAC DHR12RD Address
                                 0x40007420
V.Váňa: CanSAT Book for Students – part 2
                                     155
                                                        ARM STM32 programming
```



```
const uint16 t Pila[16] = { 0, 273, 546, 819, 1092, 1365, 1638, 1911,
2184, 2457, 2730, 3003, 3276, 3549, 3822, 4095};
const uint16 t Sinus[16] = {2048, 2831, 3495, 3939, 4095, 3939, 3495, 2831,
                         0, 156, 600, 1264};
2048, 1264, 600, 156,
                          // do této promněnné nastrkáme data pro DAC
uint32 t Dva kanaly[16];
 IO uint8_t Pozice, Stisk = 0;
 IO int16 t Zmena=0;
IO uint32 t Perioda=100;
/* Private function prototypes -----*/
void GPIO Inicializace(void); // nastavení vstupne/vystupnich pinu na kitu
void Delay( IO uint32 t nTick);
void BliknutiLEDZelena(void);
void TIM2 Inicializace(void);
int main(void)
{
 GPIO Inicializace();
 TIM2 Inicializace();
 /* Enable DAC clock */
 RCC APB1PeriphClockCmd(RCC APB1Periph DAC, ENABLE);
  /* DAC channel1 Configuration */
 DAC InitStructure.DAC Trigger = DAC Trigger T2 TRGO;
 // tady bude trigger, využijeme signál od Timeru
 DAC InitStructure.DAC WaveGeneration = DAC WaveGeneration None;
 // nepozadujeme automatickou generaci!
 DAC InitStructure.DAC OutputBuffer = DAC OutputBuffer Disable;
 // kdyz nepovolite output buffer a zatizite vystup, tak napeti bude
klesat!!
 DAC Init(DAC Channel 1, &DAC InitStructure);
  /* DAC channel2 Configuration */
 DAC Init(DAC Channel 2, &DAC InitStructure);
 STM32_Discovery_LEDOn(LED4); // LED4 - modra
 Delay(0xAAAAA);
 STM32 Discovery LEDOff(LED4); // LED4 - modra
  /* naplnime buffer daty */
 for (Pozice = 0; Pozice < 16; Pozice++)</pre>
  {
   Dva kanaly[Pozice] = ( Sinus[Pozice] << 16) + (Pila[Pozice]);</pre>
  /* v hornim halfwordu je Sinus = PA5, ve spodnim je Pila = PA4 */
```

V.Váňa: CanSAT Book for Students – part 2



```
/* Enable DMA clock */
 RCC AHBPeriphClockCmd(RCC AHBPeriph DMA1, ENABLE);
  /* DMA1 channel4 configuration */
 DMA DeInit(DMA1 Channel4);
 DMA InitStructure.DMA PeripheralBaseAddr = DAC DHR12RD Address;
  DMA InitStructure.DMA MemoryBaseAddr = (uint32 t)&Dva kanaly;
  DMA InitStructure.DMA DIR = DMA DIR PeripheralDST;
  DMA InitStructure.DMA BufferSize = 16;
  DMA InitStructure.DMA PeripheralInc = DMA PeripheralInc Disable;
     InitStructure.DMA_MemoryInc = DMA_MemoryInc Enable;
  DMA
  DMA
     InitStructure.DMA PeripheralDataSize = DMA PeripheralDataSize Word;
  DMA
     InitStructure.DMA MemoryDataSize = DMA MemoryDataSize Word;
     InitStructure.DMA Mode = DMA Mode Circular;
  DMA
  DMA
     InitStructure.DMA Priority = DMA Priority High;
  DMA
     InitStructure.DMA M2M = DMA M2M Disable;
 DMA Init(DMA1 Channel4, &DMA InitStructure);
  // uz tam máme data, muzeme pustit prenos
  /* Enable DMA1 Channel4 */
 DMA Cmd(DMA1 Channel4, ENABLE);
  /* Enable DAC Channels */
  DAC Cmd(DAC Channel 1, ENABLE);
 DAC Cmd(DAC Channel 2, ENABLE);
  /* Enable DMA for DAC Channel2 */
 DAC DMACmd(DAC Channel 2, ENABLE);
  /* TIM2 enable counter */
 TIM Cmd(TIM2, ENABLE);
  Zmena = -5;
 while (1)
  {
        //BliknutiLEDZelena(); //dva krat Delay(0x1FFFF);
        if(0 != STM32 Discovery PBGetState(BUTTON USER))
    { // je stisknuto tlacitko
      Stisk = 1; // nastavime priznak stisku tlacitka
           if (Perioda <= 150)</pre>
            {
                  Perioda += Zmena;
            } else
            {
                 Perioda += 10*Zmena; // pro nizsi frekvence urychlime
zmenu
            }
            if (Perioda < 1)
            {
                  STM32 Discovery LEDOn(LED4); // rozsvitime modrou kdyz
jsme na konci
                  Perioda = 1;
            }
```

157



```
if (Perioda > 1000)
            {
                  STM32 Discovery LEDOn(LED4); // rozsvitime modrou kdyz
jsme na konci
                  Perioda = 1000;
            }
            STM32 Discovery LEDOn(LED3); // rozsvitime zelenou ze je
stlacene tlacitko
            /* zmenime nastaveni frekvence */
            TIM TimeBaseStructure.TIM Prescaler = (uint16 t) 1200;
                                                                        11
pro 20Hz-20kHz!!
            TIM TimeBaseStructure.TIM ClockDivision = 0x0;
            TIM TimeBaseStructure.TIM CounterMode = TIM CounterMode Down;
            TIM TimeBaseStructure.TIM Period = (uint16 t) Perioda;
            TIM TimeBaseInit(TIM2, &TIM TimeBaseStructure);
            Delay(0x5FFFF);
      } else
            // neni stisknuto tlacitko
      {
            if (Stisk ==1)
            {
                  // prave doslo k uvolneni tlacitka, zhasneme ledky
                  Stisk = 0;
                  Zmena = - Zmena;
                  STM32_Discovery_LEDOff(LED3);
                  STM32 Discovery LEDOff(LED4);
            }
      }
 }
}
void GPIO Inicializace (void)
 GPIO InitTypeDef GPIO InitStructure;
 STM32 Discovery LEDInit(LED3);
 STM32 Discovery LEDInit(LED4);
 STM32 Discovery PBInit (BUTTON USER, BUTTON MODE GPIO);
 STM32 Discovery LEDOff(LED3);
   STM32 Discovery LEDOff(LED4);
  /* GPIOA Periph clock enable */
 RCC APB2PeriphClockCmd(RCC APB2Periph GPIOA, ENABLE);
  /* Once the DAC channel is enabled, the corresponding GPIO pin is
automatically connected to the DAC converter. In order to avoid parasitic
consumption, the GPIO pin should be configured in analog */
 GPIO InitStructure.GPIO Pin = GPIO Pin 4 | GPIO Pin 5;
 GPIO InitStructure.GPIO Mode = GPIO Mode AIN;
 GPIO Init(GPIOA, &GPIO InitStructure);
void BliknutiLEDZelena(void)
V.Váňa: CanSAT Book for Students – part 2
                                      158
                                                         ARM STM32 programming
```



```
STM32 Discovery LEDOn(LED3);
 Delay(0x1FFFF);
 STM32 Discovery_LEDOff(LED3);
 Delay(0x1FFFF);
}
void TIM2 Inicializace(void)
{
      /* TIM2 Periph clock enable */
     RCC APB1PeriphClockCmd(RCC APB1Periph TIM2, ENABLE);
      /* TIM2 Configuration */
     TIM TimeBaseStructInit(&TIM TimeBaseStructure);
     TIM TimeBaseStructure.TIM Period = 0x500; // 0.5s!! Pro velmi pomaly
prubeh
     TIM TimeBaseStructure.TIM Prescaler = (uint16 t) 0x24000;
     TIM TimeBaseStructure.TIM ClockDivision = 0x0;
      TIM TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
     TIM TimeBaseInit(TIM2, &TIM TimeBaseStructure);
      /* TIM2 TRGO selection */
     TIM SelectOutputTrigger(TIM2, TIM TRGOSource Update);
}
void Delay( IO uint32 t nTick)
{
 for(; nTick != 0; nTick--);
}
void assert func (const char *file, int line, const char *func, const char
*failedexpr)
{
 while(1)
 { }
}
void assert(const char *file, int line, const char *failedexpr)
{
    assert func (file, line, NULL, failedexpr);
}
```





2.10.4 DAC (program11e) – generátor trojúhelníkového a sinusového signálu pevného kmitočtu s využitím DMA

Jedná se o vlastně o předchozí program, který se od předchozího liší jen jednodušším obsahem nekonečné hlavní smyčky. Oproti předchozímu programu jsem zjednodušil tento obsah odstraněním kódu na změnu hodnoty proměnné Perioda. Tím generujeme sinusový a trojúhelníkový signál o stálém kmitočtu. Můžeme tak otestovat jak závisí kmitočet výstupního signálu na hodnotě Perioda a na dělícím poměru **TIM_TimeBaseStructure.TIM_Prescaler**.

```
while (1)
  {
/* zmenime nastaveni frekvence */
/* 12Hz pri TIM Prescaler = 1200 */
                                           //Perioda = 100;
/* 113Hz pri TIM_Prescaler = 1200 */
                                             //Perioda = 10;
/* 224z pri TIM Prescaler = 1200 */
                                     //Perioda = 1;
/* 126Hz
            pri TIM Prescaler = 120 */ //Perioda = 100;
/* 1.13kHz
            pri TIM Prescaler = 120 */
                                           //Perioda = 10;
/* 6.24 kHz pri TIM Prescaler = 120*/
                                             //Perioda = 1;
/* 243Hz
            pri TIM Prescaler = 60 */
                                             //Perioda = 100;
/* 2.2kHz pri TIM_Prescaler = 60 */
                                             //Perioda = 10;
/* 12.2 kHz pri TIM_Prescaler = 60 */
                                             //Perioda = 1;
```

V.Váňa: CanSAT Book for Students – part 2 160







2.11 Assembler – blikáme zelenou LED na PC9 (program12)



C:\ARM_priklady\program12\program12.uvproj - u	sion4	
File Edit View Project Flash Debug Periphe	als Tools SVCS Window Help	
📋 🗋 🗃 🛃 🔺 🔉 🛍 🏙 🔊 🗠 🔲 🖛	🅐 魯 魯 魯 🛊 詳 // /版 🎯 🕢 🕞 🖳 🛷 🍳 🌢 🔹 🔗 🚓 🕅	3
😂 🎬 🕮 🥪 🧮 🙀 🛛 program12	💽 🔊 📥 🗟	
Project 🛛 🗜 🔯	piny_asm.s 📓 main.c	→ ×
Program12 Program12	<pre>01 /* STM32 VL Discovery Library 02 * program12 - blikač s zelenou LED na PC9 pouzivajici asm 03 * ver 1.0 04 * Vd for SPSE Jecna 2011 lessons 05 */ 06 07 /* Includes</pre>	*/
Project 🔞 Books {} Functions 0, Templates		<u>•</u>
Build Output		4 📦
assembling startup_stm32f10x_md_vl.s compiling core_cm3.c compiling system_stm32f10x.c linking Program Size: Code=4296 RO-data=348 H FromELF: creating hex file ".\output\program12.axf" - 0 Error(s)	 W-data=52 ZI-data=1636 , O Warning(s).	* •
	ST-Link Debugger L:2 C:60	CAP NUM SCRL OVR R/W



Project	4
Image: source Imain.c Imain.c	ß
E Project Books {} Functions 0, Te	emplates

Obsah souboru piny_asm.s

; specifikace oblasti	
AREA .text , CODE, R	EADONLY
; zapis hodnoty bitu PC	9
write_pc9 PROC	
	EXPORT write_pc9 ;
; konstanty	
	LDR r1, =0x40011010
	MOV r2, #0x0200
; podle predaneho arg	umentu nastavim adresu SET nebo RESET registru
	CMP r0, #0
	ADDEQ r1, r1, #0x04
; zapisu data do ovlada	aciho registru portu C
	STR r2, [r1]
; odchod z funkce	



BX lr ENDP

29
EXPORT read_pc9;
iho registru portu C
LDR r2, =0x4001100c
LDR r1, [r2]
ANDS r1, r1, #0x0200
MOVEQ r0,#0x00
MOVNE r0, #0x01
BX lr
ENDP

Pozn. V výše uvedeném asm kódu jsou (občas) důležité i mezery, což je rozdíl od většiny vyšších jazyků

Hodnoty konstant pro jednotlivé piny najdeme např. v souboru stm32f10x_gpio.h

#define GPIO_Pin_0	((uint16_t)0x0001)
#define GPIO_Pin_1	((uint16_t)0x0002)
#define GPIO_Pin_2	((uint16_t)0x0004) /*!< Pin 2 selected */
#define GPIO_Pin_3	((uint16_t)0x0008) /*!< Pin 3 selected */
#define GPIO_Pin_4	((uint16_t)0x0010)
#define GPIO_Pin_5	((uint16_t)0x0020)
#define GPIO_Pin_6	((uint16_t)0x0040)
#define GPIO_Pin_7	((uint16_t)0x0080)
#define GPIO_Pin_8	((uint16_t)0x0100)
#define GPIO_Pin_9	((uint16_t)0x0200)
#define GPIO_Pin_10	((uint16_t)0x0400)
#define GPIO_Pin_11	((uint16_t)0x0800)
#define GPIO_Pin_12	((uint16_t)0x1000) /*!< Pin 12 selected */
#define GPIO_Pin_13	((uint16_t)0x2000) /*!< Pin 13 selected */
#define GPIO_Pin_14	((uint16_t)0x4000) /*!< Pin 14 selected */
#define GPIO Pin 15	((uint16 t)0x8000) /*!< Pin 15 selected */

V.Váňa: CanSAT Book for Students – part 2



Obsah souboru main.c

```
/* STM32 VL Discovery Library
* program12 – blikač s zelenou LED na PC9 pouzivajici asm
* ver 1.0
 * Vd for SPSE Jecna 2011 lessons
 */
#include "stm32f10x.h"
#include "STM32vldiscovery.h"
void GPIO Inicialization(void);
void Delay(__IO uint32_t nTick);
void write pc9(u8 value);
u8 read pc9(void);
int main(void)
{
  GPIO Inicialization();
  while (1)
  {
      /* prepneme stav pinu PC9 */
            write_pc9(1 - read_pc9());
            /* cekame */
            Delay(0x5ffff);
            Delay(0x5ffff);
            Delay(0x5ffff);
            Delay(0x5ffff);
            Delay(0x5ffff);
            Delay(0x5ffff);
  }
}
void GPIO Inicialization(void)
{
  STM32vldiscovery LEDInit(LED3);
 STM32vldiscovery LEDInit(LED4);
 STM32vldiscovery PBInit (BUTTON USER, BUTTON MODE GPIO);
 STM32vldiscovery LEDOff(LED3);
  STM32vldiscovery LEDOff(LED4);
}
void Delay( IO uint32 t nTick)
for(; nTick != 0; nTick--);
V.Váňa: CanSAT Book for Students – part 2
                                                          ARM STM32 programming
                                      165
```



2.12.1 Blikač s LED na PC7 (program12A)



Obsah souboru piny_asm.s

```
; specifikace oblasti
AREA |.text|, CODE, READONLY
; zapis hodnoty bitu PC7
write_pc7
               PROC
                      EXPORT write_pc7 ;
; konstanty
                      LDR r1, =0x40011010
                      MOV r2, #0x0080
; podle predaneho argumentu nastavim adresu SET nebo RESET registru
                      CMP r0, #0
                      ADDEQ r1, r1, #0x04
; zapisu data do ovladaciho registru portu C
                      STR r2, [r1]
; odchod z funkce
                                               166
```



BX lr ENDP

```
; cteni hodnoty bitu PC7
read_pc7 PROC
EXPORT read_pc7;
; nactu obsah vystupniho registru portu C
LDR r2, =0x4001100c
LDR r1, [r2]
; vyctu stav pinu 7
ANDS r1, r1, #0x0080
MOVEQ r0,#0x00
MOVNE r0, #0x01
BX lr
ENDP
```

END

Obsah souboru main.c

```
/* STM32 VL Discovery Library
 * program12a - ukazka pouziti asembleru
 * ver 1.0
 * Vd for SPSE Jecna 2011 lessons
 */
#include "stm32f10x.h"
//#define HSE //Pouziti externiho oscilatoru
#define HSI //Pouziti interniho oscilatoru
/* funkcni prototypy */
void RCC Configuration(void);
void GPIO Configuration(void);
void Delay( IO uint32 t nTick);
void write pc7(u8 value);
u8 read pc\overline{7} (void);
int main(void)
{
      /* konfigurace zdroju hodinoveho signalu */
      RCC Configuration();
      /* konfigurace I/O portu */
      GPIO Configuration();
      /* blikaci smycka */
  while (1)
  {
      /* prepneme stav pinu PC7 */
```

V.Váňa: CanSAT Book for Students – part 2



```
write pc7(1 - read pc7());
            /* cekame */
            Delay(0x5ffff);
            Delay(0x5ffff);
            Delay(0x5ffff);
            Delay(0x5ffff);
            Delay(0x5ffff);
            Delay(0x5ffff);
  }
}
/* nastaveni zdroju hodinoveho signalu (HSE) */
void RCC Configuration(void) {
#ifdef HSE
      ErrorStatus HSEStartUpStatus;
      /* reset nastaveni */
     RCC DeInit();
    /* aktivace signalu HSE (External High Speed oscillator) */
      RCC_HSEConfig(RCC_HSE_ON);
      /* kontrola stability oscilatoru */
      HSEStartUpStatus = RCC WaitForHSEStartUp();
      if (HSEStartUpStatus == SUCCESS) {
            /*Nastaveni delicek HS*/
            RCC HCLKConfig(RCC SYSCLK Div1); //HCLK = 24 MHz, AHB
            RCC_PCLK2Config(RCC_HCLK_Div1); //APB1 = 24 MHz
            RCC PCLK1Config(RCC HCLK Div1); //APB2 = 24 MHz
            /* nastaveni PLL */
            RCC PLLConfig(RCC PLLSource PREDIV1, RCC PLLMul 4); //PLLCLK =
24 MHz
      }
#endif
#ifdef HSI
      /* reset nastaveni */
      RCC DeInit();
      /* aktivace signalu HSI (Internal High Speed oscillator) */
      RCC HSICmd(ENABLE);
      while (RCC GetFlagStatus(RCC FLAG HSIRDY) == RESET);
      /* nastaveni delicek HS */
      RCC HCLKConfig(RCC SYSCLK Div1); //HCLK = 24 MHz, AHB
      RCC PCLK1Config(RCC HCLK Div1); //APB1 = 24 MHz
      RCC PCLK2Config(RCC HCLK Div1); //APB2 = 24 MHz
      /* nastaveni PLL */
      RCC PLLConfig(RCC PLLSource HSI Div2, RCC PLLMul 8); //PLLCLK = 24
MHz
#endif
      /* nastaveni FLASH */
      FLASH SetLatency (FLASH Latency 2);
      FLASH PrefetchBufferCmd(FLASH PrefetchBuffer Enable);
      RCC PLLCmd (ENABLE);
      while (RCC GetFlagStatus(RCC FLAG PLLRDY) == RESET);
      /*Zhroj hodin pro AHB*/
      RCC SYSCLKConfig(RCC SYSCLKSource PLLCLK);
      while (RCC GetSYSCLKSource() != 0x08);
```

V.Váňa: CanSAT Book for Students – part 2



```
/* aktivace zdroje HS pro periferie */
      RCC APB2PeriphClockCmd(RCC APB2Periph GPIOC , ENABLE);
}
 /* konfigurace I/O portu */
void GPIO Configuration(void) {
      /* struktura s informacemi o konfiguraci pinu */
      GPIO InitTypeDef GPIO InitStructure;
      /* konfigurujeme pin PC7 jako output push-pull */
      GPIO InitStructure.GPIO Pin = GPIO Pin 7;
      GPIO InitStructure.GPIO Speed = GPIO Speed 10MHz;
      GPIO InitStructure.GPIO Mode = GPIO Mode Out PP;
      GPIO Init(GPIOC, & GPIO InitStructure);
}
void Delay( IO uint32 t nTick)
{
  for(; nTick != 0; nTick--);
}
```

2.13.1 I2C EEPROM AT24C02 (program13)

Z hlediska univerzálnosti použití má velký praktický význam sběrnice I2C (Inter-Integrated-Circuit Bus) vytvořená firmou Philips. Podporu této sběrnice však nalezneme i u řady integrovaných obvodů jiných výrobců. Sběrnice je tvořena dvojicí vodičů, na kterých je v klidovém stavu přes pull up odpory nastavena logická jednička, viz.obr.:



Jeden z vodičů, značený SCL, přenáší hodinový signál. Druhý, značený SDA, slouží k synchronnímu přenosu dat:





Budiče s otevřeným kolektorem umožňují, aby sběrnice byla využívána skupinou rovnoprávných řadičů sběrnice I2C (multimaster konfigurace). Odpovídající arbitrážní protokol , který řeší přidělení sběrnice při současném požadavku více řadičů sběrnice I2C je popsán v dokumentaci k tomuto protokolu. V následujících dvou programech budeme uvažovat nejčastější případ, kdy sběrnici bude řídit jen jeden řadič sběrnice. Bude jím mikrořadič STM32 řízený našim programem a využívající jeho knihovní funkce pro I2C

Komunikaci zahajuje řadič prvkem START – sestupná hrana signálu SDA při jedničce signálu SCL. Dále je vysílán osmibitový znak po vodiči SDA počínajíc nejvýznamějším bitem. Vyslaný znak je potvrzován příjmačem stažením signálu SCL na úroveň nula – signál ACK (acknowledge).

Norma I2C definuje i formát přenášených dat, jejich potvrzování a předávání řízení mezi účastníky komunikace. Prvním znakem, vyslaným řadičem, je vždy řídící slovo. Jeho sedm nejvýznamějších bitů A7, A6 … A1 slouží k specifikaci podřízeného obvodu, s nímž bude probíhat komunikace. Nejméně významný bit A0 řídícího slova pak určuje směr následující komunikace, tj. směr druhého, popř. dalších slov. A0 = 0 znamená směr přenosu z řadiče do podřízeného obvodu, A0 = 1 obrácený směr přenosu. První čtyři (popř. další) bity řídícího slova A7, A6, A5, A4 jsou určeny typem obvodu , Další bity A3, A2,A1 jsou nastaveny adresačními vývody podřízeného obvodu. Těchto podřízených obvodů tedy může být ke sběrnici I2C připojeno až $2^3 = 8$.

Pro aplikace v oblasti spotřební elektroniky a telekomunikací vyrábějí firmy Philips a Siemens mnoho obvodů řízených sběrnicí I2C. Pro ilustraci si některé uvedeme:

		A7	A6	A5	A4	A3	A2	A1
PCF8574	Osmibitový vstup/výstup	0	1	0	0	а	a	a
PCF8577	64-segmentový řadič LCD	0	1	1	1	0	1	0
PCF8578	Řadič bodového LCD	0	1	1	1	1	0	a
PCF8582A	Paměť EEPROM 256 x 8	1	0	1	0	а	a	a
PCF8583	Hodiny/kalendář a paměť RAM	1	0	1	0	0	0	a
PCF8591	Osmibitový A/D, D/A převodník	1	0	0	1	а	a	a
SAA3028	Infra přijímač kódu RD5	0	1	0	0	1	1	0
TDA8444	Osminásobný D/A převodník	0	1	0	0	а	a	a
TSA5055	Syntezátor kmitočtu – PLL do 2,6 GHz	1	1	0	0	0	a	a

Protože obvodů řízených I2C je velký počet, není možné, aby každý takový obvod měl v této tabulce svoji řádku a proto některé obvody mají stejné hodnoty bitů A7, A6, A5, A4 . Jde však přitom obvody s obdobnou funkcí. Např. všechny paměti EEPROM mají hodnoty těchto bitů 1010, takže např. AT24C01 či 24C02 jsou řízeny a programovány stejně, jako PCF8582A. Tato paměť nám také poslouží v následujícím příkladě k podrobnějšímu vysvětlení funkce protokolu I2C i snadnosti naprogramování

V.Váňa: CanSAT Book for Students – part 2



mikrořadiče STM32 jako řadiče I2C. Nejprve si ukážeme jednoduchý program umožňující naprogramovat několik byte v paměti EEPROM 24C01 či 24C02, Ta bude s MCU STM32 propojena dvěma vodiči, např.:

// I2C EEPROM pripojime k PB
//SCL bude na PB6 DATA na PB7 AT24C02

Tak jako v předchozích příkladech uvedeme jen základní informace, tj. zdrojový kód **main.c** a strukturu projektu:

C:\ARM_priklady\keil\program13\priklad13.uvproj - µ\	/ision4									
File Edit View Project Flash Debug Peripheral	ls Tools S	SVCS Window Help								
	19 19 19	18月 連連 北京 29			•	🗟 🌾	• @	•	8 🏩 🔲 🔧	
🔗 🕮 🕮 🥔 🚉 🙀 priklad13 🛛 🚦	- 🌾 🛔	8								5
Project 🏨 🔛	📩 mai	in.c stm32f10x_i2c.h								▼ ×
	001 /* 002 * 003 * 004 * 005 * 007 */ 008 * 007 */ 008 * 010 #ii 010 #ii 010 #ii 010 #ii 010 #ii 010 #ii 011 #ii 013 * 014 /* 015 GP! 015 GP! 016 * 017 /* 018 #dd 020 021 022 // 022 // 022 // 022 // 022 // 022 // 022 // 022 // 024 // 027 // 027 // 027 // 027 //	I2C komunikace s EEPR dile 29. dilu serialu d priklad 13 mm Vd pro SPSE Jecna Leci nolude "stm32f10x.h" nolude "stm32f10x_12c nolude "STM32v1discove Private typedef	OM (AT24 o STM32 ain.c tures tures ary.h" InitStru DRESS OA A na PB a A7 A6 0 1 1 0 1 0 1 0	CC02)	:; //prc 24C01 =0 je 24C01 0 4 0 4 0 4 0 4 0 4 0 4 0 4 1 4	vrite 2 A3 A a a a a a a a a	je to do d a a a a	o EEPRO bobvodu, l napr. napr. napr.	M Al=1 je read) PCF8574 PCF8582 PCF8583 PCF8591	*/ */ */
<pre>compiling stm32f10x_i2c.c linking Program Size: Code=6192 RO-data=348 RW FromELF: creating hex file ".\output\priklad13.axf" - 0 Error(s),</pre>	-data=60 0 Warnin	ZI-data=1636								T III
<			Si	mulatio.	<u>r</u>				L:4 C:23	CAP NUM SCRL C



Project	Д.	X
🖃 🚵 priklad13		_
🗄 📇 source		
🗄 🖬 main.c		
庄 🔚 stm32f10x_it.c		
Image: Image		
🗄 😋 CMSIS		
🛓 🔛 core_cm3.c		
🗄 📇 startup		
startup_stm32f10x_md	vl.s	
🗄 🦳 libraries		
195 - 197 - 197 - 197 - 197 - 197 - 197 - 197 - 197 - 197 - 197 - 197 - 197 - 197 - 197 - 197 - 197 - 197 - 197		
Project Books {} Functions	0 Templates	1

```
/*
* I2C komunikace s EEPROM (AT24C02)
* dle 29. dilu serialu o STM32
 priklad 13
              main.c
* Vd pro SPSE Jecna Lectures
*
*/
#include "stm32f10x.h"
#include "stm32f10x i2c.h"
#include "STM32vldiscovery.h"
/* Private typedef ----
GPIO InitTypeDef GPIO InitStructure;
/* Private define -----
#define EEPROM_SLAVE_ADDRESS 0xA0 //protoze je to EEPROM
//SCL bude na PB6 DATA na PB7 AT24C02
```

172

V.Váňa: CanSAT Book for Students – part 2



```
//Typ obvodu A7 A6 A5 A4 A3 A2 A1
//Sbitovy vstup/vystup 0 1 0 0 a
PCF8574
                     1
                          0
                                1
                                     0
//EEPROM
                                           а
                                                а
                                                      а
                                                            napr.
PCF8582
                    1
                                     0
                                           0
//Hodiny/kalendar
                          0
                                1
                                                0
                                                      а
                                                            napr.
PCF8583
                          0
//8bitovy A/D a D/A
                     1
                               0
                                     1
                                           а
                                                а
                                                      а
                                                            napr.
PCF8591
//PLL
                     1
                          1
                               0
                                     0
                                          0
                                                a
                                                     а
                                                            napr.
TSA5055
/* Private variables -----*/
uint8 t hodnota=0;
void Delay(__IO uint32_t nTick);
void GPIO Inicializace(void); // nastavení vstupne/vystupnich pinu na kitu
void I2C Inicializace(void); // nastavení I2C
uint8 t Read EEPROM(uint8 t adresa); // cteni
void Write_EEPROM(uint8_t adresa, uint8_t data); // zapis
void Ready EEPROM(void);
                                    // polling potvrzeni
void bliknutiLEDMODRA(void); //
void bliknutiLEDZELENA(void);
void bliknutiLEDZELENA(void);
                                      11
int main(void)
{
  GPIO Inicializace();
  I2C Inicializace();
  // blikneme modrou jako priznak startu
 bliknutiLEDMODRA();
 /* pouzita EEPROM AT24C02 - 256 bajtu x 8 bitu */
 hodnota = Read EEPROM(0x00);
 hodnota = Read EEPROM(0x01);
 hodnota = Read EEPROM(0x02);
 hodnota = Read EEPROM(0x03);
 hodnota = Read EEPROM(0x04);
 Write EEPROM(0x00, 0x01);
 Ready EEPROM();
 bliknutiLEDZELENA();
 Write EEPROM(0x01, 0x23);
 Ready EEPROM();
 bliknutiLEDMODRA();
 Write EEPROM(0x02, 0x45);
 Ready EEPROM();
 bliknutiLEDZELENA();
 Write EEPROM(0x03, 0x67);
 Ready EEPROM();
 bliknutiLEDMODRA();
 hodnota = Read\_EEPROM(0x00);
```

V.Váňa: CanSAT Book for Students – part 2



```
hodnota = Read EEPROM(0x01);
 hodnota = Read EEPROM(0x02);
 hodnota = Read EEPROM(0x03);
 hodnota = Read EEPROM(0x04);
  // blikneme zelenou jako priznak konce
 bliknutiLEDZELENA();
 while (1)
 {
    bliknutiLEDMODRA();
  }
}
void GPIO Inicializace(void)
{
 STM32vldiscovery_LEDInit(LED3);
 STM32vldiscovery_LEDInit(LED4);
 STM32vldiscovery_PBInit(BUTTON_USER, BUTTON_MODE_GPIO);
 STM32vldiscovery_LEDOff(LED3);
 STM32vldiscovery LEDOff(LED4);
}
void bliknutiLEDMODRA(void)
{
     STM32vldiscovery LEDOn(LED4); // zapnem LED4 - modra
     Delay(0x5FFFF);
     STM32vldiscovery_LEDToggle(LED4); // vypnem LED4 - modra
     Delay(0x5FFFF);
}
void bliknutiLEDZELENA(void)
{
      STM32vldiscovery LEDOn(LED3); // zapnem LED3 - zelena
      Delay(0x5FFFF);
      STM32vldiscovery LEDToggle(LED3); // vypnem LED3 - zelena
      Delay(0x5FFFF);
}
void Delay( IO uint32 t nTick)
{
 for(; nTick != 0; nTick--);
ļ
void I2C Inicializace(void)
{
 GPIO InitTypeDef GPIO InitStructure;
 I2C InitTypeDef I2C InitStructure;
 I2C DeInit(I2C1);
```



```
/* I2C Periph clock enable */
 RCC APB1PeriphClockCmd(RCC APB1Periph I2C1, ENABLE);
  /* GPIO Periph clock enable */
 RCC APB2PeriphClockCmd(RCC APB2Periph GPIOB, ENABLE);
  /* Configure I2C pins: SCL and SDA */
  GPIO InitStructure.GPIO Pin = GPIO_Pin_6 | GPIO_Pin_7;
  GPIO InitStructure.GPIO Speed = GPIO Speed 10MHz;
  GPIO InitStructure.GPIO Mode = GPIO Mode AF OD; //GPIO Mode AF PP
  GPIO Init(GPIOB, & GPIO InitStructure); //I2C EEPROM pripojime k PB
  //SCL bude na PB6 DATA na PB7
                                    AT24C02
  /* I2C configuration */
  I2C_InitStructure.I2C_Mode = I2C_Mode_I2C;
I2C_InitStructure.I2C_DutyCycle = I2C_DutyCycle_2;
  I2C_InitStructure.I2C_OwnAddress1 = 0xA0; //EEPROM
  I2C_InitStructure.I2C_Ack = I2C_Ack_Enable;
  I2C_InitStructure.I2C_AcknowledgedAddress = I2C_AcknowledgedAddress_7bit;
  I2C InitStructure.I2C ClockSpeed = 10000;
  /* I2C Peripheral Enable */
 I2C Cmd(I2C1, ENABLE);
  /* Apply I2C configuration after enabling it */
 I2C Init(I2C1, &I2C InitStructure);
}
uint8 t Read EEPROM(uint8 t adresa)
{
        uint8 t Data;
       Data = 0;
       /* While the bus is busy */
        while(I2C GetFlagStatus(I2C1, I2C FLAG BUSY));
        /* Send START condition */
        I2C GenerateSTART(I2C1, ENABLE);
        /* Test on EV5 and clear it */
        while(!I2C CheckEvent(I2C1, I2C EVENT MASTER MODE SELECT));
        /* Send EEPROM address for read */
        I2C Send7bitAddress(I2C1, EEPROM SLAVE ADDRESS,
I2C Direction Transmitter);
        /* Test on EV6 and clear it */
        while(!I2C CheckEvent(I2C1,
I2C EVENT MASTER TRANSMITTER MODE SELECTED));
        /* Clear EV6 by setting again the PE bit */
        I2C Cmd(I2C1, ENABLE);
        /\star Send the EEPROM's internal address to read from: MSB of the
address first */
        I2C SendData(I2C1, adresa);
```

V.Váňa: CanSAT Book for Students – part 2 175



}

Inovace ve vzdělávání na území hl.m. Prahy 2011, proj.č.2046

```
/* Test on EV8 and clear it */
        while(!I2C CheckEvent(I2C1, I2C EVENT MASTER BYTE TRANSMITTED));
        /* Send START condition a second time */
        I2C GenerateSTART(I2C1, ENABLE);
        /* Test on EV5 and clear it */
        while(!I2C CheckEvent(I2C1, I2C EVENT MASTER MODE SELECT));
        /* Send EEPROM address for read */
        I2C Send7bitAddress(I2C1, EEPROM SLAVE ADDRESS,
I2C Direction Receiver);
        /* Test on EV6 and clear it */
        while(!I2C CheckEvent(I2C1,
I2C EVENT MASTER RECEIVER MODE SELECTED));
        /* Disable Acknowledgement */
        I2C AcknowledgeConfig(I2C1, DISABLE);
        /* Send STOP Condition */
        I2C GenerateSTOP(I2C1, ENABLE);
        while(!I2C CheckEvent(I2C1, I2C EVENT MASTER BYTE RECEIVED)){}
        /* Read a byte from the EEPROM */
        Data = I2C ReceiveData(I2C1);
        /* Enable Acknowledgement to be ready for another reception */
        I2C AcknowledgeConfig(I2C1, ENABLE);
       return(Data);
void Write EEPROM(uint8 t adresa, uint8 t data)
{
        /* Send START condition */
        I2C GenerateSTART(I2C1, ENABLE);
        /* Test on EV5 and clear it */
        while(!I2C CheckEvent(I2C1, I2C EVENT MASTER MODE SELECT));
        /* Send EEPROM address for write */
        I2C Send7bitAddress(I2C1, EEPROM SLAVE ADDRESS,
I2C Direction Transmitter);
        /* Test on EV6 and clear it */
        while(!I2C CheckEvent(I2C1,
I2C EVENT MASTER TRANSMITTER MODE SELECTED));
        /* Send the EEPROM's internal address to write to : MSB of the
address first */
        I2C SendData(I2C1, adresa);
        /* Test on EV8 and clear it */
        while (!12C CheckEvent (12C1, 12C EVENT MASTER BYTE TRANSMITTED));
V.Váňa: CanSAT Book for Students – part 2
                                   176
                                                         ARM STM32 programming
```



```
/* Send the byte to be written */
        I2C SendData(I2C1, data);
        /* Test on EV8 and clear it */
        while(!I2C CheckEvent(I2C1, I2C EVENT MASTER BYTE TRANSMITTED));
        /* Send STOP condition */
        I2C GenerateSTOP(I2C1, ENABLE);
void Ready EEPROM(void)
{
        IO uint16 t temp = 0;
        do
        {
          /* Send START condition */
         I2C_GenerateSTART(I2C1, ENABLE);
          /* Read I2C EE SR1 register to clear pending flags */
          temp = I2C ReadRegister(I2C1, I2C_Register_SR1);
          /* Send EEPROM address */
         I2C Send7bitAddress(I2C1, EEPROM SLAVE ADDRESS,
I2C_Direction_Transmitter);
        }while(!(I2C ReadRegister(I2C1, I2C Register SR1) & 0x0002));
        /* Clear AF flag */
        I2C ClearFlag(I2C1, I2C FLAG AF);
        /* STOP condition */
        I2C GenerateSTOP(I2C1, ENABLE);
```

Zkontrolovat obsah EEPROM můžeme např. free programem **PonyProg2000** <u>http://www.lancos.com/prog.html</u> a přípravkem připojeným k sériovému portu **COM1** či **COM2**.





Nové počítače už obvykle port COM nemají a tak můžeme použít převodním USB/COM a jeho driver, čímž získáme virtuální sériový port.





2.13.2 I2C – paralelní port PCF8574 (program14 a program14b)

Schéma a PCB v free EAGLE :










Nyní se již budeme věnovat programování:

Nejprve vytvoříme program pro zápis tj. pomocí I2C nastavíme logické úrovně na 8 I/O obvodu PCF8574









```
/*
* I2C komunikace s PCF8574
*
* priklad 14 main.c
* Vd pro SPSE Jecna Lectures
*
*/
#include "stm32f10x.h"
#include "stm32f10x_i2c.h"
#include "STM32vldiscovery.h"
/* Private typedef ------*/
GPIO_InitTypeDef GPIO_InitStructure;
/* Private define ------*/
//#define I2C_SLAVE_ADDRESS 0x40 //protoze je to PCF8574
#define I2C_Adr 0x40 // Adresa PCF8574 pro zapis
```

V.Váňa: CanSAT Book for Students – part 2



```
//SCL bude na PB6 DATA na PB7
//Tabulka I2C slave adres(A0=0 je write do obvodu, A1=1 je read )//Typ obvoduA7A6A5A4A3A2A1//8bitovy vstup/vystup010aaa
//Tabulka I2C slave adres
    napr. PCF8574 0x40 zapis 0x41 cteni
//EEPROM
                          1
                               0
                                      1
                                           0
                                                а
                                                      а
                                                            а
     napr. PCF8582
                                     1
                           1
                               0
                                          0
//Hodiny/kalendar
                                                0
                                                      0
                                                            а
    napr. PCF8583
//8bitovy A/D a D/A
                           1
                               0
                                     0
                                          1
                                                а
                                                     a
                                                            а
    napr. PCF8591
                          1 1
//PLL
                                    0
                                          0
                                                0
                                                     a
                                                            а
    napr. TSA5055
/* Private variables ------
___*/
uint8 t hodnota=0;
void Wait_ms(u8 Time);
void Wait us(void);
void Delay( IO uint32 t nTick);
void GPIO Inicializace(void); // nastavení vstupne/vystupnich pinu na kitu
void PCF8574 Init(void);
void bliknutiLEDMODRA(void);
                               11
                                      11
void bliknutiLEDZELENA(void);
int main(void)
{
  GPIO Inicializace();
  I2C Inicializace();
  // blikneme modrou jako priznak startu
 bliknutiLEDMODRA();
 PCF8574 Init();
 bliknutiLEDZELENA();
 //zapis 0x55
 I2C SendData(I2C1, 0x55);
          while(!I2C CheckEvent(I2C1,
I2C EVENT MASTER BYTE TRANSMITTED));
           Wait ms(1);
 bliknutiLEDMODRA();
  Delay(0xAFFFFF);
   Delay(0xAFFFFF);
   Delay(0xAFFFFF);
   Delay(0xAFFFFF);
  // Send STOP condition
     I2C GenerateSTOP(I2C1, ENABLE);
  // blikneme zelenou jako priznak konce
      bliknutiLEDZELENA();
```

184

```
V.Váňa: CanSAT Book for Students – part 2
```



```
while (1)
  {
   PCF8574 Init();
       I2C SendData(I2C1, 0x55);
            while(!I2C CheckEvent(I2C1,
I2C EVENT MASTER BYTE TRANSMITTED));
           Wait ms(1);
          bliknutiLEDZELENA();
        I2C SendData(I2C1, 0xAA);
            while(!I2C CheckEvent(I2C1,
i2C EVENT MASTER BYTE TRANSMITTED));
            Wait ms(1);
            bliknutiLEDMODRA();
      I2C GenerateSTOP(I2C1, ENABLE);
  }
}
void GPIO Inicializace(void)
{
  STM32vldiscovery LEDInit(LED3);
  STM32vldiscovery_LEDInit(LED4);
  STM32vldiscovery_PBInit(BUTTON_USER, BUTTON_MODE_GPIO);
  STM32vldiscovery_LEDOff(LED3);
  STM32vldiscovery LEDOff(LED4);
}
void bliknutiLEDMODRA(void)
{
      STM32vldiscovery LEDOn(LED4); // zapnem LED4 - modra
      Delay(0x5FFFF);
      STM32vldiscovery LEDToggle(LED4); // vypnem LED4 - modra
      Delay(0x5FFFF);
}
void bliknutiLEDZELENA(void)
{
       STM32vldiscovery LEDOn(LED3); // zapnem LED3 - zelena
       Delay(0x5FFFF);
       STM32vldiscovery_LEDToggle(LED3); // vypnem LED3 - zelena
       Delay(0x5FFFF);
}
void Delay( IO uint32 t nTick)
{
  for(; nTick != 0; nTick--);
}
```



