

Příloha článku

Mikrokontroléry STM32F prakticky

Ing. Vladimír Váňa, CSc. OK1FVV

Programování paměti flash v STM32F10x

Výsledkem tvorby software pro naši aplikaci např. pomocí některého vývojového prostředí a postupu uvedeného v [1] je např. binární či hex soubor, jejichž obsah potřebujeme nahrát do programové paměti flash mikrořadiče. K tomu můžeme použít např. JTAG programátor či ST – link programátor využívající SWD rozhraní. Výhodou je i možnost jejich použití při odlaďování programů. Nutností je ovšem některý z těchto programátorů vlastnit.

Pro využití protokolu SWD a či protokolu JTAG, musí být toto v mikrořadiči povoleno. Ale u nových mikrořadičů není defaultně toto provedeno. Povolení se provádí **programově** pomocí volání funkce (pro mikrořadiče řady STM32F10x) [16]

```
GPIO_PinRemapConfig(GPIO_Remap_SWJ_NoJTRST, ENABLE); // Full Enable SWJ
```

Toto volání funkce umístíme ve zdrojovém kódu nějakého našeho programu v jeho inicializační části, program přeložíme a získaný .hex nebo .bin soubor použijeme jako zdroj pro naprogramování paměti flash. Po spuštění programu v mikrořadiči je povoleno SWJ a poté již k programování budeme moci používat SWD. Takže zbývá ještě vyřešit, jak tento program dostaneme do mikrořadiče v době, kdy ještě SWJ není povoleno, stejně jako v případě, kdy SWD či JTAG programátor nemáme vůbec.

Využijeme k tomu výrobcem vestavěný bootloader. Bootloader přes RS232C je v mikrokontroléru STM32F10x automaticky aktivován v případě, že je na **Boot** pinech mikrořadiče konfigurace „System memory“. Jedná se o stav kdy je vývod Boot0 = High a Boot1 = Low. Hodnota pro **Boot** piny je mikrokontrolérem stanovena při náběžné hraně SYSCCLK a současně je-li aktivní **Reset** pin mikrokontroléru.

Konfigurace Boot pinů

BOOT1	BOOT0	Boot mode	Popis
X	0	User Flash memory	User Flash memory is selected as the boot space
0	1	System memory	System memory is selected as the boot space
1	1	Embedded SRAM	Embedded SRAM is selected as the boot space

Bootloader v mikrokontroléru STM32F10x je podporován přes interface USART1, vývody PA.9/USART1_Tx, PA.10/USART1_Rx. U 64 pinového STM32F100 je PA9 na pinu 42, PA10 na pinu 43, BOOT0 na pinu 60 a BOOT1 (PB2) na pinu 28. Zároveň z tabulky vidíme důvod, proč na našich řídicích (palubních) počítačích máme pomocí odporů 10k zajištěno, že na pinech BOOT1 a BOOT0 máme nastavenou úroveň logická 0 – po resetování počítače bude počítač vykonávat náš uživatelský program.

Pokud v době resetování bude na BOOT0 úroveň 1, bude mikrořadič řízen programem Bootloader a bude se pokoušet komunikovat přes UART1. Jako interfejs mezi mikrořadičem a USB portem počítače PC použijeme převodník USB <-> sériový port. Doporučuji

převodník s FT232RL např. PremiumCord Ku2-232a nebo použít převodník vlastní výroby[17] .

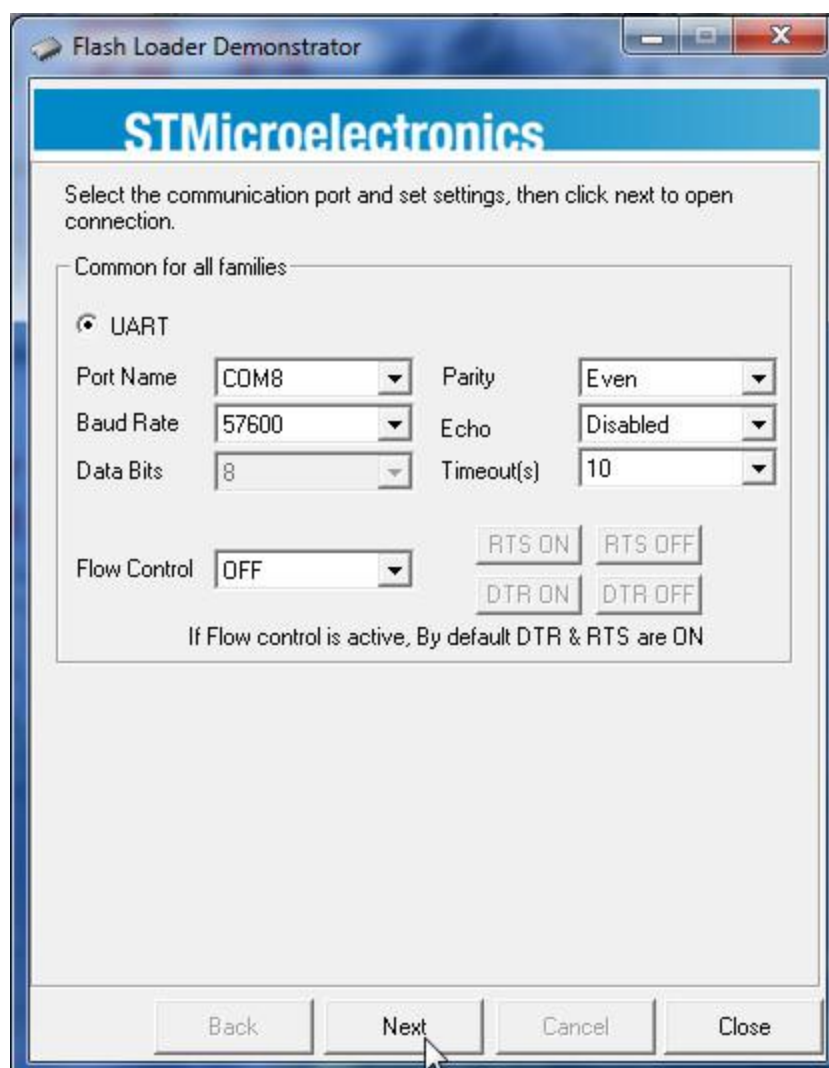
Pokud se bootloaderu podaří komunikace s PC, na kterém bude spuštěn vhodný program, můžeme pomocí tohoto programu poslat náš nový uživatelský program do mikrořadiče a poté, po odstranění 1 z jeho pinu BOOT0 a resetování, již náš program bude v mikrořadiči řídit jeho chod.

Doporučeným programem na PC pro naprogramování flash paměti obvodů STM32 je Flash Loader Demonstrator. Jeho poslední verze je 2.6 a získáme ho zdarma stažením ze stránek firmy ST jako instalační soubor Flash Loader Demonstrator v2.6.0_Setup.exe (je to program pro Windows). Po nainstalování Flash Load Demonstratoru nahrajeme náš program do programové paměti flash. Dále je uveden postup tohoto programování:

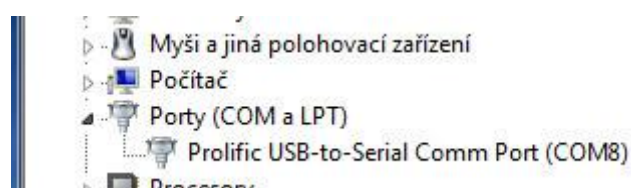
1. Propojíme palubní počítač s PC pomocí sériové linky RS232 (nemáme-li na PC RS232, lze použít i převodník USB -> RS232)
2. Připojíme napájení + 3,3 V palubního počítače
3. Stiskneme a držíme tlačítko BOOT0 připojené k palubnímu počítači a stiskneme a uvolníme tlačítko RESET, rovněž vně připojené k palubnímu počítači. Poté uvolníme i tlačítko BOOT0. Takto je mikroprocesor připraven k nahrání programu.

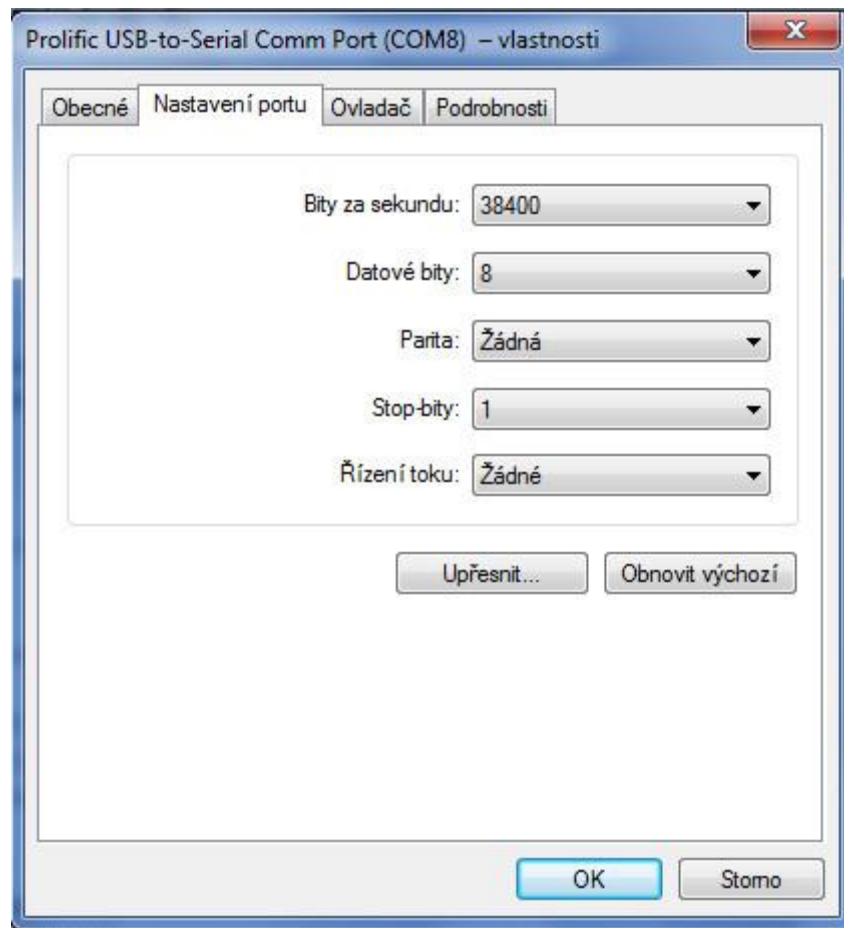
(V některých návodech je ale uvedeno, že tlačítko BOOT0 musíme mít stlačeno po celou dobu práce se SW Flash Loader Demonstrator!!! - to ale nebyl můj případ, ostatně jednou rukou držet tlačítko BOOT0 a jen druhou snímat obrazovky bych nezvládl ...)

4. Na PC spustíme aplikaci STMicroelectronics Flash Loader Demonstrator a na úvodní obrazovce vybereme komunikační port (COM0, COM1, ..),



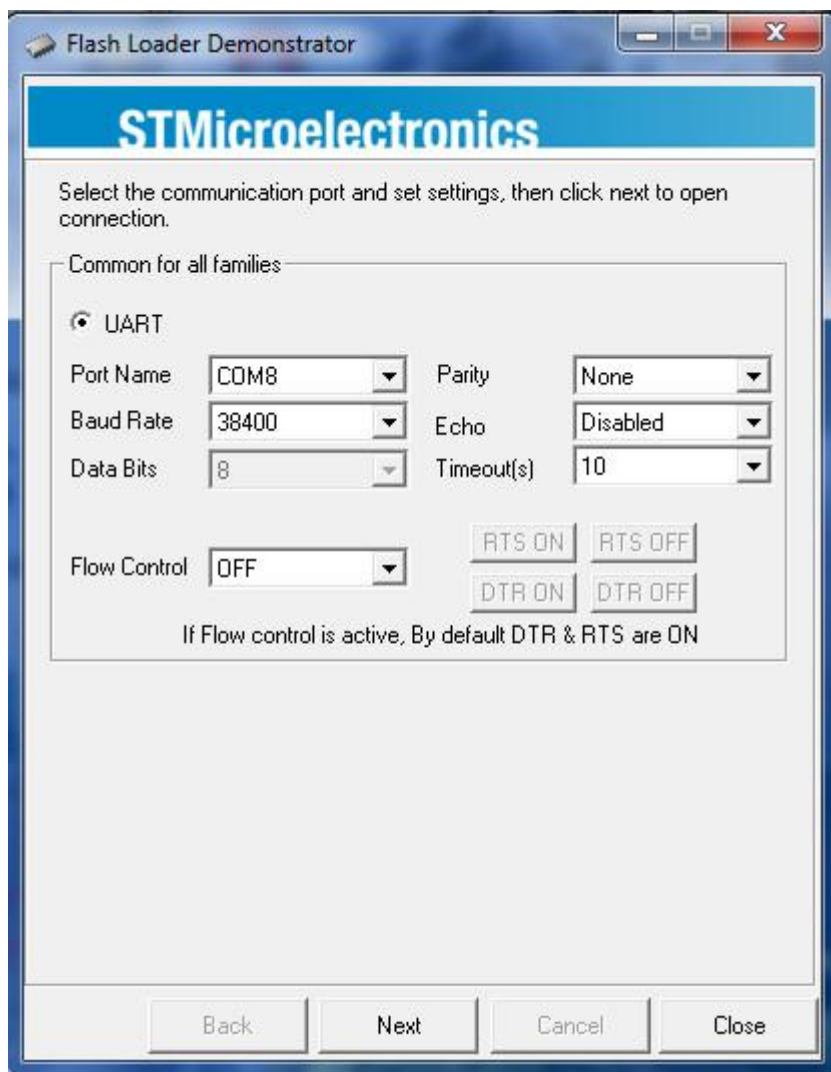
Nastavíme parametry sériového přenosu (pracoval jsem s notebookem s USB a s USB/com převodníkem PL2303) a virtuálním portem COM8



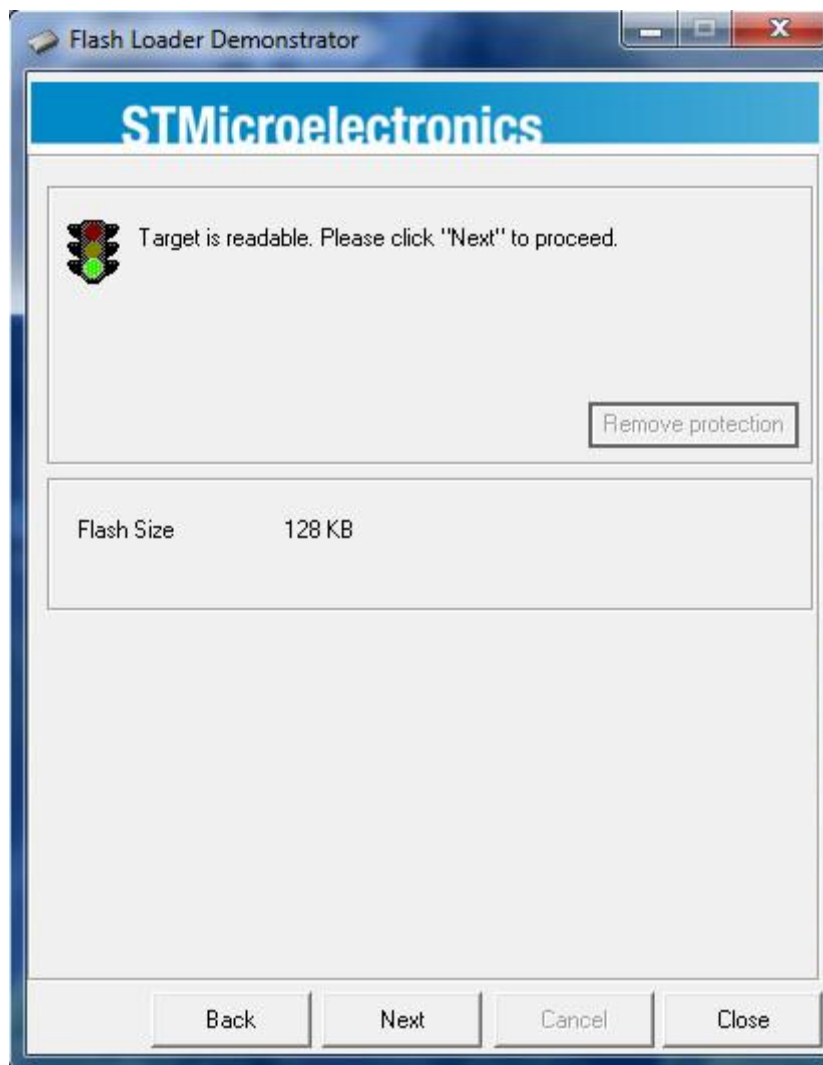


Pozn.