```
void I2C Inicializace(void) //pro zapis
  GPIO InitTypeDef GPIO InitStructure;
  I2C InitTypeDef I2C InitStructure;
  I2C DeInit(I2C1);
  /* I2C Periph clock enable */
  RCC APB1PeriphClockCmd(RCC APB1Periph I2C1, ENABLE);
  /* GPIO Periph clock enable */
  RCC APB2PeriphClockCmd(RCC APB2Periph GPIOB, ENABLE);
  /* Configure I2C pins: SCL and SDA */
  GPIO InitStructure.GPIO Pin = GPIO Pin 6 | GPIO Pin 7;
  GPIO InitStructure.GPIO Speed = GPIO Speed 10MHz;
  GPIO InitStructure.GPIO Mode = GPIO Mode AF OD; //GPIO Mode AF PP
  GPIO Init(GPIOB, & GPIO InitStructure); //I2C PCF8574 pripojime k PB
  //SCL bude na PB6 DATA na PB7
  /* I2C configuration */
  I2C InitStructure.I2C Mode = I2C Mode I2C;
  I2C_InitStructure.I2C_DutyCycle = I2C_DutyCycle_2;
  I2C InitStructure.I2C OwnAddress1 = 0x40; //PCF8574 zapis
  I2C_InitStructure.I2C_Ack = I2C_Ack_Enable;
  I2C_InitStructure.I2C_AcknowledgedAddress = I2C_AcknowledgedAddress_7bit;
  I2C InitStructure.I2C ClockSpeed = 10000;
  /* I2C Peripheral Enable */
  I2C Cmd(I2C1, ENABLE);
  /* Apply I2C configuration after enabling it */
  I2C Init(I2C1, &I2C InitStructure);
}
void PCF8574 Init(void) //pro zapis
{
     Wait ms(200);
                                                     // Pockat na nabeh
napeti
//Start komunikace s PCF8574
     I2C GenerateSTART(I2C1, ENABLE); // Generovat START
// Test on EV5 and clear it
     while(!I2C CheckEvent(I2C1, I2C EVENT MASTER MODE SELECT));
// Send I2C address for write
      I2C Send7bitAddress(I2C1, I2C Adr, I2C Direction Transmitter);
// Test on EV6 and clear it
     while(!I2C CheckEvent(I2C1,
I2C EVENT MASTER TRANSMITTER MODE SELECTED));
}
void Wait us (void)
```

V.Váňa: CanSAT Book for Students – part 2 186



```
{
vu32 nTick = 0xFFF;
for(; nTick != 0; nTick--);
}
void Wait_ms(u8 Time)
{
    if (Time > 0 )
        for(; Time != 0; Time--)
        {
            Wait_us();
        }
}
```

Program pro čtení: tj pomocí I2C přečteme logické úrovně na 8 I/O obvodu PCF8574 a přeneseme do STM32.

Bude to vyžadovat následující změny: Ve funkci I2C Inicializace (void) bude

```
I2C InitStructure.I2C OwnAddress1 = 0x40; //PCF8574 zapis
```

nahraženo

I2C InitStructure.I2C OwnAddress1 = 0x41; //PCF8574 ctení

Změní se i funkce

```
void PCF8574_Init(void) //pro cteni
{
    Wait_ms(200); // Pockat na nabeh napeti
//Start komunikace s PCF8574
    I2C_GenerateSTART(I2C1, ENABLE); // Generovat START
// Test on EV5 and clear it
    while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_MODE_SELECT));
// Send I2C address for write
    I2C_Send7bitAddress(I2C1, I2C_Adr,I2C_Direction_Receiver);
// Test on EV6 and clear it
    while(!I2C_CheckEvent(I2C1,I2C_EVENT_MASTER_RECEIVER_MODE_SELECTED));
}
```



Posloupnost příkazů pro čtení byte dat z PCF8574 bude

```
while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_BYTE_RECEIVED));
    Wait_ms(1);
    hodnota = I2C_ReceiveData(I2C1);
```

Následující výpis **main.c** je z programu **priklad14b**, ve kterém se na LCD zobrazuje číslo 0 až 255 odpovídající nastavení hodnot na vstupech obvodu PCF8574

```
/*
* I2C komunikace s PCF8574
 cteni dat z PCF8574
 priklad 14b
                      main.c
 Vd pro SPSE Jecna Lectures
*/
#include <stddef.h>
#include "stm32f10x.h"
#include "stm32f10x i2c.h"
#include "STM32vldiscovery.h"
#include "LCD.h"
uint8 t LCD text[] = "Test PCF8574.";
/* Private typedef -----
GPIO InitTypeDef GPIO InitStructure;
/* Private define ------
//#define I2C_SLAVE_ADDRESS 0x40 //protoze je to PCF8574
#define I2C_Adr 0x41 // Adresa PCF8574 pro cteni
//SCL bude na PB6 DATA na PB7
//SCL bude na PB6 DATA na PB7
//Tabulka I2C slave adres(A0=0 je write do obvodu, A1=1 je read )//Typ obvoduA7A6A5A4A3A2A1//8bitovy vstup/vystup010aaa
  napr. PCF8574 0x40 zapis 0x41 cteni
                                        1
                                             0
//EEPROM
                                  0
                           1
                                                   а
                                                         а
                                                                а
     napr. PCF8582
                            1 0
                                       1
                                             0
                                                         0
//Hodiny/kalendar
                                                   0
                                                                а
    napr. PCF8583
                            1 0
//8bitovy A/D a D/A
                                       0
                                             1
                                                   а
                                                         а
                                                                а
    napr. PCF8591
//PLL
                            1 1
                                             0
                                                   0
                                       0
                                                         а
                                                               a
     napr. TSA5055
/* Private variables ------
---*/
```

V.Váňa: CanSAT Book for Students – part 2



```
uint8 t hodnota;
void itoa(uint16_t n, int8_t s[]);
void reverse(int8 t s[]);
void Wait ms(u8 Time);
void Wait us(void);
void Delay( IO uint32 t nTick);
void GPIO Inicializace(void); // nastavení vstupne/vystupnich pinu na kitu
void I2C Inicializace(void);
                                 // nastavení I2C
void PCF8574 Init(void);
void bliknutiLEDMODRA(void);
                                    11
                                          11
void bliknutiLEDZELENA(void);
int main(void)
{
  LCD Inicialization();
  GPIO Inicializace();
  I2C Inicializace();
   // blikneme modrou jako priznak startu
  bliknutiLEDMODRA();
  printLCD(LCD text);
  PCF8574 Init();
  bliknutiLEDZELENA();
  //cteni
     while(!I2C CheckEvent(I2C1, I2C EVENT MASTER BYTE RECEIVED));
     Wait ms(1);
     hodnota = I2C ReceiveData(I2C1);
     bliknutiLEDMODRA();
     LCD Clear();
     itoa(hodnota,LCD text);
      printLCD(LCD text);
      Wait ms(100);
   // Send STOP condition
      I2C GenerateSTOP(I2C1, ENABLE);
  // blikneme zelenou jako priznak konce
     bliknutiLEDMODRA();
  while (1)
  {
   bliknutiLEDZELENA();
     PCF8574 Init();
      while (!I2C CheckEvent (I2C1, I2C EVENT MASTER BYTE RECEIVED));
     Wait ms(1);
    hodnota = I2C ReceiveData(I2C1);
    LCD Clear();
     itoa(hodnota,LCD text);
     printLCD(LCD text);
    Wait ms(100);
      I2C GenerateSTOP(I2C1, ENABLE);
  }
}
void GPIO Inicializace(void)
  STM32vldiscovery LEDInit(LED3);
                                                         ARM STM32 programming
V.Váňa: CanSAT Book for Students – part 2
                                      189
```



```
STM32vldiscovery LEDInit(LED4);
 STM32vldiscovery PBInit (BUTTON USER, BUTTON MODE GPIO);
 STM32vldiscovery LEDOff(LED3);
 STM32vldiscovery LEDOff(LED4);
void bliknutiLEDMODRA(void)
{
     STM32vldiscovery LEDOn(LED4); // zapnem LED4 - modra
     Delay(0x5FFFF);
     STM32vldiscovery LEDToggle(LED4); // vypnem LED4 - modra
     Delay(0x5FFFF);
}
void bliknutiLEDZELENA(void)
{
      STM32vldiscovery LEDOn(LED3); // zapnem LED3 - zelena
      Delay(0x5FFFF);
      STM32vldiscovery LEDToggle(LED3); // vypnem LED3 - zelena
      Delay(0x5FFFF);
}
void Delay( IO uint32 t nTick)
{
 for(; nTick != 0; nTick--);
}
void I2C Inicializace(void) //pro cteni
{
 GPIO InitTypeDef GPIO_InitStructure;
 I2C InitTypeDef I2C InitStructure;
 I2C DeInit(I2C1);
  /* I2C Periph clock enable */
 RCC APB1PeriphClockCmd(RCC APB1Periph I2C1, ENABLE);
  /* GPIO Periph clock enable */
 RCC APB2PeriphClockCmd(RCC APB2Periph GPIOB, ENABLE);
  /* Configure I2C pins: SCL and SDA */
  GPIO InitStructure.GPIO Pin = GPIO Pin 6 | GPIO Pin 7;
  GPIO InitStructure.GPIO Speed = GPIO Speed 10MHz;
  GPIO InitStructure.GPIO Mode = GPIO Mode AF OD; //GPIO Mode AF PP
  GPIO Init(GPIOB, & GPIO InitStructure); //I2C PCF8574 pripojime k PB
  //SCL bude na PB6 DATA na PB7
  /* I2C configuration */
 I2C InitStructure.I2C Mode = I2C Mode I2C;
  I2C InitStructure.I2C DutyCycle = I2C DutyCycle 2;
 I2C InitStructure.I2C OwnAddress1 = 0x41; //PCF8574 cteni
 I2C InitStructure.I2C Ack = I2C Ack Enable;
 I2C InitStructure.I2C AcknowledgedAddress = I2C AcknowledgedAddress 7bit;
  I2C InitStructure.I2C ClockSpeed = 10000;
  /* I2C Peripheral Enable */
 I2C Cmd(I2C1, ENABLE);
```

190

V.Váňa: CanSAT Book for Students – part 2



```
/* Apply I2C configuration after enabling it */
 I2C Init(I2C1, &I2C InitStructure);
}
void PCF8574 Init(void) //pro cteni
{
     Wait ms(200);
                                         // Pockat na nabeh napeti
//Start komunikace s PCF8574
     I2C GenerateSTART(I2C1, ENABLE); // Generovat START
// Test on EV5 and clear it
     while(!I2C CheckEvent(I2C1, I2C EVENT MASTER MODE SELECT));
// Send I2C address for write
     I2C Send7bitAddress(I2C1, I2C Adr, I2C Direction Receiver);
// Test on EV6 and clear it
     while(!I2C CheckEvent(I2C1,I2C EVENT MASTER RECEIVER MODE SELECTED));
}
void Wait us (void)
{
vu32 nTick = 0xFFF;
 for(; nTick != 0; nTick--);
}
void Wait ms(u8 Time)
{
     if (Time > 0)
           for(; Time != 0; Time--)
            {
                 Wait us();
            }
}
void itoa(uint16 t n, int8 t s[])
{
   int i, sign;
   if ((sign = n) < 0) / * record sign */
      n = -n;
                        /* make n positive */
    i = 0;
   s[i++] = n % 10 + '0'; /* get next digit */
} while ((n /= 10) > 0); /* delete it */
    if (sign < 0)
       s[i++] = '-';
   s[i] = ' \setminus 0';
   reverse(s);
}
void reverse(int8 t s[])
{
   int i, j, k;
   char c;
```

V.Váňa: CanSAT Book for Students – part 2



```
strlen(s)-1;
      k =
    for (i = 0, j = k; i<j; i++, j--) {</pre>
        c = s[i];
        s[i] = s[j];
        s[j] = c;
    }
 }
      assert func (const char *file, int line, const char *func, const char
void
*failedexpr)
 while(1)
  { }
}
void assert(const char *file, int line, const char *failedexpr)
{
    assert func (file, line, NULL, failedexpr);
}
```

2.13.3 I2C – čidlo teploty TI AD7416 (program15)

Předchozí programy ve kterých jsme zapisovali do EEPROM AT24C02 a zapisovali a četli data z PCF8574 nám ukázaly, jak psát kód pro I2C komunikaci STM32 s obvody I2C. Z kódu pochopitelně odstraníme kód pro blikání zelené a modré LED, který nám pomáhal při odlaďování programu a sledování jeho chodu. Pro jiné obvody I2C pak ještě použijeme informace z datesheetu tohoto obvodu (staženého z webu výrobce) a dále změníme příslušně bázovou adresu obvodu I2C. Při odlaďování můžeme opět použít kódy pro blikání LED. Po odladění programu pro obvod I2C pak kód pro blikání LED odstraníme, musíme mít ovšem na zřeteli, že pro některé obvody bude po odstranění kódu pro blikání nutné do těchto míst vložit nějaké zpoždění, aby obvod I2C správně fungoval.

Nicméně ukážeme si ještě kód pro práci s čidlem 10bitovým čidlem teploty AD7416. Jde o obvod od Analog Devices. Tyto obvody lze od AD získat jako free samples. Obvod v 8pinovém pouzdru je přímo předurčený pro konstrukce CanSATů, neboť měření teploty je při soutěžích CanSATů často požadováno (ESA CanSAT Competition). Popis tohoto obvodu najdeme v firemní dokumentaci. Pinově a funkčně kompatibilní teplotní čidla vyrábí i STMicroelectronics a to STTS75 a STDS75 s rozlišením 9bitů (default) až 12bitů (rozlišení můžeme volit, lepší rozlišení je však vykoupeno delší dobou měření teploty) , STLM75 a STCM75 s rozlišením 9bitů v pouzdrech SO8. Do CanSATů doporučuji používat právě obvody od STMicroelectrinics.



AD7416 PIN CONFIGURATION

SOIC/MSOP



Pokud na A0,A1,A2 připojíme úroveň 0, bude bázová adresa tohoto obvodu pro zápis 0x90, pro čtení 0x91. Tento obvod, má stejnou sadu příkazů jako obvody AD7417 a AD7418, které mohou pracovat nejen jako čidla teploty, ale i jako 10bitové A/D převodníky. Proto nejprve musíme do těchto obvodů zapsat tzv. *Adress Pointer Register*. V případě, kdy požadujeme na obvodu, aby pracoval jako čidlo teploty bude tato hodnota 00000000.

Proto nejprve musí být na I2C sběrnici posloupnost následujících akcí: START, poslání 0x90 (protože budeme zapisovat), poslání 0x00 (Adres Pointer Register) a STOP. Poté již může následovat přečtení hodnoty naměřené teploty a to ve dvou bytech dat, tj. následující akce na sběrnici I2C: START, poslání 0x91 (protože budeme číst), přečtení horního byte dat (více významnější byte), přečtení dolního byte dat (méně vyznamější byte) a STOP. Tím jsme získali obsah Temperature Value Register

Table III	Temperature	Value	Register	
-----------	-------------	-------	----------	--

D15	D14	D13	D12	D11	D10	D9	D8	D 7	D6
MSB	B8	B7	B 6	B5	B4	B3	B2	B1	LSB

Informaci o teplotě s přesností 0,25 ° C nese horních 10 bitů. Obsah dolních bitů D5, D4, … D0 nemá žádný význam. Následující tabulka pak ukazuje převod mezi 10bitovým datovým údajem a hodnotou teploty v ° C.

Table IV. Temperature Data Format

Temperature	Digital Output				
-128°C	10 0000 0000				
-125°C	10 0000 1100				
-100°C	10 0111 0000				
-75°C	10 1101 0100				
-50°C	11 0011 1000				
-25°C	11 1001 1100				
-0.25°C	11 1111 1111				
0°C	00 0000 0000				
+0.25°C	00 0000 0001				
+10°C	00 0010 1000				
+25°C	00 0110 0100				
+50°C	00 1100 1000				
+75°C	01 0010 1100				
+100°C	01 1001 0000				
+125°C	01 1111 0100				
+127°C	01 1111 1100				



Nejvíce významný bit je znaménkový – 0 je nezáporná čísla, 1 pro záporná. Pro záporná čísla je další bity v doplňkovém kódu.

Pozn. Pinově i funkčně kompatibilní (včetně I2C komunikace) s obvodem AD7416 je obvod LM75 firmy National Semiconductor. Jediný rozdíl je v tom, že LM75 měří teplotu s přesností 0,5 ° C a informaci o teplotě nese 9 horních bitů Temperature Value Register.

1.12 TEMPERATURE REGISTER

(Read Only):

					:		41/2	41 · · · · · ·	C		8	S	×. 3	S	20 J.
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
MSB	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	LSB	Х	Х	Х	Х	Х	X	X

D0-D6: Undefined

D7-D15: Temperature Data. One LSB = 0.5° C. Two's complement format.

Jak jsme se již zmínili, kompatibilní jsou i obvody STMicroelectronics STTS75, STDS75, STLM75 a STCM75.

Algoritmus přečtění dat o teplotě z AD7416 tedy bude

```
float ad7416_calculate()
{
    int t1,t2;
    float t;
    i2c_start();
    i2c_write( 0x90 ); // Address
    i2c_write( 0x00 ); // Address pointer register
    i2c_stop();

i2c_start();
    i2c_write( 0x91 );
t1 = (int)i2c_read(0); // horních osm bitů Temperature registru
t2 = (int)i2c_read(1); // dolních osm bitů Temperature registru
i2c_stop();
t = (float)( ( (t2>>6)+(t1<<2) ) * 0.25 );
return t;
}</pre>
```

Neboli horních 8bitů posuneme o dva bity doleva, čímž dostaneme 10bitové číslo s dolními dvěma bity nulovými. Dolních 8 bitů posuneme o 6 bitů doprava. Tím přijdeme o 6dolních (nevýznamných) bitů z t2. Zůstanou nám tak 2 významné bity z t2. Po součtu již dostaneme v 10ti bitový údaj o teplotě, kde jeden bit odpovídá 0,25 ° C, tak že po vynásobení 0,25 dostaneme v t hodnotu teploty ve stupních C.

Pozn.: Pro teploty větší nebo rovné nule obsahuje t1 rovnou údaj o teplotě s přesností 1 °C.



K otestování AD7426 a dalších obdobných obvodů (STMicroelectronics) můřeme použít PC se sériovým portem či s USB a převodníkem USB/COM (a tudíž s virtuálním sériovým portem) a zapojit ho pomocí stejného přípravku, který jsme použili pro I2C čtení obsahu EEPROM programem PonyProg2000, nebo můžeme použít přípravek z <u>http://www.amaterskaelektronika.cz/file/hw-prevodnik-i2c-rs232</u>

V obou případech můžeme na PC spustit free program z http://www.amaterskaelektronika.cz/file/teplomer-Im75a-delphi-i2c







```
/*
* I2C komunikace s AD7416 a obdobnými čidly teploty
* cteni dat z AD7416
* priklad 15
                    main.c
* Vd pro SPSE Jecna Lectures
*/
#include <stddef.h>
#include "stm32f10x.h"
#include "stm32f10x i2c.h"
#include "STM32vldiscovery.h"
#include "LCD.h"
uint8 t LCD text[] = "Test AD7416.";
/* Private typedef -----
GPIO InitTypeDef GPIO InitStructure;
I2C_InitTypeDef I2C_InitStructure;
/* Private define -
#define I2C Adr
                      0x91 // Adresa AD7416
V.Váňa: CanSAT Book for Students – part 2
                                    196
                                                       ARM STM32 programming
```



```
//SCL bude na PB6 DATA na PB7
```

//Tabulka I2C slave adro //Typ obvodu	es	А7	(A0=0 A6	je wr: A5	ite do A4	obvod A3	u, A1= A2	1 je read) A1
<pre>//8bitovy vstup/vystup PCF8574 0x40 zapis 0x42</pre>	0 1 cten	1 i	0	0	a	a	a	napr.
//EEPROM	1	0	1	0	a	a	a	napr.
//Hodiny/kalendar PCF8583	1	0	1	0	0	0	a	napr.
//8bitovy A/D a D/A PCF8591 nebo AD7416	1	0	0	1	a	a	a	napr.
//PLL TSA5055	1	1	0	0	0	a	a	napr.
<pre>/* Private variables uint8_t hodnota1; uint8_t hodnota2;</pre>								****
<pre>void itoa(uint16_t n, in void reverse(int8_t s[]</pre>	nt8_t);	s[]);						
<pre>void Wait_ms(u8 Time); void Wait_us(void); void Delay(IO uint32_' void GPIO Inicializace()</pre>	t nTic void);	k);		// nas	staven	í vstu	pne/vv	stupnich
pinu na kitu void I2C_Inicializace_R	(void)	;		// nas	staven	í I2C :	pro ct	eni
<pre>void AD7416_Init_R(void void I2C_Inicializace_W void AD7416_Init_W(void void bliknutiLEDMODRA(void)</pre>); (void)); pid);	;	//	// nas	staven	í I2C <u>:</u>	pro za	pis
void bliknutiLEDZELENA(void);		//					
<pre>int main(void) { LCD_Inicialization() GPIO_Inicializace(); printLCD(LCD_text);</pre>	;							
// blikneme modrou jako bliknutiLEDMODRA(); //zapis do registru AD	prizn 7416	ak sta	rtu					
<pre>I2C_Inicializace_W() Wait_ms(2); bliknutiLEDZELENA AD7416_Init_W(); Wait_ms(2); bliknutiLEDZELENA I2C_SendData(I2C1, 0; bliknutiLEDZELENA</pre>	; (); A(); x00); ();							

V.Váňa: CanSAT Book for Students – part 2

197



```
while(!I2C CheckEvent(I2C1, I2C EVENT MASTER BYTE TRANSMITTED));
 Wait ms(2);
 bliknutiLEDZELENA();
 I2C GenerateSTOP(I2C1, ENABLE);
 bliknutiLEDZELENA();
 // Delay(0xAFFFFF);
//cteni
  I2C Inicializace R();
 Wait ms(2);
   bliknutiLEDZELENA();
 AD7416 Init R();
 Wait ms(2);
    bliknutiLEDZELENA();
//cteni data0
    while(!I2C CheckEvent(I2C1, I2C EVENT MASTER BYTE RECEIVED));
    Wait ms(2);
    bliknutiLEDZELENA();
  hodnota1 = I2C ReceiveData(I2C1);
     bliknutiLEDZELENA();
    LCD Clear();
    itoa(hodnota1,LCD text);
    bliknutiLEDMODRA();
    printLCD(LCD_text);
    Wait ms(5000);
    //cteni data1
  while(!I2C CheckEvent(I2C1, I2C EVENT MASTER BYTE RECEIVED));
    Wait ms(1);
     bliknutiLEDZELENA();
  hodnota2 = I2C ReceiveData(I2C1);
     bliknutiLEDZELENA();
  LCD Clear();
    itoa(hodnota2,LCD text);
    printLCD(LCD text);
    Wait ms(1);
 // Send STOP condition
    I2C GenerateSTOP(I2C1, ENABLE);
 // blikneme zelenou jako priznak konce
    bliknutiLEDMODRA();
while (1)
 {
 bliknutiLEDZELENA();
 //cteni
 I2C Inicializace R();
 bliknutiLEDZELENA();
 Wait ms(2);
 AD7416 Init R();
    bliknutiLEDZELENA();
 Wait ms(2);
//cteni data0
    while(!I2C CheckEvent(I2C1, I2C EVENT MASTER BYTE RECEIVED));
    Wait ms(2);
```

```
V.Váňa: CanSAT Book for Students – part 2
```



```
bliknutiLEDZELENA();
    hodnota1 = I2C_ReceiveData(I2C1);
     LCD Clear();
     itoa(hodnota1,LCD text);
      printLCD(LCD text);
      Wait ms(500);
      //cteni data1
    while(!I2C CheckEvent(I2C1, I2C EVENT MASTER BYTE RECEIVED));
     Wait ms(1);
       bliknutiLEDZELENA();
    hodnota2 = I2C ReceiveData(I2C1);
      bliknutiLEDZELENA();
    Wait ms(1);
  // Send STOP condition
     I2C GenerateSTOP(I2C1, ENABLE);
     bliknutiLEDZELENA();
  }
}
void GPIO Inicializace(void)
{
  STM32vldiscovery_LEDInit(LED3);
  STM32vldiscovery_LEDInit(LED4);
  STM32vldiscovery_PBInit(BUTTON_USER, BUTTON_MODE_GPIO);
  STM32vldiscovery_LEDOff(LED3);
  STM32vldiscovery LEDOff(LED4);
}
void bliknutiLEDMODRA(void)
{
      STM32vldiscovery LEDOn(LED4); // zapnem LED4 - modra
     Delay(0x5FFFF);
     STM32vldiscovery LEDToggle(LED4); // vypnem LED4 - modra
     Delay(0x5FFFF);
}
void bliknutiLEDZELENA(void)
{
       STM32vldiscovery LEDOn(LED3); // zapnem LED3 - zelena
       Delay(0x5FFFF);
       STM32vldiscovery_LEDToggle(LED3); // vypnem LED3 - zelena
       Delay(0x5FFFF);
}
void Delay( IO uint32 t nTick)
  for(; nTick != 0; nTick--);
```

199



```
void I2C Inicializace R(void) //pro cteni
 GPIO InitTypeDef GPIO InitStructure;
  //I2C InitTypeDef I2C InitStructure;
 I2C DeInit(I2C1);
  /* I2C Periph clock enable */
 RCC APB1PeriphClockCmd(RCC APB1Periph I2C1, ENABLE);
  /* GPIO Periph clock enable */
 RCC APB2PeriphClockCmd (RCC APB2Periph GPIOB, ENABLE);
  /* Configure I2C pins: SCL and SDA */
 GPIO InitStructure.GPIO Pin = GPIO Pin 6 | GPIO Pin 7;
 GPIO_InitStructure.GPIO_Speed = GPIO_Speed_10MHz;
 GPIO InitStructure.GPIO Mode = GPIO Mode AF OD; //GPIO Mode AF PP
 GPIO Init(GPIOB, & GPIO InitStructure); //I2C PCF8574 pripojime k PB
 //SCL bude na PB6 DATA na PB7
  /* I2C configuration */
 I2C InitStructure.I2C Mode = I2C Mode I2C;
 I2C_InitStructure.I2C_DutyCycle = I2C_DutyCycle_2;
 I2C_InitStructure.I2C_OwnAddress1 = 0x91; //AD7416 cteni
 I2C_InitStructure.I2C_Ack = I2C_Ack_Enable;
 I2C_InitStructure.I2C_AcknowledgedAddress = I2C_AcknowledgedAddress_7bit;
 I2C InitStructure.I2C ClockSpeed = 10000;
  /* I2C Peripheral Enable */
 I2C Cmd(I2C1, ENABLE);
  /* Apply I2C configuration after enabling it */
 I2C Init(I2C1, &I2C InitStructure);
void I2C Inicializace W(void) //pro zapis
{
 GPIO InitTypeDef GPIO InitStructure;
 //I2C InitTypeDef I2C InitStructure;
 I2C DeInit(I2C1);
  /* I2C Periph clock enable */
 RCC APB1PeriphClockCmd(RCC APB1Periph I2C1, ENABLE);
 /* GPIO Periph clock enable */
 RCC APB2PeriphClockCmd(RCC APB2Periph GPIOB, ENABLE);
  /* Configure I2C pins: SCL and SDA */
 GPIO InitStructure.GPIO Pin = GPIO Pin 6 | GPIO Pin 7;
 GPIO InitStructure.GPIO Speed = GPIO Speed 10MHz;
 GPIO InitStructure.GPIO Mode = GPIO Mode AF OD; //GPIO Mode AF PP
 GPIO Init(GPIOB, &GPIO InitStructure); //I2C AD7416 pripojime k PB
 //SCL bude na PB6 DATA na PB7
```

```
/* I2C configuration */
```

V.Váňa: CanSAT Book for Students – part 2 200



```
I2C InitStructure.I2C Mode = I2C Mode I2C;
 I2C InitStructure.I2C DutyCycle = I2C DutyCycle 2;
 I2C InitStructure.I2C OwnAddress1 = 0x90; //AD7416 zapis
 I2C InitStructure.I2C Ack = I2C Ack Enable;
 I2C InitStructure.I2C AcknowledgedAddress = I2C AcknowledgedAddress 7bit;
 I2C_InitStructure.I2C_ClockSpeed = 10000;
  /* I2C Peripheral Enable */
 I2C Cmd(I2C1, ENABLE);
  /* Apply I2C configuration after enabling it */
 I2C Init(I2C1, &I2C InitStructure);
void AD7416 Init W(void)
                            //pro zapis
{
     Wait ms(200);
                            // Pockat na nabeh napeti
//Start komunikace s PCF8574
     I2C GenerateSTART(I2C1, ENABLE); // Generovat START
// Test on EV5 and clear it
    while(!I2C CheckEvent(I2C1, I2C EVENT MASTER MODE SELECT));
// Send I2C address for write
     I2C Send7bitAddress(I2C1, I2C Adr, I2C Direction Transmitter);
// Test on EV6 and clear it
    while(!I2C CheckEvent(I2C1,
I2C EVENT MASTER TRANSMITTER MODE SELECTED));
}
void AD7416 Init R(void) //pro cteni
{
                            // Pockat na nabeh napeti
    Wait ms(200);
//Start komunikace s AD7416
     I2C GenerateSTART(I2C1, ENABLE); // Generovat START
// Test on EV5 and clear it
     while(!I2C CheckEvent(I2C1, I2C EVENT MASTER MODE SELECT));
// Send I2C address for write
     I2C Send7bitAddress(I2C1, I2C Adr,I2C Direction Receiver);
// Test on EV6 and clear it
     while(!I2C CheckEvent(I2C1,I2C EVENT MASTER RECEIVER MODE SELECTED));
void Wait us(void)
vu32 nTick = 0xFFF;
 for(; nTick != 0; nTick--);
```

201

V.Váňa: CanSAT Book for Students – part 2



```
/*****
* Function Name : Wait ms
* Description : Program pro zpozdeni
* Input : Time = cislo, ktere se odcita k NULE
              : None
* Output
* Return
              : None
                 * * * * * * * * * * * * * * * * * * *
void Wait ms(u8 Time)
{
     if (Time > 0 )
          for(; Time != 0; Time--)
          {
               Wait us();
          }
}
void itoa(uint16 t n, int8 t s[])
{
   int i, sign;
   if ((sign = n) < 0) /* record sign */
                     /* make n positive */
      n = -n;
   i = 0;
   s[i++] = n % 10 + '0';  /* get next digit */
   if (sign < 0)</pre>
      s[i++] = '-';
   s[i] = '\0';
   reverse(s);
}
void reverse(int8 t s[])
{
   int i, j, k;
   char c;
    k = strlen(s) - 1;
   for (i = 0, j = k; i<j; i++, j--) {</pre>
      c = s[i];
      s[i] = s[j];
      s[j] = c;
   }
}
void assert func (const char *file, int line, const char *func, const char
*failedexpr)
{
 while(1)
 { }
}
void assert(const char *file, int line, const char *failedexpr)
    assert func (file, line, NULL, failedexpr);
```

202

V.Váňa: CanSAT Book for Students – part 2



Nejprve se zobrazily čísla 24 a 192, poté ve smyčce občas 24, občas 23 – ve smyčce zobrazujeme jen s přesností na 1 ° C. V programu pro jednoduchost neprovádíme výpočet teploty a neuvažujeme teploty pod nulou (pro CanSAT v Norsku budeme muset uvažovat i nízké teploty). Před smyčkou jsme zobrazili i hodnotu2 což bylo 192 tj 11000000. Těch dolních 6 nul je nevýznamných, neboli význam má jen 11 což odpovídá 0,75 °C, takže teplota naměřená AD7416 byla 24,75 °C.

Program ukazuje jen principy, tj není v něm prováděn výpočet teploty, obsahuje kontrolní místa s blikajícími LED. Při použití STM obvodů popř. bude třeba odprogramovat nastavení na větší přesnost měření teploty. Podle použitých obvodů popř. po odstranění kódů pro blikání LED umístit potřebné Delay.

Dále bude třeba upravit kódy typu

```
while(!I2C_CheckEvent(I2C1, I2C_EVENT_MASTER_MODE_SELECT));
```

neboť v této smyčce může program uvíznout např. v případě poruchy obvodu či I2C komunikaci. Stačí např. "studený spoj" u pull odporu 4k7, kdy bez pull odporů na SDA a SCL nefunguje I2C komunikace.

2.14 Klávesnice PS2 (program16)

V případě, kdy potřebujeme klávesnici s větším počtem kláves může být výhodné použít běžnou klávesnici používanou u počítačů PC. Důsledkem jejího rozšíření je nízká cena a snadnost jejího připojení k mikropočítači či mikrořadiči. PC AT klávesnice tvoří samostatný konstrukční celek, obsahující obyčejně 102 kláves a mikrořadič , který vykonává řadu funkcí. Tou hlavní je neustálé sledování stavu matice spínačů. V případě jakékoli změny zjistí mikrořadič souřadnice spínače, který ji způsobil, vybere v této situaci odpovídající SCAN kód, a ten po sériové lince DATA vyšle směrem k zařízení, s nímž klávesnice komunikuje. Je-li některá klávesa stisknuta déle než 0,5 s, vysílá její kód automaticky znovu. Kromě vlastních dat vysílá klávesnice při stisku klávesy i hodinový signál CLK o frekvenci 10 až 20 kHz.. Při sestupné hraně hodinového signálu CLK jsou vysílaná DATA platná. Toho můžeme využít při komunikace klávesnice s nějakým systémem jako PC, mikrořadič STM32 apod. Tuto komunikaci včetně obslužného programu napsaného v jazyce C si ukážeme na komunikaci PC klávesnice s **STM32VL Discovery**, který s klávesnicí spojíme podle obrázku:





Signál CLK z klávesnice přitom bude vyvolávat vnější přerušení mikrořadiče STM32. Přerušení vyvolané sestupnou hranou hodinového signálu CLK bude obsluhováno funkcí, která která bude snímat data vysílaná klávesnicí. Klávesnice bude k mikrořadiči připojena přes její konektor. Zapojení konektoru PC AT klávesnice popisuje následující tabulka:

klávesnice PC AT		
Signál	DIN41524 ,zásuvka 5-pin	p-pin Mini DIN PS2
Clock	1	5
Data	2	1
nezapojen	3	2,6
GND	4	3
+5V	5	4
stínění	stínění	stínění

Klávesnice vysílá data sériově vždy po 11 bitech. Nejprve je vysílán Start bit (logická nula), poté 8 datových bitů (první je LSB, poslední MSB), dále parita a nakonec Stop bit (logická jednička) :



Při obsluze přerušení se při prvním přerušení vynuluje proměnná nesoucí informaci o počtu přijatých bitů, při každém dalším přerušení se tato proměnná inkrementuje. Obsah bitů 1 až 8 (bit 0 byl Start bit) datového vodiče se načítá do jednoho znaku, bit 9 a 10 se ignorují. Poté se provádí dekódování přijatého znaku. Toto řešení s použitím přerušení se dá použít pro jakékoli jednočipové procesory a jeho výhodou je minimální režie, kterou si obsluha klávesnice z celého systému zabere.

Minimální mezera mezi dvěma kódy vyslanými po sobě je 1,2 ms.

Tato synchronní sériová komunikace je obousměrná, my však budeme klávesnici používat jen ve směru klávesnice – mikrořadič. Kromě kódů kláves vysílá klávesnice i řídící signály:

- FFh přetečení bufferu, klávesnice detekuje chybu
- FEh žádost o zaslání posledního znaku, špatně přijatý znak, parita
- FAh potvrzení ACK
- F0h kód uvolnění klávesy
- AAh úspěšný power-on test



EEh echo – klávesnice odpoví zpět také EEh jako echo – pro test

00h přetečení bufferu, klávesnice detekuje chybu

Z hlediska kódování kláves klávesnicí je možné rozdělit klávesy do tří skupin. Jsou to skupina základní, skupina rozšířená a skupina speciální. Do *základní skupiiny* patří 83 kláves. Po stisku některé z těchto kláves je vyslán kód této klávesy a po uvolnění této klávesy je vyslán kód uvolnění klávesy a znovu kód uvolněné klávesy. Při delším stisku je kód neustále vysílán až do uvolnění. Kód klávesy v hranatých závorkách (na následujícím obrázku) se vysílá dokud je klávesa stisknuta:



Rozšířená skupina zahrnuje 14 kláves. Tyto klávesy mají kódy shodné s klávesami numerické klávesnice, patřících do základní skupiny, ale před tímto kódem je předřazen kód F8, který je od těchto kláves odlišuje. Dva kódy v hranatých závorkách (na následujícím obrázku) se vysílají dokud je klávesa stisknuta:



Speciální skupina zahrnuje všechny ostatní klávesy. Jedná se o klávesy Pause a Print Screen. Při současném stisku dvou a více ostatních kláves (základní a rozšířená skupina) jsou kódy vysílány v odpovídajícím pořadí v jakém se jednotlivé akce stisku a uvolnění stanou a tyto kódy se vzájemně neovlivňují, kromě případu kdy je přerušeno cyklické vysílání kódu klávesy – autorepeat. Rozmístění všech kláves a názvy jejich kódů jsou zřejmé z obr:



Kódy kláves jsou uvedeny v následujících tabulkách: CSAN kódy kláves základní skupiny:



Klávesa	kód	Klávesa	kód	Klávesa	kód
`	8F	S	27	F1	5F
1	97	D	3B	F2	9F
2	87	F	2B	F3	DF
3	9B	G	D3	F4	CF
4	5B	Н	33	F5	3F
5	8B	J	23	F6	2F
6	93	К	BD	F7	3E
7	43	L	2D	F8	AF
8	83	;	CD	F9	7F
9	9D	,	B5	F10	6F
0	5D	Shift	B7	F11	E1
-	8D	Z	A7	F12	1F
=	55	Х	BB	Scroll lock	81
Backspace	99	С	7B	Num lock	11
Tab	4F	V	AB	*	C1
Q	57	В	B3	N-	21
W	47	N	73	+	61
E	DB	М	A3	NO	F1
R	4B	,	7D	N1	69
Т	СВ	•	6D	N2	B1
Y	53	/	AD	N3	A1
U	C3	P Shift	65	N4	29
I	3D	١	79	N5	31

V.Váňa: CanSAT Book for Students – part 2



0	DD	Ctrl	D7	N6	D1
Р	4D	Alt	77	N7	C9
[D5	Space	6B	N8	51
]	25	Enter	A5	N9	41
Caps lock	E5	Esc	91	N.	71
A	C7				

SCAN kódy kláves rozšířené skupiny:

Klávesa	kód	zkratka
R alt	77	Alt
R ctrl	D7	Ctrl
Ins	F1	NO
Home	C9	N7
PgUp	41	N9
Del	71	N5
End	69	N1
PgDw	A1	N3
Left	29	N4
Right	D1	N6
Up	51	N8
Down	B1	N2
/	AD	/?
enter	A5	enter

Speciální skupina:



Klávesa	kód
Print Screen	F8 B7 F8 C1
Break	78 D7 11 F0 D7 F0 11

Náš program provádí na LCD výpis znaků odpovídajících stisknuté klávese na běžné PS2 klávesnici připojené svým hodinovým výstupem na PA1 a datovým výstupem na PA2. PS2 klávesnici nebudeme pochopitelně používat jako periferii letícího CanSATu. Můžeme ji však používat jako vstup dat při pozemním testování CanSATu. Hlavním důvodem uvedení tohoto programu je to, že je ideální ukázkou použití vnějšího přerušení.

Klávesnice totiž vysílá dva sériové signály – datový a hodinový. Hodinový signál máme přiveden na **PA1** a náběžnou hranou tohoto signálu se vyvolá externí přerušení, jehož obsluhou je funkce EXTI1_IRQHandler. Větší část kódu PS2.c a PS2.h jsem převzal z 17. dílu seriálu z MCU serveru. Bylo přitom nutné přejmenovat jméno funkce obsluhy přerušení a místo **void Preruseni_PS2(void)** použít void **EXTI1_IRQHandler(void)**.

Zdrojové kódy:

```
ps2.h
       Author: User
#ifndef PS2 H
#define PS2 H
/* Includes ----
#include "stm32f10x.h"
/** Exported_Types
                              */
/** Exported Functions */
void PS2 Inicializace(void);
//void Preruseni PS2(void);
void EXTI1 IRQHandler(void);
uint8 t keyboard getchar(void);
void dekoduj klavesu(uint8 t scan);
void put to buff(uint8 t ch);
void Kontrola_hodin(void);
/** Exported Defines */
#define PS2 PORT
                             GPIOA
#define
          PS2 CLOCK PIN
                             GPIO Pin 1
#define PS2_DATA_PIN
#define PS2_BUFSIZE
                             GPIO Pin 2
                             16
#endif /* PS2 H */
```



```
* main.h
 *
 *
 *
 */
#ifndef MAIN H
#define MAIN H
/* Includes -----
---*/
#include "stm32f10x.h"
#include "STM32vldiscovery.h"
void ZmenaCasu(void);
void Kontrola hodin(void);
uint32_t Cas(void);
```

#endif /* MAIN_H_ */

```
* ps2.c
*
  STM32 VL Discovery kit Keyborad interface
 *
  Vd pro SPSE Jecna lectures
 *
 */
/* Includes ----
                             _____
#include "ps2.h"
#include "main.h"
static uint32 t casPS2;
                            // okamzik posledniho preruseni hodinami
z klavesnice
static uint8 t znak;
                   // bude obsahovat scan kod
volatile uint8_t buffer_pointer = 0;
volatile uint8_t buffer[PS2_BUFSIZE];
volatile uint16_t status = 0;
// Lookup tabulka pro scankody normalniho stisku
const uint8 t obyc[] = {
     0x0d,9,
    0x0e,'|',
                              209
```

V.Váňa: CanSAT Book for Students – part 2



0x15,	'q'	,
0x16.	11	
01 0	1-1	'
UXIA,	· Z ·	'
UXID,	's'	/
0x1c,	'a'	,
0x1d,	'w'	,
0x1e.	'2'	
0v21		'
022		'
UXZZ,	· X ·	'
0x23,	'd'	,
0x24,	'e'	,
0x25,	'4'	,
0x26.	131	
020	, ĭ,	'
0x29,	11	'
0x2a,	'v'	,
0x2b,	'f'	,
0x2c,	't'	,
0x2d.	1r1	
0.200	151	'
Uxze,	. 5	'
0x31,	'n'	/
0x32,	'b'	,
0x33,	'h'	,
0x34.	'a'	
0.235	9	'
0,	<u>У</u>	'
0x36,	'6'	/
0x39,	'7'	,
0x3a,	'm'	,
0x3b.	141	
0v3c	111	'
0	171	'
0x3a,	: / :	'
0x3e,	'8'	,
0x41,	171	,
0x42,	'k'	,
0x43.	141	
0		'
0X44,	.0.	'
0x45,	.0.	/
0x46,	'9'	,
0x49,	1.1	,
0x4a.	$\tau = \tau$	
0 v l h	111	<i>'</i>
		'
0X4C,	:r:	'
0x4d,	'p'	,
0x4e,	'+'	,
0x55,	1//	۰,
0x5a	13	
On Eh	1	
, acxu		'
0x5d,	. / .	' /
0x61,	'<'	,
0x66,	8,	
0x69	111	
Oveh	1 1	'
, doxo	4	'
0x6C,	171	1
0x70,	'0'	,
0x71,	1,1	,
0x72.	'2'	,
,		

V.Váňa: CanSAT Book for Students – part 2