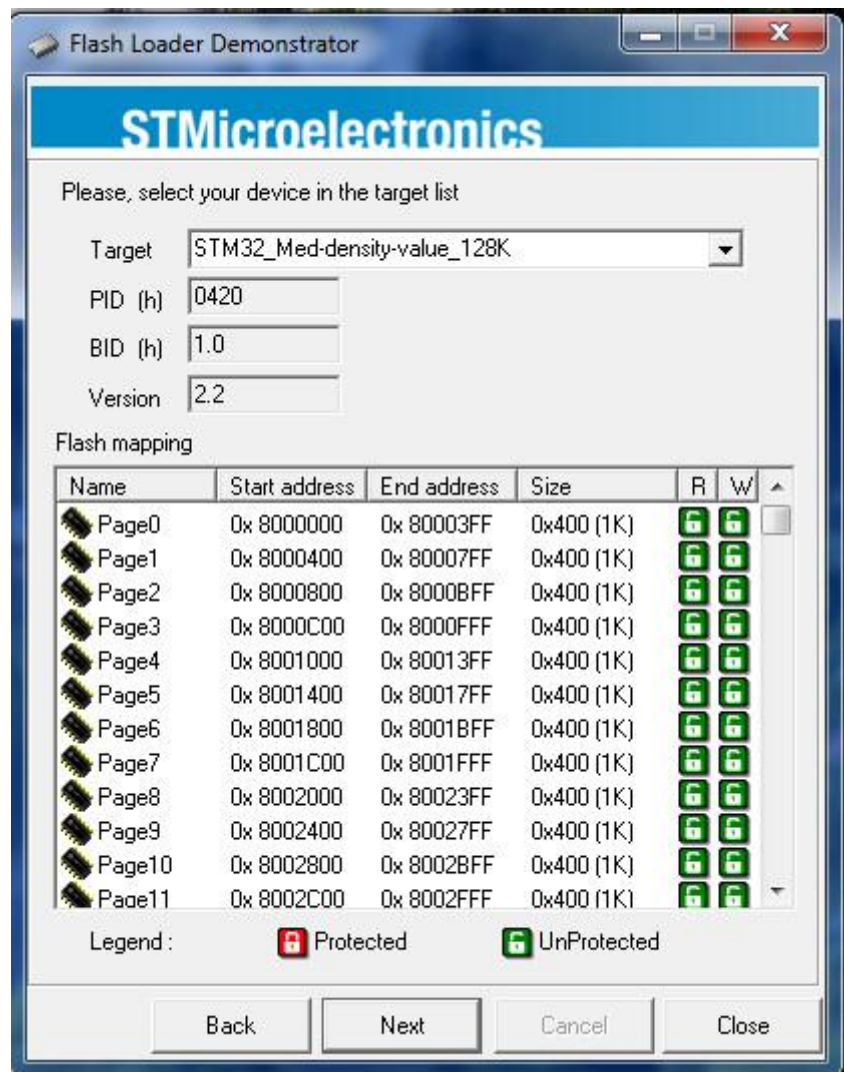
38400 Bd jsem nastavil z toho důvodu, že stejnou rychlostí se komunikuje s vysílačem a protože com port PC využívám i při testování palubního počítače kdy terminálovým programem kontroluji, co se vysílá a mám proto tento port nastavený na rychlost 38400. Kdybych měl jinou rychlost nastavenou ve Flash Loader Demonstrator, musel bych měnit *properties* com portu.



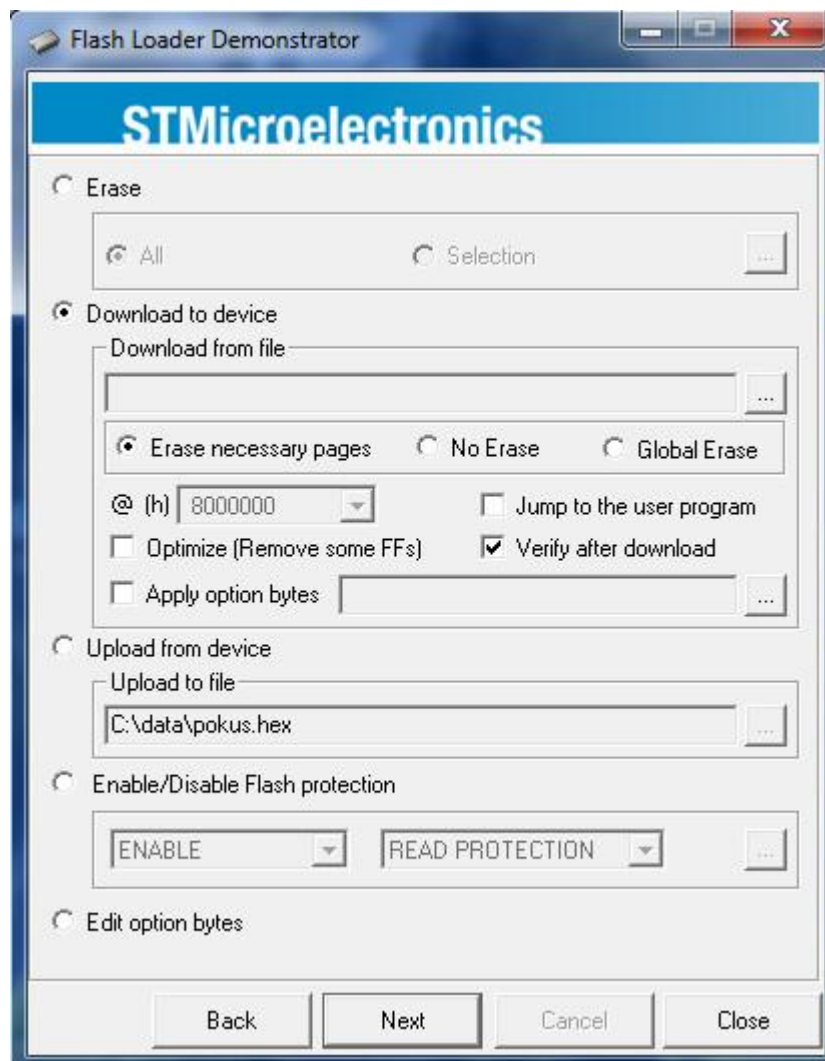
ostatní nastavení ponecháme a pokračujeme dále (tlačítko **NEXT**).



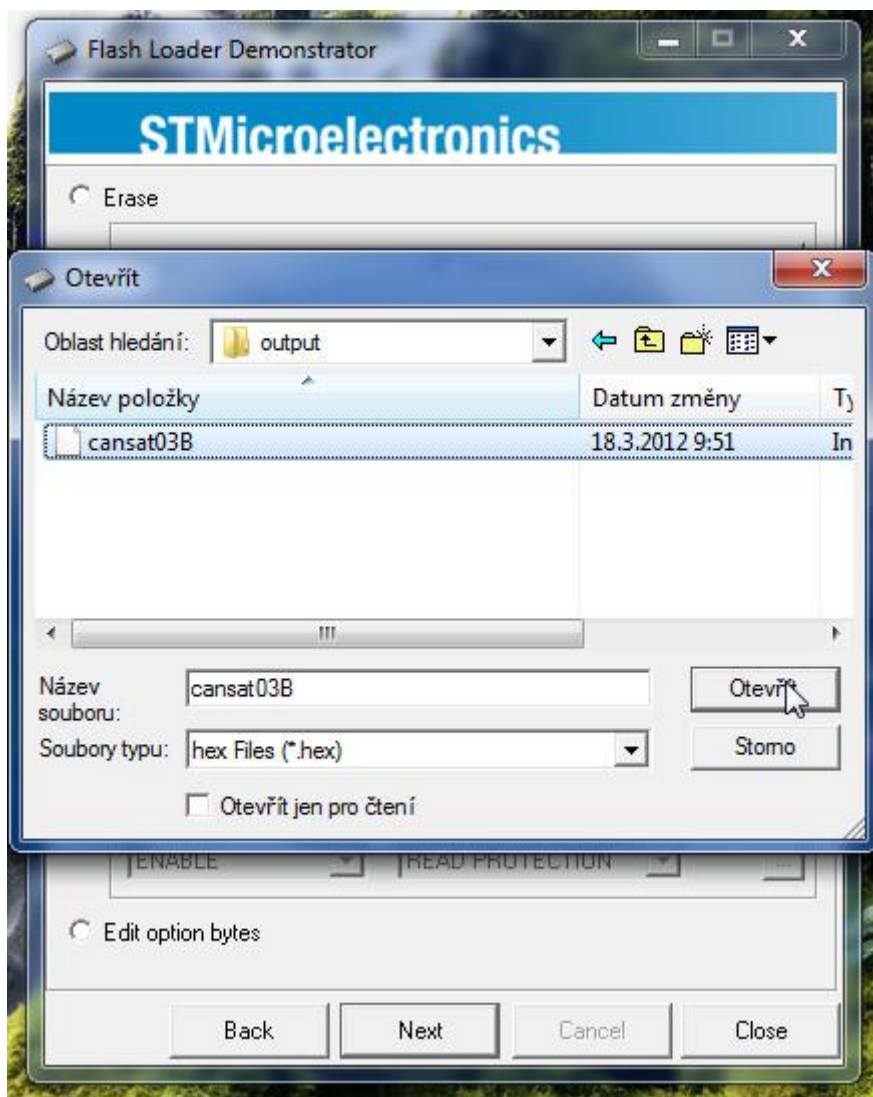
To již máme vyhráno, s navázáním spojení jsem měl totiž občas problémy. Dáme **Next**, dostaneme