```
0x73,'5',
      0x74,'6',
      0x75,'8',
      0x79, '+',
      0x7a,'3',
      0x7b, '-',
      0x7c,'*',
      0x7d, '9',
      0,0
};
// Lookup tabulka pro scankody stisku se shiftem
const uint8 t seshiftem[] = {
      0x0d,9,
      0x0e,'§',
0x15,'Q',
      0x16,'!',
      0x1a,'Z',
      0x1b,'S',
      Ox1c, 'A',
      0x1d, 'W',
      0x1e,'"',
      0x21,'C',
      0x22,'X',
      0x23,'D',
      0x24,'E',
      0x25,'¤',
      0x26,'#',
      0x29,'',
      0x2a,'V',
      0x2b,'F',
      0x2c,'T',
      0x2d,'R',
      0x2e,'%',
      0x31,'N',
      0x32,'B',
      0x33,'H',
      0x34,'G',
      0x35,'Y',
      0x36,'&',
      0x39,'L',
      0x3a,'M',
      0x3b,'J',
      0x3c,'U',
      0x3d, '/',
      0x3e,'(',
      0x41,';',
      0x42,'K',
      0x43,'I',
      0x44,'0',
      0 \times 45, '=',
      0x46,')',
      0x49,':',
      0x4a,'_',
      0x4b,'L',
```

V.Váňa: CanSAT Book for Students – part 2



```
0x4d, 'P',
      0x4e,'?',
      0x55,'`',
      0x5a,13,
      0x5b,'^',
     0x5d,'*',
      0x61, '>',
      0x66,8,
      0x69,'1',
      0x6b, '4',
      0x6c, '7',
      0x70,'0',
      0x71,',',
      0x72,'2',
      0x73,'5',
     0x74, '6',
     0x75,'8',
      0x79, '+',
      0x7a,'3',
     0x7b,'-',
     0x7c,'*',
     0x7d, '9',
     0,0
};
void PS2 Inicializace(void)
{
 GPIO InitTypeDef GPIO InitStructure;
 EXTI_InitTypeDef EXTI_InitStructure;
 NVIC InitTypeDef NVIC InitStructure;
 RCC APB2PeriphClockCmd(RCC APB2Periph GPIOA, ENABLE);
 GPIO InitStructure.GPIO Pin = PS2 CLOCK PIN | PS2 DATA PIN;
 GPIO InitStructure.GPIO Mode = GPIO Mode IN FLOATING;
 GPIO Init(PS2 PORT, & GPIO InitStructure);
 EXTI StructInit(&EXTI InitStructure);
 EXTI InitStructure.EXTI Mode = EXTI Mode Interrupt;
 EXTI InitStructure.EXTI Trigger = EXTI Trigger Falling;
 EXTI InitStructure.EXTI Line = EXTI Line1;
 EXTI InitStructure.EXTI LineCmd = ENABLE;
 EXTI Init(&EXTI InitStructure);
  /* Clear SC EXTI Line Pending Bit */
 EXTI ClearITPendingBit(EXTI1 IRQn);
  /* Configure the NVIC Preemption Priority Bits */
 NVIC PriorityGroupConfig(NVIC PriorityGroup 1);
 /* Clear the SC EXTI IRQ Pendinpbg Bit */
 NVIC ClearPendingIRQ(EXTI1 IRQn);
 NVIC InitStructure.NVIC IRQChannel = EXTI1 IRQn; //preruseni bude
  // od hrany CLK z PS2 na PA1
 NVIC InitStructure.NVIC IRQChannelPreemptionPriority = 0;
```

212

V.Váňa: CanSAT Book for Students – part 2



```
NVIC InitStructure.NVIC IRQChannelSubPriority = 0;
 NVIC InitStructure.NVIC IRQChannelCmd = ENABLE;
 NVIC Init(&NVIC InitStructure);
}
void EXTI1 IRQHandler(void) //obsluha preruseni od PA1
//void Preruseni PS2(void)
{
   uint8 t pom;
    if(EXTI GetITStatus(EXTI Line1) != RESET)
         {
           casPS2 = Cas(); // aktualizujeme cas posledniho preruseni
           pom = GPIO_ReadInputDataBit(PS2_PORT, PS2_DATA_PIN);
           if (cislo bitu==11) // prvni musi byt start bit
            {
                 znak = 0x00;
                 if (0 == pom)
                 {
                       --cislo bitu;
                 }
            } else
           if (cislo bitu<11 && cislo bitu>2)
                 // jde o datovy bit, musime ho zpracovat
            {
                 znak = (znak >> 1); // ochranime si predchozi hodnotu
                 if (0 != pom)
                       znak = znak + 0x80;
                                                   // vlozime 8. bit
rovny 1!
                 --cislo bitu;
            } else
            { // muse jit o bit paritni nebo stop, ty nezpracovavame
                 if (--cislo bitu == 0)
                 {
                       // mame vsechno nactene, musime dekodovat klavesu
                       dekoduj klavesu(znak);
                       cislo bitu = 11;
                                              // nyni prijde start bit
                 }
           /* Clear the Key Button EXTI line pending bit */
           EXTI ClearITPendingBit(EXTI Line1);
      }
}
  * @brief Dekodovani scan kodu klavesy na prislusny znak
  * @param Scan kod klavesy
  * @retval : None
  */
void dekoduj klavesu(uint8 t scan)
{
   static uint8 t is up=0, shift = 0;
```

213

V.Váňa: CanSAT Book for Students – part 2



```
uint8 t i;
    if (!is up) // posledni prijaty scan byl indetifikator up-key
    {
       switch (scan)
        {
                            // up-key
         case 0xF0 :
           is up = 1;
           break;
                            // Left SHIFT
          case 0x12 :
           shift = 1;
           break;
                            // Right SHIFT
          case 0x59 :
           shift = 1;
           break;
          /* na Caps lock, Levy Atl, Pravy Alt muzete doplnit kód
           * podle programu na http://eugenemcu.ru/publ/13-1-0-75 */
         default:
                 if(!shift) // kdyz neni shift
                 {
                       i = 0;
                       for(i = 0; obyc[i]!=scan && obyc[i]!=0; i+=2);
                       if (obyc[i] == scan) {
                            put_to_buff(obyc[i+1]);
                       }
                 } else { // se shiftem
                       for(i = 0; seshiftem[i]!=scan && seshiftem[i]!=0;
i+=2);
                       if (seshiftem[i] == scan) {
                            put to buff(seshiftem[i+1]);
                       }
                 }
           break;
        }
    } else {
        is up = 0;
        // Manual pravi ze: Two 0xF0 in a row not allowed!!
        switch (scan)
        {
         case 0x12 :
                         // Left SHIFT
           shift = 0;
           break;
                        // Right SHIFT
         case 0x59 :
           shift = 0;
           break;
       }
    }
```

214



```
* @brief Vlozeni dekodovaneho znaku do bufferu
 * @param Scan kod klavesy
 * @retval : None
 */
void put to buff(uint8 t ch)
{
   if (buffer pointer<PS2 BUFSIZE) // jeste tam mame misto
   {
       *buffer = ch; // soupneme to tam
buffer_pointer++; // posuneme ukazatel
   }
}
/**
 * @brief Cteni z klavesnice
 * @param None
 * @retval : nacteny znak nebo 0!
  */
uint8 t keyboard getchar(void)
{
   uint8_t byte = 0;
   if (buffer pointer>0) // neco tam mame
       yte = *buffer; // vytahneme to
buffer_pointer--; // posuneme ukazatel
    byte = *buffer;
    }
  return byte;
}
 * @brief Zkontrolujeme zda se nezasekla klavesnice
  * @param None
  * @retval : None
 */
void Kontrola hodin(void)
{
       IO uint32 t delta;
     if (cislo bitu != 11)
      {
                                // zatim jen nacteme cas
           delta = Cas();
           if (delta < casPS2)</pre>
            { // citac pretekl
                 delta = (0xFFFFFFFF - casPS2) + delta;
           } else
            { // citac nepretekl
                 delta = delta - casPS2;
            }
           {
                 cislo_bitu = 11; // zkusime cekat na novy znak
```

V.Váňa: CanSAT Book for Students – part 2 215



```
}
}
```

A testovací main.c

```
/* STM32 VL Discovery Library
* priklad 16 - PS2 hlavesnice ... PS2 clk PA1 PS2 data PA2
* ver 1.0 for SPSE Jecna lectures - ukazka obsluhy preruseni
* Created Vd on: Jul 20, 2011
*/
#definePS2_PORTGPIOA#definePS2_CLOCK_PINGPIO_Pin_1#definePS2_DATA_PINGPIO_Pin_2
/* Includes -----
                       _____
#include <stddef.h>
#include "stm32f10x.h"
#include "STM32vldiscovery.h"
#include "LCD.h"
#include "PS2.h"
 IO uint8 t c;
uint8 t LCD Text[] = "Test PS2 klavesnice.";
static uint32 t CasTick = 0; // centralni casove jednotky
/* Private function prototypes -----
void GPIO Inicialization(void);
void Delay( IO uint32 t nTick);
                             ||
void bliknutiLEDMODRA(void);
void bliknutiLEDZELENA(void);
//void PinInputZ b (u32 *adr, u8 bit);
int main(void)
{
  /* Nastaveni casovace SysTick na 1 msec preruseni */
 if (SysTick Config(SystemCoreClock / 1000))
  {
     /* Capture error */
     while (1);
  }
  LCD Inicialization();
  GPIO Inicialization();
  PS2 Inicializace();
  bliknutiLEDMODRA();
  printLCD(LCD Text);
  Delay(0xAFFFF);
  LCD Clear();
 while (1)
```

216


```
{
    bliknutiLEDZELENA();
      c = keyboard getchar();
    if(c != 0)
    {
      if(c != 13)
      {
        SendCharLCD(c);
      } else
                                   // po entru vymazeme LCD
           LCD Clear();
    }
                                   // kontrola na chybu klavesnice
     Kontrola hodin();
  }
}
void GPIO Inicialization(void)
{
 STM32vldiscovery_LEDInit(LED3);
 STM32vldiscovery_LEDInit(LED4);
 STM32vldiscovery_PBInit(BUTTON_USER, BUTTON MODE GPIO);
 STM32vldiscovery_LEDOff(LED3);
 STM32vldiscovery_LEDOff(LED4);
}
void bliknutiLEDMODRA(void)
{
     STM32vldiscovery_LEDOn(LED4); // zapnem LED4 - modra
     Delay(0x5FFFF);
     STM32vldiscovery LEDToggle(LED4); // vypnem LED4 - modra
     Delay(0x5FFFF);
}
void bliknutiLEDZELENA(void)
{
      STM32vldiscovery_LEDOn(LED3); // zapnem LED3 - zelena
      Delay(0x5FFFF);
      STM32vldiscovery LEDToggle(LED3); // vypnem LED3 - zelena
      Delay(0x5FFFF);
}
void Delay( IO uint32 t nTick)
{
 for(; nTick != 0; nTick--);
}
void ZmenaCasu(void)
{
   CasTick++;
}
uint32 t Cas(void)
{
    return CasTick;
```

V.Váňa: CanSAT Book for Students – part 2

ARM STM32 programming



2.15 GPS

GPS moduly, které lze v současné době zakoupit jsou převážně moduly, které se připojují k PC pomocí USB. Před několika lety bylo možné zakoupit takové moduly připojitelné k PC přes COM1 či COM2 se signály dle RS232. Vzhledem k tomu, že nové PC již obvykle nejsou rozhraním COM vybaveny, je obtížné získat GPS modul připojitelný k PC přes sériový port. Bohužel právě takový GPS potřebujeme pro náš CanSAT.

Je tu ještě jedna možnost a to GPS modul připojitelný k PDA. Tuto možnost jsme využili. Následující příklady budou využívat právě takový GPS modul, v našem případě jsem použil Navilock NL-303P GPS PDA Reciever SiRF III.



Pro použití v CanSatu ovšem nebudeme potřebovat ani kryt, ani těžký magnet.





Modul je napájen 5V a zapojení přívodního kabelu s konektory ukazuje obr.



Pro účely testování a vývoje zatím ponecháme modul v původním stavu a připojíme se k němu pomocí PS2 konektoru. Jak vidíme, je tento konektor zapojen jinak, než PS2 konektory v PC a nesmíme ho do PC přes PS2 připojovat. Má navíc jiné úrovně signálů, než klávesnice či myš připojované přes PS2 k PC.

Signály RX a TX mají úroveň RS232 a k našemu STM32 ho připojíme přes MAX3232. Můžeme ho připojit i k PC, pokud má náš počítač ještě COM rozhraní. K tomuto účelu si zhotovíme redukci a nezapomeneme připojit 5V napájení pro GPS modul. Pak si jeho funkci můžeme ověřit např. programem SiRFDemo.

Pozn. Při realizaci výše uvedených propojení však nic nefungovalo. Teprve zobrazením signálu TX z GPS modulu se ukázalo, že se tento signál po připojení k MAX3232 či COM portu PC zkreslí natolik, že přijímaný signál je interpretován jako odlišné znaky. Pomohlo vřazení odporu 220 Ω mezi TX GPS modulu a RxD COM portu PC či STM32 (PA10). TxD u STM32 je PA9.









Inicializace GPS

```
void init gps(void)
{
    Delay(0x2FFFFF);
      Strcpy (T Buff, "\r\n");
      USART ITConfig(USART1, USART IT TXE, ENABLE); //odeslani prazdneho
//retezce a CR LF
    Delay(0x0FFFFF);
    Strcpy (T Buff, "\r\n");
      USART ITConfig(USART1, USART IT TXE, ENABLE); //odeslani prazdneho
//retezce a CR LF
    Delay(0x0FFFFF);
// send_str("$LTC,1000,0*6A\r\n");
                                               // GPS Interrupt
// Strcpy (T_Buff, "$LTC,1000,0*6A\r\n"); // nastavime co se odesle po
//USART
          USART ITConfig(USART1, USART IT TXE, ENABLE);
11
       Delay(0x0FFFFF);
```



```
11
    send str("$PSRF100,1,4800,8,1,0*0E\r\n"); // SetSerialPort,
//4800Bd, 8N1
       Strcpy (T Buff, "$PSRF100,1,4800,8,1,0*0E\r\n"); // nastavime co
//se odesle po USART
           USART ITConfig(USART1, USART IT TXE, ENABLE);
Delay(0x0FFFFF);
/* SiRF nastaveni s*/
/* GGA enable rate=1 s*/
     Strcpy (T Buff, "$PSRF103,00,00,01,01*25\n\r");// nastavime co se
//odesle po USART
           USART ITConfig(USART1, USART IT TXE, ENABLE);
       Delay(0x0FFFFF);
/* GLL disable
                       */
     Strcpy (T Buff, "$PSRF103,01,00,00,01*25\n\r");// nastavime co se
//odesle po USART
           USART ITConfig(USART1, USART IT TXE, ENABLE);
Delay(0x0FFFFF);
/* GSA enable rate = 1s */
     Strcpy (T Buff, "$PSRF103,02,00,01,01*27\n\r"); // nastavime co se
//odesle po USART
           USART ITConfig(USART1, USART IT TXE, ENABLE);
       Delay(0x0FFFFF);
/* GSV enable rate = 1s
                            */
     Strcpy (T Buff, "$PSRF103,03,00,01,01*26\n\r"); // nastavime co se
//odesle po USART
           USART ITConfig(USART1, USART IT TXE, ENABLE);
       Delay(0x0FFFFF);
/* RMC enable rate 1 sec */
     Strcpy (T Buff, "$PSRF103,04,00,01,01*21\n\r");// RMC enable rate 1
//sec - nastavime co se odesle po USART
          //Strcpy (T Buff, "$PSRF103,04,00,00,01*20\n\r"); // RMC disble
//- nastavime co se odesle po USART
           USART ITConfig(USART1, USART IT TXE, ENABLE);
       Delay(0x0FFFFF);
/* VTG disable
                                   */
     Strcpy (T Buff, "$PSRF103,05,00,00,01*21\n\r");// nastavime co se
//odesle po USART
           USART ITConfig(USART1, USART IT TXE, ENABLE);
       Delay(0x40FFFF);
}
```



RealTerm: Serial Capture Program 2.0.0.57		
[18 \$PSRF100,1,4800,8,1,0*0E(8 \$PSRF103,00,00,01,01*25[F08 \$PSRF103,01,00,00,01*25[F08 \$PSRF103,02,00,01,01*25[F08 \$PSRF103,02,00,01,01*25[F08 \$PSRF103,02,00,01,01*26[F08 \$PSRF103,04,00,01,01*21[F08 \$PSRF103,05,00,00,01*21[F08		
Display Port Capture Pins Send Echo Port 121 Baud 4800 Port 8 8 6 7 6 6 7 6 6 6 6 6 6 7 6 6 6 6 6 7 6 7 6 7 6 7 6 7 7 6 7 6 7 8 8 7 7 8 8 8 7 8 8 7 8 9 9 9 9 9 9 9 9	C 12C-2 12CMisc Misc Dpen Spy Change Control Software Flow Control Receive Xon Char: 17 Transmit Xoff Char: 19 Winsock is: C Raw C Telnet	Clear Freeze
UART receiver framing error	Char Count:354	CPS:0 Port: 8 4800 8N1 None

Pozn.:

Posláním odpovídajících řetězců do GPS modulu se povolí/zakáže posílání daných typů zpráv SiRF tvaru \$PSRF103,05,00,01,01*20 kde

\$PSRF103	
05	00=GGA
	01=GLL
	02=GSA
	03=GSV
	04=RMC
	05=VTG
00	mode, 0=set rate, 1=query
01	rate in seconds, 0-255
01	checksum 0=no, 1=yes
*20	checksum

 Takže máme následující možnosti:

 GGA povolit (rate = 1s):
 "\$PSRF103,00,00,01,01*25\n"

 GGA zakázat (rate = 0s):
 "\$PSRF103,00,00,00,01*24\n"

GLL povolit (rate = 1s):	"\$PSRF103,01,00,01,01*26\n"
GLL zakázat (rate = 0s):	"\$PSRF103,01,00,00,01* 25 \n"

GSA	povolit (rate = 1s):	"\$PSRF103,02,00,01,01*27\n"
GSA	zakázat (rate = 0s):	"\$PSRF103,02,00,00,01*26\n"

V.Váňa: CanSAT Book for Students – part 2 223



GSV povolit (rate = 1s):	"\$PSRF103,03,00,01,01*26\n"
GSV zakázat (rate = 0s):	"\$PSRF103,03,00,00,01*27\n"
RMC povolit (rate = 1s):	"\$PSRF103,04,00,01,01*21\n"
RMC zakázat (rate = 0s):	"\$PSRF103,04,00,00,01*20\n"

V našem příkladu jsem GGA, GSA, GSV a RMC povolil, GLL a VTG zakázal.

Pozn. GPS defaultně je na 4800Bd 8N1 Po připojení GPS modulu, tento vysílá např.

RealTerm: Serial Capture Program 2.0.0.57	and some of	
<pre>\$GPGGA.143909.000.5007.8173.N.01439.6 \$GPGSA.A.3.12.25.21.29</pre>	5383, E, 1, 04, 2.3, 240.5, .3, 1.0×39 ^{(klf} , .23, 31, 45, 297, 12, 40, 1 .22, 10, 07, 049, 21, 06, 1 .23, 00, 351, *41 ^{(klf}) .6383, E, 0.24, 15.39, 28 5390, E, 1, 04, 2.3, 239.6, .3, 1.0×39 ^{(klf}) .6390, E, 0.41, 351.44, 2 5399, E, 1, 04, 2.3, 238.6, .3, 1.0×39 ^{(klf}) .6399, E, 0.41, 350.04, 2 6400, E, 1, 04, 2.3, 238.4, .3, 1.0×39 ^{(klf}) .6400, E, 0.45, 341.53, 2	M, 45.3, M, 0000*50 (klf 04, 33*73 (klf 88, 41*7B (klf 0711, *3E (klf M, 45.3, M, 0000*52 (klf 80711, *0B (klf M, 45.3, M, 0000*54 (klf 80711, *09 (klf M, 45.3, M, 0000*55 (klf 80711, *00 (klf
Baud 4800 ▼ Port 8 ▼ Parity Data Bits Stop Bits ● 1 bit C 2 bits Odd C 7 bits C 6 bits ● 1 bit C 2 bits Mark C 5 bits C 0 TR/DSR C RS485-rts	Open Spy Change Software Flow Control Receive Xon Char: Transmit Xoff Char: W9 Winsock is: Raw Telnet	Status Disconnect RXD (2) TXD (3) CTS (8) DCD (1) DSR (6) Ring (9) BREAK Error
UART receiver framing error	Char Count:4240	CPS:480 Port: 8 4800 8N1 None

\$GPGGA,145058.000,5007.8020,N,01439.6458,E,1,05,3.4,278.0,M,45.3,M,,0000*54

zeme**pisná šírka:** 5007.8020 severní šírky => 50°7,8020'N = 50°7'48,12"N **zem**e**pisná délka:** 01439.6458 východní délky => 14°39,6458'E = 14°39'38,74"E **nadmorská výška:** 278.0m

\$GPRMC,151952.000,A,5007.7931,N,01439.6520,E,0.19,27.22,280711,,*38

Čas: 15:19:52 UTC tj. 17:19:52 letního času v CZ Datum: 28.7.2011

Program main.c zobrazující na LCD veškerá přijatá data (obdobně jako např. RealTerm na PC)V.Váňa: CanSAT Book for Students – part 2224ARM STM32 programming



/* STM32 VL Discovery Library * vysilaní a prijem pres USART PA9 je Tx PA10 je Rx * ver 1.0 datum a cas z GPS druzice * Vd for SPSE Jecna lessons - prijem 4800Bd znaku - do/z GPS */ /* Includes -----___*/ #include <stddef.h> #include <stdio.h> #include <string.h> #include "stm32f10x.h" #include "STM32vldiscovery.h" #include "LCD.h" /* Private typedef --------*/ /* Private define -----_____ ___*/ #define BUFF SIZE 100 /* Private macro ---------*/ /* Private variables --------*/ // buffery na prijimane a odesilane znaky uint8 t R Buff[BUFF SIZE]=""; uint8_t T_Buff[BUFF_SIZE]=""; uint8 t LCD Message1[] = "Prijem GPS 4800"; /* Private function prototypes · ---*/ void USART Inicializace(void); void GPIO Inicialization(void); void init gps(void); int fputc(int ch, FILE * f); void Delay(IO uint32 t nTick); uint8 t Strcmp (const uint8 t * s1, const uint8 t * s2); // porovnani str void Strcpy(uint8 t *d1, const uint8 t *s1); // kopirovani str void vynulovaniRXregistru(void); /* Private functions --------*/ int main(void) { GPIO Inicialization(); // aktivujeme USART USART Inicializace(); V.Váňa: CanSAT Book for Students – part 2 225 ARM STM32 programming



```
LCD Inicialization();
 STM32vldiscovery LEDOn(LED4); // LED4 - modra dioda
 printLCD(LCD Message1);
  init gps();
 Delay(0xAFFFFF);
 STM32vldiscovery LEDOn(LED3); // LED3 - zelena dioda pridal jsem
 LCD Clear();
 while (1)
  {
     if (RX prijato == 1) // neco jsme prijali - zakoncene CR
     LCD Clear(); // vymazani LCD a navrat kurzoru na prvni pozici prvniho
radku
     Strcpy (LCD Message1,R Buff );
     printLCD(LCD Message1);
     vynulovaniRXregistru();
     RX prijato = 0;
                            // vymazeme flag ukonceni prijmu
      }
  }
}
void GPIO Inicialization(void)
{
 STM32vldiscovery LEDInit(LED3);
 STM32vldiscovery_LEDInit(LED4);
 STM32vldiscovery_PBInit(BUTTON_USER, BUTTON_MODE_GPIO);
 STM32vldiscovery_LEDOff(LED3);
 STM32vldiscovery LEDOff(LED4);
}
void USART Inicializace(void)
{
     GPIO InitTypeDef GPIO InitStructure;
     USART InitTypeDef USART InitStructure;
     NVIC InitTypeDef NVIC InitStructure;
      /* pustime hodiny do periferie (protoze jde o alternativni funkci,
musi
      * jit hodiny i do AFIO periferie! */
      RCC APB2PeriphClockCmd(RCC APB2Periph GPIOA | RCC APB2Periph USART1 |
RCC APB2Periph AFIO, ENABLE);
     // PA9 je Tx
     GPIO InitStructure.GPIO Pin = GPIO Pin 9;
     GPIO InitStructure.GPIO Mode = GPIO Mode AF PP; // alternate
function!
     GPIO InitStructure.GPIO Speed = GPIO Speed 50MHz;
     GPIO Init(GPIOA, &GPIO InitStructure);
     // PA10 je Rx
     GPIO InitStructure.GPIO Pin = GPIO Pin 10;
     GPIO InitStructure.GPIO Mode = GPIO Mode IN FLOATING;
     GPIO Init(GPIOA, &GPIO InitStructure);
```



```
USART InitStructure.USART BaudRate = 4800;
      /* Hodnota 4800 je defaultne pouzivana GPS */
      USART InitStructure.USART WordLength = USART WordLength 8b;
      //8N1
      USART InitStructure.USART StopBits = USART StopBits 1;
      USART InitStructure.USART Parity = USART Parity No;
      USART InitStructure.USART HardwareFlowControl =
USART HardwareFlowControl None;
      USART InitStructure.USART Mode = USART Mode Rx | USART Mode Tx;
      USART
           _Init(USART1, &USART_InitStructure);
      USART
           ITConfig(USART1, USART IT RXNE, ENABLE);
      USART Cmd (USART1, ENABLE);
      // konfigurace preruseni
      NVIC PriorityGroupConfig(NVIC PriorityGroup 1);
     NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQn;
     NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
     NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
     NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
     NVIC Init(&NVIC InitStructure);
}
void init gps(void)
{
     Delay(0x2FFFFF);
      Strcpy (T Buff, "\r\n");
      USART ITConfig(USART1, USART IT TXE, ENABLE); //odeslani prazdneho
//retezce a CR LF
    Delay(0x0FFFFF);
     Strcpy (T_Buff, "\r\n");
      USART ITConfig(USART1, USART IT TXE, ENABLE); //odeslani prazdneho
//retezce a CR LF
    Delay(0x0FFFFF);
// send str("$LTC,1000,0*6A\r\n");
                                               // GPS Interrupt
// Strcpy (T Buff, "$LTC,1000,0*6A\r\n"); // nastavime co se odesle po
//USART
11
           USART ITConfig(USART1, USART IT TXE, ENABLE);
       Delay(0x0FFFFF);
     send str("$PSRF100,1,4800,8,1,0*0E\r\n"); // SetSerialPort,
//4800Bd, 8N1
       Strcpy (T Buff, "$PSRF100,1,4800,8,1,0*0E\r\n"); // nastavime co
//se odesle po USART
           USART ITConfig(USART1, USART IT TXE, ENABLE);
Delay(0x0FFFFF);
/* SiRF nastaveni s*/
/* GGA enable rate=1 s*/
     Strcpy (T Buff, "$PSRF103,00,00,01,01*25\n\r");// nastavime co se
//odesle po USART
           USART ITConfig(USART1, USART IT TXE, ENABLE);
        Delay(0x0FFFFF);
/* GLL disable
                       */
```

227



```
Strcpy (T Buff, "$PSRF103,01,00,00,01*25\n\r");// nastavime co se
//odesle po USART
            USART ITConfig(USART1, USART IT TXE, ENABLE);
Delay(0x0FFFFF);
/* GSA enable rate = 1s */
     Strcpy (T Buff, "$PSRF103,02,00,01,01*27\n\r");// nastavime co se
//odesle po USART
            USART ITConfig(USART1, USART IT TXE, ENABLE);
        Delay(0x0FFFFF);
/* GSV enable rate = 1s
                              */
     Strcpy (T Buff, "$PSRF103,03,00,01,01*26\n\r");// nastavime co se
//odesle po USART
           USART ITConfig(USART1, USART IT TXE, ENABLE);
       Delay(0x0FFFFF);
/* RMC enable rate 1 sec
                             */
     Strcpy (T_Buff, "$PSRF103,04,00,01,01*21\n\r");// RMC enable rate 1
//sec - nastavime co se odesle po USART
           //Strcpy (T Buff, "$PSRF103,04,00,00,01*20\n\r"); // RMC disble
//- nastavime co se odesle po USART
           USART ITConfig(USART1, USART IT TXE, ENABLE);
       Delay(0x0FFFFF);
/* VTG disable
                                    * /
     Strcpy (T Buff, "$PSRF103,05,00,00,01*21\n\r"); // nastavime co se
//odesle po USART
           USART ITConfig(USART1, USART IT TXE, ENABLE);
        Delay(0x40FFFF);
}
uint8 t Strcmp (const uint8 t * s1, const uint8 t * s2)
{
    for(; *s1 == *s2; ++s1, ++s2)
       if(*s1 == 0 || *s1 == 0x0d)
           return 0;
    return *(unsigned char *)s1 < *(unsigned char *)s2 ? -1 : 1;
}
void Strcpy(uint8 t *d1, const uint8 t *s1)
 uint8 t i;
  for (i=0; s1[i] != '\0'; ++i)
   d1[i] = s1[i];
     d1[i] = ' \setminus 0';
}
void Delay( IO uint32 t nTick)
  for(; nTick != 0; nTick--);
```

228

V.Váňa: CanSAT Book for Students – part 2

ARM STM32 programming



```
/* Zapis znaku 'ch' do souboru 'f' */
int fputc(int ch, FILE * f)
 {
  /* Predani znaku pro odeslani UARTem */
  USART SendData(USART1, (u8) ch);
  /* Cekame, dokud neni prenos dokoncen */
       while (USART GetFlagStatus(USART1, USART FLAG TXE) == RESET);
        return ch;
 }
void vynulovaniRXregistru(void)
{
  uint8 t i;
 for(i=0; i<40; i++)</pre>
  R Buff[i] = ' \setminus 0';
  ;
}
```

Protože např. \$GPGGA,155153.000,5007.7974,N,01439.6481,E,1,05,1.4,290.9,M,45.3,M,,0000*51 má délku 75 znaků, proto jsem zvolil

#define BUFF SIZE 100

K získání a zobrazení dat jako je čas, datum, souřadnice musíme ovšem přijaté textové řetězce dále zpracovat. **Program17b** např. zobrazuje jen zprávy GGA. Tento příklad je stejný jako **Program17**, liší se jen funkcí **main(void)**

```
int main(void)
{
 GPIO Inicialization();
 // aktivujeme USART
 USART Inicializace();
 LCD Inicialization();
 STM32vldiscovery_LEDOn(LED4); // LED4 - modra dioda
 printLCD(LCD Message1);
  init gps();
 Delay(0x0FFFFF);
 STM32vldiscovery LEDOn(LED3); // LED3 - zelena dioda pridal jsem
 LCD Clear();
 while (1)
  {
     if (RX prijato == 1) // neco jsme prijali - zakoncene CR
     LCD Clear(); // vymazani LCD a navrat kurzoru na prvni pozici prvniho
radku RMC
      if ((R Buff[0] == '$')&(R Buff[1] == 'G')&(R Buff[2] == 'P'))
```

229

V.Váňa: CanSAT Book for Students – part 2



```
if ((R Buff[3] == 'R')&(R Buff[4] == 'M')&(R Buff[5] == 'C')) //RMC
          {
           Strcpy (LCD Message1,R Buff );
           printLCD(LCD Message1);
          }
      if ((R Buff[3] == 'G')&(R Buff[4] == 'G')&(R Buff[5] == 'A')) //GGA
          {
           Strcpy (LCD Message1,R Buff );
           printLCD(LCD Message1);
          }
       }
       vynulovaniRXregistru();
       RX prijato = 0; // vymazeme flag ukonceni prijmu
      }
 }
}
```

Další **program17c** ukazuje hodiny řízené GPS. Od předchozích **program17** a **program17b** se opět liší jen funkcí **main(void)**

```
int main(void)
{
 GPIO Inicialization();
 // aktivujeme USART
 USART Inicializace();
 LCD Inicialization();
 STM32vldiscovery LEDOn(LED4); // LED4 - modra dioda
 printLCD(LCD Message1);
  init_gps();
 Delay(0x0FFFFF);
 STM32vldiscovery LEDOn(LED3); // LED3 - zelena dioda pridal jsem
 LCD Clear();
 while (1)
  {
     if (RX prijato == 1) // neco jsme prijali - zakoncene CR
     //LCD Clear(); // vymazani LCD a navrat kurzoru na prvni pozici
prvniho radku
                 RMC
      if ((R Buff[0] == '$')&(R Buff[1] == 'G')&(R Buff[2] == 'P'))
       {
       if ((R Buff[3] == 'R')&(R Buff[4] == 'M')&(R Buff[5] == 'C')) //RMC
          {
           LCD Message1[0] = R Buff[7];
```

V.Váňa: CanSAT Book for Students – part 2 230





LCD pak ukazuje čas ve formátu 14:58:12 UTC

Na dvouřádkovém LCD bychom ještě na druhém řádku mohli zobrazovat datum, který je rovněž zprávou RMC přenášen, museli bychom ovšem R_Buff procházet a "odpočítat" příslušný počet oddělovacích čárek a poté zkopírovat a zobrazit údaj o aktuálním datumu. Ze zpráv RMC a GGA můžeme též určit zeměpisné souřadnice.



3 Práce v prostředí IAR Embedded Workbench

3.1 Stažení instalačního souboru

Na stránkách firmy IAR Systems přejdeme na download a dále vybereme produkt **Kick Start edition of IAR Embedded Workbench for ARM**:



Klikneme na Continue, dostaneme registrační formulář



Product Registra	tion and Download		
IAR Embedded Workbe	nch for ARM, v. 6.20, 32K K	ckstart Edition	
When you have register that is required to instal address correctly!	ed below, you will receive an the software. Please make	e-mail containing informa sure to spell your e-mail	tion
First name *			
Last name *			
Last name *			
Last name * Title			
Last name * Title Please specify:	•		
Last name * Title Please specify: Email *	•		
Last name * Title Please specify: Email *	•		
Last name * Title Please specify: Email *	•		

Po vyplnění a odeslání formuláře se objeví





Poté obdržíme e-mail s licenčním číslem a linky pro stažení sw.



Thank you for your registrati	ion of the product:
AR Embedded Workbench	for ARM, v. 6.20, 32K Kickstart Edition
Download software	
• English version (EXE, 478 I	MB)
Download from:	
Europe (HTTP site)	
Europe (FTP site)	
Japan	
Japanese version (EXE, 4	78 MB)
Europe (HTTP site)	
Europe (FTP site)	
Japan	
When installing the softwar License Key. Please use th	e, you will be asked for a License Number and a ese values:
License Number:	
9551-027-658-0108	

Stáhneme instalační program (např. EWARM-KS-CD-6204-2.exe)



EWARM-KS-CD-6204-2.	exe z supp.iar.com	
Zbývající čas (odhad): Cíl stahování:	10 hod 20 min (zkop írováno 2,58 MB z 478 MB) C:\Data \FWARM-KS-CD-6204-2 exe	
Přenosová rychlost:	13,0 kB/s	
Tento dialog po dokor	nčení stahování zavřít	
	Otevřít Otevřít složku Stomo	
Tato etažoná n	oložka byla zkontrolována filtrem SmartScreen a není	

Tím jsme získali instalační soubor EWARM-KS-CD-6204-2.exe Jeho instalací získáme(popis v eng. z IAR webu)

IAR Embedded Workbench® for ARM - KickStart edition

IAR Embedded Workbench is an Integrated Development Environment with a complete and easy-to-use set of C/C++ cross compiler and debugger tools for professional embedded applications.

- KickStart 32KB IAR C/C++ Compiler for ARM
- Project manager
- Editor
- Linker and librarian tools
- C-SPY® debugger

• Full integration with IAR J-Link

- Example projects for STM32 include:
- GettingStarted

This example project shows basic use of I/O, timer and the interrupt controllers. Displays running lights on the board LED's.

• LCD_Demo

This example project shows basic use of parallel I/O, timer, interrupt controller, ADC and interface to a LCD HD44780 compatible module. The LCD will display the position of the potentiometer in percents.

USBMouse

This example project shows how to implement a USB HID mouse. When host install needed driver a mouse's cursor begin to move in a rectangle shape. The WAKE-UP button is used for USB resume when the device is suspended.

The KickStart edition of IAR Embedded Workbench is fully functional, but the code size is limited.



3.2 Instalace IAR

c:\inst*.*			*	•
Název	Přípona	Velikost	Datum	Atr
\$ []		<dir></dir>	13.07.2011 13:55	9—
EWARM-KS-CD-6204-2	exe 50	389 490	12.07.2011 11:53	3-a-

Spustíme instalační program

IAR Embedded V	Vorkbench
• • • =	Destination folder
O IAR	C:\Users\vladimir\AppData\Local\Temp\RarSFX0 Browse
SYSTEMS	Installation progress
	Install Cancel

Probíhá extrakce potřebných souborů





V výše uvedené nabídce již zvolíme vlastní instalace





Klikneme na Next



IAR Embedded Workbench f	r ARM 6.20.4 Kickstart
Online Registration	OIAR SYSTEMS
If you already have click Next and prod	registered and received your license number and key, eed with the installation.
If you don't have a Register	icense, register now to get your license number and key.
InstallShield	< Back Net

Opět **Next**

icense Agreement Please read the following license agreement carefully.	SYSTEM
SOFTWARE LICENSE AGREEMENT (concerning Products: IAR Embedded Workbench®, visualSTAT PowerPac™) from IAR Systems AB	E® and IAR
PREAMBLE THIS SOFTWARE LICENSE AGREEMENT (THE "AGREEMEN" PRODUCTS LICENSED BY IAR SYSTEMS AB INCLUDING AN" SUBSIDIARIES ("IAR SYSTEMS") OR DISTRIBUTORS ("IAR D YOU ("THE LICENSEE").	T") APPLIES TO Y OF ITS VISTRIBUTORS"), TO
 I accept the terms of the license agreement I do not accept the terms of the license agreement 	Print
< Back	Next



Odsouhlasíme licenční podmínky a opět klikneme na Next

nter User	Information Information SYS
Enter your Kickstart lic	name, the name of your company and your IAR Embedded Workbench for ARM cense number.
Name:	Microsoft
Company:	Microsoft
	Can be found on the CD cover, or via e-mail registration
License#:	
illShield —	

Vyplníme uživatelské informace a licenční číslo



inter User	Information
Enter your Kickstart lie	name, the name of your company and your IAR Embedded Workbench for ARM cense number.
Name:	Vladimir Vana
Company:	SPSE
	Can be found on the CD cover, or via e-mail registration
License#:	9551-027-658-0108
allShield	

Klikneme **Next**

nter Licen	se Key	I A SYSTE
The license QuickStart I If you have	key can be either y key (found on the C received the perma	your QuickStart key or your permanent key. If you enter the D cover), you have 30 days to try the product out. anent key via email, you paste it into the License Key textbox.
License #:	9551-027-658-01	08
License Key	r;	
Read Lice	ense Key From File	Browse
Read Lice C:	ense Key From File	Browse
Read Lice C: allShield —	ense Key From File	Browse

V.Váňa: CanSAT Book for Students – part 2

ARM STM32 programming



Vložíme klíč

	OIAI SYSTEM
The license key can be eit QuickStart key (found on t If you have received the p	ther your QuickStart key or your permanent key. If you enter the the CD cover), you have 30 days to try the product out. ermanent key via email, you paste it into the License Key textbox.
License #: 9551-027-65	8-0108
License Key:	
MK96QUHWFGT1ISCMM 5BS4NSJ03NBHFIWV02	48QSRKQ7NH4258HLTP92PXMOKUAFY1HBXFKDKZZCVTDFUZN 2YWDNOME34SLZVVIRZS0S01VRPY45V1Z1E2J1JFVCXTRZC#
Feature: EWARM-KS32 V expires 2036-07-12	/ersion: U2_WIN_I emporary license (Licno: 9551-027-658-0108),
Feature: EWARM-KS32 V expires 2036-07-12 Read License Key From	File
Feature: EWARM-KS32 V expires 2036-07-12 Read License Key From C:	File Browse
Feature: EWARM-KS32 V expires 2036-07-12 Read License Key From C: allShield	File Browse

Stiskneme Next



IAR Embedd	ed Workbench for ARM 6.20.4 Kickstart	x
Choose D Select fo	Destination Location Ider where setup will install files.	R
	Install IAR Embedded Workbench for ARM Kickstart to: C:\\Embedded Workbench 6.0 Kickstart Change	
InstallShield -	< Back Next > Cancel	

Znovu Next



elect Program Folder Please select a program folder.		OIA SYSTE
Setup will add program icons to the Pro	ogram Folder listed below. You ma olders list. Click Next to continue	y type a new folder
Program Folder	Olders list, Chert MEAL to Continue.	
IAB Embedded Workbench for ABM	6.20 Kickstart	
Existing Folders:		
Administrative Tools		
aSc Rozvrhy		
ASUS Utility		
Avira Deletiti 2253		
i bakalari 32010 Fasu Vusvědčení		
EMS		
E. JOE. J		*
EXCESO		
EXt2FS0		
IShield		
IIShield		

Opět Next



IAR Embedded Workbench for ARM 6.20.4 Kickstart	x
Ready to Install the Program The wizard is ready to begin installation.	AR
Click Install to begin the installation.	
If you want to review or change any of your installation settings, click Back. Click Cancel to the wizard.	exit
Install©hield	
Install5hield Cance	1

Konečně můžeme tlačítkem Install spustit vlastní instalaci



Setup Status					DIA
IAR Embedded Workb	ench for ARM	Kickstart is config	uring your new sol	tware install	ation.
Installing					
C:\\Embedded Work	bench 6.0 Kic	kstart\arm\lib\dl5	E_tbn.a		
					3
tallShield					

Instalace probíhá

Setup Status			I SYS	AR
IAR Embedded Work	pench for ARM Kicks	tart is configuring your ne	ew software installation.	
	Unpackin	g vendor documentation		
tallShield —				

V.Váňa: CanSAT Book for Students – part 2

ARM STM32 programming







Klikneme na Finish

Select the language f Workbench IDE	or the IAR Embedded	ок 🖓
Angličtina (Spojené sl	táty)	•
Angličtina (Spojené st	táty)	

Vybereme angličtinu pro prostředí IDE





IAR Embedded Workbench IDE máme nainstalovaný,





Ještě nainstalujeme drivery ST link pro USB



spořádat 🔻 🛛 Zahrnout	do knihovny 🔻	Sdílet s 🔻	Vypálit	Nová složka			
<pre>{E00AE62A- ^</pre>	Název položky Jlink Macraigor pemicro STComstickFT StellarisFTDI ST-Link	TDI		Datum změny 21.6.2011 15:57 21.6.2011 15:57 21.6.2011 15:57 21.6.2011 15:57 21.6.2011 15:57 21.6.2011 15:57	Typ Složka souborů Složka souborů Složka souborů Složka souborů Složka souborů	Velikost	

					-		
Data	Local Temp	► RarSFX0 ►	drivers 🕨	ST-Link	👻 🍫 Prohlec	lat: ST-Link	Q
nrnout	t do knihovny 🔻	Sdílet s 🔻	Vypálit	Nová složka		≣≡ ▼ [9
a_ ^ ¥	Název položky	<u>^</u>		Datum změny	Тур	Velikost	
	ST-Link_V2_U	JSBdriver		10.6.2011 20:42	Aplikace	11 696 kB	



Bronaving Solun		
Please wait while the InstallShield	Wizard prepares the setup.	L
STLinkDriver Setup is preparing the the setup process. Please wait.	e InstallShield Wizard, which will guide you through the	rest of
	1	
stallShield		

STLinkDriver	- InstallShield Wizard			×
Choose D Select fo	Destination Location older where setup will install files.			N
	Install STLinkDriver to: C:\\STMicroelectronics\st_tools	set	Ch	ange
InstallShield -		< Back	Next > 💦 🗌 C	ancel

V.Váňa: CanSAT Book for Students – part 2

ARM STM32 programming


TLinkDriver - InstallShield Wizard	
Ready to Install the Program The wizard is ready to begin installation.	
Click Install to begin the installation.	
If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.	
nstallShield Cancel]



STLinkDriver - InstallShield Wiz	ard
	InstallShield Wizard Complete The InstallShield Wizard has successfully installed STLinkDriver. Click Finish to exit the wizard.
	< <u>B</u> ack Finish Cancel

Instalace USB driveru pro ST je ukončena, zbývá kliknout na tlačítko **Finish**. Po připojení **STM32VL Discovery** se spustí Instalace software ovladače zařízení





🖉 👝 🕨 Počítač 🕨	Vyměnitelný disk (G:)		🝷 🍫 Prohledat: V	yměnitelný disk (G:)
spořádat 🔻 Sdílet s 🔻	Vypálit		and a strength		
Oblíbené položky	Název položky	Datum změny	Тур	Velikost	
归 Naposledy navštíver	STM8S - 8-bit Microcontrollers	11.12.2008 13:50	Zástupce internet	1 kB	N
📃 Plocha	🔊 STM32 (CORTEX M3) - 32-bit Microcontr	11.12.2008 13:21	Zástupce internet	1 kB	5
〕 Stažené soubory	STMicroelectronics - Microcontrollers - 8	11.12.2008 13:20	Zástupce internet	1 kB	

3.3 Začínáme pracovat s existujícím IAR projektem

IAR Embedded Workbench IDE se skládá z několika částí:



Naší první ukázkou práce s *IAR Embedded Workbench* bude práce s již existujícím projektem **Demo** otevíraným pomocí souboru **DISCOVER.eww**. Najdeme jej v **stm32vldiscovery_package**





v podadresáři Projects, v jeho podsložce Demo, v podsložce EWARMv5

Název	1Pří	Velikost	Datum	Atribut
	90303030303030	<dir></dir>	12.07.2011 09:28	
DISCOVER]		<dir></dir>	12.07.2011 09:28	
[settings]		<dir></dir>	12.07.2011 09:28	
DISCOVER	dep	12 066	13.07.2011 12:04	-a-
DISCOVER	ewd	22 885	12.07.2011 09:25	-a
DISCOVER	ewp	25 404	13.07.2011 12:04	-a
DISCOVER	eww	162	13.09.2010 13:17	-a
stm32f100_flash	icf	1 369	13.09.2010 13:17	-a

Všimněme si, že adresář **Demo** obsahuje podadresáře **EWARMv5, MDK-ARM** a **TrueSTUDIO** a že stejné podadresáře najdeme i ve všech podadresářích v příkladech **Examples**. Tím spolu s umístěním společných adresářů **Libraries a Utilities** a dílčích adresářů **src** a **inc** je docíleno toho, že tyto adresáře jsou společné pro projekty v prostředí **IAR Embedded Workbench**, stejně jako v prostředí **Keil uVision4** a v prostředí **Atolic TrueSTUDIO**.

V těchto prostředích najdeme i nastavení cest k souborům v výše uvedených adresářích. Dobře nám poslouží jako vzor pro nastavování cest ve vlastních projektech.



Balíček **stm32vldiscovery_package** získáme <u>http://www.st.com/stm32vldiscovery</u> v dolní části **Design Support** <u>http://www.st.com/internet/evalboard/product/250863.jsp</u> pod nápisem firmware jako **stm32vldiscovery_package.zip**.

Jeho popis pak najdeme v aplikační poznámce **AN3268 Application Note STM32VLDISCOVERY firmware package** ze <u>http://www.st.com/stonline/products/literature/an/17901.pdf</u>

Po stažení **stm32vldiscovery_package.zip** z tohoto souboru nejprve získáme adresář *stm32vldiscovery_package*, který umístíme do nějaké složky s našimi projekty. Poté spustíme **IAR Embedded Workbench** a v něm v menu **File – Open -- Workspace**



Poté vybereme workspace DISCOVER



			10.
Uspořádat 🔻 🛛 Nová s	ložka	8≡ ▼ [0
Plocha ^	Název položky	Datum změny	
Stazene soubory	DISCOVER	12.7.2011 9:28	
C Kaihaway	퉳 settings	12.7.2011 9:28	
	DISCOVER	13.9.2010 13:17	
Domácí skupina Počítač			k dispozi
🏭 Místní disk (C:)			
👝 Místní disk (E:) 🗸		Þ	
	and the second		
👝 Místní disk (E:) 👻	· · · · · · · · · · · · · · · · · · ·	F	

Klikneme na **Otevřít** , dostaneme



🔏 IAR Embedded Workbench IDE			- • ×
File Edit View Project Tools Wind	dow Help		
0 🖻 🖬 🕼 🎒 🐇 🖻 🛍 🗠	CH		
Workspace	×	main c	+ x
DISCOVER	•	/* Includes*/	
Files	₹: B]	<pre>#include "stm32F10x.h" #include "SIM32vldiscovery.h"</pre>	*
Hei CMSIS Hei CMSIS Hei CDC Hei EWARMv5 Hei StdPeriph_Driver Hei StdPeriph_Driver Hei StdPeriph_Driver Hei Output		<pre>/* Private typedef</pre>	Β
DISCOVER		<pre></pre>	•
× Messages		File	Line
801 ×			• •
Ready		Errors 0, Warnings 0 NLn 50 Col 1	

V levé části si všimneme obsahu okna souborů



File Edit View Project Tools Window	v H	elp
	 .	cip
	• }	~
		-î
DISCUVER		•
Files	82	D.
DISCOVER - DISCOVER	~	
H-p 🗀 CMSIS		
Core_cm3.c		
System_stm32f10x.c		
tartup_stm32f10x_md_vi.s		
HH StdPeriph_Driver		_
⊞ [C] misc.c		
THE CI Stm32110x_extl.C		_
Figure String 10x_gpic.c		
He stm32vldiscoverv		
STM32vldiscoverv.c		
📙 🗕 🖻 main.c		
📙 🖵 🖸 stm32f10x_it.c		
🛛 🖵 🖸 Output		
03		
		_
DISCOVER		

V něm označíme text **DISCOVER – DISCOVER** a pravým tlačítkem rozvineme místní menu



🔀 IAR Embedded Workbench IDE	A COLUMN TWO	and the later of t
File Edit View Project Tool	s Window Help	
D 🚅 🖬 🕼 🈹 🖪 I	8 N N N	
Workspace	×	main.c
DISCOVER		/* Includes
Files	8:: Bi	<pre>#include "stm32F10x.h" #include "STM32vldiscovery.h"</pre>
□ □	Options Make Compile Rebuild All Clean Stop Build Add Remove Rename Source Code Contro File Properties Set as Active	<pre>#include "STM32vldiscovery.h" Vate typedef Vate define LSE_FAIL_FLAG 0x80 Vate macro Vate consts Vate consts Vate variables Vate variables Vate variables Vate variables Vate = 0; Vate = 0; Vate function prototypes Vate function prototypes Vate functions V</pre>
DISCOVER		

V něm vybereme položku **Options**, dostaneme okno se záložkami. Jednotlivé záložky otevíráme a nastavujeme jejich obsah dle následujících obrázků.



General Options					
C/C++ Compiler Assembler	-				
Output Converter	Target	Output	Library Configuration	Library Options	MISRA-C:200 1
Build Actions Linker	Proc	essor va	riant		
Debugger	00	Core	Cortex-M3	*	
Simulator Angel GDB Server	0)evice	ST STM32F100xB		T +
IAR ROM-monitor J-Link/J-Trace TI Stellaris FTDI	End	an mode	FPI	J	
Macraigor	0 L	ittle	No	ne	*
PE micro	O E	Sig			
ST-LINK		BE32			
Third-Party Driver	9	DE0			

Vybereme CPU



Category: General Options C/C++ Compiler	Multi-file Compilation Discard Unused Publics					
Assembler Output Converter Custom Build	Code Optimizations Output List Preprocessor Diagnostics					
Build Actions Linker	Additional include directories: (one per line)					
Simulator Angel GDB Server IAR ROM-monitor	<pre>\$PROJ_DIR\$\\inc\ \$PROJ_DIR\$\\\Libraries\STM32F10x_StdPeriph_Driver\inc\ \$PROJ_DIR\$\\\Utilities\ \$PROJ_DIR\$\\\Libraries\CMSIS\CM3\CoreSupport\ \$PROJ_DIR\$\\\Libraries\CMSIS\CM3\DeviceSupport\ST\STM: -</pre>					
J-Link/J-Trace TI Stellaris FTDI	Preinclude file:					
Macraigor PE micro RDI ST-LINK Third-Party Driver	Defined symbols: (one per line) USE_STDPERIPH_DRIVER STM32F10X_MD_VL Preserve comments Generate #line directives					
10						

Důležité pro budoucí projekty (budeme upravovat) Additional include directories:

\$PROJ_DIR\$\..\inc\
\$PROJ_DIR\$\..\..\Libraries\STM32F10x_StdPeriph_Driver\inc\
\$PROJ_DIR\$\..\..\Utilities\
\$PROJ_DIR\$\..\..\Libraries\CMSIS\CM3\CoreSupport\
\$PROJ_DIR\$\..\..\Libraries\CMSIS\CM3\DeviceSupport\ST\STM32F10x

Defined symbols:

USE_STDPERIPH_DRIVER STM32F10X_MD_VL



Category:	Factory S	ettings
General Options C/C++ Compiler		
Assembler		
Output Converter	Jutput	
Custom Build Build Actions	Generate additional output	
Linker	Otation	
Debugger	bisser	
Simulator	Dinary	
Angel	Output file	
IAR ROM-monitor	Vovemide default	
J-Link/J-Trace	DISCOVER.bin	
TI Stellaris FTDI		
Macraigor		
PE micro		
ST-LINK		
Third-Party Driver		

Zvolíme formát výstupního souboru bingy a klikneme na **OK**



ategory:							Factory 9	Settings
eneral Options								
C/C++ Compiler								
Output Converter	Config	Library	Input	Optimizations	Output	List	#define	D + >
Custom Build	9				10000	1		
Build Actions	link	er config	iration fi	e				
Linker)vorido (dofoult					
Debugger		Annue (
Simulator		\$PROJ_DIR\$\stm32f100_flash.icf						
Angel		Eda						
GDB Server	1	COL						
J-Link/J-Trace	Config	uration fil	e symbo	definitions: (one	e per line)			1.1
TI Stellaris FTDI								
Macraigor								<u> </u>
PE micro								
RDI								
ST-LINK								
Third-Party Driver								
								-

Ještě konfigurace linkeru

/ector Table	Memory Regions	Stack/Heap Sizes	
.intvec start	0x08000000		



ector Table	Memory Regions	Stack/Heap Sizes
	Start:	End:
ROM	0x08000000	0x0801FFFF
RAM	0x20000000	0x20001FFF

A nastavení Debugeru na ST-Link

ategory:						Factory Setting:
eneral Options						<u></u>
C/C++ Compiler						
Assembler	Setup	Download	Images	Extra Ontione	Plugine	
Custom Build		Download	inages	Data Options	riugina	
Build Actions	Driver			Run to		
Linker	STI	INK	-			
Debugger	0.0			j main		
Simulator	Setu	ip macros				
Angel		Use macro f	le(s)			
GDB Server						
IAR ROM-monitor						
TI Stellaris FTDI						
Macraigor	Dev	ice descriptio	n file			
PE micro						
RDI		Overnae der	auit			
ST-LINK	ST	OOLKIT_DI	R\$\CONF	IG\debugger\S	T\iostm32f100	kdb.ddf
mird-Party Driver						
			13			



Category:	Factory Setting
C/C++ Compiler Assembler	
Output Converter Custom Build Build Actions	Setup Download Images Extra Options Plugins
Eulid Actions Linker Debugger Simulator Angel GDB Server IAR ROM-monitor J-Link/J-Trace TI Stellaris FTDI Macraigor PE micro RDI ST-LINK Third-Party Driver	 Suppress download Use flash loader(s) Override default board file \$TOOLKIT_DIR\$\config\flashloader\ST\RashSTM3. Edit

Nezapomeneme na SWD



Category:		Factory Settings
General Options C/C++ Compiler Assembler Output Converter Custom Build Build Actions Linker Debugger Simulator Angel GDB Server IAR ROM-monitor J-Link/J-Trace TI Stellaris FTDI Macraigor PE micro	ST-LINK Interface Interface ITAG I SWD	
RDI ST-LINK Third-Party Driver	2	

Nyní máme prohlédnutá veškerá nastavení, která s výhodou použijeme později ve vlastních projektech. Můžeme přistoupit k překladu, linkování a sestavení.



File Edit View Pr	oject Tools Window Help		
D 😂 🖬 🎒 Workspace DISCOVER	Add Files Add Group Import File List		
Files	Edit Configurations Remove		M32vldi:
└── Ĉ core_ └── Ĉ syste └── DOC	Create New Project Syste DOC EWARN Options Alt+F7		PASS_FI
⊢⊖ 🗀 EWARN			acro consts -
	Source Code Control	+	ariable
-⊞ C stm32	Make Compile	F7 Ctrl+F7	y = 0; 0;
	Rebuild All		red = 0; = 0;
-⊕ <mark>C</mark> stm32∨l	Clean Batch build	F8	uint32_ function lint32_t
l -⊞ 🖸 main.	Stop Build	Ctrl+Break	elay_De
□ □ □ 0utput	Download and Debug Debug without Downloading	Ctrl+D	unction
	Make & Restart Debugger	Ctrl+R	
Message:	Restart Debugger Download	Ctrl+Shift+R ▶	

V menu proto vybereme Project – Rebuild All . Po chvíli dostaneme





Přes USB připojíme startkit a



Po chvíli dostaneme





Klikneme a ikonku Stop Debugging

IAR Embedded Workbench IDE	And the second s
File Edit View Project Debug	Disassembly ST-LINK Tools Window Help
0 🚅 日 🗗 🎒 🐰 🗅 🛍	L トロコー
5 8 8 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	
Workspace ×	ma Stop Debugging - × D
DISCOVER	void Delay(uint32_t nTime);
Files 👫 📑	void TimingDelay_Decrement (void);
	/* Private functions
	/**
EWARMv5	* @brief Main program.
Hara Hara StdPeriph	* gparam None
Harring Harris	* Gretval None
User	*/
└─⊞ 🗀 Output	ALL CONTRACTOR
	{

Pokud startkit odpojíme a znovu připojíme, spustí se program – blikání zelené LED **PC9**. Je to vlastně ten program, se kterým je **STM32VL DISCOVERY** dodán od výrobce.

3.4.1 První vlastní IAR projekt (priklad01)



K IAR Embedded Wo	orkbench IDE	1				
File Edit View P	roject Tools	Window Help				
🗅 🖙 🖬 🕼	Add Files					
Workspace	Add Group.					
<u></u>	Import File	List				
Files	Edit Configu	urations	Pe			
	Remove		Ŷ			
	Create New	Create New Project				
	Add Existing	Add Existing Project Di Options Alt+F7				
	Options					
	Source Cod	e Control	+ uid			
	Make		F7			
	Compile		Ctrl+F7			
	Rebuild All					
	Clean					
	Batch build	F8 prod				
	Stop Build Ctrl+Brea					
	Download a	nd Debug	Ctrl+D n yo			
	Debug with	out Downloading				
	Make & Res	tart Debugger	Ctrl+R			
	Restart Deb	ugger Ctr	l+Shift+R 🗘			
	Download		+ PPO			
		Tutorials to make you familiar with the IDE and the features of the IAR C-SPY debugger.	For questions to use your IAI reporting a pro finding suppor resources.			
		fo a <				
Create a new project a	nd insert it into	the workspace				

V menu vybereme Project – Create New Project ...



Tool chain:	ABM	•
Draigat tamplata		
Froject template	5.	
Empty proje	ect	
ittin asm		
DLIB Estemally L		
Externally t	Julit executable	
Description:		
Creates an emp	ty project.	

Vybereme ARM a klikneme na **OK**



🔀 Uložit jako	1 1 1 1 1 1 1		a a	1	×
	M_priklady → IAR → priklad0	· ·	↓ Prohleda	ıt: priklad01	Q
Uspořádat 🔻 N	lová složka			• ==	0
 Knihovny Dokumenty Hudba Obrázky Videa Domácí skupina 	E	Název položk Hlec	y lání neodpovídaj	í žádné položky.	Datum z
🏭 Místní disk (C	.)				
👝 Místní disk (E:)	• •	III		•
Název souboru:	program01				-
Uložit jako typ:	Project Files (*.ewp)				•
🔿 Skrýt složky			UI	t Stor	10

Zvolíme název projektu

Reference with the second seco	
File Edit View Project Simula	ator Tools Window Help
🖸 🗲 🖬 🕼 🌆 🕹 🕯 🖻	2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
Workspace ×	IAR Information Center for ARM
Debug ▼ Files #: □ program01 ✓	
	IAR Information Center for ARM Here you will find all the information you need to get started: tutorials, example projects, user and reference guides, support information, and release notes. Systems

Nyní projekt uložíme File -- Save All





Jsme dotázáni na název workspace



🔀 Save Workspace As		6		an an	×
AR	M_priklady ► IAR ► priklad	01	• • 49	Prohledat: priklad01	٩
Uspořádat 🔻 N	lová složka			833 -	. 0
🚖 Oblíbené položi	y	^	Název položky	8 8	Datum z
🖳 Naposledy na	vštívené		🔰 inc		12.7.2011
📃 Plocha			🍶 settings		12.7.2011
🚺 Stažené soubo	ory	Ε	🎳 src		12.7.2011
 Dokumenty Hudba Obrázky Videa 					
🍓 Domácí skupina	I	-	•		P.
Název souboru:	program01				•
Uložit jako typ:	Workspace Files (*.eww)				•
le Skrýt složky				Uložit	orno

Zvolíme např. program01 a klikneme na **Uložit**. Podíváme se na vytvořené soubory a adresáře projektu

Název	Přípo	Velikost	Datum		Atribut
雀[]		<dir></dir>	12.07.2011 1	1:40	
[inc]		<dir></dir>	12.07.2011 1	1:38	
[settings]		<dir></dir>	12.07.2011 1	1:40	—
🔄 [src]		<dir></dir>	12.07.2011 1	1:38	
<pre>%program01</pre>	eww	163	12.07.2011 1	1:40	-8-
program01	ewp	45 958	12.07.2011 1	1:39	-a
program01	ewd	45 477	12.07.2011 1	1:39	-a
program01	dep	309	12.07.2011 1	1:40	-a-

Pozn. Adresáře **inc** a **src** jsem předem vytvořil.





Dále vytvoříme nový soubor výběrem v menu File – New -- File

🔀 IAR Embedded Workbench IDE	and the second
File Edit View Project Sim	ulator Tools Window Help
D 🛩 🖬 🕼 🐇 🖻 (¶ר אר
Workspace	IAR Information Center for ARM Untitled1
Debug 🔹	
Files 😤 🛤	
🗐 program01 🖌	

Do Untitlet1 napíšu krátký kód



File Edit View Pro	ject Simu	lator Tools Window Help	100
	X 🖻 🕻	3 0 0 0	. 4
workspace	×	IAR Information Center for ARM Untitled1 *	
Debug	•	int main (void)	
Files	8 B.	{ return (0) :	
🗐 program01	. v	iecum(op,	



Poté uložíme File – Save As...



Uspořádat 👻 Nová složka				•
🔆 Oblíbené položky	-	Název položky	*	Datum
📃 Naposledy navštívené		鷆 inc		12.7.201
Nocha Plocha		퉬 settings		12.7.201
📕 Stažené soubory	E	鷆 src		12.7.201
Dokumenty				
J Hudba				
🔄 Obrázky				
🛃 Videa				
🜏 Domácí skupina	*	•		•
Název souboru: Untitled1				•
Uložit jako typ: IDE Files (*.c;*.cpp;*.c	cc;*.h;*.s*;*.m	sa;*.asm;*.lst;*.map;	*.log;*.xcl;*.icf;*.mac;	;*.prj;*.pew) 🔻
				the court of the c

Zvolíme název nového souboru jako main.c



Save As			×
🕒 🗢 📕 « IAI	R ▶ priklad01 ▶ src	🗸 🍫 Prohledat: src	٩
Uspořádat 🔻 🛛 N	lová složka	}≡ ▼	0
 Knihovny Dokumenty Hudba Obrázky Videa Domácí skupina Počítač 	E	Název položky Hledání neodpovídají žádné položky.	Datum z
🏭 Místní disk (C 👝 Místní disk (E:	;))	<	•
Název souboru:	main.c		
Uložit jako typ:	C/C++ Files (*.c;*.cpp;*.cc;*.h)		•
Skrýt složky		Uložit	•

Klikneme na **Uložit.** Dále tento nový soubor přidáme do projektu. Proto název projektu označíme, pravým tlačítkem rozvineme menu



File E	dit View Project Simula	tor	Tools Window Help			
		10	C#		• 1 1	9
Workspa	ace ×	IAR I	nformation Center for ARM	main.c		
Debug	<u> </u>	in	t main(void)			
Files	2:: B i	1	return(0);			
	Options					
	Make					
	Compile					
	Rebuild All					
	Clean					
	Stop Build					
	Add	•	Add Files			
	Remove		Add "main.c"			
	Rename	4	Add Group			
	Source Code Control	•				
	File Properties					
	Set as Active					

V místním menu vybereme **Add – Add Files...**



Add Files - program01					×
🔾 🗢 – 🕌 « IAR 🕨	priklad01 🕨 src		▼ 4 ₇	Prohledat: src	Q
Uspořádat 👻 Nová	složka			8	• 🗆 🔞
🔆 Oblíbené položky	 Název polož 	ky		Datum změn	int main (void)
🖳 Naposledy navšt	📄 main			12.7.2011 11:4	<pre>i return(0); </pre>
💻 Plocha 🚺 Stažené soubory					3
 Knihovny Dokumenty Hudba Obrázky Videa 	E				
🍓 Domácí skupina					
počítač		III		۶.	
Ná	zev souboru: mai	in	•	Source Files (*.c; Otevčit	*.cpp;*.cc;*.h;* 🔻

Vybereme soubor **main.c** a klikneme na **Otevřít**

File Edit View Pro	oject	Simu	lator Tools Window Help
🗅 😂 🖬 🕼 🌆	1 %] v u [
Workspace		×	IAR Information Center for ARM main.c
Debug		•	int main(void)
Files	<u>e</u>	87: 10	{
🗆 🗊 program01			recurn(0);
- 🗗 💽 main.c		*	1.
📙 🖵 🗀 Output			
🖵 🖸 Output			



× IAR Information Center for ARM main.c int main (void) *
int main (void) [§] :: Bisse return (0); Options Make
Image: Second
Options 😡
Make
Rebuild All Clean Stop Build Add Remove Rename
Source Code Control File Properties
RRR

Znovu rozvineme místní menu a vybereme položku Options



ategory:						
C/C++ Compiler						
Assembler Output Converter	Target	Output	Library Confi	auration	Library Options	MISRA-C:200
Custom Build	1					
Build Actions	Proc	essor va	riant			
Debugger	00	Core	ARM7TDMI		•	
Simulator	1 1 2 3					
Angel CDB Server	0	Device	None			0 +
IAR ROM-monitor	115					
J-Link/J-Trace	End	an mode	<u> </u>	FPU		
TI Stellaris FTDI		ittle		Ner		_
Macraigor PE micro				INON	e	
RDI		a neaa				
ST-LINK	9	BE32				
Third-Party Driver		BES				
CONTRACTOR OF A						

Překlikneme radiobutton Processor variant na Device



Coporal Options						
C/C++ Compiler						
Assembler Output Converter	Target	Output	Library Con	figuration	Library Options	MISRA-C:200
Custom Build	NAME OF BRIDE	a sup su				
Build Actions	Proc	essor va	riant			
Linker		-	ADM7TDM	1		
Simulator	O	ore			×	
Angel	0	Device	None			Ē.
GDB Server						
J-Link/J-Trace	105 3	117.787		22/2010		
TI Stellaris FTDI	Endi	ian mode		FPU	0	
Macraigor	0 L	little		Non	e	*
PE micro		Big				
ST-LINK	0) BE32				
Third-Party Driver		BE8				
1 1 1 1 1 1 1 1 1 2 2 1 1 1 2 1	1					N

Klikneme na ikonku napravo od *textboxu* **device**



2			
Category: General Options C/C++ Compiler Assembler Output Converter Custom Build Build Actions Linker Debugger Simulator	Target Output Library C Processor variant © Core ARM7TE	Configuration Library Options	MI None Actel AnalogDevices Atmel Cirrus
Angel GDB Server IAR ROM-monitor J-Link/J-Trace TI Stellaris FTDI Macraigor PE micro RDI ST-LINK Third-Party Driver	 Device None Endian mode Little Big BE32 BE8 	FPU None	EnergyMicro Epson Faraday Freescale Fujitsu Hilscher Intel Marvell
		Ш. Ш.	Micronas NetSilicon Nuvoton NXP OKI Samsung Socle

V rozvinutém menu vybereme ST



K IAR Embedded Workbench IDE		State of the local division of the local div	A		
File Edit View Project Simulator Tools Window	ST SPEAR300SNOR				
0 📽 🖬 🕼 🐛 📾 📾 🖬 🐿 🖬	ST SPEAr320				
Workspace × IAR Information Center f	ST STM32F100x4				
Debug	<pre>int main(void)</pre>				
Files 😤 📴 Options for node "prog	ST STM32F100x8				
🗆 🗇 program01 🗸			ST STM32F100xB		
			ST STM32F100xC		
Category:			ST STM32F100xD		
General Options			ST STM32F100xE		
C/C++ Compiler			ST STM32F105x8		
Output Converter	Target Output Library Configuration Library Options MI	None	ST STM32F105xB		
Custom Build	10	Actel •	ST STM32F105xC		
Build Actions	Processor variant	AnalogDevices +	ST STM32F107xB		
Debugger	Core ARM7TDMI -	Atmel +	ST STM32F107xC		
Simulator	None Fb	Cirrus 🕨	ST STM32F10xx4		
GDB Server	O Device Hono	EnergyMicro 🕨	ST STM32F10xx6		
IAR ROM-monitor		Epson •	ST STM32F10xx8		
J-Link/J-Trace TT Stellaris ETDI	- Endian mode FPU	Faraday 🕨	ST STM32F10xxB		
Macraigor	Little None	Freescale +	ST STM32F10xxC		
PE micro	🗇 Big	Fujitsu 🕨	ST STM32F10xxD		
ST-LINK	@ BE32	Hilscher 🕨	ST STM32F10xxE		
Third-Party Driver	© BE8	Intel 🕨	ST STM32F10xxF		
		Marvell +	ST STM32F10xxG		
		Micronas 🕨	ST STM32F205rx		
		NetSilicon	ST STM32F205zx		
	UK	Nuvoton +	ST STM32F207xx		
		NXP +	ST STM32F215rx		
		OKI +	ST STM32F215zx		
		Samsung 🕨	ST STM32F217xx		
program01		Socle +	ST STM32L151x8		
	ST STM32L151xB				

A v něm **STM32F100xB**



ategory:						
C/C++ Compiler						
Assembler Output Converter	Target	Output	Library Configuration	Library Options	MISRA-C:200 1	
Custom Build						
Build Actions	Processor variant					
Debugger	Core Cortex-M3		Cortex-M3	*		
Simulator					-	
GDB Server	0	Jevice	ST STMSZETUUKD			
IAR ROM-monitor						
J-Link/J-Trace	End	ian mode	FPU			
Macraigor	01	little	Non	e	*	
PE micro	© E	Big				
RDI		BE32				
Third-Party Driver	0) BE8				
	- 64					

Poté vybereme v podokně Category položku Linker, dostaneme


Category:							Factory 9	Settings
General Options C/C++ Compiler								
Assembler	0.0	1		1-			Lass	T-D-D-
Output Converter	Config	Library	Input	Optimizations	Output	List	#define	
Custom Build								
Build Actions	Link	er config	uration fi	e				
Debugger		Override o	default					
Simulator		STOOLK	T DIR	\CONFIG\gene	ric corte	cicf	5	
Angel			8	2	-152		8	
GDB Server		Edit						
IAR ROM-monitor	3	5	- 10					
J-Link/J-Trace	Config	juration fil	e symbo	l definitions: (one	e per line)			
TI Stellaris FTDI								
Macraigor								
PE micro								
STJ INK								
Third-Party Driver								
man arey briver				T				
								*

Zaškrtneme v Linker configuration file políčko Override default



							Factory S	Settings
General Options C/C++ Compiler Assembler	Config	Library	Input	Optimizations	Output	List	#define	
Custom Build Build Actions	Link	er config	uration fi		output	Dot	Monine	
Debugger Simulator Angel GDB Server IAR ROM-monitor J-Link/J-Trace	Config	STOOLK STOOLK Edit	erauit KIT_DIR: le symbo	\$\CONFIG\gene I definitions: (one	ric_cortex per line)	cicf		
11 Stellaris FTD1 Macraigor PE micro RDI ST-LINK Third-Party Driver								*

Vyplníme textbox dle obrázku a klikneme poté na Edit



Contraction That (This)	Eactory Settings
General Options	
C/C++ Compiler	
Assembler	Config Library Input Ontimizations Output List #define D 1
Custom Build	
Build Actions	
Linker	
Debugger	Vector Table Memory Regions Stark/Hean Sizes
Simulator	
Angel	intyec start 0x0000000
GDB Server	
IAR ROM-me	
J-Link/J-Trad	
	A
TI Stellaris F	
TI Stellaris F Macraigor	
TI Stellaris F Macraigor PE micro	Save Storno
TI Stellaris F Macraigor PE micro RDI	Save Storno
TI Stellaris F Macraigor PE micro RDI ST-LINK	Save Storno
TI Stellaris F Macraigor PE micro RDI ST-LINK Third-Party Driv	er Storno

Vyplníme dle obrázků

ector rable	Memory Regions	Stack/Heap Sizes	
.intvec start	0x08000000		
	1	<u> </u>	



ector Table	Memory Regions	Stack/Heap Sizes
	Start:	End:
ROM	0x00000000	0x0007FFFF
RAM	0x20000000	0x2000FFFF

ector Table	Memory Regions	Stack/Heap Sizes
	Start:	End:
ROM	0x08000000	0x08020000
RAM	0x20000000	0x20002000

Nakonec klikneme na tlačítko Save, objeví se okno Uložit jako



	•			
Uspořádat 👻 Nová složka			855 -	• 🔞
 Knihovny Dokumenty Hudba Obrázky Vídea Domácí skupina Počítač 	E	Název položky Debug inc settings rc program01.dep program01.ewp program01.ewp program01.ewp program01.ewp		Datum 12.7.20 12.7.20 12.7.20 12.7.20 12.7.20 12.7.20 12.7.20
🏭 Místní disk (C:) 👝 Místní disk (E:)	•	٠ [
Název souboru: program01.icf Uložit jako typ:				

Vyplníme a stiskneme tlačítko Uložit



Category: General Options C/C++ Compiler	Multi-file Compilation
Assembler Output Converter Custom Build Build Actions Linker	Language 2 Code Optimizations Output List Preprocessor Ignore standard include directories Additional include directories: (one per line) Image: Code directories directories directories directories directories: (one per line) Image: Code directories dir
Simulator Angel GDB Server IAR ROM-monitor J-Link/J-Trace TI Stellaris FTDI Macraigor PE micro RDI ST-LINK Third-Party Driver	Preinclude file: Defined symbols: (one per line) Preprocessor output to file Preserve comments Generate #line directives

V Options ještě vyplníme vlastnosti C/C++ překladače



Category: General Options C/C++ Compiler	Multi-file Compilation	Factory Settings
Output Converter	Language 2 Code Optimizations Output	List Preprocessor
Eustoin Build Build Actions Linker Debugger Simulator Angel GDB Server IAR ROM-monitor	Ignore standard include directories Additional include directories: (one per line) \$PROJ_DIR\$\.\inc	*
J-Link/J-Trace	Preinclude file:	
TI Stellaris FTDI Macraigor PE micro RDI ST-LINK Third-Party Driver	Defined symbols: (one per line)	Preprocessor output to file Preserve comments Generate #line directives

A dále nastavíme vlastnosti Debugeru



eneral Ontions						Factory Setting
C/C++ Compiler						
Assembler Output Converter	Setup	Download	Images	Extra Options	Plugins	
Custom Build	2.0			_		
Build Actions	Driver			Run to	1	
Debugger	Simul	ator	•	main		
Simulator	Setu	ip macros				
Angel	[FT]	Use macro fi	le(s)			
GDB Server						
J-Link/J-Trace	1					
TI Stellaris FTDI	10-					
Macraigor	Dev	ice descriptio	n file			
PE micro		Override def	ault			
ST-LINK	ST	OOLKIT DI	RS\CONF	IG\debuaaer\S	Niostm32f10	Oxb.ddf
Third-Party Driver						

Vybereme driver **ST-Link**



Contract						Factory Settings
C/C++ Compiler						
Assembler	Cature	1				
Output Converter	Setup	Download	Images	Extra Options	Plugins	
Custom Build Build Actions	Driver			Run to		
Linker	CCT I	INUZ	2			
Debugger	51-2	INK		j main		
Simulator	Setu	up macros				
Angel		Use macro f	ile(s)			
GDB Server						
IAR ROM-monitor						
TI Stellaris ETDI						
Macraigor		8. 1 2. 3 1 2	CI-			N
PE micro	Dev	ice descriptio	on file			13
RDI		Override def	ault			
ST-LINK	S	TOOLKIT_DI	R\$\CONF	IG\debugger\S	T\iostm32f100	bdo.ddf
Third-Party Driver						



Category:	Factory Settings
General Options C/C++ Compiler Assembler	Sature Download Income Etan Ontione Diverse
Custom Build Build Actions Linker Debugger Simulator Angel GDB Server IAR ROM-monitor J-Link/J-Trace TI Stellaris FTDI Macraigor PE micro RDI ST-LINK Third-Party Driver	Attach to program Verify download Suppress download Use flash loader(s) Override default .board file \$TOOLKIT_DIR\$\config\flashloader\ST\FlashSTM3} Edit

V záložce **Download** zaškrtneme **Use flash loaders**



Category:						Factory Settings
C/C++ Compiler						
Assembler Output Converter	Setup	Download	Images	Extra Options	Plugins	
Custom Build Build Actions Linker	Att	ach to progra rify download	i I			
Simulator Angel	Su Su	ppress down e flash loade	load r(s)			
GDB Server IAR ROM-monitor J-Link/J-Trace TI Stellaris FTDI Macraigor PE micro RDI ST-LINK		STOOLKIT	auit boar _DIR\$\co	a t∎e mfig √lashloader	\ST\FlashST	M3]
Third-Party Driver	-					

A poté vybereme v **Category** položku **ST-Link**



Category:		Factory Settings
General Options C/C++ Compiler		
Assembler	STIME	
Output Converter		
Build Actions		
Linker	Interface	
Debugger	O ITAG	
Simulator	O sind	
GDB Server	SWD	
IAR ROM-monitor		
J-Link/J-Trace		
TI Stellaris FTDI		
Macraigor PE micro		
RDI		
ST-LINK		
Third-Party Wyver		

A radiobutton interface překlikneme na SWD



Category:		Factory Settings
General Options C/C++ Compiler		
Output Converter	ST-LINK	
Custom Build Build Actions	Interface	
Linker Debugger Simulator	© JTAG	
Angel GDB Server	⊚ SWD	
J-Link/J-Trace		
TI Stellaris FTDI Macraigor		
PE micro		
ST-LINK		
Third-Party Driver		

Poté již zbývá kliknout na tlačítko **OK**.





Soubor main.c zatím obsahuje jen minimální kód. V menu vybereme Project – Rebuild All



File Edit View	Project Tools Window Help	
D 🛩 🖬 🕼	Add Files	
Workspace	Add Group	
Debug	Import File List	
Files	Edit Configurations	
□ □ program(Remove	
ULE Cutp	Create New Project	
	Add Existing Project	
	Options	ALT+F7
	Source Code Control	•
	Make	F7
	Compile	CTRL+F7
	Rebuild All	
	Clean	
	Batch build	F8
	Stop Build	CTRL+Break
	Download and Debug	CTRL+D
	Debug without Downloading	
	Make & Restart Debugger	CTRL+R
	Restart Debugger	CTRL+SHIFT+R
	Download	*

V tak jednoduchém programu pochopitelně nejsou chyby a tak se v okně zpráv objeví





Nyní je možné Download and Debug:



Náš **main.c** však zatím neobsahoval žádný užitečný program a tak **Download and Debug** zatím vynecháme a doplníme kód **main.c** a dále přidáme linky na potřebné knihovny a další soubory.

3.4.2 Blikač LED PB5 (priklad02)

Začneme tím, že vytvoříme (postupem z předchozího příkladu) nový **priklad02** a workspace **priklad02**. Poté do něj přidáme i **main.c**



K IAR Embedded Workbench IDE		
File Edit View Project Tools Window Help		
Workspace ×	IAR Information Center for ARM main.c*	* x
Debug 👻		
Files 😤 📴	#include "stm32f10x.h"	- i
🗆 🗇 priklad02 - Debug * 🛛 🗸	//priklad02	
Here and the second sec	and the second sec	
Uutput	void InitAll(void);	
	void Delay(unsigned int val);	
	//	
	int main (woid) (
		E
	<pre>InitAll();</pre>	
	while(1) /	
	// Set PB.5 bit	
	GPIOB->BSRR = GPIO_BSRR_BS5;	
	Delay(600000);	
	// Reset PB.5 bit	
	GPIOB->BSRR = GPIO_BSRR_BR5;	
	Delay(600000);	
	1	
	}	
	0	
	void InitAll(void) {	
	// Enable PORTE Periph clock	
	RCC->APB2ENR = RCC_APB2ENR_IOPBEN;	
	2.4 2.4	
priklad02		
× ×	m	· · · ·
Ready	Errors 0, Warnings 0 Ln 4, C	ol 12 🗾

Zdrojový kód souboru main.c

```
#include "stm32f10x.h"
//priklad02-----
void InitAll( void);
void Delay( unsigned int Val);
//-----
int main( void) {
    InitAll();
    while(1) {
        // Set PB.5 bit
        GPIOB->BSRR = GPIO_BSRR_BS5;
        Delay( 600000);
        // Reset PB.5 bit
        GPIOB->BSRR = GPIO_BSRR_BR5;
        Delay( 600000);
    }
}
```

305