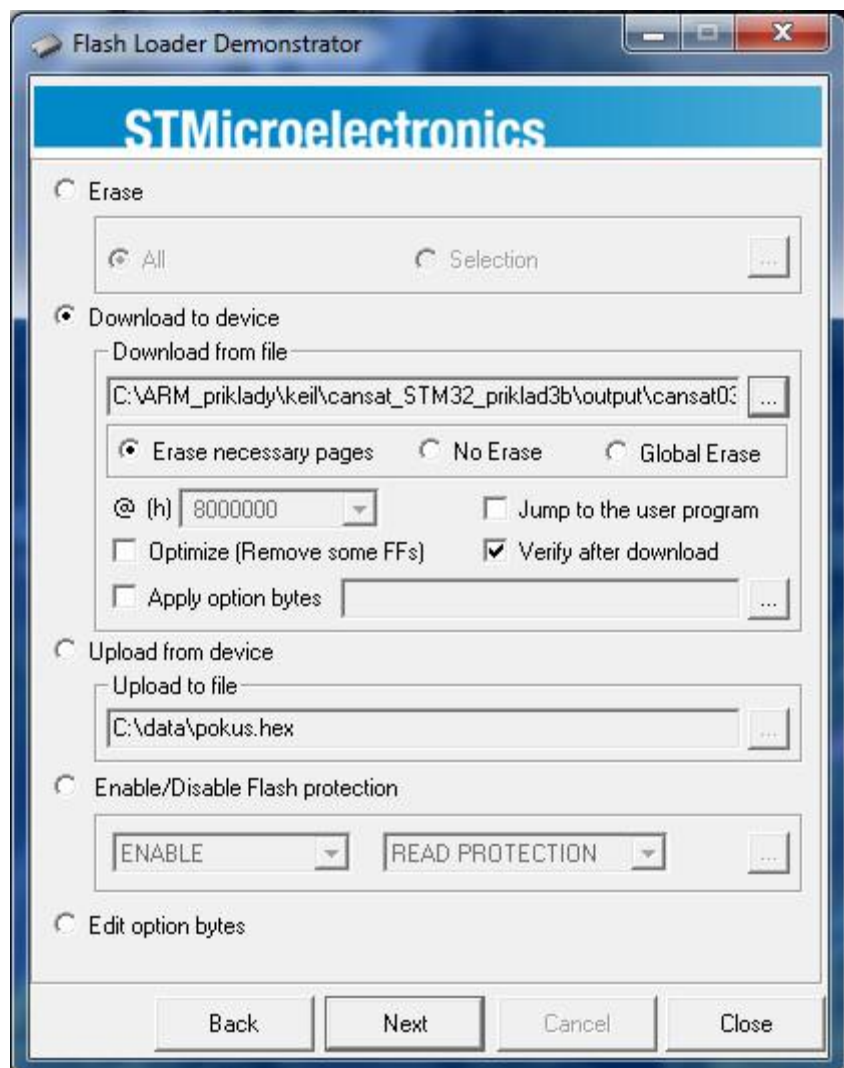
Opět dáme **Next** (předtím se pro jistotu podíváme, že to opravdu poznalo STM32 – viz textbox Target). Dostaneme



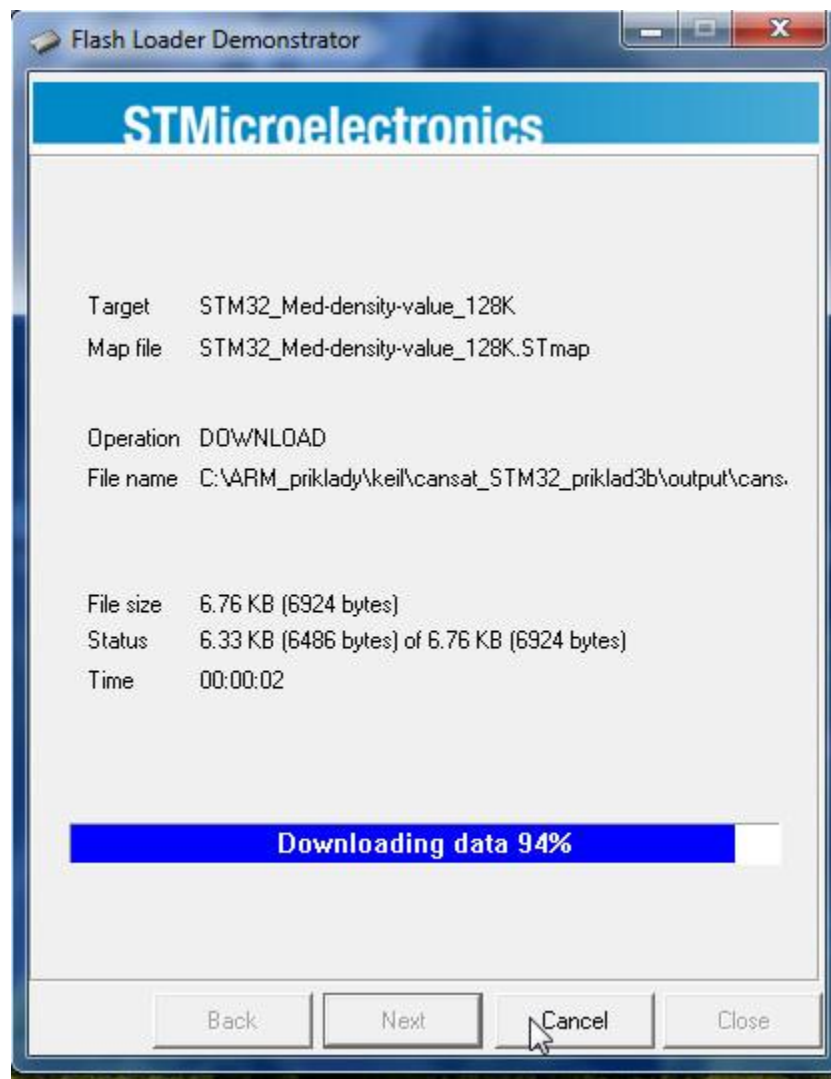
Klikneme na ikonku s třemi tečkami napravo od textboxu **Download to device** a najdeme náš soubor hex