```
}
                        _____
void InitAll( void) {
 // Enable PORTB Periph clock
 RCC->APB2ENR |= RCC_APB2ENR_IOPBEN;
 // Clear PB.5 control register bits
 GPIOB->CRL &= ~(GPIO CRL MODE5 | GPIO CRL CNF5);
 // Configure PB.5 as Push Pull output at max 10Mhz
 GPIOB->CRL |= GPIO_CRL_MODE5_0;
  return;
}
//-----
void Delay( unsigned int Val) {
 for( ; Val != 0; Val--) {
   ___NOP();
 }
}
//----
```

Do projektu musíme přidat knihovny a další soubory uložené v podadresářích. V místním menu (pravé tlačítko myši nad názvem projektu rozvine toto místní menu) vybereme **Add – Add Group**





Dostaneme

×
ОК
Cancel
1 0202
L X

Klikneme na OK. Dostaneme



Debug		12
Files	82	07
🗆 🗇 priklad02 - Debug *	~	
├── 🖸 main.c		
🖵 📮 🗀 Output		
└─⊞ 📄 priklad02.out		

Tím jsme do projektu přidali podadresář CMSIS . Obdobně

Group name:	ОК.
inc	

dd Group - priklad02	X
Group name:	
sic	
	Cancel

Files	20	Ø.
🗆 🗇 priklad02 - Debug *	~	10
🗀 inc		
- 🗀 src		
⊞ 🖸 main.c		
Ц 🔁 Output		
└─⊞ 🛄 priklad02.out		

Do přidaných podadresářů ještě přidáme soubory. Rozvineme místní nabídku nad názvem CMSIS





A v ní vybereme Add – Add Files...





Přidáme soubor core_cm3.c

Debug		
Files	22	0
🗆 🗊 priklad02 - Debug *	4	
└─── C core_cm3.c		
🗀 inc		
C src		
- 🖽 💽 main.c		
🖵 🔁 Output		
└─⊞ 🗋 priklad02.out		

Obdobně doplníme o další soubory

V.Váňa: CanSAT Book for Students – part 2 310



Files	22	Di
🗆 🗊 priklad02 - Debug *	~	
│ └─⊞ 💽 core_cm3.c		
📔 🖵 🔝 system_stm32f10x.h		
- Src		
- 🕀 🖸 main.c		
🖵 🔁 Output		
└─⊞ 🗋 priklad02.out		

V Options nastavíme vlastnosti compilátoru



Category:	Facto	ory Settings
General Options	Multi-file Compilation	
C/C++ Compiler	Discard Unused Publics	
Assembler Output Converter	Language 1 Language 2 Code Optimizations Output Lis	t F.
Custom Build		
Build Actions		
Linker	The second shall be be a street	
Debugger	Ignore standard include directories	
Angel	Additional include directories: (one per line)	i and
GDB Server	SPROJ_DIRS\CMSIS	*
IAR ROM-monitor		
J-Link/J-Trace		-
Macraigor	Preinclude file:	
PE micro		
RDI	5 5 1 1 1 2 1 3 1 S	
ST-LINK Third-Barty Driver	Defined symbols: (one per line)	ofile
Third Party Driver	Preserve comment	ts
	Generate #line dire	ectives

Ostatní nastavení Options for node "priklad02" budou stejné jako u prvního příkladu, tj



Category: General Options C/C++ Compiler					
Assembler Output Converter	Target	Output	Library Configuration	Library Options	MISRA-C:200 +
Custom Build Build Actions Linker	Proc	essor va	iant		
Debugger	00	Core	Cortex-M3	*	
Angel GDB Server	۹ (Device	ST STM32F100xB		∎.
IAR ROM-monitor J-Link/J-Trace TI Stellaris FTDI Macraigor PE micro	End I	ian mode .ittle Big	FPU	ne	*
RDI ST-LINK Third-Party Driver) BE32) BE8			



Category:							Factory 9	Settings
General Options C/C++ Compiler Assembler	Coofig	1.1		0.5.5.5	0	11.1		
Output Converter Custom Build	Coning	Library	Input	Optimizations	Output	List	Haetine	U
Linker	Link	er config	uration fi	le				
Debugger		Override (default					_
Simulator		\$PROJ_	DIR\$\pri	iklad02.icf				
Angel		- La						
GDB Server	1	EOIL						
J-Link/J-Trace	Config	uration fi	e symbo	l definitions: (one	e per line)			
TI Stellaris FTDI								*
Macraigor								
PE micro								
ST-LINK			I					
Third-Party Driver								
								-



Lategory:						Factory Settings
General Options C/C++ Compiler						
Output Converter	Setup	Download	Images	Extra Options	Plugins	
Custom Build Build Actions	Driver	15		🔽 Run to	40	
Linker	ST-L	INK	-	main		
Angel GDB Server IAR ROM-monitor J-Link/J-Trace TI Stellaris FTDI Macraigor PE micro	- Dev	Use macro f	ile(s) on file			
RDI ST-LINK Third-Party Driver	S	TOOLKIT_DI	R\$\CONF	IG\debugger\S	T\iostm32f100	kb.ddf



Category:		Factory Settings
General Options C/C++ Compiler		
Assembler Output Converter	ST-LINK	
Custom Build		
Build Actions Linker	Interface	
Debugger	O ITAG	
Simulator	© SWD	
GDB Server	0 3WD	
IAR ROM-monitor J-Link/J-Trace		
TI Stellaris FTDI		
Macraigor PE micro		
RDI		
Third-Party Driver		
Marca A. A.		
L		



Category: General Options C/C++ Compiler Assembler Output Converter Custom Build Build Actions Linker Debugger Simulator Angel GDB Server IAR ROM-monitor J-Link/J-Trace TI Stellaris FTDI Macraigor PE micro RDI	Output Image: Contrast additional output Output format: binary Output file Image: Override default priklad02 bin
ST-LINK Third-Party Driver	

Nakonec v menu vybereme Project – Rebuil All



IAR Embedded Wo	orkbench IDE		Section Section 2 and the sec
File Edit View P	roject Tools Window Help		
0 🛩 🖬 🕼	Add Files		· < > > > > > 20 ≥ = + + + + + + + + + + + + + + + + + +
Workspace	Add Group		ter for ARM main.c*
Debug	Import File List		PORTB Periph clock
Files	Edit Configurations		NR = RCC_APB2ENR_IOPBEN;
□ <mark>□ priklad02</mark> 	Remove		B 5 control register bits
└── ͡ core_ └── ͡ inc └── ि stm3	Create New Project Add Existing Project		<pre>s= ~ (GPIO_CRL_MODE5 GPIO_CRL_CNF5); re PB.5 as Push Pull output at max 10Mhz</pre>
syste	Options	Alt+F7	= GPIO_CRL_MODE5_0;
Here Content	Source Code Control	+	
L-⊞ D prikla	Make	F7	
	Compile	Ctrl+F7	
	Rebuild All	12	compared a straight
	Clean Batch build	F8	unsigned int Val) { != 0; Val) {
	Stop Build	Ctrl+Break	
	Download and Debug Debug without Downloading	Ctrl+D	
	Make & Restart Debugger	Ctrl+R	
	Restart Debugger	Ctrl+Shift+R	
	Download	•	
priklad02			III
×			III .
Clean and make the a	ctive project		Errors 0, Warnir

Projekt je úspěšně přeložen a spojen (linked)



AR Embedded Workbench IDE	A REAL PROPERTY AND A REAL	
File Edit View Project Tools Window Help		
D 📽 🖬 🕼 🕼 🕼 🖬 🛍 🗠 🗠 🗌		
Workspace	× IAR Information Center for ARM main.c	÷ 3
Debug	// Enable PORTE Periph clock	
Files 80 B	RCC->APB2ENR = RCC_APB2ENR_IOPBEN;	^
E Coriklad02 - Debug		
	// Clear PB 5 control register hits	
📔 🖾 🗁 core_cm3.c	GPIOB->CRL 4= ~(GPIO CRL MODE5 GPIO CRL CNF5);	
⊨		
stm32f10x.h	// Configure PB.5 as Push Pull output at max 10Mhz	
System_stm32f10x.h	GPIOB->CRL = GPIO_CRL_MODE5_0;	
	return;	
Let D priklad02 out		
		E
	world Delay (unclement int Mal) (
	for(: Val != 0: Val) {	
	NOP();	
priklad02		
* Messages	File	Line
Building configuration: priklad02 - Debug		
Updating build tree		
6 file(s) deleted.		
Updating build tree		
core_cm3.c		
main.c		
priklad02 out		
piniduse.out		
Total number of errors: 0		
Total number of warnings: 0		
	N	
Ready	From Q Warnings ()	
neug	Lifes of Wannings o	

Zbývá připojit přes USB startkit STM32VL Discovery. Dále

K IAR Embedded Workbench IDE		
File Edit View Project Tools Window Help		
🗅 🖙 🖬 🗿 🚳 👗 🖻 💼 🖘 🖂		
Workspace ×	IAR Information Center for ARM main.c Download and Debug	* x
Debug 🗸 🗸	// Enable FORTB Periph clock	
Files 8: B	RCC->APB2ENR = RCC_APB2ENR_IOPBEN;	C
Priklad02 - Debug ✓ CMSIS Gore_cm3.c finc finstrat2110x.h System_stra32110x.h Src Output priklad02.out	<pre>// Clear PB.5 control register bits GPIOB->CRL = -(GPIO_CRL_MODE5 GPIO_CRL_CNF5); // Configure PB.5 as Push Pull output at max 10Mhz GPIOB->CRL = GPIO_CRL_MODE5_0; return; } //</pre>	ш
priklad02	<pre>void Delay(unsigned int Val) { for(; Val != 0; Val) { NOP(); </pre>	



IAR Embedded Workbench IDE	A REAL PROPERTY AND A REAL		×
File Edit View Project Debug	Disassembly ST-LINK Tools Window Help		
0 🚅 🖬 🕼 🎒 👗 🖻 🛍	N CA 🚽 🔸 🌾 📜 🔤	🖻 🐢 🍓 🌬 🛤 👷 🗶 🎂 🕁	
5 - 526833	×		
Workspace ×	IAR Information Center for ARM main.c	Disassembly	×
Debug 👻		Go to 🔹 Memory 💌 🗈	
Files 😤 🗠	<pre>#include "stm32f10x.h"</pre>	main:	-
🗆 🗇 priklad02 🖌	//priklad02	0x8000040: 0xb580 PUSH {r7, 1r}	
		0x8000042: 0xf000 0xf80e BL InitAll	
	<pre>void InitAll(void);</pre>	GPIOB->BSRR = GPIO BSRR BS5;	
u u src u u u src	void Delay(unsigned int Val);	??main_0:	
L D Output	//	0x8000048: 0x2120 MOV5 r1, #32	÷
The Alexander of		0x800004a: 0x6001 STR r1, [r0]	
	⇔int main(void) {	Delay(600000);	
	InitAll():	0x800004c: 0x480f LDR.N r0, ??DataTable1	_1
	Initial (),	GPTOR-SBSRR = GPTO BSRR BR5.	
	while(1) {	0x8000052: 0x480d LDR.N r0, ??DataTable1	
	// Set PB.5 bit	0x8000054: 0xf45f 0x1100 MOVS.W r1, #2097152	-1
	GPIOB->BSRR = GPIO_BSRR_BS5;	0x8000058: 0x6001 STR r1, [r0]	
	Delay(600000);	Delay(600000);	
	// Reset PR 5 hit	0x800005a: 0x480c LDR.N r0, ??DataTable1	_1
	GPIOB->BSRR = GPIO BSRR BR5:	0x8000060: 0xe7f1 B N 22main 0	
	Delay(600000);	RCC->APB2ENR = RCC APB2ENR IOPBEN;	-
priklad02		<	P.
* Messages		File Lir	e 🔺
Updating build tree			
core cm3.c			
main.c			
Linking			
priklad02.out			=
Fotal number of errors: 0		N	
i otal number of warnings: l	Jő	Log Log	-
Debug Log Build			×
Ready		Errors 0, Warnings 0 Ln 11, Col 1	1

Program ukončíme, startkit resetujeme. LED připojená na PB5 podle očekávání bliká.

3.4.3 Blikač LED PC8 (priklad03)

Nyní vytvoříme další projekt a workspace priklad03. Nejprve do adresáře priklad03 zkopírujeme adresáře Libraries a Utilities (včetně obsahu) z stm32vldiscovery_package a src a inc z projektu Demo (Discover)

Název	↑Př	ip Velikost	Datum		Atribut
^ []		<dir></dir>	13.07.2011	19:51	
🛄 [inc]		<dir></dir>	13.07.2011	19:50	
🛄 [Libraries]		<dir></dir>	13.07.2011	19:50	
[src]		<dir></dir>	13.07.2011	19:50	75 - 8 0
🗀 [Utilities]		<dir></dir>	13.07.2011	19:50	24 - 2 2
readme	txt	2 427	13.09.2010	14:37	-a-

Poté již v IAR Embedded Workbench



File Edit View	Project Tools Wind	low Help		
Workspace	Add Files Add Group Import File List Edit Configuratio Remove Create New Proje	ns		
	Add Existing Proj	ect		nn Ce
	Options		Alt+F7	ation you n
	Source Code Con	trol	•	uides, sup
	Make Compile		F7 Ctrl+F7	
	Clean Batch build		F8	t GUIDES product
	Stop Build	trl+Break	ation in PDF es you all the	
	Download and De Debug without D	ebug ownloading	Ctrl+D	eference in you need.
	Make & Restart D Restart Debugger	ebugger Ctrl	Ctrl+R + Shift+ <mark>R</mark>	4
	Download		+	DDODT.
	T fa al	utorials to make you miliar with the IDE nd the features of the NR C-SPY debugger.	For que to use y reporting finding	stions about how our IAR product, g a problem, or support

V menu vybereme **Project – Create New Project**



Tool chain:	ARM	•	
Project template	s:		
- Empty proje	ect		
🗄 - asm			
⊕- C			
- DLIB			
Externally b	puilt executable		
Description:			
Creates an emp	ty project.		
	-		

Vybereme **ARM** a klikneme na **OK**



🖁 Uložit jako	1.5.6	d.	and the	A date	×
🕒 🔍 🗢 📕 « Af	M_priklady ► IAR ► pril	klad03 ၊	• • • •	Prohledat: priklad03	٩
Uspořádat 🔻 🛛 N	ová složka			855 🗣	0
 Knihovny Dokumenty Hudba Obrázky Videa Domácí skupina Počítač 	Ĩ	E	Název položky inc Libraries src Utilities		Datum 2 13.7.201 13.7.201 13.7.201 13.7.201
🏭 Místní disk (C)				
🕞 Místní disk (E:)	-	٠ [•
Název souboru:	priklad03				
Uložit jako typ:	Project Files (*.ewp)				•
🔿 Skrýt složky				Uložit	rno

Zvolíme název projektu a klikneme na **Uložit**





Uložíme nový workspace File – Save Workspace






ARM_priklady > IAR > priklad	d03	▶ • + ₂	Prohledat: priklad03	Q
Uspořádat 🔻 Nová složka				0
ARM_priklady DAC Demo IAR demo Libraries misc pozn priklad01 priklad02	* III I	Název položky		Datum z 13.7.2011 13.7.2011 13.7.2011 13.7.2011 13.7.2011
Název souboru: priklad05 Uložit jako typ: Workspace Files (*.eww)	-	< [Uložit Sto	rno

Vybereme název Workspace





Přidáme soubor výběrem v místní nabídce Add – Add Files...



Add Files - priklad03	
Uspořádat 🔻 Nová složka	₩ • □ 0
Plocha ^ Název položky	Datum změny
main	14.9.2010 15:48 *****
Stm32f10x_it Stm32f10x_it Stm32f10x_it Stm32f10x_it Stm32f10x_it	14.9.2010 15:48
	src/m
Mistni disk (E:) + +	iii ≯ ain.c ·
Název souboru: main	▼ Source Files (*.c;*.cpp;*.cc;*.h;* ▼
	Otevřít 🔽 Storno

Vybereme main.c

File Edit View Pr	oject	Simul
D 🗲 🖬 🕼 🎒	1	h f
Workspace		×
Debug		•
Files	۶2	07: 10
🗆 🗇 priklad03	. 4	1
H-🕀 💽 main.c		*

V místní nabídce vybereme **Options**





Přidáme podadresáře (Groups)

Group name:	
CMSIS	
	Cancel



File Edit View Pro	ject	Sin
D 🛩 日 🕼 🎒	X	
Vorkspace	"- <u>-</u> _	
Debug		,
Files	22	D.
🗆 🗇 priklad03	4	1
- CMSIS		10
🗕 🖸 main.c		
Let Output		

Group name:	OK .
startup	
area who	Canad

Group name:	OK N
StdPeriph Driver	- WE
	Cancel

Group name:	ОК
stm32vldiscovery	
ano. Tormar dives you all'a	ne naroware
nno. – – tormat olves vou all to d Group - priklad03	ne i naroware
king. – Format gives vou all t Id Group - priklad03 Group name:	



Files	\$2	0
🗆 🗇 priklad03 - Debug *	~	
🗀 src		
🛏 🗀 startup		
- 🖽 🖸 main.c		*
🖵 🔁 Output		

Dále soubory

🔀 IAR Embedded	Workbench IDE	-		-
File Edit View	Project Simulato	or Tools W	indow Help	
0 😂 🖬 🕼	6 X B C	IN CA	• ·	1 2 2 3
Workspace		×	IAR Information Center for AR	M
Debug			q	٩
Files			0 12 90 12 0	BR. (
🗆 🗇 priklad0	3 - Debug *	~	Nîrîl kî l (kî le Nîrî	111161
Options				
StdF StdF StdF StdF Stm Compile Stm Compile Stm Compile Compile Stm Compile Com			IAR Infor Here you will find all projects, user and re	matio the informa eference gu
-	Add	•		
	Remove Rename Source Code Con	trol 🕨	Add "index.ENU.html" Add Group ourdennes for setting op your project, adding files, compiling, linking, and debugging it.	USER documplete p documenta format give user and re information
	Set as Active			

V.Váňa: CanSAT Book for Students – part 2



Add Files - CMSIS	CoreSupport	• 4,	Prohledat: CoreSupport	× A
Uspořádat 👻 Nová složka				
E Plocha Název	položky ^		Datum změny 7.9.2010 11:12	/**** ***** *****
 Knihovny Dokumenty Hudba Obrázky Videa Domácí skupina Počítač 				***** ***** ***** ***** ***** ***** ****
Místní disk (C:)	III		,	@brie f
Název souboru:	core_cm3	•	C/C++ Source Files (*.c;* Otevřít	.cpp;*. ▼ :orno



Nyní již máme potřebnou strukturu projektu



Debug		
Files	82	D:
🗆 🗇 priklad03 - Debug	~	
│		*
│ └─⊞ 🖸 system_stm32f10x.c		*
│ └─⊞ 📴 stm32f10x_it.c		*
│ └─⊞ 歸 startup_stm32f10x_md_vl.s		*
│		*
│		*
⊨ 🖽 🖻 stm32f10x_gpio.c		*
│		*
L→⊞ 🖸 stm32f10x_rcc.c		*
□ 🖵 🖸 STM32∨ldiscovery.c		*
- 🖽 🖸 main.c		*
🖵 🖸 Output		

Všimněme si červených teček. Později se jimi budeme zabývat. Nyní ale nastavíme **Options**







General Options						
C/C++ Compiler Assembler Output Converter	Target	Output	Library Co	nfiguration	Library Options	MISRA-C:200 4
Custom Build Build Actions Linker	Proc	essor va	riant			
Debugger	00	Core	Cortex-M3		*	
Simulator Angel GDB Server	0)evice	ST STM32	F100xB		Ē.
GDB Server IAR ROM-monitor J-Link/J-Trace TI Stellaris FTDI Macraigor PE micro RDI ST-LINK Third-Party Driver	Endi © L © E	an mode ittle 3ig 9 BE32 9 BE8		FPU Non	e	T

Vybereme CPU STM32F100xB (podrobněji v předchozích kapitolách)



General Options C/C++ Compiler	Factory Setting Multi-file Compilation Discard Unused Publics
Assembler Output Converter Custom Build Build Actions Linker	Language 2 Code Optimizations Output List Preprocessor
Debugger Simulator Angel GDB Server IAR ROM-monitor	Preinclude file:
J-Link/J-Trace TI Stellaris FTDI Macraigor PE micro RDI ST-LINK Third-Party Driver	Defined symbols: (one per line) Preprocessor output to file Preserve comments Generate #line directives

Additional include directories:

\$PROJ_DIR\$\inc\
\$PROJ_DIR\$\Libraries\STM32F10x_StdPeriph_Driver\inc\
\$PROJ_DIR\$\Utilities\
\$PROJ_DIR\$\Libraries\CMSIS\CM3\CoreSupport\
\$PROJ_DIR\$\Libraries\CMSIS\CM3\DeviceSupport\ST\STM32F10x

Defined symbols:

USE_STDPERIPH_DRIVER STM32F10X_MD_VL



Category:	Eactory Settings		
General Options	Multi-file Compilation		
C/C++ Compiler	Discard Unused Publics		
Assembler			
Output Converter	Language I Language 2 Code Optimizations Output List		
Build Actions			
Linker			
Debugger	Ignore standard include directories		
Simulator	Additional include directories: (one per line)		
Angel	\$PROJ_DIR\$\Libraries\STM32F10x_StdPeriph_Driver\inc\		
GDB Server	\$PROJ_DIR\$\Utilities\ \$PROJ_DIR\$\Utilities\		
J-Link/J-Trace	\$PROJ_DIR\$\Libraries\CMSIS\CM3\DeviceSupport\ST\STM32F10x		
TI Stellaris FTDI			
Macraigor	Preinclude file:		
PE micro			
ST-LINK	Defined symbols: (one per line)		
Third-Party Driver	USE_STDPERIPH_DRIVER Preprocessor output to file STM32F10X_MD_VL Preserve comments Generate #line directives		

Vlastnosti překladače



Category: General Options C/C++ Compiler Assembler Output Converter Custom Build Build Actions Linker Debugger	Output Output Image: Contrast additional output Output format:
Simulator Angel GDB Server IAR ROM-monitor J-Link/J-Trace TI Stellaris FTDI Macraigor PE micro RDI ST-LINK Third-Party Driver	binary ▼ Output file ✓ Override default priklad03.bin

Nastavíme výstup



General Options C/C++ Compiler Assembler Output Converter	
Custom build Build Actions Linker Debugger Simulator Angel	Linker configuration file Unverside default STOOLKIT_DIR\$\CONFIG\generic_cortex.icf
GDB Server IAR ROM-monitor J-Link/J-Trace TI Stellaris FTDI Macraigor PE micro RDI ST-I INK	Configuration file symbol definitions: (one per line)

Nastavme linker

ector Table	Memory Regions	Stack/Heap Sizes	
.intvec start	0x0800000		



ector Table	Memory Regions	Stack/Heap Sizes	
	Start:	End:	
ROM	0x08000000	0x0801FFFF	1
RAM	0x20000000	0x20001FFF	-

Po kliknutí na **Save** se objeví

Uspořádat 👻 Nová složka		83 🔹 🔞
Nocha		
 Stažené soubory Knihovny Dokumenty Hudba Obrázky Videa Počítač 	Název položky	Datur * 13.7.2 13.7.2 13.7.2 13.7.2 13.7.2 13.7.2 13.7.2 13.7.2 13.7.2 13.7.2 13.7.2 13.7.2
🏭 Místní disk (C:) 👝 Místní disk (E:)	riklad05.ewp	13.7.2
Název souboru: priklad03.icf Uložit jako typ:		•

Vyplníme a klikneme na Uložit



ategory:	Factory Setting
General Options	
Assembler	
Output Converter	Setup Download Images Extra Options Plugins
Custom Build	Déux
Build Actions	Driver V Run to
Debugger	ST-LINK main
Simulator	Setup macros
Angel	Use macro file(s)
GDB Server	
IAR ROM-monitor	
J-Link/J-Trace	
Macraigor	
PE micro	Device description file
RDI	Override default
ST-LINK	\$TOOLKIT_DIR\$\CONFIG\debugger\ST\iostm32f100xb.ddf
Third-Party Driver	

Dále nastavíme Debugger



General Options C/C++ Compiler Assembler Output Converter Custom Build Build Actions Linker Debugger Simulator Angel GDB Server IAR ROM-monitor J-Link/J-Trace TI Stellaris FTDI Macraigor PE micro RDI ST-LINK Third-Party Driver	Setup Atta	Download ach to progra ify download opress down e flash loade Override def STOOLKIT Edit	images am load r(s) fault_boar _DIR\$\co	Extra Options d file	Plugins	M3
--	---------------	---	---	-------------------------	---------	----



Category:		Factory Settings
General Options C/C++ Compiler Assembler Output Converter Custom Build Build Actions Linker Debugger Simulator Angel GDB Server IAR ROM-monitor J-Link/J-Trace TI Stellaris FTDI Macraigor PE micro RDI ST-LINK Third-Party Driver	ST-LINK Interface Interface SWD	

A ST-Link na SWD. Klikneme na **OK**. Poté v IDE spustíme překlad a sestavení (Build).





Překlad a sestavení proběhlo **OK**, zároveň vymizely červené tečky.







Odpojíme a připojíme startkit. Zelená led **PC9** bliká. Nyní upravíme kód **main.c** tak, že vzájemně zaměníme výskyty LED3 a LED4

```
/* Includes -----
---*/
#include "stm32F10x.h"
#include "STM32vldiscovery.h"
/* Private typedef --
  -*/
/* Private define --
  -*/
#define LSE FAIL FLAG 0x80
#define LSE PASS FLAG 0x100
/* Private macro -----
___*/
/* Private consts ---
---*/
/* Private variables -----
___*/
u32 LSE Delay = 0;
u32 \text{ count} = 0;
u32 BlinkSpeed = 0;
```

V.Váňa: CanSAT Book for Students – part 2

ARM STM32 programming



```
u32 KeyState = 0;
static __IO uint32_t TimingDelay;
/* Private function prototypes -
---*/
void Delay(uint32 t nTime);
void TimingDelay Decrement(void);
/* Private functions -----
---*/
/**
 * @brief Main program.
* @param None
  * @retval None
  */
int main(void)
{
  /* Enable GPIOx Clock */
 RCC APB1PeriphClockCmd(RCC APB1Periph_PWR, ENABLE);
  /* Initialise LEDs LD3&LD4, both off */
  STM32vldiscovery LEDInit(LED3);
  STM32vldiscovery LEDInit(LED4);
  STM32vldiscovery LEDOff(LED3);
  STM32vldiscovery LEDOff(LED4);
  /* Initialise USER Button */
  STM32vldiscovery PBInit (BUTTON USER, BUTTON MODE GPIO);
  /* Setup SysTick Timer for 1 msec interrupts */
  if (SysTick Config(SystemCoreClock / 1000))
  {
   /* Capture error */
   while (1);
  }
  /* Enable access to the backup register => LSE can be enabled */
  PWR BackupAccessCmd(ENABLE);
  /* Enable LSE (Low Speed External Oscillation) */
  RCC LSEConfig(RCC LSE ON);
  /* Check the LSE Status */
  while(1)
  {
    if(LSE Delay < LSE FAIL FLAG)</pre>
      /* check whether LSE is ready, with 4 seconds timeout */
     Delay (500);
      LSE Delay += 0x10;
      if(RCC GetFlagStatus(RCC FLAG LSERDY) != RESET)
      {
        /* Set flag: LSE PASS */
```

346

V.Váňa: CanSAT Book for Students – part 2



```
LSE Delay |= LSE PASS FLAG;
      /* Turn Off Led4 */
      STM32vldiscovery LEDOff(LED3);
      /* Disable LSE */
      RCC LSEConfig(RCC LSE OFF);
     break;
    }
  }
  /* LSE FAIL FLAG = 0x80 */
  else if(LSE Delay >= LSE FAIL FLAG)
  {
    if(RCC GetFlagStatus(RCC FLAG LSERDY) == RESET)
    {
      /* Set flag: LSE FAIL */
     LSE Delay |= LSE FAIL FLAG;
      /* Turn On Led4 */
     STM32vldiscovery LEDOn(LED3);
    }
    /* Disable LSE */
    RCC LSEConfig(RCC LSE OFF);
    break;
  }
}
/* main while */
while(1)
{
  if(0 == STM32vldiscovery PBGetState(BUTTON USER))
    {
      if(KeyState == 1)
      {
         if(0 == STM32vldiscovery PBGetState(BUTTON USER))
        {
          /* USER Button released */
           KeyState = 0;
          /* Turn Off LED4 */
           STM32vldiscovery LEDOff(LED3);
        }
      }
    }
  else if(STM32vldiscovery PBGetState(BUTTON USER))
    {
      if(KeyState == 0)
      {
         if(STM32vldiscovery PBGetState(BUTTON USER))
        {
          /* USER Button released */
           KeyState = 1;
          /* Turn ON LED4 */
          STM32vldiscovery LEDOn(LED3);
          Delay(1000);
          /* Turn OFF LED4 */
          STM32vldiscovery_LEDOff(LED3);
          /* BlinkSpeed: 0 -> 1 -> 2, then re-cycle */
```

V.Váňa: CanSAT Book for Students – part 2

ARM STM32 programming



```
BlinkSpeed ++ ;
          }
        }
      }
        count++;
        Delay(100);
      /* BlinkSpeed: 0 */
      if(BlinkSpeed == 0)
          {
            if(4 == (count % 8))
            STM32vldiscovery_LEDOn(LED4);
            if(0 == (count % 8))
            STM32vldiscovery LEDOff(LED4);
         }
           /* BlinkSpeed: 1 */
           if(BlinkSpeed == 1)
          {
            if (2 == (count % 4))
            STM32vldiscovery_LEDOn(LED4);
            if(0 == (count % 4))
            STM32vldiscovery LEDOff(LED4);
          }
          /* BlinkSpeed: 2 */
          if(BlinkSpeed == 2)
          {
            if(0 == (count % 2))
            STM32vldiscovery LEDOn(LED4);
            else
            STM32vldiscovery LEDOff(LED4);
          }
          /* BlinkSpeed: 3 */
          else if(BlinkSpeed == 3)
        BlinkSpeed = 0;
 }
}
 **
 * @brief Inserts a delay time.
 * Oparam nTime: specifies the delay time length, in milliseconds.
 * @retval None
  */
void Delay(uint32 t nTime)
{
 TimingDelay = nTime;
 while(TimingDelay != 0);
}
  * @brief Decrements the TimingDelay variable.
  * @param None
  * @retval None
  */
void TimingDelay_Decrement(void)
{
```

V.Váňa: CanSAT Book for Students – part 2



```
if (TimingDelay != 0x00)
 {
    TimingDelay--;
  }
}
#ifdef USE FULL ASSERT
/**
  \star @brief Reports the name of the source file and the source line number
 *
            where the assert param error has occurred.
 * @param file: pointer to the source file name
* @param line: assert_param error line source number
 * @retval None
 */
void assert failed(uint8 t* file, uint32 t line)
{
 /* User can add his own implementation to report the file name and line
number,
  ex: printf("Wrong parameters value: file %s on line %d\r\n", file,
line) */
  /* Infinite loop */
 while (1)
  {
  }
#endif
```



CIAR Embedded Wo	orkbench IDE	Section 1
File Edit View P	roject Iools Window Help	
	Add Flies	-
Debug	Import File List	P
Files	Edit Configurations	P.
Files	Lucconngulations	
	Remove	
	Create New Project	
	Add Existing Project	
L⊕ C stm32	Options	Alt+F7
L ⊕ Bastartu	Source Code Control	+
⊕	Make	F7
	Compile	Ctrl+F7
	Rebuild All	
	Clean	15
	Batch build	F8 al
-⊞ C]SIM. -⊞ C] main.c	Stop Build	Ctrl+Break
L-⊞ 🔁 Output	Download and Debug	Ctrl+D
	Debug without Downloading	
	Make & Restart Debugger	Ctrl+R
	Restart Debugger	Ctrl+Shift+R
	Download	• •
		Tutoria familia and the IAR C-
priklad03		

Po překladu a sestavení a spuštění programu v startkitu bliká modrá LED, po stisknutí USR tlačítka zelená a zrychlí se blikání modré.

POZNÁMKA:

V.Váňa: CanSAT Book for Students – part 2 350



Debug		
Files	?	D.
🗆 🗊 priklad03 - Debug	~	
└─⊞ 🖸 stm32f10x_it.c		
–⊕ 🔁 StdPeriph_Driver		
H 🔁 🗀 stm32∨ldiscovery		
⊢⊞ 🖸 main.c		
🖵 🔁 Output		
└─⊞ 🗋 priklad03.out		

Všimněme si popisu priklad03 – **Debug**. Pokud již budeme s naším programem spokojeni, budeme jistě chtít verzi **Release**.



File Edit View	Project Tools Window Help	
Debug	Add Files Add Group Import File List	
Files	Edit Configurations	
□	Remove	
-⊕ ⊡ src -⊕ ⊡ stm3; -⊕ ⊡ startup	Create New Project Add Existing Project	
H⊞ C StdPeri	Options	Alt+F7
-⊞ C main.c	Source Code Control	•
Uutput	Make	F7
	Compile	Ctrl+F7
	Rebuild All	
priklad03	Clean	
	Batch build	F8
Messages	Stop Build	Ctrl+Break
STM32vld	Download and Debug	Ctrl+D
core_cm3	Debug without Downloading	
main.c	Make & Restart Debugger	Ctrl+R
startup_str	Restart Debugger	Ctrl+Shift+R
stm32f10x	Download	•

onfigurations:	ОК
Debug Release	New
	Remove

Přepneme na **Release**, potvrdíme **OK**.

V.Váňa: CanSAT Book for Students – part 2 352



Files	\$2	D.
🗆 🗇 priklad03 - Release	~	
│ └─⊞ 💽 stm32f10x_it.c		
⊨⊞ 🔁 startup		
H 🖸 StdPeriph_Driver		
H 🔁 🗀 stm32vldiscovery		
H 🔁 💽 main.c		
🖵 🔁 Output		
└─⊞ 🗋 priklad03.out		

Ve skutečnosti to tak jednoduché není. Výše uvedený obrázek se objeví s červenými tečkami, překlad hlásí velké množství chyb. Musíme totiž nastavit **Options**, stejně jako jsme to prováděli pro **Debug** verzi.

3.4.4 Další blikač (priklad04)

Od předchozího příkladu se liší jen obsahem souboru **main.c**, takže postup bude stejný a uvedeme jen odlišnosti.





```
/**
priklad04 - stridave blikani zelene a modre LED
 */
/* Includes -----*/
#include "stm32F10x.h"
#include "STM32vldiscovery.h"
/* Private typedef -----*/
/* Private define -----*/
#define LSE_FAIL_FLAG_0x80
#define LSE_PASS_FLAG_0x100
/* Private macro ----*/
```

354

V.Váňa: CanSAT Book for Students – part 2

ARM STM32 programming



```
/* Private consts ------
---*/
/* Private variables -----
---*/
u32 LSE Delay = 0;
u32 count = 0;
u32 BlinkSpeed = 0;
u32 KeyState = 0;
static IO uint32 t TimingDelay;
/* Private function prototypes
---*/
void Delay(uint32 t nTime);
void TimingDelay Decrement(void);
/* Private functions -----
---*/
/**
 * @brief Main program.
 * @param None
 * @retval None
 */
int main(void)
{
 /* Enable GPIOx Clock */
 RCC APB1PeriphClockCmd(RCC APB1Periph PWR, ENABLE);
  /* Initialise LEDs LD3&LD4, both off */
 STM32vldiscovery LEDInit(LED3);
 STM32vldiscovery LEDInit(LED4);
 STM32vldiscovery LEDOff(LED3);
 STM32vldiscovery LEDOff(LED4);
  /* Setup SysTick Timer for 1 msec interrupts */
 if (SysTick Config(SystemCoreClock / 1000))
  {
   /* Capture error */
   while (1);
  }
  /* Enable access to the backup register => LSE can be enabled */
 PWR BackupAccessCmd(ENABLE);
  /* Enable LSE (Low Speed External Oscillation) */
 RCC LSEConfig(RCC LSE ON);
  /* main while */
 while(1)
  {
```

355

V.Váňa: CanSAT Book for Students – part 2

ARM STM32 programming



```
STM32vldiscovery LEDOff(LED3);
    STM32vldiscovery_LEDOn(LED4);
   Delay(1000);
   STM32vldiscovery LEDOff(LED4);
   STM32vldiscovery LEDOn(LED3);
   Delay(1000);
  }
}
/**
  * @brief Inserts a delay time.
 * @param nTime: specifies the delay time length, in milliseconds.
  * @retval None
  *,
void Delay(uint32 t nTime)
{
 TimingDelay = nTime;
 while(TimingDelay != 0);
}
/**
 * @brief Decrements the TimingDelay variable.
 * @param None
  * @retval None
  */
void TimingDelay Decrement(void)
{
 if (TimingDelay != 0x00)
   TimingDelay--;
 }
}
#ifdef USE FULL ASSERT
 * @brief Reports the name of the source file and the source line number
 *
    where the assert param error has occurred.
 * @param file: pointer to the source file name
 * @param line: assert param error line source number
  * @retval None
  */
void assert failed(uint8 t* file, uint32 t line)
 /* User can add his own implementation to report the file name and line
number,
    ex: printf("Wrong parameters value: file %s on line %d\r\n", file,
line) */
 /* Infinite loop */
 while (1)
  {
  }
```



#endif



Odpojíme a znovu připojíme startkit – zelená a modrá LED se opravdu střídavě rozsvěcují a zhasínají.

3.4.5 LCD displej poprvé (priklad05)

Použijeme stejné LCD.c a LCD.h, jako v programech s uVision4 Keil a stejné adresáře a podadresáře, jako v předchozích programech, takže jen stručně: přidáme stm32f10x_flash.c a stm32f10x_tim.c





358

ARM STM32 programming



```
/* Private typedef -----
---*/
/* Private define -----
___*/
#define LSE FAIL FLAG 0x80
#define LSE PASS FLAG 0x100
/* Private macro -----
---*/
/* Private consts -----
---*/
/* Private variables ------
___*/
u32 LSE Delay = 0;
u32 \text{ count} = 0;
u32 BlinkSpeed = 0;
u32 KeyState = 0;
static __IO uint32_t TimingDelay;
uint8 t LCD Message1[] = "Test displeje.";
/* Private function prototypes -----
___*/
void Delay(uint32 t nTime);
void TimingDelay Decrement(void);
/* Private functions -----
---*/
/**
 * @brief Main program.
 * @param None
 * @retval None
 */
int main(void)
{
 /* Enable GPIOx Clock */
 RCC APB1PeriphClockCmd(RCC APB1Periph PWR, ENABLE);
 /* Initialise LEDs LD3&LD4, both off */
 STM32vldiscovery LEDInit(LED3);
 //STM32vldiscovery LEDInit(LED4);
 STM32vldiscovery LEDOff(LED3);
 //STM32vldiscovery LEDOff(LED4);
 /* Setup SysTick Timer for 1 msec interrupts */
 if (SysTick Config(SystemCoreClock / 1000))
 {
   /* Capture error */
   while (1);
 }
```

359



```
/* Enable access to the backup register => LSE can be enabled */
 PWR BackupAccessCmd(ENABLE);
  /* Enable LSE (Low Speed External Oscillation) */
 RCC LSEConfig(RCC LSE ON);
 LCD Inicialization();
 printLCD(LCD Message1);
  /* main while */
 while(1)
  {
   STM32vldiscovery LEDOff(LED3);
   STM32vldiscovery LEDOn(LED4);
   Delay(1000);
   STM32vldiscovery_LEDOff(LED4);
   STM32vldiscovery LEDOn(LED3);
   Delay(1000);
  }
}
/ * *
 * @brief Inserts a delay time.
 * @param nTime: specifies the delay time length, in milliseconds.
  * @retval None
  */
void Delay(uint32 t nTime)
{
 TimingDelay = nTime;
 while(TimingDelay != 0);
}
/**
 * @brief Decrements the TimingDelay variable.
 * @param None
 * @retval None
 */
void TimingDelay Decrement(void)
{
 if (TimingDelay != 0x00)
 {
   TimingDelay--;
  }
}
#ifdef USE FULL ASSERT
 * @brief Reports the name of the source file and the source line number
           where the assert param error has occurred.
 * @param file: pointer to the source file name
  * @param line: assert param error line source number
  * @retval None
  */
```

360

V.Váňa: CanSAT Book for Students – part 2


```
void assert_failed(uint8_t* file, uint32_t line)
{
    /* User can add his own implementation to report the file name and line
number,
    ex: printf("Wrong parameters value: file %s on line %d\r\n", file,
line) */
    /* Infinite loop */
    while (1)
    {
    }
}
#endif
```

A ještě LCD.c a LCD.h

```
/* STM32 VL Discovery Library
 * LCD driver LCD.c
 * ver 1.0
 * SPSE Jecna 2011 Vladimir Vana
 */
/* Includes ----
 __*/
#include "LCD.h"
#include "stm32f10x tim.h"
/** @defgroup Private TypesDefinitions */
TIM TimeBaseInitTypeDef TIM TimeBaseStructureMili =
{36000,TIM CounterMode Up,0,0,0};
/** @defgroup Private Defines
                                          */
                                          */
/** @defgroup Private Macros
/** @defgroup Private Variables
                                                */
/** @defgroup Private FunctionPrototypes*/
void delay ms(uint16 t time);
void LCD Inicialization(void);
void SendCommandLCD(uint8_t cmd);
void SendCharLCD(uint8 t character);
void LCD Clear(void);
void LCD CursorHome(void);
void printLCD(uint8_t* message);
void SendBitsLCD(uint8 t bits);
void Send4bitLCD(uint8 t cmd);
void FirstRow(void);
void SecondRow(void);
/** @defgroup Private Functions
                                                */
```



```
* @brief Nastaveni LCD a pinu.
  * @retval : None
  */
void LCD Inicialization(void)
  GPIO InitTypeDef GPIO InitStructure;
 RCC APB2PeriphClockCmd(LCD PORT CLK, ENABLE);
 GPIO InitStructure.GPIO Pin = LCD E | LCD RS | LCD D4 | LCD D5 | LCD D6 |
LCD D7;
 GPIO InitStructure.GPIO Mode = GPIO Mode Out PP;
  GPIO InitStructure.GPIO Speed = GPIO Speed 50MHz;
  GPIO Init(LCD PORT, & GPIO InitStructure);
 GPIO ResetBits (LCD PORT, LCD RS);
     //RS=0:Instruction
  GPIO ResetBits (LCD PORT, LCD E);
     //E=0:Disable
  //delay us(200000);
  delay ms(200);
 Send4bitLCD(0x30);
  //delay_us(8000);
                      // dle datasheetu cekat vice nez 4.1 ms
  delay ms(8);
  GPIO ResetBits(LCD PORT, LCD E);
                                                            //E=0:Disable
 Send4bitLCD(0x30);
  //delay_us(1000);
 delay ms(1);
 GPIO ResetBits (LCD PORT, LCD E);
                                                            //E=0:Disable
 Send4bitLCD(0x30);
  //delay us(1000);
 delay ms(1);
 GPIO ResetBits (LCD PORT, LCD E);
                                                            //E=0:Disable
 // delay us(1000);
 delay ms(1);
 // ted by mel byt display v 8-bitovem modu a definitivne ho prepneme do
4-bitoveho modu!
 Send4bitLCD(0x20);
  //delay us(1000);
 delay ms(1);
 GPIO ResetBits (LCD PORT, LCD E);
                                                            //E=0:Disable
  //delay us(1000);
 delay ms(1);
  // nyni jsme ve 4-bitovem rezimu
  // Prikaz 0 0 1 DL N F - -
  // N = pocet radek 1=2 radky, 0=1 radka
 // DL = bitu komunikace 1=8bitu, 0=4bity
  // F = Font 1=5x10bodu, 0=5x8 bodu
 SendCommandLCD(0x28);
 //delay us(1000);
 delay ms(1);
  // zvedej sam adresu, posunuj kurzor
  SendCommandLCD(0x0E);
                          //nyni OK
 //delay us(1000);
V.Váňa: CanSAT Book for Students – part 2
                                                         ARM STM32 programming
                                     362
```



```
delay ms(1);
  //PosliPrikazLCD(0b00000110);
 LCD Clear(); // vymazani LCD a navrat kurzoru na prvni pozici prvniho
radku
}
/**
  * @brief Smaze LCD a nastavi LCD kurzor na home pozici
 * @retval : None
 */
void LCD Clear(void)
{
   SendCommandLCD(0x01);
}
/**
 * @brief Nastavi LCD kurzor na home pozici
 * @retval : None
 */
void LCD CursorHome(void)
{
   SendCommandLCD(0x02);
}
/**
 * @brief Posle prikaz do LCD
 * @param cmd prikaz do LCD
 * @retval : None
 */
void SendCommandLCD(uint8 t cmd) //v promenne cmd mame zaslany prikaz
{
     GPIO ResetBits (LCD PORT, LCD RS);
     //RS=0:Instruction
     SendBitsLCD(cmd); // posle bity jako instrukci
}
/**
 * @brief Posle znak do LCD
 * @param character znak
 * @retval : None
 */
void SendCharLCD(uint8 t character) //v promenne cmd je znak poslany na LCD
{
     GPIO SetBits(LCD PORT, LCD RS);
                                     //RS=1:Data
     SendBitsLCD(character); // posle bity jako data
}
  * @brief Posle bity do LCD
  * @param bits znak nebo prikaz
  * @retval : None
  */
void SendBitsLCD(uint8_t bits)
```

V.Váňa: CanSAT Book for Students – part 2

ARM STM32 programming



```
{
     Send4bitLCD(bits); // zaslani bitu 4-7
     bits = bits<<4; // bitovy posun na zaslani bitu 0 az 3</pre>
     Send4bitLCD(bits); // zaslani bitu 0-3
}
/**
  * @brief Posle retezec znaku do LCD
 * @param message pointer na retezec
 * @retval : None
 */
void printLCD(uint8 t* message)
{
     uint8 t i=0;
     if (!message) return;
     while (message[i]!='\0')
      {
           if(i%20==0){
                 if(i%40==0){ // pokud je na konci druheho radku prejde na
zacatek prvniho
                       FirstRow();
                       delay ms(1);
                                         // pokud je na konci prvniho radku
                 }else{
prejde na zacatek druheho
                    SecondRow();
                 }
           }
           SendCharLCD(message[i]);
           i++;
     }
}
/**
 * @brief Prejde na prvni radek
 * @retval : None
 */
void FirstRow(void)
{
     SendCommandLCD(0x80); // prejde na zacatek prvniho radku - na adresu
prvniho znaku
    delay ms(1);
}
/**
 * @brief Prejde na druhy radek
 * @retval : None
 */
void SecondRow(void)
{
    SendCommandLCD(0xC0); // prejde na zacatek druheho radku - na
adresu prvniho znaku
     delay ms(1);
}
  * @brief Posle po 4 bitech prikaz nebo znak do LCD
```

V.Váňa: CanSAT Book for Students – part 2 364

ARM STM32 programming