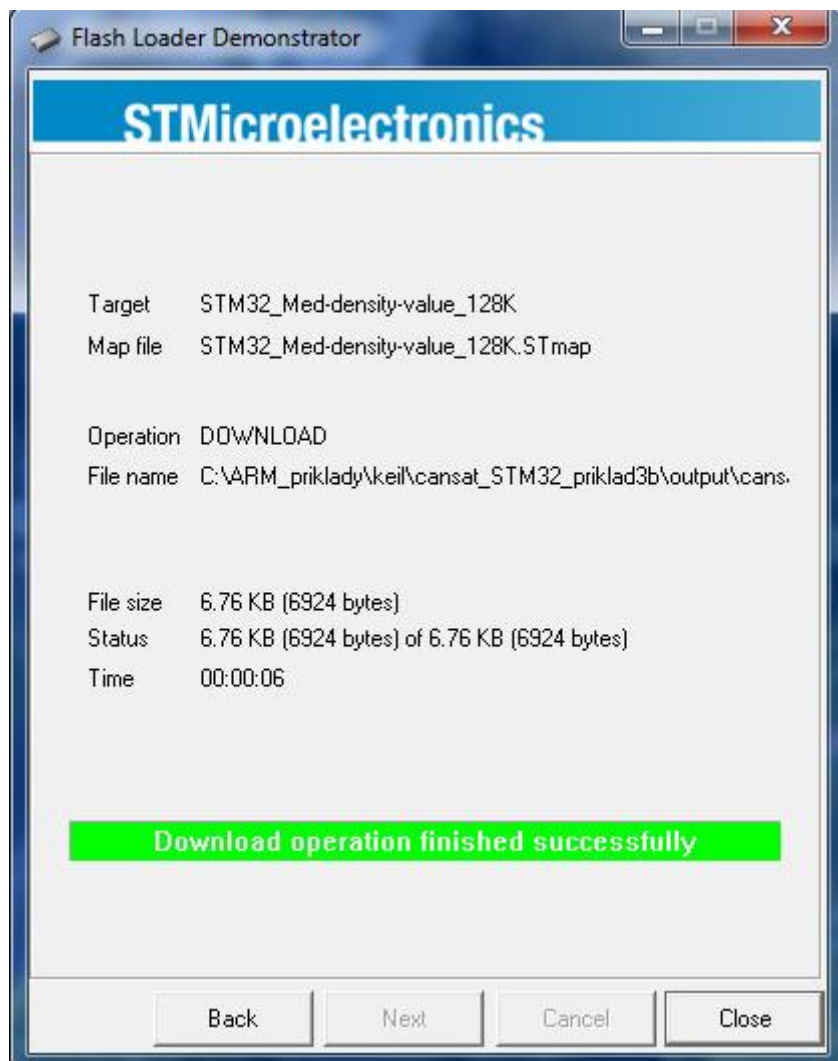
Klikneme na tlačítko **Otevřít**, máme



Klikneme na **Next** a začne se nahrávat náš program do flash paměti procesoru



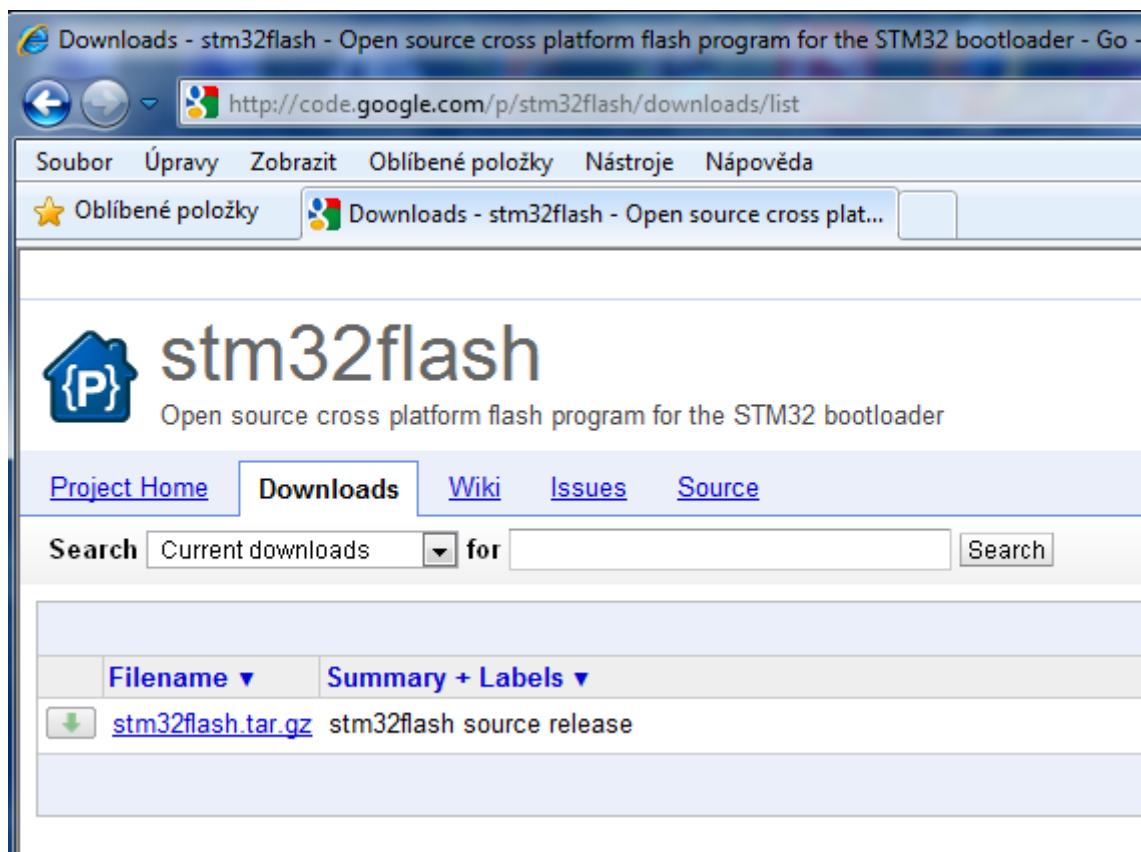
Poté bychom měli dostat



Stačí již jen kliknout na **Close**. Po resetování by už měl pracovat procesor pod nahráním uživatelským programem.

S nahráváním programu do paměti flash pomocí bootloaderu a programu FlashLoader Demonstrator bývají občas problémy s navázáním spojení s programovaným mikrořadičem. Někdy stačí jen snížit rychlost COM např. na 9600 Bd , popř. použít jinou verzi **Flash Loader Demonstrator**. Mě se však nejvíce osvědčilo pracovat pod OS Linux a používat program STM32Flash.

Proto si teď práci s STM32 pod Linuxem také ukážeme. Nejprve si z <http://code.google.com/p/stm32flash/downloads/list>



Stáhneme soubor **stm32flash.tar.gz** a poté z něj vyextrahujeme (ve windows nebo v Linuxu) adresář STM32flash jehož obsah je

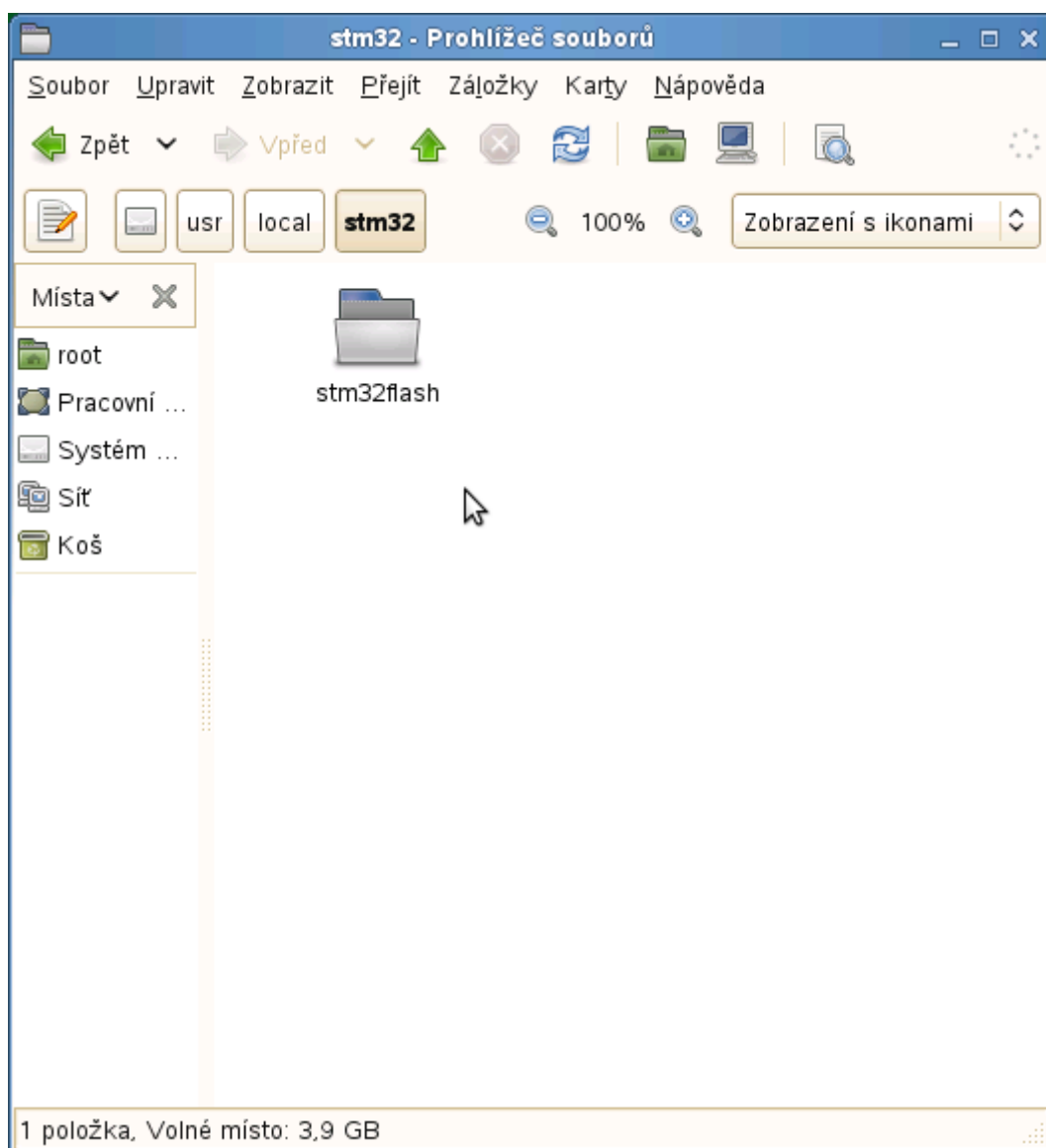
c:\stm32flash*.*				
Název	↑ Příp	Velikost	Datum	Atribut
[.]		<DIR>	26.10.2012 12:19	—
[parsers]		<DIR>	26.10.2012 12:19	—
[stm32]		<DIR>	26.10.2012 12:19	—
Makefile		357	04.05.2011 12:06	-a-
main	c	10 537	26.05.2011 06:48	-a-
serial_common	c	2 539	04.05.2011 12:06	-a-
serial_platform	c	86	04.05.2011 12:06	-a-
serial_posix	c	5 597	04.05.2011 12:06	-a-
serial_w32	c	6 133	26.05.2011 06:57	-a-
stm32	c	8 865	26.05.2011 06:49	-a-
utils	c	1 174	04.05.2011 12:06	-a-
parser	h	2 032	04.05.2011 12:06	-a-
serial	h	2 540	04.05.2011 12:06	-a-
stm32	h	1 965	26.05.2011 06:49	-a-
utils	h	946	04.05.2011 12:06	-a-
protocol	pdf	606 186	04.05.2011 12:06	-a-
gpl-2.0	txt	18 092	04.05.2011 12:06	-a-

To může být překvapením pro uživatele windows zvyklé na instalační soubory msi, exe apod. U linuxu totiž často dostaneme jen zdrojový kód programu a jeho binární verzi, potřebnou ke spouštění programu si musíme vytvořit sami. Obvykle jde o překlad jednotlivých souborů zdrojového kódu free překladačem jazyka C **gcc**, slinkováním takto vzniklých object souborů, knihoven apod. Naštěstí na našem linuxu již obvykle máme vše potřebné

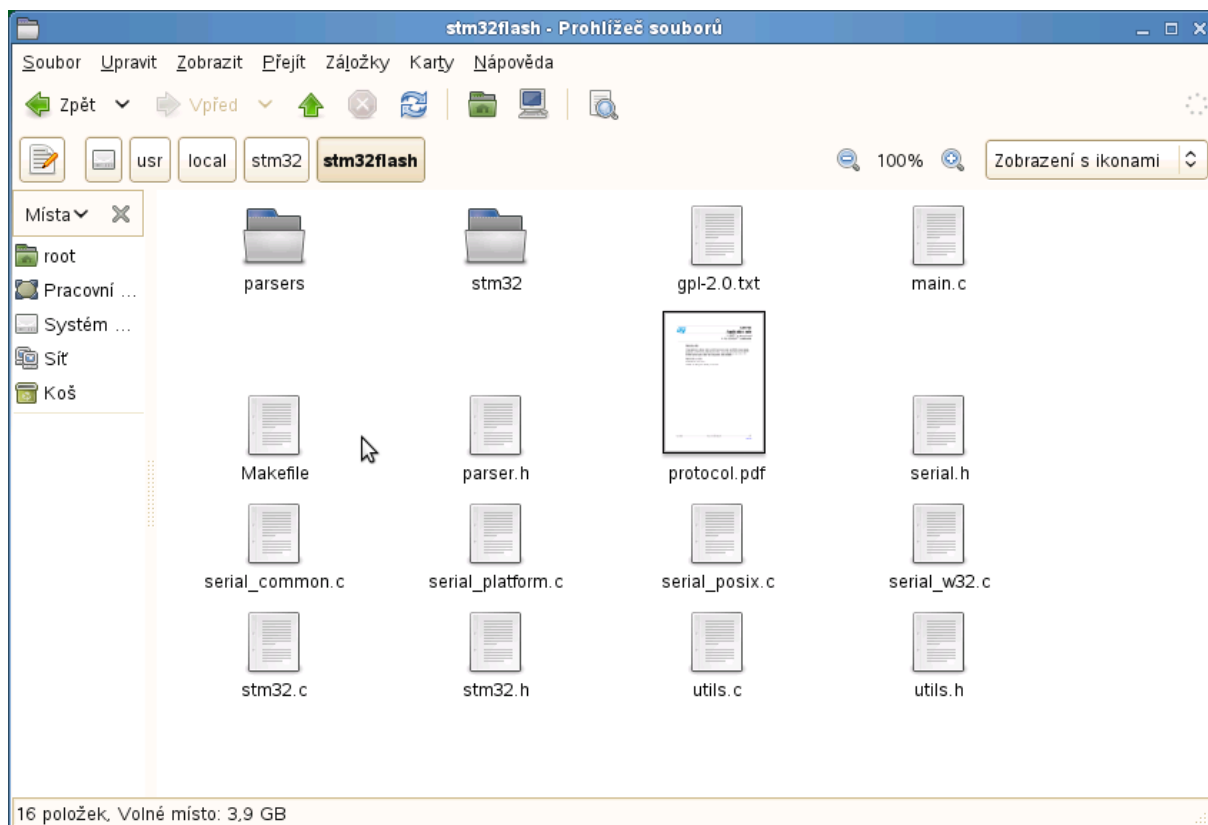
nainstalované a hlavně, jak je vidět na obrázku výše, kromě zdrojového kódu jsme k instalaci dostali i soubor **Makefile**, obsahující potřebné informace k překladu a sestavení výsledného binárního kódu. Soubor **Makefile** totiž slouží jako podklad pro program **make**, který tuto činnost provede za nás. Na nás zbývá v linuxu, v režimu příkazového řádku (terminálu) v adresáři, ve kterém máme umístěn soubor **Makefile** i soubory se zdrojovými kódy spustit příkaz