```
* @param command
  * @retval : None
 */
void Send4bitLCD(uint8 t cmd)
{
    GPIO ResetBits(LCD PORT, LCD E);
     delay ms(1);
      if (cmd & 0x10) // vymaskovani 4 bitu
      {
            GPIO SetBits(LCD PORT, LCD D4);
      } else {
            GPIO ResetBits(LCD PORT, LCD D4);
      }
      if (cmd & 0x20) // vymaskovani 5 bitu
      {
           GPIO SetBits (LCD PORT, LCD D5);
      } else {
           GPIO ResetBits (LCD PORT, LCD D5);
      }
      if (cmd & 0x40) // vymaskovani 6 bitu
      {
           GPIO SetBits (LCD PORT, LCD D6);
      } else {
           GPIO ResetBits (LCD PORT, LCD D6);
      }
      if (cmd & 0x80) // vymaskovani 7 bitu
      {
           GPIO SetBits (LCD PORT, LCD D7);
      } else {
           GPIO ResetBits (LCD PORT, LCD D7);
      }
     GPIO SetBits(LCD PORT, LCD E);
      //E=1:Enable
     delay ms(1);
     GPIO ResetBits(LCD PORT, LCD E);
                                                               //E=0:Disable
}
 **
 * @brief Cekat n milisekund
  * @param time kolik ms se ma cekat
  * @retval None
  */
void delay ms(uint16 t time)
 RCC APB1PeriphClockCmd(RCC APB1Periph_TIM4, ENABLE);
 TIM TimeBaseStructureMili.TIM Period = ((time+1) * 2)-1;
 TIM TimeBaseInit(TIM4, &TIM TimeBaseStructureMili);
 TIM SelectOnePulseMode(TIM4, TIM OPMode Single);
 TIM SetCounter(TIM4,2);
 TIM Cmd(TIM4, ENABLE);
 while (TIM GetCounter(TIM4)){};
 TIM Cmd(TIM4, DISABLE);
```



LCD.h

```
/* STM32 VL Discovery Library
 * LCD driver LCD.c
 * ver 1.0
 * Vd for SPSE Jecna 2011 lessons
 */
#ifndef LCD H
#define LCD H
/* Includes -----
---*/
#include "stm32f10x.h"
/** @defgroup Exported Types
                                        */
/** @defgroup Exported Functions */
void delay ms(uint16 t time);
void LCD Inicialization(void);
void SendCommandLCD(uint8 t cmd);
void SendCharLCD(uint8 t cmd);
void LCD Clear(void);
void LCD CursorHome(void);
void printLCD(uint8 t* message);
void Send4bitLCD(uint8 t cmd);
void FirstRow(void);
void SecondRow(void);
/** @defgroup Defines */
// piny pro ovladani displeje - zapojeni stejne jako na Katedre mereni CVUT
#defineLCD_EGPIO_Pin_14#defineLCD_RSGPIO_Pin_15#defineLCD_D4GPIO_Pin_10#defineLCD_D5GPIO_Pin_11#defineLCD_D6GPIO_Pin_12#defineLCD_D7GPIO_Pin_13
#define LCD PORT GPIOB
#define LCD PORT CLK RCC APB2Periph GPIOB
#endif /* LCD H */
```

3.4.6 LCD displej podruhé (priklad06)

Trochu jiný přístup najdete v projektu na ruském serveru <u>http://eugenemcu.ru/publ/13-1-0-75</u>



a IAR Embedded Workbench IDE	
File Edit View Project Tools Window	Help
Workspace ×	ain.c hd44780_emcu.c hd44780_emcu.h
Debug ▼ Files \$2 Bb STM32HD44780 ✓ H 0 hd44780_emcu.c ✓ H 0 main.c ✓ H 0 system_stm32f10x.c ✓ H 0 system_stm32f10x.c ✓ H 0 system_stm32f10x.c ✓ H 0 system_stm32f10x.d ✓ H 0 system_stm32f10x.d ✓ H 1 system_stm32f10x.d ✓ H 1 system_stm32f10x.d ✓ H 2 system_stm32f10x.d ✓ H 3 system_stm32f10x.d ✓ H 3 system_stm32f10x.d ✓ H 4 system_stm32f10x.d ✓ H 3 system_stm32f10x.d ✓ H 4 system_stm32f10x.d ✓ H 3 system_stm32f10x.d ✓ H 4 system_stm32f10x.d ✓ H 5 system_stm32f10x.d ✓ H 5 system_stm32f10x.d ✓ H 6 system_stm32f10x.d ✓ H 7 system_stm32f10x.d ✓ H 7 system_stm32f10x.d ✓ H 7 system_stm32f10x.d ✓ H 7 system_stm32f10x.d ✓ <th><pre>#include "stm32F10x.h" #include "gpio_emcu.h" #include "hd44780_emcu.h" #include <stdio.h> void InitRCC (void); void InitGPIO (void); int main(void) { InitGPIO(); // InitGPIO(); // IcdE_INIT (); CREATE_CHAR (0, 0x0F, 0x1F, 0x11, 1, 1, 0x11, 0x1F, 0x0F); CREATE_CHAR (1, 0, 0, 0x0F, 0x1F, 0x1F, 0x0F, 0, 0); Icd_VIEW_MODE (OFF, OFF, 0x1; 0x1F, 0x0F, 0, 0); Icd_VIEW_MODE (OFF, OFF, 0x1; 0x1F, 0x0F, 0, 0); Icd_VIEW_MODE (OFF, 0FF, 0x1; 0x1F, 0x0F, 0, 0); Icd_UIEW_SCLK_FREQ/50); while (tmp0) tmp0; Icd_GOTO (0); printf ("STM32F100 s"); Icd_STRING ("LCD HD44780"); for (u0 n=27; n!=0;n) </stdio.h></pre></th>	<pre>#include "stm32F10x.h" #include "gpio_emcu.h" #include "hd44780_emcu.h" #include <stdio.h> void InitRCC (void); void InitGPIO (void); int main(void) { InitGPIO(); // InitGPIO(); // IcdE_INIT (); CREATE_CHAR (0, 0x0F, 0x1F, 0x11, 1, 1, 0x11, 0x1F, 0x0F); CREATE_CHAR (1, 0, 0, 0x0F, 0x1F, 0x1F, 0x0F, 0, 0); Icd_VIEW_MODE (OFF, OFF, 0x1; 0x1F, 0x0F, 0, 0); Icd_VIEW_MODE (OFF, OFF, 0x1; 0x1F, 0x0F, 0, 0); Icd_VIEW_MODE (OFF, 0FF, 0x1; 0x1F, 0x0F, 0, 0); Icd_UIEW_SCLK_FREQ/50); while (tmp0) tmp0; Icd_GOTO (0); printf ("STM32F100 s"); Icd_STRING ("LCD HD44780"); for (u0 n=27; n!=0;n) </stdio.h></pre>
	fol M
 Messages Total number of errors: 0 Total number of warnings: 0 	



Carl Embedded Workbench IDE	
File Edit View Project Debug Disassembly ST-LINK Tools Window Help	
· D 🖨 🖬 🕼 🕼 🗠 이 👘 🗸 💊 🧏 🧏 🛙	v 🗗 🗗 🦛 📴 😳 😥 🦺 🧄
与▲뎡ኈ솒쑝푱삥¥	
Workspace × main c hd44780 emou c hd44780 emou h	Disassembly x Register x
Debug	Go to
Files \$2.02; Image: Sine Sine Sine Sine Sine Sine Sine Sine	main: R0 = 0x0000000 0x8001dea: 0xb086 R1 = 0x0000000 InitRCC(); // R1 = 0x0000000 0x8001dea: 0xf000 0xf8d5 R1 = 0x0000000 InitGPIO(); // R2 = 0x2000000 0x8001dea: 0xf000 0xf8d5 R4 = 0xFE07AB2 InitGPIO(); // // R4 = 0xFE07AB2 0x8001df0: 0xf70f 0xf8c6 R6 = 0xF556C9FF 0x8001df1: 0x200f R6 = 0xF556C9FF 0x8001df2: 0x200f R8 = 0x539D4EEC 0x8001df2: 0x200f R9 = 0xE46F6F4F 0x8001df2: 0x201f R11 = 0x5000000 0x8001df2: 0x201f R12 = 0x8000000 0x8001df2: 0x2011 R12 = 0x8000000 0x8001e00: 0x2011 R13 (SP) = 0x2000408 R14 (LR) 0x60002183 HAPSR = 0x6000000
volatile u32 tmp0; tmp0=(5YSCLK_TREQ/50); while (tm LCD_GOTO (0); printf ("STM32F100 c"); LCD_GOTO (16); STM32HD44780_emcu_iar F@ < m + 10000000000000000000000000000000000	0x8001e0s: 0x8001 0x8001e0s: 0x8001 0x8001e0s: 0x8001 0x8001e0s: 0x801 0x8001e10: 0x211f 0x8001e12: 0x2000 0x8001e12: 0x7ff 0x8001e14: 0xfaff
Thu Jul 1411:30:50 2011: 8880 bytes downloaded into FLASH and verified (3.68 Kbyth Thu Jul 1411:30:50 2011: Loaded debugee: C:\ARM_priklady\IAR\priklad06\projec Thu Jul 1411:30:51 2011: Target reset	es/sec) t\Debug\Exe\c.out
Ready	Ln 9, Col 5

LCD zapojíme

(6)	PC12
(5)	PC11
(4)	PC10
(11)	PA8
(12)	PA9
(13)	PA10
(14)	PA11
	 (6) (5) (4) (11) (12) (13) (14)

3.4.7 LCD displej a klávesnice PS2 (priklad07)

převzato z ruského serveru http://eugenemcu.ru/publ/13-1-0-75

3.4.8 I2C komunikace s EEPROM AT24C022 (priklad08)



Value IAR Embedded Workbench IDE	
File Edit View Project Tools Window Help	
0 😂 🖬 🕼 🙈 🗼 🖻 🖬 😂 🖸	
Workspace	x main a
Debug	
	void T2C Inicializace(void): // nestavení T2C
Files č~	uint8_t Read EEFROM(uint8 t adresa); // cteni
🗆 🗇 priklad08 - Debug 🗸 🗸	void Write_EEPROM(uint8_t adresa, uint8_t data); // zapis
	void Ready_EEPROM(void); // polling potvrzeni
Htt C core_cm3.c	E E
System_stm32f10x.c	
	void bliknutilEDMODRA(void); //
	void bitknutileDzelena(void); //
H Startup	int wain (word)
Here and the startup_stm 32110x_md_Vi.s	
E Starenpn_Unver	
H H C misc.c	GPIO Inicializace();
E stm 32/10x_extl.c	I2C Inicializace();
the stm32f10x_gpio.c	// blikneme modrou jako priznak startu
□ □ □ ================================	bliknutiLEDMODRA();
E Stm32riux_pwr.c	/* pouzita EEPROM AT24C02 - 256 bajtu x 8 bitu */
□ □ □ stm 32110x_rcc.c	
	hodnota = Read_EEPROM(0x00);
	hodnota = Read_EEPROM(0x01);
	hodnota = Read_EEPROM(0x02);
priklad08	hodnota = Read EEPROM(0x03);
Messages	File Line ^
stm32f10x it.c	
stm32f10x_pwr.c	
stm32f10x_rcc.c	
system_stm32f10x.c	
Linking	
priklad08.out	
	=
Total number of errors: 0	
Total number of warnings: 0	
a l 5	
Ready	Errors 0, Warnings 0



V.Váňa: CanSAT Book for Students – part 2

ARM STM32 programming



```
* I2C komunikace s EEPROM (AT24C02)
* dle 29. dilu serialu o STM32
* priklad 08 main.c
* Vd pro SPSE Jecna Lectures
                    _____
/* Includes -----
#include "stm32F10x.h"
#include "stm32f10x i2c.h"
#include "STM32vldiscovery.h"
/* Private typedef -----
GPIO InitTypeDef GPIO InitStructure;
#define EEPROM SLAVE ADDRESS 0xA0 //protoze je to EEPROM
//SCL bude na PB6 DATA na PB7 AT24C02
                            (A0=0 je write do obvodu, A1=1 je read )
//Tabulka I2C slave adres
                              A6A5A4A3A2A100aaanapr
//Typ obvodu
                         Α7
//8bitovy vstup/vystup 0 1
                              0 0
                                                  a napr.
PCF8574
                 1 0 1 0 a a a napr.
//EEPROM
PCF8582
//Hodiny/kalendar 1 0
//8bitovy A/D a D/A 1
                         1
                              0
                                   0
                                        0
                                             a napr. PCF8583
                         0
                              0
                                   1
                                        а
                                             а
                                                  a napr.
PCF8591
                   1 1 0 0 0 a a napr.
//PLL
TSA5055
/* Private variables -----
uint8 t hodnota=0;
static __IO uint32_t TimingDelay;
/* Private function prototypes --
void Delay(__IO uint32_t nTick);
void TimingDelay_Decrement(void);
void GPIO Inicializace(void); // nastavení vstupne/vystupnich pinu na kitu
void I2C Inicializace(void); // nastavení I2C
uint8 t Read EEPROM(uint8 t adresa); // cteni
void Write EEPROM(uint8 t adresa, uint8 t data); // zapis
                                  // polling potvrzeni
void Ready EEPROM(void);
void bliknutiLEDMODRA(void);
                             11
                                   11
void bliknutiLEDZELENA(void);
int main(void)
{
GPIO Inicializace();
I2C Inicializace();
// blikneme modrou jako priznak startu
V.Váňa: CanSAT Book for Students – part 2 370
                                               ARM STM32 programming
```



```
bliknutiLEDMODRA();
  /* pouzita EEPROM AT24C02 - 256 bajtu x 8 bitu */
 hodnota = Read EEPROM(0x00);
 hodnota = Read EEPROM(0x01);
 hodnota = Read EEPROM(0x02);
 hodnota = Read EEPROM(0x03);
hodnota = Read_EEPROM(0x04);
 Write EEPROM(0x00, 0xA1);
 Ready EEPROM();
 bliknutiLEDZELENA();
 Write EEPROM(0x01, 0xB2);
 Ready EEPROM();
 bliknutiLEDMODRA();
 Write EEPROM(0x02, 0xC3);
 Ready EEPROM();
 bliknutiLEDZELENA();
 Write EEPROM(0x03, 0xD4);
 Ready_EEPROM();
 bliknutiLEDMODRA();
 hodnota = Read EEPROM(0x00);
 hodnota = Read EEPROM(0x01);
 hodnota = Read\_EEPROM(0x02);
 hodnota = Read\_EEPROM(0x03);
 hodnota = Read EEPROM(0x04);
  // blikneme zelenou jako priznak konce
 bliknutiLEDMODRA();
  /* main while */
 while(1)
 {
    bliknutiLEDZELENA();
  }
}
void Delay( IO uint32 t nTick)
{
 for(; nTick != 0; nTick--);
}
void TimingDelay Decrement(void)
{
 if (TimingDelay != 0x00)
  {
    TimingDelay--;
  }
}
void GPIO Inicializace(void)
{
 STM32vldiscovery_LEDInit(LED3);
```

371



```
STM32vldiscovery LEDInit(LED4);
 STM32vldiscovery PBInit (BUTTON USER, BUTTON MODE GPIO);
 STM32vldiscovery LEDOff(LED3);
 STM32vldiscovery LEDOff(LED4);
void bliknutiLEDMODRA(void)
{
     STM32vldiscovery LEDOn(LED4); // zapnem LED4 - modra
     Delay(0x5FFFF);
     STM32vldiscovery LEDToggle(LED4); // vypnem LED4 - modra
     Delay(0x5FFFF);
}
void bliknutiLEDZELENA(void)
{
      STM32vldiscovery LEDOn(LED3); // zapnem LED3 - zelena
      Delay(0x5FFFF);
      STM32vldiscovery_LEDToggle(LED3); // vypnem LED3 - zelena
      Delay(0x5FFFF);
}
void I2C Inicializace(void)
 GPIO_InitTypeDef GPIO_InitStructure;
 I2C InitTypeDef I2C InitStructure;
 I2C DeInit(I2C1);
  /* I2C Periph clock enable */
 RCC APB1PeriphClockCmd(RCC APB1Periph I2C1, ENABLE);
  /* GPIO Periph clock enable */
 RCC APB2PeriphClockCmd(RCC APB2Periph GPIOB, ENABLE);
  /* Configure I2C pins: SCL and SDA */
 GPIO InitStructure.GPIO Pin = GPIO Pin 6 | GPIO Pin 7;
 GPIO InitStructure.GPIO Speed = GPIO Speed 10MHz;
 GPIO InitStructure.GPIO Mode = GPIO Mode AF OD; //GPIO Mode AF PP
 GPIO Init(GPIOB, &GPIO InitStructure); //I2C EEPROM pripojime k PB
  //SCL bude na PB6 DATA na PB7 AT24C02
  /* I2C configuration */
 I2C InitStructure.I2C Mode = I2C Mode I2C;
 I2C InitStructure.I2C DutyCycle = I2C_DutyCycle_2;
 I2C_InitStructure.I2C OwnAddress1 = 0xA0; //EEPROM
 I2C InitStructure.I2C Ack = I2C Ack Enable;
 I2C InitStructure.I2C AcknowledgedAddress = I2C AcknowledgedAddress 7bit;
 I2C InitStructure.I2C ClockSpeed = 10000;
 /* I2C Peripheral Enable */
 I2C Cmd(I2C1, ENABLE);
 /* Apply I2C configuration after enabling it */
 I2C Init(I2C1, &I2C InitStructure);
```



```
uint8 t Read EEPROM(uint8 t adresa)
{
       uint8_t Data;
       Data = 0;
       /* While the bus is busy */
        while(I2C_GetFlagStatus(I2C1, I2C FLAG BUSY));
        /* Send START condition */
        I2C GenerateSTART(I2C1, ENABLE);
        /* Test on EV5 and clear it */
       while(!I2C CheckEvent(I2C1, I2C EVENT MASTER MODE SELECT));
        /* Send EEPROM address for read */
        I2C Send7bitAddress(I2C1, EEPROM SLAVE ADDRESS,
I2C Direction Transmitter);
        /* Test on EV6 and clear it */
       while(!I2C CheckEvent(I2C1,
I2C EVENT MASTER TRANSMITTER MODE SELECTED));
        /* Clear EV6 by setting again the PE bit */
        I2C Cmd(I2C1, ENABLE);
       /\star Send the EEPROM's internal address to read from: MSB of the
address first */
       I2C SendData(I2C1, adresa);
        /* Test on EV8 and clear it */
       while(!I2C CheckEvent(I2C1, I2C EVENT MASTER BYTE TRANSMITTED));
        /* Send START condition a second time */
       I2C GenerateSTART(I2C1, ENABLE);
        /* Test on EV5 and clear it */
        while(!I2C CheckEvent(I2C1, I2C EVENT MASTER MODE SELECT));
        /* Send EEPROM address for read */
        I2C Send7bitAddress(I2C1, EEPROM SLAVE ADDRESS,
I2C Direction Receiver);
        /* Test on EV6 and clear it */
       while(!I2C CheckEvent(I2C1,
I2C EVENT MASTER RECEIVER MODE SELECTED));
        /* Disable Acknowledgement */
       I2C AcknowledgeConfig(I2C1, DISABLE);
        /* Send STOP Condition */
       I2C GenerateSTOP(I2C1, ENABLE);
        while(!I2C CheckEvent(I2C1, I2C EVENT MASTER BYTE RECEIVED)){}
        /* Read a byte from the EEPROM */
        Data = I2C ReceiveData(I2C1);
```



```
/* Enable Acknowledgement to be ready for another reception */
        I2C AcknowledgeConfig(I2C1, ENABLE);
        return(Data);
void Write EEPROM(uint8 t adresa, uint8 t data)
{
        /* Send START condition */
        I2C GenerateSTART(I2C1, ENABLE);
        /* Test on EV5 and clear it */
        while(!I2C CheckEvent(I2C1, I2C EVENT MASTER MODE SELECT));
        /* Send EEPROM address for write */
        I2C Send7bitAddress(I2C1, EEPROM SLAVE ADDRESS,
I2C Direction Transmitter);
        /* Test on EV6 and clear it */
        while(!I2C CheckEvent(I2C1,
I2C EVENT MASTER TRANSMITTER MODE SELECTED));
        /* Send the EEPROM's internal address to write to : MSB of the
address first */
       I2C SendData(I2C1, adresa);
        /* Test on EV8 and clear it */
        while(!I2C CheckEvent(I2C1, I2C EVENT MASTER BYTE TRANSMITTED));
        /* Send the byte to be written */
        I2C SendData(I2C1, data);
        /* Test on EV8 and clear it */
        while(!I2C CheckEvent(I2C1, I2C EVENT MASTER BYTE TRANSMITTED));
        /* Send STOP condition */
        I2C GenerateSTOP(I2C1, ENABLE);
void Ready EEPROM(void)
{
        IO uint16 t temp = 0;
        do
        {
          /* Send START condition */
         I2C GenerateSTART(I2C1, ENABLE);
          /* Read I2C EE SR1 register to clear pending flags */
          temp = I2C ReadRegister(I2C1, I2C Register SR1);
          /* Send EEPROM address */
          I2C Send7bitAddress(I2C1, EEPROM SLAVE ADDRESS,
I2C Direction Transmitter);
```



```
}while(!(I2C_ReadRegister(I2C1, I2C_Register_SR1) & 0x0002));
/* Clear AF flag */
I2C_ClearFlag(I2C1, I2C_FLAG_AF);
/* STOP condition */
I2C GenerateSTOP(I2C1, ENABLE);
```

PonyProg	2000	- Se	eria	l De	vice	Pro	ogra	mme	er										
File Edit Dev	vice (Comm	nand	Sci	ript	Utilit	y S	ietup	?	Wi	ndow	ŧ.							
1			E)				5	ļ	6	থ	8		•	12C	Bust	3bit ee	prom 💌 2402 💌	
6	Ē	2			E	<u>¢</u>	₽		⇔		0								
🏂 No Name)																		^
00000000000000000000000000000000000000	HT FF FF FF FF FF FF FF FF FF FF FF FF FF	BZ FF FF FF FF FF FF FF FF FF FF FF FF	C3 FF FF FF FF FF FF FF FF FF FF FF FF	FF FF FF FF FF FF FF FF FF FF FF FF FF	FF	FF	FF	FF	E E EM N A A A E E EM N A A	FFF FFF FFF FFF FFF FFF FFF FFF FFFFFFF	FF	FF	FF	FF	FF	FF	FF FF FF FF FF FF FF FF FF FF FF FF FF	·····	
0000F0)	FF	FF	FF	FF	FF	FF	FF	FF	-	FF	FF	FF	FF	FF	FF	FF	FF	•••••	~
<							3	1111											>
PonyProg200	0	3	2402			Size	256	6 Byte	es	CR	C 70	3Ah							

4 Práce v prostředí Atollic TrueStudio STM32 Lite

Atollic TrueSTUDIO[®]/STM32 Lite v2.1.0 je vývojové prostředí založené na Eclipse. Lze ho zdarma stáhnout z <u>http://www.atollic.com/index.php/targets/stm32</u>. Lite verze je free, není omezená časově ani délkou kódu, ale neobsahuje některé funkčnosti TrueSTUDIO Profesional.

4.1 Instalace Atollic TrueStudio STM32 Lite

Spustíme instalační soubor, dostaneme





Klineme na Next



icense Agreement Please review the license terms before TrueSTUDIO(R) STM32 Lite.	e installing Atollic	100	
Press Page Down to see the rest of the	e agreement.		
ATOLLIC END USER SOFTWARE LICEN	NSE AGREEMENT		
December 04, 2010			
Atollic AB, Reg. No. 556615-0172, a li organised under the laws of Sweden a Jönköping, Gjuterigatan 9, SE-55318 Atollic TrueSTUDIO or Atollic TrueANA	imited liability company duly i and having its principal office Jönköping, Sweden ("Atollic LYZER or Atollic TrueINSPEC	ncorporated and at Science Park ") hereby licenses TOR or Atollic	Ŧ
If you accept the terms of the agreem agreement to install Atollic TrueSTUDIC	ent, click I Agree to continue O(R) STM32 Lite.	. You must accept the	
		LArran R Conce	

Odsouhlasíme licenční podmínky I Agree



Choose Install Location	A AMER
Choose the folder in which to install Atollic TrueSTUDIO(R) STM32 Lite.	
Setup will install Atollic TrueSTUDIO(R) STM32 Lite in the following folder different folder, click Browse and select another folder. Click Next to cor	. To install in a ntinue.
Destination Folder	
C:\Program Files (x86)\Atollic\TrueSTUDIO STM32 Lite 2.1.0	Browse
C:\Program Files (x86)\Atollic\TrueSTUDIO STM32 Lite 2.1.0 Space required: 533.0MB	Browse
C:\Program Files (x86)\Atollic\TrueSTUDIO STM32 Lite 2.1.0 Space required: 533.0MB Space available: 90.0GB	Browse

Ponecháme nabízené nastavení místa instalace a klikneme na Next. Dostaneme



License Registration	18 M	**
Please register your copy of Atollic	c TrueSTUDIO(R) STM32 Lite	
Your computer id is:		
EPIJFHMGJMAIJAPEEIGNNILOLE. FBBAMKEGLNKPLGKAEEMFNCPKK	JBMDABIGPPNPIHEBMPOLOCNHIJCFKGANDAONKB (HLJNCBIKFKNAJAJJGJK	*
4	•	Ŧ
Please register your computer id a registration form will be opened au immediately. Copy the installation	t http://www.atollic.com/index.php/registerlite (the utomatically). An installation key will then be e-mailed to key from the e-mail and paste it here:	you
		*

Obsah horního okna přeneseme do registračního formuláře na stránkách www.atolic.com



🕒 🕘 🔻 🖪 http://	www. atollic.com /index.php/regi	sterlite		• * ×
🚖 Oblíbené položky 📗	👍 🚺 Navrhované weby 🔻			
a Register Lite version			🖄 🔹 🖸	• 🖃
	Markets	Regist	er Lite version	
	Benefits	Please enter	your contact information and Computer id (generated by the installation	
	Global	program) in t	ne form below. Then press the "Register" button below to register your free se	
	Newsletter	An Installation	a key is then depended and e-mailed to you automatically. You must enter	
	Buy now	the Installatio	n key in the installation program to complete the product installation.	
	Request evaluation			
	Products	First name	Vladimir	
	Overview	Last name	Vana	
	TrueSTUDIO	E-mail	vana@spsejecna.cz	
	TrueINSPECTOR	Company	SPSE Jecha	
		Postal code	120.00	
	TrueANALYZER	City	Praque	
	TrueVERIFIER	State	Czech Republic	
	Target support	Phone	224 942 065	
	Download	Country	Czech Republic	
	Register license	Customer	Academic use	
	Resources	type	EPIJFHMGJMAIJAPEEIGNNILOLEJBMDABIGPPNPIHEBMPOLOCNHIJ 🔺	
	Brochures	Computer id		
	White papers	(from installer)	PERMINEGENTRE GRAEEMPTOP RATESTOCIAL AND ASSOCIA	
	Video tutorials		*	
	Knowledge base		Register	

Po vyplnění všech ůdajů klikneme na register. E- mailem dostaneme řetězec, který vložíme do dolního textboxu v



		A BEACH
Please register your copy of Atollic TrueST	IUDIO(R) STM32 Lite	A CARA
Your computer id is:		
EPIJFHMGJMAIJAPEEIGNNILOLEJBMDAB FBBAMKEGLNKPLGKAEEMFNCPKKHLJNCB	IGPPNPIHEBMPOLOCNHIJCFKGANDAG BIKFKNAJAJJGJK	ONKB 🔺
(*
Please register your computer id at http:// registration form will be opened automatic immediately. Copy the installation key from	/www.atollic.com/index.php/registerlit :ally). An installation key will then be e m the e-mail and paste it here:	e <mark>(</mark> the -mailed to you
	VIJPIEIEMFEDEBCKNKAMLAIEIHKIPBA	NPI 🔺
ADHBKKIHMDFMANKFCHJNAIBLJFNGLCN HCGEDHOPDAOGJEODALABFGMABM		*
ADHBKKIHMDFMANKFCHJNAIBLJFNGLCN HCGEDHOPDAOGJEODALABFGMABM <		•

Nyní již můžeme spustit instalaci tlačítkem Install



Atollic TrueSTUDIO(R) STM32	Lite Setup	
Installing Please wait while Atollic TrueSTU installed.	DIO(R) STM32 Lite is being	1000
Extract: libgcov.a 100%		
Show details		
	Sack N	ext > Cancel

Instalace probíhá



Atollic TrueSTUDIO(R) STM32 Lite Setup	
Installation Complete Setup was completed successfully.	
Completed	
Show details	
	Back Next > Cancel

Klikneme na Next



Atollic TrueSTUDIO(R) STM32 Lite Setup	
ST-LINK Driver Installation ST-LINK Driver Installation	
You are highly recommended to install the ST-LINK Drivers now. The you are planning to use an ST-LINK debug probe with TrueSTUDIO.	ese drivers are required if
☑ Install the ST-LINK Drivers when you click next. This will open a (recommended)	separate installer.
	Next > Cancel

Zaškrtneme požadavek na instalaci ST-Link driverů a klikneme na Next



STLinkDriver - InstallShield Wizard		
Preparing Setup Please wait while the InstallShield Wizard prepares the s	setup.	
STLinkDriver Setup is preparing the InstallShield Wizard, the setup process. Please wait.	. which will guide you l	through the rest of
InstallShield		
	Q	Cancel

Nainstaluji se drivery a poté se objeví

STLinkDriver - I	nstallShield Wizard
Welcome Modify, repa	ir, or remove the program.
Welcome to current insta	the STLinkDriver Setup Maintenance program. This program lets you modify the illation. Click one of the options below.
Modify	Select new program features to add or select currently installed features to remove.
O Repair	Reinstall all program features installed by the previous setup.
© Remove	Remove all installed features.
	< Back Next > > Cancel

V.Váňa: CanSAT Book for Students – part 2

ARM STM32 programming



Klikneme na **Next**

Select Features Select the features setup will install.	
Select the features you want to install, and dese	lect the features you want to uninstall. Description STLink description file for the Operating System
0.00 MB of space required on the C drive 91384.68 MB of space available on the C drive allShield	

Opět **Next**



STLinkDriver - InstallShield Wiz	ard
	Maintenance Complete InstallShield Wizard has finished performing maintenance operations on STLinkDriver.
	K Back Filtsh Cancel

A nakonec tlačítko Finish



Atollic TrueSTUDIO Professional is the premier choice for embedded developers, with hundreds of new powerful features for editing, building and debugging, revision control and bug management, graphical modeling, source code review and much, much, more.	
Contact your Atollic sales representative for information on how to upgrade to the Professional version!	

Opět budeme klikat na **Next**



inde inspection!

Znovu Next



Dynamic code coverage analysis Improve your software quality with code coverage	e analysis!
Atollic TrueANALYZER is a tool for dynamic code analysis. The product performs code coverage measurements up to the same level as required for flight control systems. A demo version of Atollic TrueANALYZER is included in this installation. Try it out!	
	< Back Next > C

Další **Next**



Embedded test automation Improve your software quality with test automatio	n!
Atollic TrueVERIFIER is a tool for advanced test automation. The product auto-generate unit test cases that are executed with execution-path monitoring in the target board. A demo version of Atollic TrueVERIFIER is included in this installation. Try it out!	
	< Back Next > Cancel

Opět Next





A Finish.

Spustíme nainstalovaný program. Zeptá se nás na umístění workspace (pokud znáte Eclipse, tak víte, že toto dělá Eclipse vždy po prvním spuštění a Atolic využívá právě Eclipse)





Místo nabízeného umístění workspace můžeme vybrat jiné

Workspace	e Launcher
Select a w Atollic Trues Choose a wo	rorkspace STUDIO®/STM32 Lite stores your projects in a folder called a workspace. orkspace folder to use for this session.
Workspace:	C:\Programy Browse
☑ Use this a	is the default and do not ask again

Klikneme na **OK**





Spustí se vývojové prostředí Atolic v Lite verzi. Ta nás občas bude obtěžovat okny, která hned zavřeme.





C/C++ - Atollic TrueSTUDIO®/STM32 Lite						J	_ D X
File Edit Source Refactor Navigate Se	arch Project Run Manuals	Window Help					
<mark>13</mark> ・日頃色 タ・ジ・や や・ウ・	• 🖻 • 🞯 • 🚿 • 🕅 •	☆ • ○ •	Q	• 🙋 🛷 •		12 10 C/	C++
Project Explorer 🕴 🗖 🗖	(🗄 Out 🖾 🤇 🖉	Ma = -
	3					An outline is not	available.
	Problems 🕅 🔎 Tasks	E Console E	Properties				
	0 items		-openies				
	Description		Resource	Path	Location	Туре	
							2
				-			
U [™] 0 items selected							

4.2 První program v TrueStudio – import hotového programu

V menu vybereme File --Import



	/C++ - Atollic TrueSTUDIO®	/STM32 Lite														L G	
ile	Edit Source Refactor	Navigate Search	Project	Run	Manual	ls Windo	w Help										
	New	Alt+Shift+N	0	- 1 %	• 🛞 •	- 恭 -	0 -	Q 📢	🋊 - 🤹	- 10	9 🔗 •	•	1		11 参 1	ebug [C/
	Open File																
	Close	Ctrl+W										- 6	🗄 🗄 Outlin	ie 🖾	 Mal 	ke Targe	-
	Close All	Ctrl+Shift+W											An outlin	e is not a	vailable.		
	Save	Ctrl+S															
	Save As																
6	Save All	Ctrl+Shift+S															
	Revert																
	Move																
	Rename	F2															
8	Refresh	F5															
	Convert Line Delimiters To																
8	Print	Ctrl+P															
	Switch Workspace	4															
	Restart																
2	Import																
4	Export	10															
	Properties	Alt+Enter															
	Exit																
_			Pi	oblems	🖉 Tasl	ks 📮 Cor	nsole 🖾	P	roperties						e E] - [9	• -
			No co	nsoles t	o display	at this tim	ie.										
E.S	Q itoms colosted											1					
1	o items selected																

Objeví se okno Select


a subors	
Select Choose import source.	2
Select an import source:	
type filter text General Control Archive File Existing Projects into Workspace File System Preferences C/C++ Run/Debug Feam	

V General vybereme Existing Projects into Workspace, objeví se



Import	
Import Projects Select a directory to search for existing Eclipse projects.	
Select root directory: Select archive file: Projects:	Browse Browse
	Select All
	Deselect All
	Refresh
Copy projects into workspace Working sets	
Add project to working sets	
Working sets:	* Select
Kext > Finish	Cancel

Klikneme na Browse. Máme



 Select root directory Select archive file: 	/: Browse Browse
Projects:	/yhledat složku
Copy projects ir Working sets Add project to Working sets:	Demo EWARMv5 inc MDK-ARM src TrueSTUDIO metadata Složka: TrueSTUDIO Vytvořit novou složku OK Storno

Klikneme na **OK**



mport Projects Select a directory to sear	ch for existing Eclipse projects.	
Select root directory:	C:\ARM_priklady\stm32vldiscovery_package\Pi	Browse
Select archive file:		Browse
Projects:	RM_priklady\stm32vldiscovery_package\Project\[Select All
2.7.7.7.5.4. residentiale contraction and		Deselect All
		Refresh
•		
Copy projects into we Working sets	orkspace	
Copy projects into we Working sets Add project to work	III orkspace ing sets	
Copy projects into we Working sets Add project to work Working sets:	III + orkspace ting sets	Select
Copy projects into we Working sets Add project to work Working sets:	III orkspace ing sets	Select

Zkontrolujeme, popř. opravíme dle obr. A klikneme na Finish. Dostaneme



C/C++ - Atollic TrueSTUDIO®/STM32 Lite	
File Edit Source Refactor Navigate Search	Project Run Manuals Window Help
<mark>[]</mark> ▼] () () () () () () () ()	- ③ - 《 - ◎ - 参 - ○ - Q - 後 - 参 - 参 - ◎ 『 □ Q 話 弥 Debug โ급 C/C++
陷 Project Explorer 🛛 📄 🔄 🌄 🗆	🗖 🗖 🗄 Outline 🛛 🛞 Make Targe 🔤 🗖
😂 DISCOVER	An outline is not available.
	😰 Problems 🕢 Tasks 🖳 Console 🖾 🔲 Properties 🕹 🗘 😨 🖓 😨 🔹 👘 🖛 🖄 😨 🗸 😁 🗖
	C-Build [DISCOVER]
	**** Rebuild of configuration Debug for project DISCOVER **** I
	**** Internal Builder is used for build ****
	-DSTM32F10X_MD_VL -DUSE_STM32100B_EVAL
	-I/\\Libraries\CMSIS\CM3\CoreSupport
	_T / \ \ \ Librariae\STM32F10v StdDarinh Driver\inc _T / / /inc

V project Exploreru se objeví projekt DISCOVER





Project Explorer 🛛	□ 🔄 🏹	- 0)
A 🔊 Includes		-
C:/ARM_priklady/stm32vldiscovery_package/Libraries/CMSIS/CM3/CoreSupport		
core_cm3.c		
⊳ h core_cm3.h		
C:/ARM_priklady/stm32vldiscovery_package/Libraries/CMSIS/CM3/DeviceSupport/ST/STM3	2F10x	
C:/ARM_priklady/stm32vldiscovery_package/Libraries/STM32F10x_StdPeriph_Driver/inc		
C:/ARM_priklady/stm32vldiscovery_package/Project/Demo/inc		
C:/ARM_priklady/stm32vldiscovery_package/Utilities		
🔋 🕒 C:/Program Files/Atollic/TrueSTUDIO STM32 Lite 2.1.0/ARMTools/arm-atollic-eabi/include		
👂 🕒 C:/Program Files/Atollic/TrueSTUDIO STM32 Lite 2.1.0/ARMTools/arm-atollic-eabi/usr/inclu	ıde	
🛛 🕞 📴 C:/Program Files/Atollic/TrueSTUDIO STM32 Lite 2.1.0/ARMTools/lib/gcc/arm-atollic-eabi/4	.5.1/include	
🕞 🕒 C:/Program Files/Atollic/TrueSTUDIO STM32 Lite 2.1.0/ARMTools/lib/gcc/arm-atollic-eabi/4	.5.1/include-fixed	
🔺 🔄 CMSIS		
⊳ 📑 core_cm3.c		
system_stm32f10x.c		
🗅 🗁 Debug		
👂 🚰 Doc		=
🔺 🛅 StdPeriph_Driver		
👂 者 misc.c		
▷ A stm32f10x_exti.c		
⊳ 🕞 stm32f10x_gpio.c		
⊳ 📑 stm32f10x_pwr.c		
▶ 📑 stm32f10x_rcc.c		
Cm STM32_Discovery		
STM32vldiscovery		
STM32vldiscovery.c		
TrueSTUDIO		
startup_stm32f10x_md_vl.s		
🔺 🔁 User		
🔒 main.c		
▶ 📑 stm32f10x_it.c		
STM32F100RB_FLASH.Id		25

4 6	User
	main.c
⊳	stm32f10x_it.c
····· 34	CTL INDELADOR FL LOUI



▲ <mark>()</mark> C:/	New	CMSIS/CM3/CoreSupport
⊳ <mark>.0</mark> ⊳ . 1 ⊳ . ./	Open Open With	CMSIS/CM3/DeviceSupport/ST/STM32F10x
▷ <mark>(</mark>) C:/ () ▷ () C:/ () ▷ () C:/ ()	Copy Ctrl+C Paste Ctrl+V	STM32F10x_StdPeriph_Driver/inc)emo/inc
⊳ 🕒 C:/ 💥 ⊳ 📴 C:/	Delete Delete Move	/ARMTools/arm-atollic-eabi/include /ARMTools/arm-atollic-eabi/usr/include
	Rename F2 Import	ARMTools/lib/gcc/arm-atollic-eabi/4.5.1/include ARMTools/lib/gcc/arm-atollic-eabi/4.5.1/include-fixed
 children in the system children in the system children in the system 	Export	
> 🔁 Debug 🖏 > 🔁 Doc	Refresh F5	
I 者 StdPer ▷ 🔐 mi: ▷ 🔐 stn	Make Targets Resource Configurations	
⊳ 📑 stn ⊳ 📑 stn ⊳ 📑 stn	Clean Selected File(s) Build Selected File(s)	
CTM32 C CTM32 C CTM32	Team Compare With Replace With	
i 者 TrueST	Inspection	
User	Properties Alt+Enter	

Nastavíme Properties pomocí místní nabídky



Properties for main.c		
type filter text	Resource	⇔ - ⇔
Resource C/C++ Build C/C++ General Run/Debug Settings Testing	Path: /DISCOVER/User/main.c Type: Linked File Location: C:\ARM_priklady\main.c - (does not exist) Resolved location: C:\ARM_priklady\main.c - (does not exist) Size: <resource does="" exist="" not=""> Last modified: <resource does="" exist="" not=""> Attributes: Read only Prived Text file encoding Operault (inherited from container: Cp1250) Other: Cp1250</resource></resource>	Edit
۲		OK Cancel

V Resource klikneme na Edit

Edit Link Location	
Edit a Link Location Path target is of wrong type. The path n	ust point to a file.
Location: Resolved Location:	File Variable
?	OK Cancel

Vybereme Location kliknutím na tlačítko File



Jspořádat 👻 Nov	á složka	8=	- 🗆 🌘
Nocha	 Název položky 	Datum změny	/**
🚺 Stažené soubory	main	14.9.2010 15:48	****
 Knihovny Dokumenty Hudba Obrázky Videa Domácí skupina Počítač 	stm32f10x_it	14.9.2010 15:48	
🏭 Místní disk (C:)			Demo/
👝 Místní disk (E:)	+ • III		⊧ ain.c
Ν	lázev souboru: main	✓ *.* Otevřít ✓	Storno
reme main.c			
Edit Link Location			x
dit a Link Locatic dit link location.	n		4

Klikneme na **OK**

?

Cancel

OK

23





Obdobně s dalšími soubory







Spustíme Project – Rebuil All





Zbývá program (ze souboru *.elf) umístit do flash startkitu Klikneme na **F11** (nebo na ikonku zeleného brouka), objeví se



Edit Configuration	State of the local division of the	And the second second	×
Edit launch configuration properties			×
Name: STM32100B-EVAL.elf	ze) 🕨 Startup Unit Test) 🤯 Source) 🗔	Common	
Debug/STM32100B-EVAL.elf		Search Project	B <u>r</u> owse
Project: DISCOVER			Browse
Build (if required) before launching Build configuration: Use Active			•
 Enable auto build Use workspace settings 	Disable auto build <u>Configure Workspace Settings</u>		9
		Apply	Revert
?		ОК	Cancel

Klikneme na **OK**



Debug Configurations		2000041	Contraction of the local division of the loc	×
Create, manage, and run c	onfigurations			T.
Embedded C/C++ Application STM32100B-EVAL.elf Launch Group	Name: STM32100B-EVAL.elf Main Startup D C/C++ Application: Debug/STM32100B-EVAL.elf Project: DISCOVER Build (if required) before launching Build configuration: Use Active © Enable auto build © Use workspace settings	ebug Startup Analyze O Disable auto <u>Configure Work</u>	 Startup Unit Test Search Project) build space Settings 	»2 Browse Browse
← III → Filter matched 3 of 3 items			Apply	Revert
?			Debug	Close

4.3 První vlastní program v TrueStudio

Založíme si první projekt začneme tím, že z menu vybereme File, pak New a nakonec C Project.



File	Edit Source Refactor Navigate	Search Proiec	piect Run Manuals Window Help	
	New Open File Close Close All Save Save As Save All Revert Move	Alt+Shift+N > Ctrl+W Ctrl+Shift+W Ctrl+Shift+S Ctrl+Shift+S	Image: Convertion of Convert to a C/C++ Project Image: Convert to a C/C++ Pro	
2) 4)	Rename Refresh Convert Line Delimiters To Print	F2 F5 Ctrl+P	 File from Template Class Other 	
2	Switch Workspace Restart Import Export	•	▶	
	Properties 1 main.c [DISCOVER/User] 2 core_cm3.c [ARM_priklady/] Exit	Alt+Enter	r Tasks 📮 Console 🖾 🔲 Properties o display at this time.	

(předtím ovšem zavřeme všechny již existující projekty, např. DISCOVER)



C Project	
C Project Create C project of selected type	
Project name: priklad01	
Location: C:\ARM_priklady\atolic\prikl	Toolchains:
 Executable Empty Project STM32 C Project Makefile project 	Atollic ARM Tools
Back	Next > Finish Cancel

C Projekt nastavíme dle obr. a klikneme na Next. Objeví se



Target	· · · · · · · · · · · · · · · · · · ·
Evaluation board:	None
Microcontroller family:	All
Microcontroller:	STM32F101RC -
Floating point:	Software implementation 👻
Code location:	FLASH 🗸
Endianess	endian
Optimization	
Remove unused cod	e (dead code removal)
Disable C++ runtime	a (dead data removal)
	cype mornation (k) in

Nastavíme hardwarovou konfiguraci podle obrázku (viz níže)



Target	
Evaluation board:	STM32_Discovery
Microcontroller family:	All
Microcontroller:	STM32F100RB
Floating point:	Software implementation 👻
Code location:	FLASH 🗸
🔿 Big endian 🔘 Little	endian
Big endian () Little	endian
Optimization	
Remove unused cod	e (dead code removal)
Disable Cooperations	a (dead data removal)
	an bandling
Dirable () avecantic	

Klineme na Next



C Project		x
TrueSTUDIO® Misc Settings		
Select miscellaneous project settings		
Debug probe selection		
JTAG Probe: ST-LINK -		
ST-LINK		
Atollic TrueSTUDIO Lite only supports one type of JTAG probe. Please purchase the Professional version to get support for a large number of other types of JTAG probes.		
Runtime library		
 Use tiny printf/sprintf/fprintf (small code size) 		
Ose full printf/sprintf/fprintf (large code size)		
Generate system calls file (enable I/O redirection and O/S integration)		
(2) C Pack Next > Einich	Cance	-
T Sack Ivext > Finish	Cance	=

Nastavíme ST_link a klikneme na Finish



E C Project				
Select Configues Select platforms	urations and configurations yo	u wish to deplo	oy on	
Project type: Toolchains: Configurations:	Executable Atollic ARM Tools			
 ✓ Some Debug ✓ Some Release 				Select all Deselect all
Additional config Use "Manage co	gurations can be added nfigurations" buttons o	d after project d either on toolba Next >	creation. ar or on property pa Finish	ages.

Klikneme na Finish





Rozvinu priklad01 v **Project Explorer**. V okně **Console** také vidíme, že proběhl překlad a sestavení projektu.

C/C++ - Atollic TrueSTUDIO®/STM32 Lite		_ I ×
File Edit Source Refactor Navigate Search Pro	ject Run Manuals Window Help	
·····································	- ≪ - ⊗ - ☆ - ⊘ - ≪ - ☆ - ⊘ / - ◎ □ Q	😭 📴 Unit Test 🔅
🔁 Project Explorer 🛛 📄 🔄 🏹 🖓 🗖	- 8	🗄 Outline 🛛 💿 Make Targe 📃 🗖
TISCOVER		\bigtriangledown
a 👺 priklad01		An outline is not available.
Binaries		
a 🗊 Includes		
D C:/Program Files/Atollic/TrueSTUDIO STN		
Description: De		
Description: De		
D C:/Program Files/Atollic/TrueSTUDIO STN		
priklad01/Libraries/CMSIS/CM3/CoreSupp		
priklad01/Libraries/CMSIS/CM3/DeviceSu		
priklad01/Libraries/STM32F10x_StdPeriph.		
priklad01/src		
曾 priklad01/Utilities/STM32_EVAL/STM32_D		
Utilities		
STM32_EVAL		
STM32_Discovery		
STM32vIdiscovery.c		
b In STM32vidiscovery.h		
Libraries		
STM32F10x_StdPeriph_Driver		
A 😝 SIC		
⊳ ic main.c		
stm2210x_cont.n		
b stm32f10x_it.c	🖹 Problems 🖉 Tasks 📮 Console 🛛 💭 Properties 🛛 🕹 🏠	· 🛐 📓 📓 🛃 🖳 + 🗂 + 🗂 🗖
b g system stm32f10x c	C-Build [priklad01]	
N Ca Debug	Libraries(CMSIS(CM3/DeviceSupport(SI/SIM32flox/startup/irueSlobio/startup_stm32flox_ma_v.	1.0
stm32 flash id	Libraries(CMSIS(CMS)Coresupport(core_cm3.0 -0 pri/iadul.eir -mthumb -mcpu=cortex-m3 -1	\stm32_flash.id -static
	-w, ore, u, west_nammer -w, mappirkadut.map -wi, -ge-sections -wi, -detsymmatrice_	les htollic TrueSTUDIO STM22
	Lite 21 Ollolelarmatollicarenots iar sizeinfo nrikladd alf	ies (#collic (lideStobio Simsz
	Report and converter tools only available in TrueSTUDIO Professional	
	Build complete for project priklad01	
	Time consumed: 17253 ms.	
<		*
Ҏ 👺 priklad01		

Kliknu na main.c