Make install

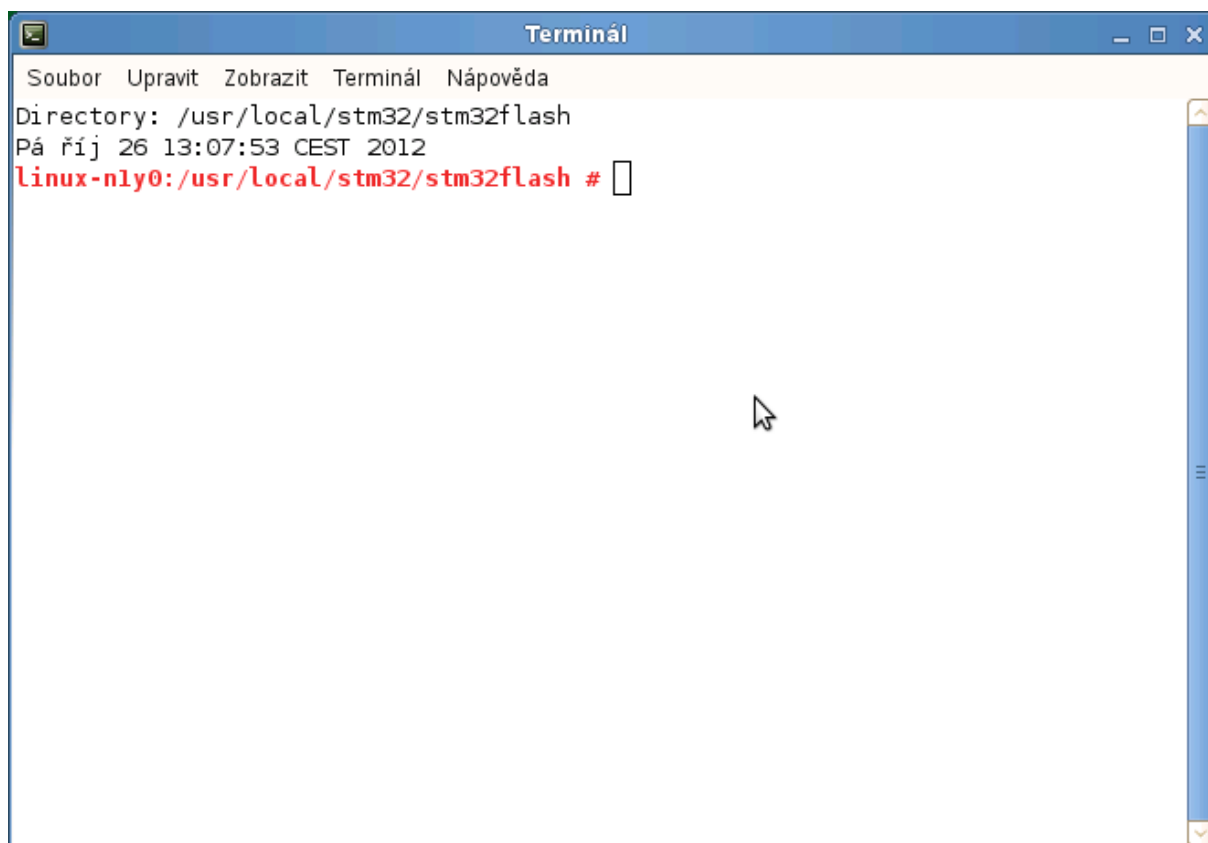
Ukážeme si to nyní na konkrétním případě. Pod OS linux vytvoříme v **usr/local** adresář **stm32** a do něj zkopírujeme adresář **stm32flash** i s jeho obsahem:



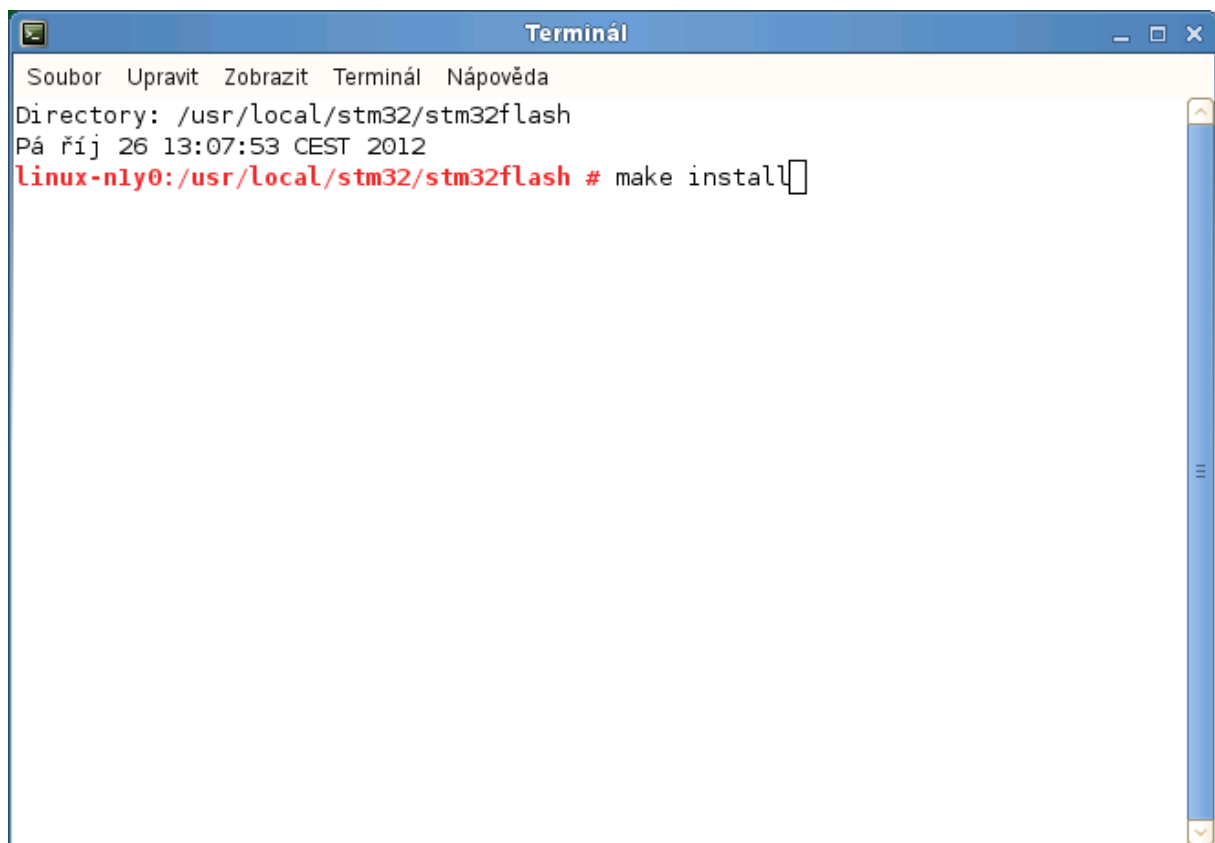
Obsah adresáře **stm32flash** je



Kliknutím pravým tlačítkem myši na ploše **Prohlížeče souborů** spustíme lokální menu. V něm vybereme položku **Terminál** a jejím výběrem ho spustíme