```
- 0
c main.c 🛛
  10/**
                                                                      14
  2 ****
           ***********************
  3 **
                                                                      Ξ.
  4 ** File
                : main.c
  5 **
  6 ** Abstract : main function.
    **
  7
  8
    **
       Functions : main
  9
    **
 10 ** Environment : Atollic TrueSTUDIO/STM32
 11 **
                   STMicroelectronics STM32F10x Standard Peripherals Library
 12 **
 13 ** Distribution: The file is distributed "as is," without any warranty
 14 **
                   of any kind.
 15 **
 16 ** (c)Copyright Atollic AB.
 17 ** You may use this file as-is or modify it according to the needs of your
 18 ** project. Distribution of this file (unmodified or modified) is not
 19 ** permitted. Atollic AB permit registered Atollic TrueSTUDIO(R) users the
 20 **
       rights to distribute the assembled, compiled & linked contents of this
 21 ** file as part of an application binary file, provided that it is built
                                                                       22 ** using the Atollic TrueSTUDIO(R) toolchain.
 23 **
 24 **
 26 */
 27
 28 /* Includes */
 29 #include <stddef.h>
 30 #include "stm32f10x.h"
    4
          III
                                                                    ×.
```

Podíváme se, co nám wizard vytvořil.

Název	Příp	oc Velikost	Datum		Atribu
<u>د.</u>]	0.0	<dir></dir>	15.07.2011	07:11	<u></u> 8
[.settings]		<dir></dir>	15.07.2011	07:11	<u></u> 2
[Debug]		<dir></dir>	15.07.2011	07:11	
[Libraries]		<dir></dir>	15.07.2011	07:11	—
[src]		<dir></dir>	15.07.2011	07:11	—
[Utilities]		<dir></dir>	15.07.2011	07:11	<u></u> 8
stm32_flash	ld	4 789	15.07.2011	07:11	-a
.project		2 148	15.07.2011	07:11	-a-
.cproject		31 212	15.07.2011	07:11	-a-
and a subject water state of the					



Soubor Označit P	říkazy Síť Zobrazit	Konfigurace S	tart
2 👯 📳 🖬	89 89 *	* * 🖨	
[-c-] - [] 10 044	140 kB z 147 353 59	6 kB volných	×
Debug Debug			
c:\ARM_priklady\ato	olic/priklad01/Debug/	*.*	* •
c:\ARM_priklady\ato ↓Název	Přípc Velikost	N*.* Datum	* ▼ Atribu
c:\ARM_priklady\ato ↓Název 1≩[]	plic\priklad01\Debug Přípc Velikost <dir></dir>	Datum 15.07.2011 07:	× ▼ Atribu
c:\ARM_priklady\ato ↓Název 1}[]][Libraries]	blic\priklad01\Debug Přípc Velikost <dir> <dir></dir></dir>	Datum 15.07.2011 07: 15.07.2011 07:	* ▼ Atribu 11 — 11 —
c:\ARM_priklady\ato ↓Název [] [Libraries] [src]	blic\priklad01\Debug Přípc Velikost <dir> <dir> <dir></dir></dir></dir>	Datum 15.07.2011 07: 15.07.2011 07: 15.07.2011 07: 15.07.2011 07:	* Atribu 11 — 11 — 11 — 11 —
c:\ARM_priklady\ato ↓Název [] [Libraries] [src] [Utilities]	blic\priklad01\Debug Přípc Velikost <dir> <dir> <dir> <dir> <dir></dir></dir></dir></dir></dir>	Datum 15.07.2011 07: 15.07.2011 07: 15.07.2011 07: 15.07.2011 07: 15.07.2011 07:	* Atribu 11 — 11 — 11 — 11 — 11 — 11 — 11 —
c:\ARM_priklady\ato ↓Název (] (Libraries) [src] (Utilities) [priklad01	blic\priklad01\Debug Přípc Velikost <dir> <dir> <dir> <dir> <dir> map 196 693</dir></dir></dir></dir></dir>	Datum 15.07.2011 07: 15.07.2011 07: 15.07.2011 07: 15.07.2011 07: 15.07.2011 07: 7 15.07.2011 07:	* ▼ Atribu 11 — 11 — 11 — 11 — 11 —

Priklad01.elf je výsledný soubor který použijeme jako zdroj k naprogramování flash startkitu. Soubory bin a hex vytváří až **TrueSTUDIO Professional**.

Poznámka.

Z <u>http://www.jukie.net/~bart/utils/elf2hex</u> lze stáhnout utilitu pro převod elf na hex. Utilita je sice určená pro Linux, lze však stáhnout i její zdrojový kód elf2hex.c

Našel jsem i na

http://www.stm32.eu/index.php?option=com_content&view=article&id=174:konwersja-plikowelf2hex-i-elf2bin&catid=35:porady&Itemid=57

Konverze elf na hex ARM-GCC (Sourcery G++ Lite): arm-none-eabi-objcopy -O ihex plik_zrodlowy.elf plik_wynikowy.hex ARM-GCC (Yagarto): arm-elf-objcopy -O ihex plik_zrodlowy.elf plik_wynikowy.hex Konverze elf na bin ARM-GCC (Sourcery G++ Lite): arm-none-eabi-objcopy -O binary plik_zrodlowy.elf plik_wynikowy.bin ARM-GCC (Yagarto): arm-elf-objcopy -O binary plik_zrodlowy.elf plik_wynikowy.bin

Klikneme na ikonu zeleného brouka (Debug)



C/C++ - priklad01/src/mai	in.c - Atollic TrueSTUDIO®/	STM32 Lite		
ile Edit Source Refacto	or Navigate Search Pro	oject Run Manuals Windo	w Help	
📬 🕶 🖾 🖌 💼	₫ • 63 • € • 6	• • • •) - Q - & - 🏇 - 😕 🖉	8 -
🍐 Project Explorer 🛛		C main.c 🛛 Debug	priklad01	
DISCOVER		10/**		
priklad01		2 *************	******************	***
includes		4 ** File	: main.c	
ostaneme				
Edit Configuration				x
Edit launch configuration	properties		Ŕ	ñ
Name: priklad01.elf				
📄 Main 🔅 Debugger 🕨	Startup Debug 🕟 Startup Ana	alyze) 🐌 Startup Unit Test) 🧤 Sour	ce) 🔲 Common	
C/C++ Application:				
Debug/priklad01.elf			Search Project Browse	ור
Project:				_
priklad01			Browse	
Build (if required) before laun	nching			
Build configuration: Use Ac	tive			
C Enable auto build	N	🔘 Disable auto build		
Over the setting of the setting o	NT	Configure Workspace Setti	ngs	
-				
			Apply Revert	
(?)			OK Cancel	



Edit Configuration					X
Edit launch configuration prope	erties				Ś
Name: priklad01.elf					
Main 🕸 Debugger 🌔 Startu	p Debug 🐌 Startup Analyz	ze 🕟 Startup Unit 1	Test 🧤 Source 🔳	Common	
JTAG Probe: ST-LINK 🔻					
GDB Connection Settings					
Autostart local GDB server	Host name or IP address:	localhost			
Connect to remote GDB server	Port number:	61234	1		
GDB Server Command Line Options					
© SWD ○ ITAG					
Verify flash download					
C:\ARM_priklady\at	tolic\priklad01\Debug\st-lin	k_gdbserver_log.bt			Browse
ß				Apply	Revert
?				ОК	Cancel

GDB Server musí být nastaven na SWD. Dále musíme na firewalu Windowsu povolit příslušný port - v mém případě 61234. Poté klikneme **OK**. Objeví se

Launching priklad01.elf			
Launching delegate			
🕅 Always run in background			
	Run in Background	Cancel	Details >>

A poté již



Debug - priklad01/src/main.c - Atollic TrueSTUDIO®/STM32 Lite					. 🖬 🗙
File Edit Source Refactor Navigate Search Project Run Manuals Window Help					
📑 • 🖬 🗟 🎒 📓 🔅 • 🔕 • 🍇 • 🌞 • 🧶 • 🦉 • 🚺 📿 🖞 • 🤤 🗢 -	•			😭 🕸 Deb	ug »
🇚 Debug 🛛 🦓 🖉 🦉 🖩 🖷 🖉 🔍 🕾 🛼 🔛 🛠 🍸 🖓 🕬 Va	riables 🛿 💊 Br	eakpoints 888 Regi	isters 🛋 Modu	ules) 🖄 🏘 🖪 🖉 🗶	% ▽ □ □
Resume (F8)					- 0
<pre>113 ** 114 ** Abstract: main program 115 ** 116 ** 116 ** 116 ** 117 */ 1189 int main(void) 119 { 120 int i = 0; 121 121 *ifdef USE_LED 123 /* Initialize LEDs */ 123 /* Initialize LEDs */ 124 STM_EVAL_LEDInit(LED1); 125 STM_EVAL_LEDInit(LED2); 126 STM_EVAL_LEDInit(LED3); 127 STM_EVAL_LEDInit(LED3); 128 128 129 /* Turn on LEDs */ 130 STM_EVAL_LEDOn(LED1); 131 STM_EVAL_LEDOn(LED2); 131 STM_EVAL_LEDON(L</pre>					A H
132 STM_EVAL_LEDOn(LED3); 133 STM_EVAL_LEDOn(LED4); 134 135 Welif defined USE STM32 DISCOVERY					
Canada 22 B Mamaa					
nriklad01 elf (Embedded C/C++ Application) C\/ARM priklad01 atolic/priklad01\Debug/priklad01 elf (7/15/11 8/06 AM)					<u> </u>
pinaooxin pinaooxin (//J)11000 Any STM32 Successufully completed reset operation					*
					+
	Writable	Smart Insert	120:1		

Klikneme na ikonku Resume (F8) . Tím se spustí program v startkitu. Svítí zelená LED. Po stisknutí tlačítka USR zhasne a rozsvítí se modrá LED.

4.4 Druhý vlastní program v TrueStudio

V předchozím, prvním, "vlastním" programu jsme vlastně jen použili wizart **TrueStudia**, který vytvořil i zdrojový kód **main.c**, takže jsme žádný kód sami nepsali. Proto nyní znovu a lépe. Zavřeme priklad01.



a C/C++ - Atollic T	rueSTUDIO®/STM32 Lite	All Contractions
File Edit Source	Refactor Navigate Search	Project Run Manu
i 🖻 🗕 🗐 🍋 🖉	1 P :	Q . 1
	New	> <u>-</u>
	Go Into	
▲ E priklac	Open in New Window	
Þ 🐝 Bir	Сору	Ctrl+C
Uti	Paste	Ctrl+V
- C- X	Delete	Delete
	Move	
	Rename	F2
🔺 🤔 Lit 🚵	Import	
	Export	
a 😕 src	Build Project	
⊳ .€	Clean Project	
▶ <u>.h</u> @	Refresh	F5
	Close Project	
Þ C	Close Unrelated Projects	
⊳ 🔁 De	Build Configurations	•
str	Make Targets	•
	Index	•
	Convert To	
	Unit test As	→
	Analyze As	•
	Run As	•
	Debug As	→ [
	Profile As	•
	Team	🕨 ms 🖉
	Compare With	► ed> pril
	Restore from Local History	ucces
۰ Q	Inspection	
0♦ 1	Properties	Alt+Enter

Nyní založíme druhý projekt. Z menu vybereme File, pak New a nakonec C Project.



File	Edit Source Refactor Navigate	Search Projec	t R	un Manuals Window Help		-
	New	Alt+Shift+N ►	C++	Makefile Project with Existing Code		10
	Open File			C++ Project		F
	Close	Ctrl+W	C #	C Project		H
	Close All	Ctrl+Shift+W	1	Unit test		L
	Save	Ctrl+S	El	Project		L
	Save As		C++	Convert to a C/C++ Project		
1	Save All	Ctrl+Shift+S	63	Source Folder		L
	Revert			Folder		L
	Maur		C	Source File		L
	Parama	E2	h	Header File		L
54	Refrech	F2 E5		File from Template		L
0	Convert Line Delimiters To		G	Class		L
	Convert Line Demniters To	,	2	Other	Ctrl+N	
	Print	Ctrl+P	Γ			1
	Switch Workspace	+				
	Restart					
2	Import					
4	Export					
	Properties	Alt+Enter				
	1 main.c [DISCOVER/User]					
	2 core_cm3.c [ARM_priklady/]					
	Exit					

Objeví se okno C project



E C Project	
C Project Create C project of selected type	
Project name: priklad02	
Location: C:\ARM_priklady\atolic\priklad0	2 Browse
Project type:	Toolchains:
 Executable Empty Project STM32 C Project Makefile project Makefile project 	Atollic ARM Tools
(?) < Back	lext > Cancel

Vyplníme a klikneme na Next



Target	
Evaluation board:	STM32_Discovery
Microcontroller family:	All 👻
Microcontroller:	STM32F100RB 👻
Floating point:	Software implementation
Code location:	FLASH 👻
Big endian Elittle	endian
Big endian O Little	endian
Optimization	
Remove unused code	e (dead code removal)
	i (dead data removal)
Remove unused data	the information (DTTD)

Nastavíme dle obr. A opět Next



a C Project	
TrueSTUDIO® Misc Settings	
Select miscellaneous project settings	
Debug probe selection	
JTAG Probe: ST-LINK	
ST-LINK	
Atollic TrueSTUDIO Lite only supports one type of JTAG probe. Please purchase the Professional version to get support for a large number of other types of JTAG probes.	
Runtime library	
O Use tiny printf/sprintf/fprintf (small code size)	
 Use full printf/sprintf/fprintf (large code size) 	
Generate system calls file (enable I/O redirection and O/S integration)	
(?) < Back Next > Finish	Cancel

ST-Link a Next





Obsah main.c vymažeme a nahradíme novým kódem.



V.Váňa: CanSAT Book for Students – part 2 428



}

{

```
// tj. když není stisknuto tlacitko
STM32vldiscovery_LEDOff(LED3); // vypnem LED3 - zelena
}
void GPIO_Inicializace(void)
STM32vldiscovery_LEDInit(LED3);
STM32vldiscovery_LEDInit(LED4);
STM32vldiscovery_PBInit(BUTTON_USER, BUTTON_MODE_GPIO);
STM32vldiscovery_LEDOff(LED3);
STM32vldiscovery_LEDOff(LED4);
}
void Delay(__IO uint32_t nTick)
{
for(; nTick != 0; nTick--);
}
```





DIO®/S	STM32	Lite:
n Proj	ject R	un Manuals Window Help
	- 4	• • • • • • • • • • • • • • • • • • •
- 0	C m	ain.c 🖾 🖟 Debug priklad02 y.h
	1	/* priklad02
	2	<pre>#include <stddef.h></stddef.h></pre>
	3	#include "stm32f10x.h"
	4	<pre>#include "STM32vldiscovery.h"</pre>

Spustíme **Debug** (ikonka zeleného brouka), máme

Edit Configuration			X
Edit launch configuration properties			Ť
Name: priklad02.elf			
Main The Debugger Startup Debug Startup Analyze	e) 🔈 Startup Unit Test 🛛 🦗 Source 🕅	Common	
Debug/priklad02.elf		Search Project	Browse
Project:		·	
priklad02			Browse
Build (if required) before launching			
Build configuration: Use Active			•
Enable auto build	🔘 Disable auto build		
Over the settings	Configure Workspace Settings		
		Apply	Revert
•		<u>Сук</u>	Cancel

Klikneme na **OK**





Klikneme na Resume (F8)

File Edit	Source	Refactor	Navigat	e Searc	h Proje	ect R	un I	Manuals	Window	Help
C • [010	恭•0	- 0	- 🔅 -	☆.	- 6	9 19 -		Q
🎋 Debug	1 23				× 14	20	• 00	1	3.3	_()_ =
C main.	1 23	STM32v	ldiscovery	.h				Termin	ate (Ctrl+F	2)

Program nyní běží – bliká modrá dioda. Pokud stisknu a držím tlačítko USR bude blikat i zelená dioda. Protože **Atollic TrueSTUDIO** je využíván v seriálu na **MCU serveru**, můžeme další programy převzít z tohoto seriálu a v těchto skriptech se jim již nebudeme příliš zabývat. Jen si ještě ukážeme, jak importovat do TrueSTUDIA projekty v souborech zip. Předtím ovšem nezapomeneme zavřít projekt priklad02.

4.5 Import programu v TrueStudio

Použijeme LCD.zip ze 14 dílu seriálu z MCU serveru.



File	Edit Source R	efactor	Navigate	Search	Project	Run I
	New			Alt+Sh	ift+N ▶	
	Open File					
	Close			Ct	rl+W	-
	Close All			Ctrl+S <mark>h</mark> i	ft+W	
	Save			C	trl+S	
	Save As					
6	Save All			Ctrl+Sh	nift+S	
	Revert					
	Move					
	Rename				F2	
2	Refresh				F5	
	Convert Line Delin	niters T	D		•	
	Print			C	trl+P	
	Switch Workspace				+	
	Restart					
2	Import	\ \				
4	Export	0				
	Properties			Alt+	Enter	
	1 main.c [DISCOV	ER/Use	r]			
	2 core_cm3.c [ARI	M_prikla	ady/]			Problem
	Exit					erminated
-	NOTION .	_		_	1100	M32 Su


Import	
Select Create new projects from an archive file or directory.	Ľ
Select an import source:	
General Archive File Existing Projects into Workspace File System Preferences C/C++ Run/Debug Team	
(?) < Back Next > Finish	Cancel



Ispořádat 🔻 Nová	složka		
Nocha 🚽	 Název položky 	Datun	
\rm Stažené soubory	LCD	21.4.20	
B R H	lcd	11.3.20	
Dokumenty			
👌 Hudba			
Solorázky			Náhled není k dispozici
H Videa			
🍣 Domácí skupina			
Počítač			
🏭 Místní disk (C:)			
👝 Místní disk (E:)		<u>ب</u>	
			[+· + · +· +· +·



a Import	18× 8 4 + 1 + 10	
Import Projects Select a directory to sear	ch for existing Eclipse projects.	
 Select root directory: Select archive file: Projects: 	C:\zaloha\arm\STM32\STM32MCU\14\lcd.zip	Browse Browse
LCD (LCD)		Select All Deselect All Refresh
Copy projects into we Working sets	orkspace ing sets	Select
?	< Back Next > Finis	Cancel





Projekt se přeložil. Poté jsem v Project Explorer rozvinul LCD a otevřel main.c.

Všimněme si, že v STM32_EVAL najdeme STM32_Discovery, kdežto pomocí wizardu jsme dostali STM32vldiscovery. To musíme mít na zřeteli, pokud budeme např. používat wizard a dále některé soubory z MCU seriálu a upravit příslušná místa v kódu

Pozn.

STM32VL Discovery má i/o signály na 3V logice, což může způsobit problémy při spolupráci s TTL obvody. U LCD řadiče HD44780 dále záleží na nastavení správných velikostí prodlev. To můžou být důvody, proč se mi nepodařilo zprovoznit LCD displej s **LCD.c** a **LCD.h** z MCU serveru. Proto, podobně jako u uVision4 Keil a IAR WorkBench používám tyto soubory ve verzi z Katedry měření ČVUT.

4.6 ukázka programu v TrueStudio s I2C

Jde o program z 29.dílu seriálu na MCU s tím, že program jsme doplnili o dvě funkce provádějící bliknutí zelené či modré LED a tyto funkce jsme umístili na několik míst v kódu main.c, čímž si při běhu programu ověříme, že provedl postupně všechny funkce a nikde se "nezasekl". Úkolem programu je prostřednictvím I2C naprogramovat EEPROM AT24C02. Při komunikaci I2C provádí komunikující obvod (slave) potvrzování přijatých dat signálem Ack, STM32VL Dicsovery je v komunikaci I2C master. V jeho programu v main.c můžeme vidět několik smyček, čekajících právě na potvrzení Ack. **Není** zde např. naprogramováno, aby po určité době, do které nedojde potvrzení Ack program např. na LCD vypsal, že došlo k chybě.



Program je napsán tak, že v čekací smyčce je tak dlouho, dokud nedojde potvrzení Ack. Pokud nedojde, např. proto, že k STN32VL Discovery nemáme připojenu EEPROM, nebo sice máme, ale bez Pull Up odporů 4k7, tak program bude v nekonečné smyčce, Dojde k jeho uvíznutí (tzv. "zaseknutí").

C/C++ - I2C_EEPROM/src/main.c - Atollic TrueSTUDIC	0@/STM32 Lite	100	
File Edit Source Refactor Navigate Search Pro	sject Run Manuals Window Help		
📑 • 🖬 🖷 🚔 🖬 🛛 🎯 • 🚳 • 🕃 • 🎯	• 🔦 • 🕸 • 🗳 • 🖉 • 🍇 • 🌞 • 🍅 • 😕 🛷 • 🗾 🗐 🔹 🔍 🐓 • 🖗 • 👳	- ¢ •	🛱 🛅 C/C++ 🛛 👋
Project Explorer 🛛 📄 😫 🍸 🗖 🗖	🖻 main.c 🛛 💽 stm32f10x_i2c.h	- 0	🗄 Outline 🛛 💿 Make Targe 📃 🗆
Comperence of the second seco	<pre>19/MAR to g sinth Action 19/* 2 * 12C komunikacg s EEFROM (AT24C02) 3 * dig 29. dil serialu o STM32 4 main.o 5 * Vd pro SPSE Jacna Lectures 6 * 7 */ 8 9 /* Includes 10 #include "stm32flox.h" 11 #include "STM32_Discovery.h" 12 13 14 15 /* Private typedef 16 GPIO_InitTypeDef GPIO_InitStructure; 17 18 /* Private define 19 #define EEFROM SLAVE_ADDRESS OxA0 //protoze je to EEFROM 20 //ScL bude na PB6 DATA na PB7 AT24C02 21 23 24 24 24 24 24 24 24 24 24 24 24 24 24</pre>	••••••••••••••••••••••••••••••••••••••	B Outline So Image Targe Image Targe Image Targe <td< th=""></td<>
priklad01			< III >
Pinda02 Pinda03 pinda03 pinda04 PS2klavesnice STM32_Discovery_Demo	Problems Tasks Console Properties C-Build [22_EEPROM] **** C:\Program Files\Atollic\TrueSTUDIO SIM32 Lite 2.1.0\ide\jre\bin\java -jar C:\Pr Lite 2.1.0\Tools\arm-atollic-reports.jar sizeinfo I2C_EEPROM.elf Report and converter tools only available in TrueSTUDIO Profeesional arm-atollic-rebit-objoopy -0 inex I2C_EEPROM.elf I2C_EEPROM.ex Flash Converter only available in TrueSTUDIO Profeesional Build complete for project I2C_EEPROM Time consumed: 442 ms.	ပို 😯	es\Atollic\TrueSTUDIO STM32
	Writable Smart Insert 22:1		





/*
* I2C komunikace s EEPROM (AT24C02)
* dle 29. dil serialu o STM32
* main.c
* Vd pro SPSE Jecna Lectures
*
*/
/* Includes ------*/
#include "stm32f10x.h"
#include "STM32_Discovery.h"
/* Private typedef ------*/

V.Váňa: CanSAT Book for Students – part 2

ARM STM32 programming



```
GPIO InitTypeDef GPIO InitStructure;
```

```
/* Private define -----
#define EEPROM SLAVE ADDRESS 0xA0 //protoze je to EEPROM
//SCL bude na PB6 DATA na PB7 AT24C02
//Tabulka I2C slave adres
//Typ obvodu
//Typ obvodu
//Typ obvodu
A7 A6 A5 A4 A3 A2 A1
//8bitovy vstup/vystup
0 1 0 0 a a a
    napr. PCF8574
//EEPROM
                           1
                               0
                                     1
                                          0
                                                a
                                                     а
                                                           а
    napr. PCF8582
                          1
//Hodiny/kalendar
                               0
                                     1
                                          0
                                                0
                                                     0
                                                           а
    napr. PCF8583
//8bitovy A/D a D/A
                          1
                               0
                                     0
                                          1
                                               a
                                                     а
                                                           а
    napr. PCF8591
//PLL
                          1 1 0
                                          0
                                               0
                                                     а
                                                           a
    napr. TSA5055
/* Private variables ------
uint8 t hodnota=0;
/* Private function prototypes ------
void GPIO Inicializace(void);
                                     // nastavení vstupne/vystupnich
pinu na kitu
void I2C Inicializace(void); // nastavení I2C
uint8_t Read_EEPROM(uint8_t adresa); // cteni
void Write EEPROM(uint8 t adresa, uint8 t data); // zapis
                                     // polling potvrzeni
void Ready EEPROM(void);
void Delay(__IO uint32_t nTick); // cekani
void bliknutiLEDMODRA(void); //
void bliknutiLEDZELENA(void);
                                      11
int main(void)
{
 GPIO Inicializace();
 I2C Inicializace();
  // blikneme modrou jako priznak startu
 bliknutiLEDMODRA();
 /* pouzita EEPROM AT24C02 - 256 bajtu x 8 bitu */
 hodnota = Read EEPROM(0x00);
 hodnota = Read EEPROM(0x01);
 hodnota = Read EEPROM(0x02);
 hodnota = Read EEPROM(0x03);
 hodnota = Read EEPROM(0x04);
```



```
Write EEPROM(0x00, 0x01);
 Ready EEPROM();
 bliknutiLEDZELENA();
 Write EEPROM(0x01, 0x12);
 Ready EEPROM();
 bliknutiLEDMODRA();
 Write EEPROM(0x02, 0x34);
 Ready EEPROM();
 bliknutiLEDZELENA();
 Write EEPROM(0x03, 0x56);
 Ready EEPROM();
 bliknutiLEDMODRA();
 hodnota = Read EEPROM(0x00);
 hodnota = Read EEPROM(0x01);
 hodnota = Read EEPROM(0x02);
 hodnota = Read EEPROM(0x03);
 hodnota = Read EEPROM(0x04);
  // blikneme zelenou jako priznak konce
 bliknutiLEDZELENA();
 while (1)
  {
       bliknutiLEDMODRA();
        // tady uz jenom blikame modrou LED
  }
}
void GPIO Inicializace(void)
{
      STM32 Discovery LEDInit(LED3);
      STM32_Discovery_LEDInit(LED4);
     STM32 Discovery PBInit (BUTTON USER, BUTTON MODE GPIO);
     STM32 Discovery LEDOff(LED3);
     STM32 Discovery LEDOff(LED4);
}
void bliknutiLEDMODRA(void)
{
     STM32 Discovery LEDOn(LED4); // zapnem LED4 - modra
     Delay(0x5FFFF);
      STM32_Discovery_LEDToggle(LED4); // vypnem LED4 - modra
     Delay(0x5FFFF);
}
void bliknutiLEDZELENA(void)
{
       STM32 Discovery LEDOn(LED3); // zapnem LED3 - zelena
       Delay(0x5FFFF);
       STM32 Discovery LEDToggle(LED3); // vypnem LED3 - zelena
      Delay(0x5FFFF);
```

V.Váňa: CanSAT Book for Students – part 2 440

ARM STM32 programming



}

```
void I2C Inicializace(void)
  GPIO InitTypeDef GPIO InitStructure;
  I2C InitTypeDef I2C InitStructure;
  I2C DeInit(I2C1);
  /* I2C Periph clock enable */
  RCC APB1PeriphClockCmd(RCC APB1Periph I2C1, ENABLE);
  /* GPIO Periph clock enable */
  RCC APB2PeriphClockCmd (RCC APB2Periph GPIOB, ENABLE);
  /* Configure I2C pins: SCL and SDA */
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6 | GPIO_Pin_7;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_10MHz;
  GPIO InitStructure.GPIO Mode = GPIO Mode AF OD; //GPIO Mode AF PP
  GPIO Init(GPIOB, &GPIO InitStructure); //I2C EEPROM pripojime k PB
  //SCL bude na PB6 DATA na PB7 AT24C02
  /* I2C configuration */
  I2C InitStructure.I2C Mode = I2C Mode I2C;
  I2C InitStructure.I2C_DutyCycle = I2C_DutyCycle_2;
  I2C_InitStructure.I2C_OwnAddress1 = 0xA0; //EEPROM
  I2C_InitStructure.I2C_Ack = I2C_Ack_Enable;
  I2C_InitStructure.I2C_AcknowledgedAddress = I2C_AcknowledgedAddress_7bit;
  I2C InitStructure.I2C ClockSpeed = 10000;
  /* I2C Peripheral Enable */
  I2C Cmd(I2C1, ENABLE);
  /* Apply I2C configuration after enabling it */
  I2C Init(I2C1, &I2C InitStructure);
}
uint8 t Read EEPROM(uint8 t adresa)
{
       uint8 t Data;
       Data = 0;
       /* While the bus is busy */
       while(I2C GetFlagStatus(I2C1, I2C FLAG BUSY));
       /* Send START condition */
       I2C GenerateSTART(I2C1, ENABLE);
        /* Test on EV5 and clear it */
        while(!I2C CheckEvent(I2C1, I2C EVENT MASTER MODE SELECT));
        /* Send EEPROM address for read */
       I2C Send7bitAddress(I2C1, EEPROM SLAVE ADDRESS,
I2C Direction Transmitter);
        /* Test on EV6 and clear it */
```



```
while(!I2C CheckEvent(I2C1,
I2C EVENT MASTER TRANSMITTER MODE SELECTED));
        /* Clear EV6 by setting again the PE bit */
        I2C Cmd(I2C1, ENABLE);
 /* Send the EEPROM's internal address to read from: MSB of the address
first */
        I2C SendData(I2C1, adresa);
        /* Test on EV8 and clear it */
        while(!I2C CheckEvent(I2C1, I2C EVENT MASTER BYTE TRANSMITTED));
        /* Send START condition a second time */
        I2C GenerateSTART(I2C1, ENABLE);
        /* Test on EV5 and clear it */
        while(!I2C CheckEvent(I2C1, I2C EVENT_MASTER MODE_SELECT));
        /* Send EEPROM address for read */
        I2C Send7bitAddress(I2C1, EEPROM SLAVE ADDRESS,
I2C Direction Receiver);
        /* Test on EV6 and clear it */
        while(!I2C CheckEvent(I2C1,
I2C EVENT MASTER RECEIVER MODE SELECTED));
        /* Disable Acknowledgement */
        I2C AcknowledgeConfig(I2C1, DISABLE);
        /* Send STOP Condition */
        I2C GenerateSTOP(I2C1, ENABLE);
        while(!I2C CheckEvent(I2C1, I2C EVENT MASTER BYTE RECEIVED)){}
        /* Read a byte from the EEPROM */
        Data = I2C ReceiveData(I2C1);
        /* Enable Acknowledgement to be ready for another reception */
        I2C AcknowledgeConfig(I2C1, ENABLE);
       return(Data);
}
void Write EEPROM(uint8 t adresa, uint8 t data)
{
        /* Send START condition */
        I2C GenerateSTART(I2C1, ENABLE);
        /* Test on EV5 and clear it */
        while(!I2C CheckEvent(I2C1, I2C EVENT MASTER MODE SELECT));
        /* Send EEPROM address for write */
        I2C Send7bitAddress(I2C1, EEPROM SLAVE ADDRESS,
I2C Direction Transmitter);
```

V.Váňa: CanSAT Book for Students – part 2 442



```
/* Test on EV6 and clear it */
       while(!I2C CheckEvent(I2C1,
I2C EVENT MASTER TRANSMITTER MODE SELECTED));
        /* Send the EEPROM's internal address to write to : MSB of the
address first */
       I2C SendData(I2C1, adresa);
        /* Test on EV8 and clear it */
        while(!I2C CheckEvent(I2C1, I2C EVENT MASTER BYTE TRANSMITTED));
        /* Send the byte to be written */
        I2C SendData(I2C1, data);
        /* Test on EV8 and clear it */
       while(!I2C CheckEvent(I2C1, I2C EVENT MASTER BYTE TRANSMITTED));
        /* Send STOP condition */
        I2C GenerateSTOP(I2C1, ENABLE);
}
void Ready EEPROM(void)
{
        IO uint16 t temp = 0;
        do
        {
         /* Send START condition */
         I2C GenerateSTART(I2C1, ENABLE);
          /* Read I2C EE SR1 register to clear pending flags */
         temp = I2C ReadRegister(I2C1, I2C Register SR1);
          /* Send EEPROM address */
         I2C Send7bitAddress(I2C1, EEPROM SLAVE ADDRESS,
I2C Direction Transmitter);
        while(!(I2C ReadRegister(I2C1, I2C Register SR1) & 0x0002));
        /* Clear AF flag */
       I2C ClearFlag(I2C1, I2C FLAG AF);
        /* STOP condition */
        I2C GenerateSTOP(I2C1, ENABLE);
}
void Delay( IO uint32 t nTick)
  for(; nTick != 0; nTick--)
  { ; }
```

443

V.Váňa: CanSAT Book for Students – part 2



Obsah naprogramované EEPROM můžeme např. zkontrolovat pomocí free programu PonyProg2000

PonyProg	32000 -	Serial De	vice Programm	mer 📃	
File Edit De	vice Cor	mmand Sci	ript Utility Setu	up 7 Window	
2 6			(L)	🔃 🗐 🔍 🦹 🐐 🛛 12C Bus 8bit eeprom 💌 2402 💽	1
6	B		2 8 :::		
🛐 No Name	1				^
999999) 999919) 999929) 999929) 999929 999929 999929 999929 999929 999929 999929 999929 999929 999929 999929 999929 999929 999929 999929 999929	01 12 FF FF FF FF	34 56 FF FF FF FF	FF FF	$\begin{array}{rrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr$	
<					>
PonyProg200)0	2402	Size 256 By	lytes CRC 4D87h	

Vidíme, že na první čtyři adresy se skutečně zapsalo 01, 12, 34, 56. Další programy pro TrueStudio najdeme v již zmiňovaném seriálu na MCU serveru.

5 Pokročilé programování, další projekty

5.1 RTOS (podle AT&P journal)

Operační systémy pro řízení v reálném čase RTOS (Real Time Operating System) lze rozdělit do několika skupin podle aplikační oblasti, uplatnění ve vestavěných systémech a využití standardních operačních systémů osobních počítačů.

Jako **první skupinu** lze uvést RT operační systémy s obecnějším uplatněním. Zde především patří tradiční oblast řízení technologických procesů, kde se jedná často o kritické úlohy reálného času, ale také o informační systémy pracující v reálném čase, zde se jedná i o nekritické systémy reálného času.

Druhou skupinu operačních systémů pro řízení jsou RTOS pro vestavěné (Embedded Systems) nebo jednodeskové systémy. U těchto aplikací lze konstatovat, že se jedná o nejrychlejší nárůst použití RTOS ze všech aplikačních oblastí. Aplikační oblasti použití vestavěných řídicích systémů s mikropočítači se neustále rozšiřují a v současné době je lze nalézt ve všech oblastech lidské činnosti . Nejrozšířenějšími aplikacemi jsou systémy v automobilní a dopravní technice. Mezi tyto RTOS patří i FreeRTOS, který budeme dále používat.



Třetí skupinu RT operačních systémů tvoří ty, které jsou orientovány pouze na skupinu procesorů Intel jako u PC, ale jsou použity pro vestavěné PC systémy.

Čtvrtou skupinu RTOS tvoří doplňující moduly jádra, které umožňují operačním systémům používaných u osobních počítačů PC, které tvoří nejrozšířenější skupinu výpočetní techniky, řešení úloh reálného času, často to však nejsou kritické úlohy reálného času.

5.2 FreeRTOS

Při přechodu z 8 nebo 16-bitových mikrokontrolérů hledají specialisté rychlejší, jednodušší a chybám méně náchylnou cestu vývoje, takže použití RTOS je důležitým krokem k využití plného výkonu mikrokontroléru při dodržení maximálního zjednodušení kódování.

RTOS je určen pro embedded aplikace v reálném čase, kde je efektivita provádění a kompaktnost kódu důležitými požadavky. RTOS se obvykle vyznačuje vysokou přenositelností, kompaktní velikostí kódu a hlavně, jeho architektura bývá optimalizovaná pro extrémně efektivní přepínání kontextu. Obvyklé funkce RTOS:

- Efektivní a přenositelné preemptivní jádro.
- Statická architektura, vše je staticky přidělené již v době kompilace.
- Dynamické rozšíření, dynamické objekty jsou podporovány volitelnou vrstvou postavenou nad statickým jádrem.
- Bohatá sada primitiv: vlákna, virtuální časovače, semafory, mutexy, stavové proměnné, zprávy, mailboxy a flagy událostí.
- Podpora algoritmu prioritního dědictví pro mutexy.
- HAL komponenty podporující různé abstraktní ovladače zařízení: Port, Serial, ADC, CAN, I2C, MAC, MMC, PWM, SPI, UART.
- Podpora pro externí komponenty UIP, IwIP, FatFs.
- Mnoho podporovaných architektur.
- Rozsáhlé testovací sady s benchmarky

HAL

Hardware Abstraction Layer je důležitou součástí přenositelnosti kódu a jeho rychlou tvorbou. HAL ChibioS/RT obsahuje:

- Configuration
- ADC Driver
- CAN Driver
- MMC over SPI Driver
- PAL Driver
- PWM Driver
- Serial Driver
- SPI Driver
- UART Driver

Drivery

Kromě použití HAL (což je doporučeno) obsahuje RTOS často i řadu driverů pro STM32 mcu. Jedná se o následující sadu:

• STM32 Initialization Support

V.Váňa: CanSAT Book for Students – part 2 445



- STM32 ADC Support
- STM32 CAN Support
- STM32 DMA Support
- STM32 GPIO Support
- STM32 PWM Support
- STM32 SPI Support
- STM32 USART Support (buffered)
- STM32 USART Support (unbuffered)

Pro ARM existuje řada RTOS pro ARM, např. ChibiOS/RT, ArmExe RTOS, eCOS či FreeRTOS. Zvolil jsem FreeRTOS.

Pozn.

Použití RTOSu nám na jedné straně usnadní práci, na druhé straně je náročnější na prostředky MCU. Tyto prostředky sice máme, ale jsou využívány obvykle rozsáhlejším výsledným programem, než umožňují free education verze nástrohů uVision4 Keil a IAR Embedded Workbench. Toto omezení nemáme např. u Eclipse, včetně verzí na Eclipce postavených, jako je Atollic TrueSTUDIO. Proto pro FreeRTOS použijeme právě toto prostředí.

5.3 instalace FreeRTOS a demo program

Z <u>http://www.freertos.org/index.html</u> stáhneme FreeRTOSV7.0.1.exe , pak extrahuji adresář FreeRTOSV7.0.1 včetně obsahu.

Zkopíruji do workspace atollic TrueStudio

Název	1Příp Velikost	Datum
金[]	<dir></dir>	15.07.2011
🗀 [.metadata]	<dir></dir>	14.07.2011
FreeRTOS]	<dir></dir>	15.07.2011
🗀 [LCD]	<dir></dir>	15.07.2011
🗀 [priklad01]	<dir></dir>	15.07.2011
🗀 [priklad02]	<dir></dir>	15.07.2011
🗀 [priklad03]	<dir></dir>	15.07.2011
🗀 [priklad04]	<dir></dir>	15.07.2011
[PS2klavesnice]	<dir></dir>	15.07.2011



Název	1 Příp Velikost	Datum
^ []	<dir></dir>	15.07.2011
Demo]	<dir></dir>	15.07.2011
[License]	<dir></dir>	15.07.2011
[Source]	<dir></dir>	15.07.2011
[TraceCon]	<dir></dir>	15.07.2011
readme	txt 873	18.08.2009

	U	*
Název	↑Příp Velikost	Datum
<u>د.</u>]	<dir></dir>	15.07.20
[ARM7_AT91FR40008_GCC]	<dir></dir>	15.07.20
[ARM7_AT91SAM7S64_IAR]	<dir></dir>	15.07.20
[ARM7_AT91SAM7X256_Eclipse]	<dir></dir>	15.07.20
[ARM7_LPC2106_GCC]	<dir></dir>	15.07.20
[ARM7_LPC2129_IAR]	<dir></dir>	15.07.20
[ARM7_LPC2129_Keil_RVDS]	<dir></dir>	15.07.20
[ARM7_LPC2138_Rowley]	<dir></dir>	15.07.20
[ARM7_LPC2368_Eclipse]	<dir></dir>	15.07.20
[ARM7_LPC2368_Rowley]	<dir></dir>	15.07.20
[ARM7_STR71x_IAR]	<dir></dir>	15.07.20
[ARM7_STR75x_GCC]	<dir></dir>	15.07.20
[ARM7_STR75x_IAR]	<dir></dir>	15.07.20
[ARM9_AT91SAM9XE_IAR]	<dir></dir>	15.07.20
[ARM9_STR91X_IAR]	<dir></dir>	15.07.20
[AVR_ATMega323_IAR]	<dir></dir>	15.07.20
[AVR_ATMega323_WinAVR]	<dir></dir>	15.07.20
[AVR32_UC3]	<dir></dir>	15.07.20
ColdFire_MCF51CN128_CodeWarrior] <dir></dir>	15.07.20
ColdFire_MCF52221_CodeWarrior]	<dir></dir>	15.07.20
ColdFire_MCF52233_Eclipse]	<dir></dir>	15.07.20
[ColdFire_MCF52259_CodeWarrior]	<dir></dir>	15.07.20
ColdFire_MCF5282_Eclipse]	<dir></dir>	15.07.20
Common]	<dir></dir>	15.07.20

Vymažeme zbytečné podadresáře



lázev	1Příp	Velikost	Datu	m
È[]		<dir></dir>	15.0	7.201
[Common]		<dir></dir>	15.0	7.201
[CORTEX_STM32F100_Atollic]		<dir></dir>	15.0	7.201
readme	txt	803	18.0	8.200

Obsah adresáře CORTEX_STM32F100_Atollic je

↓Název	Přípo	Velikost	Datum	
▲ []		<dir></dir>	13.05.2011	
[.settings]		<dir></dir>	13.05.2011	
🗀 [Libraries]		<dir></dir>	13.05.2011	
[Simple_Demo_Source]		<dir></dir>	13.05.2011	
🗀 [Utilities]		<dir></dir>	13.05.2011	
stm32_flash	Id	4 486	04.04.2011 2	
CreateProjectDirectoryStructure	bat	1 352	02.04.2011	
project		2 170	02.04.2011	
Cproject		25 280	02.04.2011	

Předtím, než budeme pracovat s **TrueStudiem** spustíme **CreateProjectDirectoryStructure.bat** Jeho název je plně vypovídající.

Název	1Příp	Velikost	Datum
金[]		<dir></dir>	15.07.2011
🗀 [.settings]		<dir></dir>	15.07.2011
FreeRTOS_Source]	05050505050	<dir></dir>	15.07.2011
🗀 [Libraries]		<dir></dir>	15.07.2011
Simple_Demo_Source]		<dir></dir>	15.07.2011
🗀 (Utilities)		<dir></dir>	15.07.2011
cproject		25 280	02.04.2011
project		2 170	02.04.2011
CreateProjectDirectoryStructure	bat	1 352	02.04.2011
stm32 flash	Id	4 486	04.04.2011

Vidíme, že se vytvořil FreeRTOS_Source

Nyní již spustíme Atollic TreuSTUDIO

Nezapomeneme zavřít všechny projekty





File --- Import ---- Existing Project Into Workspace



File	Edit Source Refactor Navigat	e Search Projec
	New	Alt+Shift+N ►
	Open File	
	Close	Ctrl+W
	Close All	Ctrl+Shift+W
	Save	Ctrl+S
3	Save As	
ġ.	Save All	Ctrl+Shift+S
	Revert	
	Move	
	Rename	F2
	Refresh	F5
	Convert Line Delimiters To	•
5	Print	Ctrl+P
	Switch Workspace	•
	Restart	
2	Import	
3	Export	
	Properties	Alt+Enter
	1 LCD.h [priklad04/src]	
	2 PS2KB_emcu.c [priklad04/src]	
	3 PS2Kb_emcu.h [priklad04/src]	
	4 main.c [priklad04/src]	
	Evit	



Import	
Select Create new projects from an archive file or directory.	Ľ
Select an import source:	
type filter text	
 General Archive File Existing Projects into Workspace File System Preferences C/C++ Run/Debug Team 	
Back Nex Fin	nish Cancel



Import Projects Select a directory to search for existing Eclipse projects.	
Select root directory:	Browse
Projects:	Browse
	Select All
	Deselect All
	Refresh
Copy projects into workspace	
Copy projects into workspace Working sets	N
Copy projects into workspace Working sets Add project to working sets Working sets:	▼] Select
Copy projects into workspace Working sets Working sets:	▼ Select
Copy projects into workspace Working sets Add project to working sets Working sets:	Select

Do Select root directory dáme FreeRTOS/Demo/CORTEX_STM32F100_Atollic



Import		
Import Projects Select a directory to sear	ch for existing Eclipse projects.	
 Select root directory: Select archive file: Projects: 	C:\ARM_priklady\atolic\FreeRTOS\Demo\CORTEX_STM32F100_Atollic	Browse
▼ FreeRTOS-Simple	e-Demo (C:\ARM_priklady\atolic\FreeRTOS\Demo\CORTEX_STM32F100_Atollic)	Select All Deselect All Refresh
Copy projects into we Working sets	orkspace ting sets	Select
?	< Back Next > Finish	Cancel





V souboru FreeRTOSConfig.h vymažeme řádku

#error Ensure CreateProjectDirectoryStructure.bat has been executed before building. See comment immediately above.

a projekt přeložíme





Naprogramujeme do startkitu

C/C++ - FreeRTOS-Simple-Demo/	Simple_Demo_Source/main.c - Atollic TrueSTUDIO®/STM32 Lite
File Edit Source Refactor Navi	gate Search Project Run Manuals Window Help
	🚳 • 🖸 • 🞯 • 🕺 • 🗞 • 📎 • 🎼 • 🖉 • 🍇 • 🕸 •
Project Explorer 🛛 🗖 🗖	Debug
	149
TISCOVER	150 /* The queue used by both tasks. */
😂 FreeRTOS-Simple-Demo	151 static xQueueHandle xQueue = NULL;
Rinaries	152
🗊 Includes	1530 /* The LED software timer. This uses v
冯 Libraries	154 * function.
😕 Utilities	155 */
FreeRTOS Source	156 static xTimerHandle xLEDTimer = NULL;
Simple_Demo_Source	157



Edit Configuration			X
Edit launch configuration properties			Ť.
Name: FreeRTOS-Simple-Demo.elf		~ >	
C/C++ Application:	e 📂 Startup Unit Test 🐶 Source 🔛	Common	
Debug/FreeRTOS-Simple-Demo.elf		Search Project	Browse
Project:			
FreeRTOS-Simple-Demo			Browse
Build (if required) before launching			
Build configuration: Use Active			•
© Enable auto build	⑦ Disable auto build		
Use workspace settings	Configure Workspace Settings		
		Apply	Revert
?		ок	Cancel





Program běží . bliká zelená LED. Pokud stiskneme USR tlačítko, rozsvítí se na cca 10s i modrá LED

5.4 programy pod FreeRTOS

Typická funkce main() mívá následující strukturu:

int main(void)

V.Váňa: CanSAT Book for Students – part 2 4



```
{
 /* Setup the microcontroller hardware for the demo. */
 prvSetupHardware();
 /* Create the common demo application tasks, for example: */
 vCreateFlashTasks();
 vCreatePollQTasks();
 vCreateComTestTasks();
 atd.
 /* Create any tasks defined within main.c itself, or otherwise specific to the
 demo being built. */
 xTaskCreate( vCheckTask, "check", STACK_SIZE, NULL, TASK_PRIORITY, NULL );
 Etc.
  /* Start the scheduler, this function should not return as it causes the execution
 context to change from main() to one of the created tasks. */
 vTaskStartScheduler();
 /* Should never get here! */
 return 0;
```

}

Partest.c je jméno souboru jehož název je dán historickými důvody 'parallel port test' Další soubory:

File	Features Demonstrated	
main.c	 Starting/Stopping the kernel Using the trace visualisation utility Allocation of priorities 	
dynamic.c	 Passing parameters into a task Dynamically changing priorities Suspending tasks Suspending the scheduler 	
BlockQ.c	 Inter-task communications Blocking on queue reads Blocking on queue writes Passing parameters into a task Pre-emption Creating tasks 	
ComTest.c	 Serial communications Using queues from an ISR Using semaphores from an ISR Context switching from an ISR Creating tasks 	



CRFlash.c	 Creating co-routines Using the index of a co-routine Blocking on a queue from a co-routine Communication between co-routines
CRHook.c	 Creating co-routines Passing data from an ISR to a co-routine Tick hook function Co-routines blocking on queues
Death.c	 Dynamic creation of tasks (at run time) Deleting tasks Passing parameters to tasks
Flash.c	 Delaying Passing parameters to tasks Creating tasks
Flop.c	Floating point mathTime slicingCreating tasks
Integer.c	Time slicingCreating tasks
PollQ.c	 Inter-task communications Manually yielding processor time Polling a queue for space to write Polling a queue for space to read Pre-emption Creating tasks
Print.c	Queue usage
Semtest.c	 Binary semaphores Mutual exclusion Creating tasks

5.5 Nastavení vysílače s ADF7012

V prvním díle tohoto výukového materiálu jsme uvedli základní údaje o integrovaném vysílačovém obvodu AD7012. Je použit např. v CanSAT stavebnici firmy PrattHobbies. Obvod lze získat od Analog Devices mj i jako free samples. Pomocí řídících signálů clk, dat a le se nejprve tento obvod nakonfiguruje, tj sériovým signálem do něj pošleme čtyři 32bitová řídící slova. V následujícím příkladu Program20 je ukázka inicializace tohoto obvodu. Řídící signály generujeme na PC4, PC5 a PC6. Pro demonstrační účely jsem umístil inicializační kód do nekonečné smyčky, abychom ho mohli prozkoumat pomocí osciloskopu.





```
/* STM32 VL Discovery Library
 * program20 - inicializace FM vysilace s ADF7012
 * ver 1.0
 * Vd for SPSE Jecna 2011 lessons
 */
#include "stm32f10x.h"
#include "STM32vldiscovery.h"
#include <stdio.h>
#define uchar unsigned char //dodal jsem
uchar acc;
uchar dat 1[]={0x02,0x08,0x5E,0x10}; //R Register naplnime 0x02085E10
uchar dat_2[]={0x00,0x08,0xD4,0x01}; //N-Counter Latch naplnime 0x0008D401
uchar dat_3[]={0x00,0x00,0x37,0xE2}; //Modulation Register napl.0x000037E2
uchar dat 4[]={0x00,0x5A,0xA0,0x57}; //Function Register napl. 0x005AA057
                            //PC4
void signal_clk(u8 vystup);
void signal_dat(u8 vystup);
                              //PC5
void signal le(u8 vystup);
                              //PC6
```



```
void GPIO Inicialization(void);
void Delay( IO uint32 t nTick);
void delay us(u8 Time);
void delay ms(u8 Time);
void write reg(uchar *dat1);
int main(void)
{
  GPIO Inicialization();
  STM32vldiscovery LEDOn(LED4); // LED4 - modra dioda
  STM32vldiscovery LEDOn(LED3); // LED3 - zelena dioda pridal jsem
 /* inicializace ADF7012 */
  signal_le(1);
  signal_dat(1);
 signal clk(0);
/* inicializace ADF7012 */
    STM32vldiscovery_LEDOff(LED4); // LED4 - modra dioda
     STM32vldiscovery_LEDOff(LED3);
    Delay(0x2FFFFF);
  while (1)
  {
   /* inicializace ADF7012 */
       write reg(dat 1);
       delay_us(50);
       write_reg(dat_2);
       delay us(50);
       write reg(dat 3);
       delay us(50);
       write reg(dat 4);
       delay us(50);
      Delay(0x0000FF);
  }
}
void GPIO Inicialization(void)
{
     STM32vldiscovery LEDInit(LED3);
     STM32vldiscovery LEDInit(LED4);
     STM32vldiscovery PBInit(BUTTON USER, BUTTON MODE GPIO);
     STM32vldiscovery LEDOff(LED3);
     STM32vldiscovery_LEDOff(LED4);
      GPIOC->CRL &= ~(GPIO CRL MODE4 | GPIO CRL CNF4);
      GPIOC->CRL |= GPIO CRL MODE4 0;
      GPIOC->CRL &= ~(GPIO CRL MODE5 | GPIO CRL CNF5);
      GPIOC->CRL |= GPIO CRL MODE5 0;
```

461

V.Váňa: CanSAT Book for Students – part 2



```
GPIOC->CRL &= ~(GPIO CRL MODE6 | GPIO CRL CNF6);
     GPIOC->CRL |= GPIO_CRL_MODE6_0;
}
void signal clk(u8 vystup) //PC4
{
 if (vystup==0)
  {
       GPIOC->BSRR = GPIO BSRR BR4;
  }
  else
  {
      GPIOC->BSRR = GPIO BSRR BS4;
  }
}
void signal_dat(u8 vystup) //PC5
{
 if (vystup==0)
  {
      GPIOC->BSRR = GPIO BSRR BR5;
  }
 else
  {
     GPIOC->BSRR = GPIO BSRR BS5;
  }
}
void signal le(u8 vystup) //PC6
{
 if (vystup==0)
  {
       GPIOC->BSRR = GPIO BSRR BR6;
  }
 else
  {
      GPIOC->BSRR = GPIO BSRR BS6;
  }
}
void Delay( IO uint32 t nTick)
{
 for(; nTick != 0; nTick--);
}
void delay us(u8 Time)
{
     if (Time > 0 )
           for(; Time != 0; Time--)
```

V.Váňa: CanSAT Book for Students – part 2

ARM STM32 programming



```
Delay(0x00FF);
            }
};
void delay ms(u8 Time)
{
     if (Time > 0 )
           for(; Time != 0; Time--)
            {
                 delay us(60);
            }
};
void write reg(uchar *dat1) //vysilame data pro registry ADF7012
{
 int i,j;
 signal_le(0);
 for(j=0;j<4;j++) // mame 4 x 8 = 32 bitova ridici slova obvodu ADF7012</pre>
 {
  acc=dat1[j]; //do acc zkopiruje j-tou osmici bitu z ridiciho slova
 for(i=0;i<8;i++) // vysilame po osmicich bitu</pre>
  {
   if (acc>127 ) signal_dat(1); //jestlize acc>127 tak v acc je 1xxxxxxx
   else
                           //jestlize neni acc>127 tak v acc je 0xxxxxxx
   signal_dat(0);
   acc=acc<<1;</pre>
   delay_us(10); //na clk tj PC4 posleme postupne 0 1 0 a tim udelame
                  //hodinovy pulz
   signal clk(1);
   delay us(10);
   signal clk(0);
     delay us(10);
   }
 }
}
```

Na osciloskopu vidíme





Zobrazovaný úsek vysílaných dat je 011010 - horní modrý průběh ukazuje vysílaná data **dat**, dolní zelený průběh pak hodinový signál **clk**.

Na dalším obrázku si ukážeme na osciloskopu zobrazené celé 32bitové řídící slovo





Na tomto obrázku vidíme na dat vyslaný signál 00000000 01011010 10100000 01010111 tj. 0x00 0x5A 0xA0 0x57 což je obsah Function Registru.

6.1 Příloha – Úvod do architektury cortex-m3 (převzato z pandatron.cz)

6.2.1 Úvod do architektury Cortex-M3 - díl. 1

Rodina procesorů založených na architektuře ARM Cortex poskytuje potřebné zdroje k řešení i náročných průmyslových, automobilových, bezdrátových a mnoha dalších technologií.

Rodina obvodů je složena ze tří hlavních typů architektur: ARMv7 je typ pro velmi náročné aplikace, běžící na složitých operačních systémech. Typ R je určen pro použití v real-time systémech a typ M je naproti tomu optimalizován pro použití v levných embedded aplikacích.

Rodiny procesorů Cortex-M3 jsou první procesory architektury ARMv7-M, které byly navrženy s cílem dosáhnout vysokého výkonu systému a maximální efektivity levných vestavěných aplikací, jako jsou například průmyslové řídicí systémy, automobilová elektronika, drátové a bezdrátové telekomunikační systémy, systémy elektrického ovládání atd. K dosažení tohoto cíle však bylo potřeba provést v základní architektuře řadu zásadních změn. Zejména velmi zjednodušit proces tvorby softwarového balíku. Výsledkem je cenově efektivní využití procesorů architektury Cortex-M3 a to i v těch nejjednodušších aplikací.





Obr. 1: Blokové schéma obvodu řady Atmel SAM3U

Chcete-li zvýšit výkon, musíme procesory donutit pracovat buď s vyšší taktovací frekvenci, nebo použít sofistikovanější metody zvyšování výkonnosti algoritmů. Zvýšením hodinové frekvence lze v zásadě zvyšovat produktivitu, ale zároveň tímto krokem dojde i ke zvýšení spotřeby energie a tím spojené složitosti celé aplikace, vycházející ze zajištění potřebného napájení. Naproti tomu zvýšení efektivity zpracování strojového kódu na nižší taktovací frekvenci je mnohem výhodnější. Ve svém jádru pracují procesory Cortex-M3 v závislosti na použité Harvardské technologii se třemi fázemi zpracování dat (pipeline), které nabízí řadu příležitosti. Množství jedno-cyklových operací přináší až 1,25 MIPS / MHz (v praktickém testu).

V procesorech Cortex-M3 je k dispozici nová sada příkazů s označením Thumb-2, která programátorovi umožňuje dosažení až o 70 % vyššího výkonu na jeden megahertz než klasické ARM procesory, založené na jádře ARM7TDMI a vybavené pouze klasickou Thumb instrukční sadou. Navíc je možné dosáhnout až o 35 % vyššího výkonu, než u shodných procesorů, které vykonávají základní sadu instrukcí ARM (testováno na shodném testu).

Zkrácení doby uvedení konečného produktu na trh a snížení nákladů na vývoj, jsou dnes ty nejdůležitější kritéria při výběru procesoru. Stejně jako schopnost rychle a snadno vytvořit strojový kód, jsou všechny uvedená kritéria klíčovým požadavkem aplikovaným v tomto směru.

Procesory založené na architektuře Cortex-M3, jejichž cílem je zajistit rychlé a jednoduché vytvoření efektivního strojového kódu bez použití vložek assembleru, netlačí na programátora zbytečně hluboké znalosti své architektury a to ani při vytváření průměrně náročných aplikací. Tyto procesory využívají zjednodušený programovací model zásobníku, ve kterém se podařilo velmi efektivně provázat ARM jádro s myšlenkami aplikovanými ve standardních 8- a 16-bitových mikrokontrolérech. Například tak

V.Váňa: CanSAT Book for Students – part 2 466

ARM STM32 programming



funkce řízení přerušení umožňuje produkovat velmi jednoduchý kód a to zcela bez použití assembleru nebo složité manipulace s registry.



Obr. 2: Blokové schéma jádra Cortex-M3

Zbrusu novým, klíčovým prvkem systému instrukcí Thumb-2 je účinnější kompilátor C s možností operací s bity a bitovými poli, hardwarovým tříděním a účinnějším provádění strukturálních podmínek (jakými jsou kupříkladu if / then). Z pohledu programátora, umožňuje instrukční soubor Thumb-2 generovat strojový kód mnohem rychleji, jednodušeji a efektivněji. Navíc pomocí odpovídajícího kompilátoru není třeba řešit ani volbu mezi optimalizací kódu pro rychlost nebo objem - jednoduše je možné zvolit komplexní optimalizaci objemu a rychlosti. To dále zrychluje i samotný proces přípravy a generování kódu, neboť nyní vývojáři nemusí předkompilovávat kritické části kódu a přilinkovávat je na hlavní program v podobě samostatných knihoven.

Hlavním omezujícím faktorem v masivním nasazování efektivnějších procesorů je jejich vyšší výrobní cena. Moderní technologie jsou drahé a proto je rozhodujícím faktorem při snižování nákladů na výrobu procesoru jedině velikost použitého křemíku. Z toho důvodu procesory vyrobené technologií Cortex-M3 obsahují nejmenší jádro typu ARM, které je k dnešnímu dni průmyslově aplikovatelné. Obsahuje pouze 33 tisíc tranzistorů, vyrobených technologií 0,18 mikronů a stejně tak i okolní periferie snížily počet svých prvků na rozumnou velikost. Další obměna proběhla i v přístupu jádra procesoru do paměťového prostoru a to zavedením technologie uchování pracovních dat, změnou práce bitových operací a systémových instrukcí Thumb-2.

Výsledkem těchto změn je o více než 25 % nižší počet přístupů do hlavního paměťového prostoru ve srovnání s klasickým jádrem ARM. Na následujících obrázcích jsou uvedeny grafy výkonu a objemu strojového kódu v různých oblastech využití procesorů Cortex-M3 a ARM7TDMI.





Obr. 3: Srovnání výkonu procesorů ARM7TDMI a Cortex-M3 v jednotlivých aplikacích

ARM/TDMI Cortex-M3



Obr. 4: Srovnání objemu strojového kódu procesorů ARM7TDMI a Cortex-M3 v jednotlivých aplikacích

Přestože procesory s jádrem ARM nejsou na trhu příliš dlouho, i tak již nalezly uplatnění v celé řady embedded aplikací. Dnes jsou však nejprogresivnější obvody založené na architektuře Cortex-M3 a v blízké budoucnosti se zřejmě stanou nejpoužívanější ARM architekturou. Především je tomu tak v důsledku vyšší produktivity, nižší složitosti programovacího modelu, lépe propracovaným systémem přerušení a v neposlední řadě i díky své nízké ceně. Základní výhody a srovnání obvodů založených na architekturách Cortex-M3 a ARM7TDMI jsou uvedeny v následující tabulce.

Parametr	Jádro	
	ARM7TDMI-S	Cortex-M3


Jádro	ARMv4T	ARMx7-M	
Architektura	von Neumann	Harvard	
Podporované instrukční sady	Thumb	Thumb / Thumb-2	
Přerušení	FIQ /IRQ	NMI + 1 až 240 fyz. přerušení	
Čas přerušení	24 42 cyklů	12 cyklů	
Stepping režim	Ne	Integrované	
Ochrana paměti	Ne	8 regionálních jednotek	
Výkon	0,95 DMIPS / MHz (ARM)	1,25 DMIPS / MHz	
Příkon	0,28 mW / MHz	0,19 mW / MHz	
Plocha čipu v mm2	0,62 (pouze jádro)	0,86 (základní a standardní periferie)	

Tab. 1: Srovnání základních vlastností obvodů Cortex-M3 a ARM7TDMI

6.2.2 Úvod do architektury Cortex-M3 - díl. 2

Rodina procesorů založených na architektuře ARM Cortex poskytuje potřebné zdroje k řešení i náročných průmyslových, automobilových, bezdrátových a mnoha dalších technologií.

Procesory založené na architektuře Cortex-M3 jsou v podstatě hierarchickou skupinou. Obsahují základní CM3Core rozšířené periferie, obsahující mechanismy pro řízení přerušení, ochranu a přístup k paměti a mnoho dalších. Tento soubor je do značné míry volně konfigurovatelný, což umožňuje snadné použití procesoru pro skutečně širokou škálu úkolů, které většinou plně uspokojí veškeré požadavky. Jádro Cortex-M3 a jeho začlenění do zbylé architektury (obr. 1) jsou navrženy pro splnění všech požadavků a zároveň minimalizaci požadované velikosti paměti, spotřeby energie apod.





Obr .1: Blokový diagram obvodu s jádrem ARM Cortex M3

Jádro Cortex-M3 je založeno na architektuře, vzniklé na Harvardově univerzitě. To je hlavní rozdíl od ostatních ARM procesorů typu von Neuman, který spočívá v oddělené instrukční a datové sběrnici. Při vykonávání povelů čte procesor data i instrukce zároveň a díky zřetězení je schopen vykonávat i několik operací současně. Tím je samozřejmě dosaženo vyššího výkonu systému.

Tří-stupňový pipeline vykonává naráz v jediném hodinovém cyklu následující operace:

- Fetch vyzvednutí z pamětí instrukce č.3
- Decode dekódování instrukce č.2
- Execute zpracování nejstarší instrukce č.1





Uvedený způsob spolu se základní schopností predikce vede ke značnému zrychlení vykonávání instrukcí v porovnání s jinými architekturami. Je-li tak již při načítání instrukcí z paměti známo místo pokračování kódu, není potřeba čekat na vykonání skokové instrukce a do zásobníku je možné ihned začít číst data z vyžadované oblasti. To dále snižuje zbytečné prostoje a urychluje přísun dat výkonnému jádru.

Procesorové jádro Cortex-M3 obsahuje dekodér pro standardní instrukce a instrukce Thumb-2. Ty přinášejí mnohem lepší využití dostupných prostředků, především paměti, řídicí logiky, rozhraní a ALU s podporou hardwarového násobení a dělení.

Použitá architektura procesoru Cortex-M3 je 32-bitová s 32-bitovou datovou sběrnicí, pomocnými registry a periferiemi a pamětí. Jádro obsahuje celkem 13 registrů pro všeobecné použití, dva ukazatele zásobníku, registr vazby, programový čítač, stav rejstříku a sadu speciálních registrů.

Z pohledu režimu provozu podporuje procesor Cortex-M3 dva pracovní režimy. Thread a Handler, stejně jako dvě úrovně přístupu ke kódu (privilegované a bez privilegií), umožňují snadno vytvářet i složité struktury a to bez ztráty kvality. Neprivilegovaná úroveň přístupu omezuje nebo zcela eliminuje přístup a použití některých zdrojů, jako jsou kupříkladu určité paměťové oblasti apod. Režim vlákna, tedy Thread je standardní pracovní režim, podporující distribuci úrovní přístupu. Režim Handler je využíván pouze omezeně, v opodstatněných případech, kdy celý kód získá privilegovanou úroveň přístupu.

6.2.3 Úvod do architektury Cortex-M3 - díl. 3

Rodina procesorů založených na architektuře ARM Cortex poskytuje potřebné zdroje k řešení i náročných průmyslových, automobilových, bezdrátových a mnoha dalších technologií.

V předchozích dvou dílech úvodu do architektury Cortex-M3 jsme se zaměřili na popsání hlavních principů jádra a obecné struktury. Dále byly uvedeny hlavní rozdíly mezi jádrem Cortex-M3 a tradičním jádrem architektury ARM7 a popsány základní principy interakce mezi organizačními jednotkami procesorů vystavěných na architektuře Cortex-M3.

Jak bylo uvedeno dříve, jádro Cortex-M3 obsahuje dekodér pro obě tradiční instrukční sady a pro novou generaci Thumb-2. Rozšířené ALU rovněž podporuje hardwarové násobení a dělení, stejně jako obsahuje blok řídicí logiky a rozhraní s ostatními komponentami systému procesoru. Procesor postavený na architektuře jádra Cortex-M3 má jednoduchou, pevnou paměťovou plochu s maximálním adresovatelným prostorem až do 4 GB. Její součástí je pevně daný adresní prostor pro programovou paměť Flash, pracovní paměť RAM, externí paměť, periferie a další (obr. 1). Jak je z obrázku patrné, součástí adresovatelného paměťového prostoru jsou rovněž adresy, které jsou vyhrazeny výrobcem.



0xE00FFFFF	DOMESTIC			OxFFFFFFFF
0xE00FF000	ROM table			
0xE0042000	External PPB		Vendor Specific	
0xE0041000	ETM		Private Parinheral Pus External	0~500400000
0xE0040000	TPIU	-7	Private Peripheral Bus - External	0xE003FFFF
			Private Peripheral Bus - Internal	0xE0000000 0xDFFFFFFF
0xE003FFFF	Burned			
0xE000F000	Keserved	/		
0×E000E000	NVIC		External Device 1 GB	
0*E0003000	Reserved	/		
0	FPB	/		
UXEUUU2UUU	DWT			0xA0000000 0x9FFFFFFF
0xE0001000				
0xE0000000	IIM			
			External RAM 1 GB	
0x43FFFFFF				
	Bit band alias	、		060000000
0x42000000		\backslash		0x5FFFFFFF
0x41FFFFFF			Perinheral 0.5 GB	
0x40100000 0x40000000	Bit band region		respictar 0.5 GD	0-40000000
				0x40000000
0x23FFFFFF			SPAM 0.5 GB	
	Bit band alias		SKAM 0.5 GB	0~20000000
0x22000000				0x1FFFFFFF
0x21FFFFFF			Code 0.5 GP	
0x20100000	Bit band region		Code 0.5 GB	
0x20000000				0x00000000

Obr. 1: Adresovatelný paměťový prostor obvodů architektury Cortex-M3

Procesory postavené na jádře architektury Cortex-M3 poskytují rovněž přímý přístup k datovým bitům a to pomocí jednoduchého mechanismu (obr. 2). Všimněte si, že uvedené operace jsou základní a nemohou být přerušený dalšími operacemi a přerušením.





Obr. 2: Srovnání mechanismu bitových operací v tradiční architektuře ARM7 (a) a Cortex-M3 (b)

Tradiční procesory založené na architektuře ARM7 podporují pouze přístup k celým datovým slovům, což znamená, že veškeré datové práce musí být zarovnány na hranice 32 bitových slov. Zpracování stejných instrukcí na architektuře Cortex-M3 umožňuje přístup k nezarovnaným datům, což výrazně minimalizuje časové prodlevy spojené s přístupem k takovým datům. V případě, že je požadován přístup k nezarovnaným datům, je možné provést jejich rozdělení do několika paralelních vláken se zarovnanými daty, takový proces je transparentní i pro uživatele programu, jako je tomu automaticky uvnitř jádra.

Krom toho, architektura obvodů Cortex-M3 podporuje 32-bit násobení v jediném cyklu, stejně jako práci se signed a unsigned daty, což ve svém důsledku vyžaduje 2 až 12 hodinových taktů v závislosti na velikosti operandů. Operátor dělení je samozřejmě rychlejší v případě, že dělitel a dělenec jsou malé. Takové zlepšení v provádění matematických operací, poskytované architekturou obvodů CortexM3, je významnou výhodu oproti procesorům založených na standardu architektury ARM7 a to především pro řešení problémů, které vyžadují rozsáhlé matematické výpočty.

Mikroprocesory architektury ARMv7-M jsou rozšířením architektury ARM7, které se od předchozích ARM architektur liší podporou systému instrukcí Thumb-2. Thumb-2 je technologie 16 - a 32-bit instrukční sady, poskytující kapacitu 32-bit ARM instrukcí a velice malou velikost výsledného kódu. Je třeba rovněž poznamenat, že systém instrukcí Thumb-2 je zpětně kompatibilní s 16-bit instrukcemi Thumb. Obr. 3 je uveden výsledek testu jednotlivých instrukčních sad, což svědčí o výkonnosti systému Thumb-2 ve srovnání s jinými instrukčními systémy obvodů ARM. V předchozích verzích procesorů architektury ARM7 bylo pro efektivní kód, maximální výkon a celkově kompaktní kód potřeba vytvořit různé části programu buď pomocí ARM instrukční sady a nebo sady Thumb. Současně procesory architektury CortexM3 nevyžaduje tak složité manipulace, stejně jako jsou maximálně tolerantní k 16-bit a 32-bit instrukcím, vykonávaným ve stejném režimu. Vzhledem k tomu, že systém Thumb-2 je vytvořen z 16-bit instrukční sada Thumb, je možné na architektuře



Cortex-M3 spustit i programy napsané pomocí instrukční sady Thumb stanovené pro starší verze procesorů. Stejně tak jsou obvody Cortex-M3 kompatibilní i s kódem dřívějších verzí sérií ARM7.



Obr. 3: Srovnání jednotlivých instrukčních sad

6.2.4 Úvod do architektury Cortex-M3 - díl. 4

Rodina procesorů založených na architektuře ARM Cortex poskytuje potřebné zdroje k řešení i náročných průmyslových, automobilových, bezdrátových a mnoha dalších technologií.

V předchozích dílech Úvodu do architektury Cortex-M3 jsme se zaměřili na popsání hlavních principů jádra a obecné struktury. Dále byly uvedeny hlavní rozdíly mezi jádrem Cortex-M3 a tradičním jádrem architektury ARM7 a popsány základní principy interakce mezi organizačními jednotkami procesorů vystavěných na architektuře Cortex-M3 i postup přístupu procesoru k datům.



Obr. 1: Připomenutí srovnání jednotlivých instrukčních sad

Instrukční sada Thumb-2 obsahuje povely, které umožňují zjednodušit a snížit množství kódu ve spoustě aplikací. To znamená, že kupříkladu instrukce BFA a BFC, které jsou určeny pro práci s bitovými poli, jsou často používány i k řešení problémům zpracování, jako jsou datové pakety sítí v telekomunikačních aplikacích. Instrukce SBFX a UBFX zjednodušují práci s bity v registrech, což je V.Váňa: CanSAT Book for Students – part 2 474 ARM STM32 programming



velmi užitečné v automobilovém průmyslu. Instrukce RBIT přeskupuje bity v datovém slově a v důsledku toho se velmi často využívá při provádění DSP algoritmů, jako jsou Fourierovy transformace. Instrukce pro práci s tabulkami - TBB a TBH naproti tomu umožňují dosažení rovnováhy mezi kompaktností kódu a vysokým výkonem. V neposlední řadě je rovněž možné zmínit novou implementaci instrukcí IF / THEN v instrukční sadě Thumb-2, která umožňuje předpovídání pořadí provedení až do čtvrté podmínky.

Nedílnou součástí architektury Cortex-M3 procesorů je konfigurovatelný blok NVIC, nabízející vynikající podmínky pro obsluhu přerušení. Ve výchozím nastavení generuje nemaskované přerušení a 32 fyzických utilit s 8 úrovněmi priority. Zároveň však může být nakonfigurován i tak, aby vygeneroval libovolný počet 1 až 240 fyzických přerušení s až 256 úrovněmi priority a to při velice jednoduché konfiguraci.

Architektura procesorů CortexM3 obsahuje handler tabulky vektorů přerušení, která obsahuje adresy funkcí, vykonávaných při daném požadavku. V případě přerušení činnosti procesoru je tato adresa vektoru využita s pomocí rozhraní instrukční sběrnice k přesunu v programové paměti. Tabulka vektorů probíhajících přerušení se výchozím stavu nachází na nulové hodnotě, ale umožňuje přesun s předprogramovanými kontrolními registry.

V případě, že nastane situace, kdy je voláno jedno přerušení po druhém, opakují procesory s tradiční architekturou celý cyklus zálohování dat a obnovení vždy dvakrát (při vstupu a ukončení přerušení), s výstupem ze zpracovávaného přerušení, při vstupu do dalšího. To má však za následek značné zpoždění a ve svém důsledku i výrazný pokles celkového výkonu systému. Architektura procesorů CortexM3 zjednodušuje přechod z aktivního do nového přerušení uplatněním moderních technologií tzv. "dokování přerušení" a hardwarovou implementaci kontroly NVIC.

Technologií dokování přerušení lze dosáhnout mnohem menších časových prodlev tím, že nahradí sekvenční obnovu a ochranu dat, která obvykle trvá zhruba 30 cyklů, jednoduchým mechanismem, který celkem vyžaduje pouhých 6 cyklů (Obr. 2). Stavové registry procesoru jsou automaticky uloženy při vstupu do prvního přeruší a obnoví se až po odchodu s využitím jen několika cyklů, což je mnohem rychlejší než plně softwarová implementace obsluhy přerušení, známá z klasických procesorových architektur. Takové zdokonalení umožňuje získat velmi vysoký výkon, což je obzvlášť důležité v situacích, kdy je na standardní programový kód vyžadováno zpracování velkého počtu přerušení.





Obr. 2: Mechanismus pro manipulaci s přerušením v procesoru architektury Cortex-M3

Jednotka NVIC je rovněž zodpovědná za obsluhu napájení systému a podporu pro úsporné režimy. NVIC obsahuje integrovaný 24-bitový systém a časovač, který se používá pro získání přerušení. Pravidelné časové intervaly jsou nepostradatelné při běhu operačních systémů a real-time funkcí. MPU (jednotka ochrany paměti) je volitelná součást procesorové architektury Cortex-M3, která však může výrazně zvýšit systémovou spolehlivost či ochranou kritických částí kódu. Nejčastěji je využívána pro ochranu operačních systémů, rozdělených na procesy, ve kterých zabraňuje

nepovolenému přístupu k některým částem pracovní paměti. Zároveň umožňuje vymezení určitých oblastí paměti jako "read only" a detekci nežádoucích účinků přístupu do paměti, které by mohly narušit celý systém.

MPU tak aplikacím umožňuje pohodlné rozložení do souboru procesů. Každý takový proces má přidělenu svou vlastní oblast paměťového prostoru (včetně přidělené paměti programu, datové paměti, zásobníku atd.) a určených periferií, přičemž zároveň může odkazovat i do společné paměťové oblasti a společných periferií. MPU tak zajišťuje privilegovaný přístup dle stanovených úrovní. K těm patří i spuštění kódu s odpovídajícími preferencemi a práci s vyhrazenou pamětí a periferiemi. Zároveň s možností přístupu do sdíleného paměťového prostoru je možné i zde aplikovat ochranu, aby se zabránilo neoprávněnému přístupu. MPU podporuje až 8 takových domén, z nichž každá může být rozdělena do 8 suboblastí. Minimální velikost pole je 32 bytů a zvyšuje se v krocích dělitelných dvěma, maximálně však do adresovatelné paměti 4 GB. Přístup k paměti není zahrnut v oblastech určených dle MPU a nepovolený přístup do takové paměti povede k vyvolání odpovídající chyby.

Ochrana oblastí paměti před neautorizovaným přístupem je prováděna dle pravidel, která jsou naopak založeny na typech operací (čtení, zápis, spuštění procesu) a stupni prioritní oblasti programového kódu, který provádí přístup. Každá oblast má sadu bitů, které reprezentují povolené typy činností oblasti, které jsou odpovědné za druhy akcí.

Mezi další výhody MPU patří rovněž i podpora překrývajících se oblastí paměťového prostoru. Tato vlastnost poskytuje velmi výrazný přínos k ochraně uložených informací. Vzhledem k tomu, že velikost oblastí je násobkem dvou, je zde příležitost plného vstupu do jedné nebo více oblastí i v rámci dalších



oblastí paměti, realizaci speed multi-level struktury ochrany či řízení přístupu, stejně jako provádění vnořeného překrývání paměťových oblastí.

Literatura a další zdroje:

[1] Váňa V.: ARM pro začátečníky. BEN, Praha 2009, ISBN 978-80-7300-246-6

[2] http://www.st.com

[3] <u>http://www.st.com/internet/evalboard/product/250863.jsp</u>

- [4] http://cz.mouser.com/stm32discovery
- [5] Pokorný V.: Vývojový modul s 32bitovým procesorem typu ARM. Bakalářská práce VUT Brno 2009
- [6] Jůn L.: Vývojový modul s 32bitovým procesorem typu ARM. Diplomová práce VUT Brno 2009

[7] serial na <u>http://www.mcu.cz/</u>

[8] <u>http://pandatron.cz/?1252&uvod_do_architektury_cortex-m3__dil_1</u>

[9] http://pandatron.cz/?1281&uvod_do_architektury_cortex-m3 - dil. 2

- [10] http://pandatron.cz/?1389&uvod do architektury cortex-m3 dil. 3
- [11] http://pandatron.cz/?1465&uvod do architektury cortex-m3 dil. 4
- [12] http://measure.feld.cvut.cz/vyuka/predmety/A4M38AVS
- [13] <u>http://neuron.feld.cvut.cz/micro/stm32/index.html</u>
- [14] http://www.emcu.it/STM32.html
- [15] <u>http://embeddednewbie.blogspot.com/2011/01/free-toolchains-for-stm32vldiscovery.html</u>

[16] Günther Pregartner a Oliver Reisky : Sensordatenerfassungsmodul, Höheren technischen

Bundes-, Lehr-, und Versuchsanstalt\Bulme Graz-Gösting Abteilung Elektronik