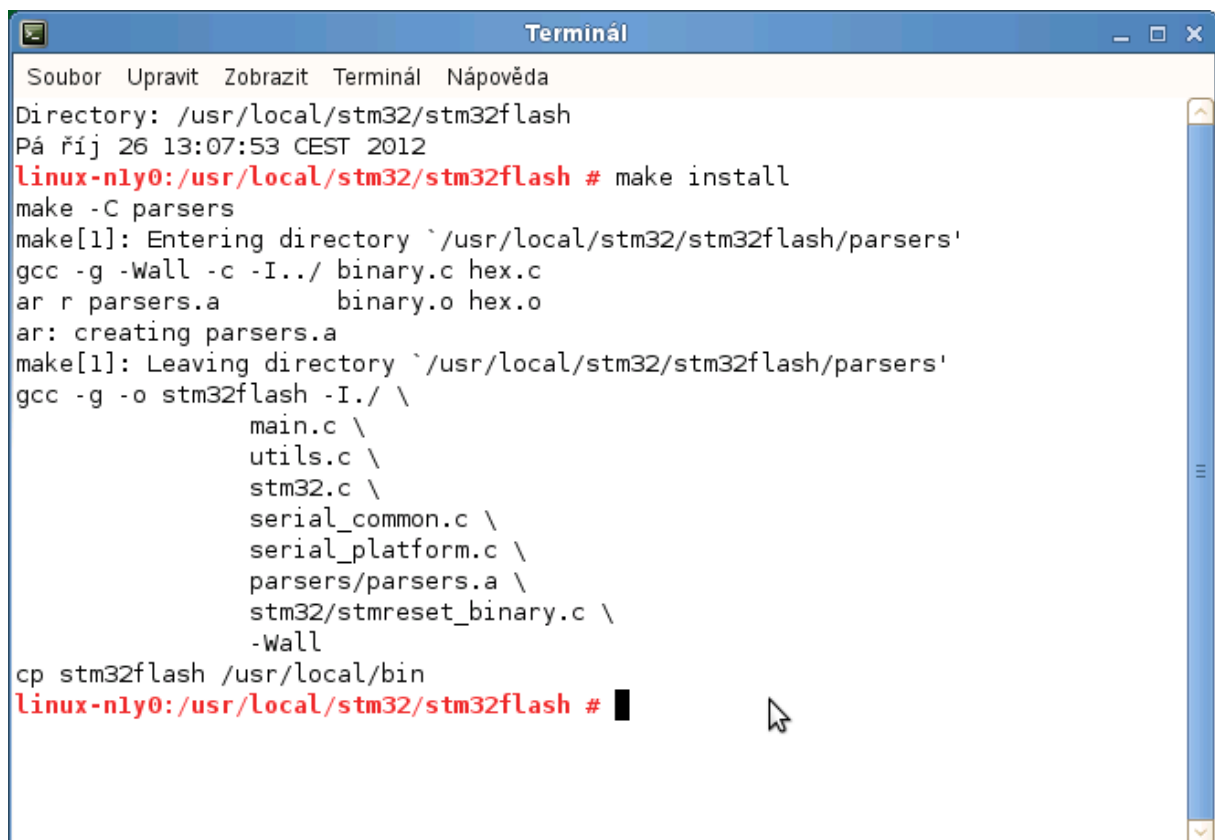


Nyní napíšeme **make install**



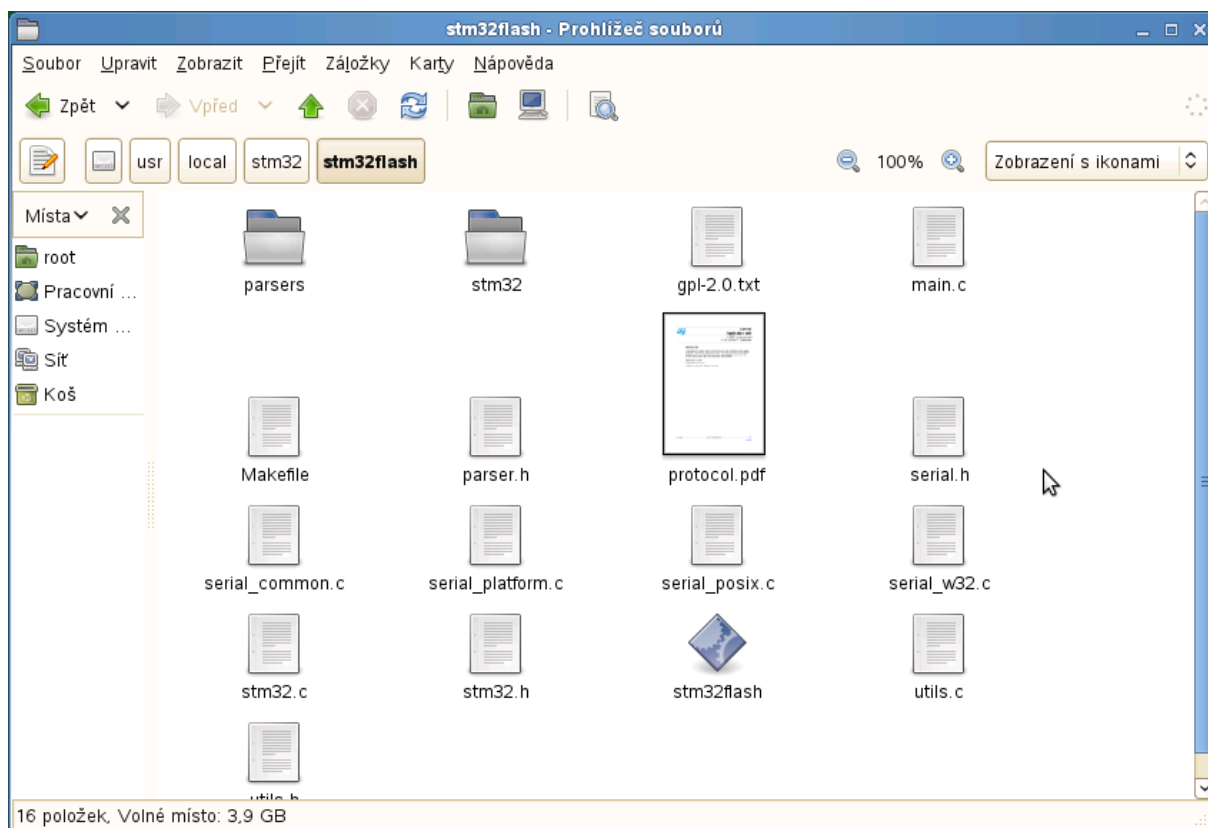
```
Terminál
Soubor Upravit Zobrazit Terminál Nápověda
Directory: /usr/local/stm32/stm32flash
Pá říj 26 13:07:53 CEST 2012
linux-nly0:/usr/local/stm32/stm32flash # make install
```

A tento příkaz spustíme, po chvíli dostaneme

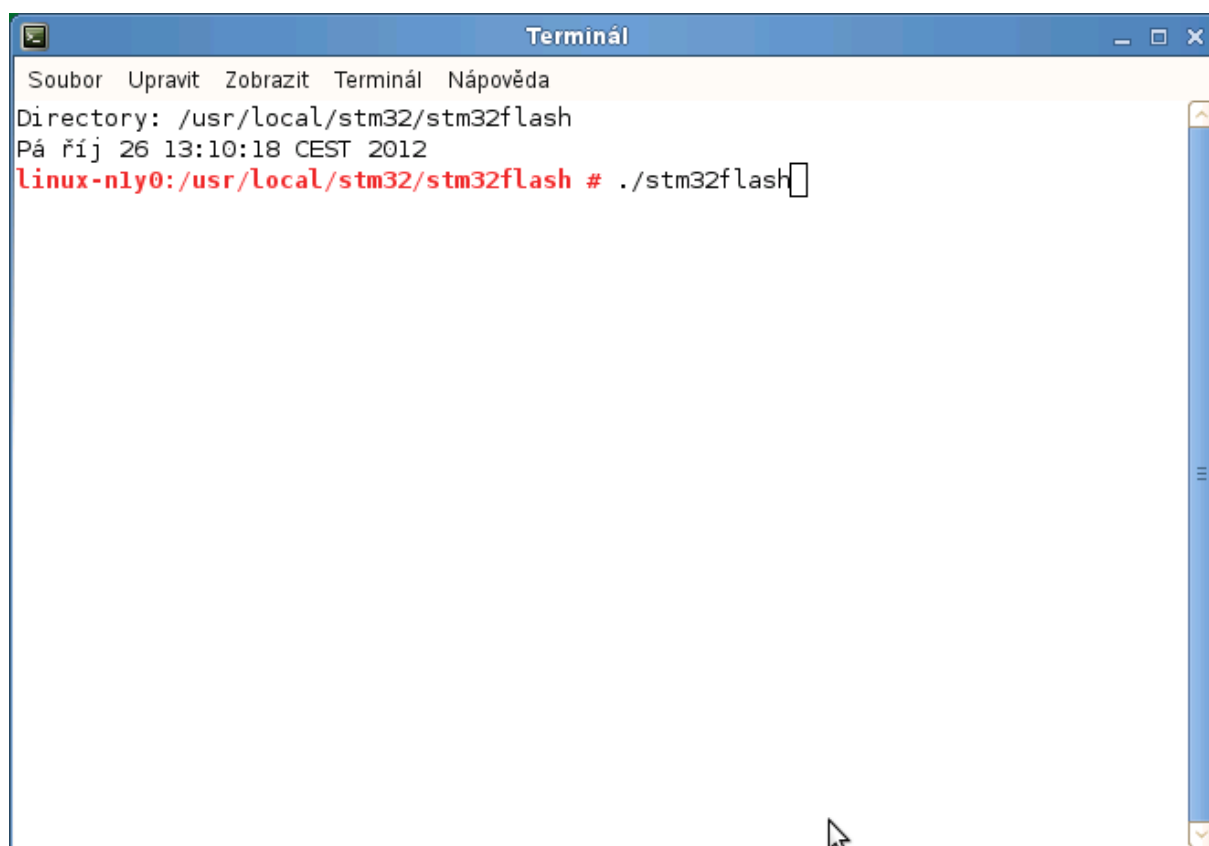


```
Terminál
Soubor Upravit Zobrazit Terminál Nápověda
Directory: /usr/local/stm32/stm32flash
Pá říj 26 13:07:53 CEST 2012
linux-nly0:/usr/local/stm32/stm32flash # make install
make -C parsers
make[1]: Entering directory `/usr/local/stm32/stm32flash/parsers'
gcc -g -Wall -c -I../ binary.c hex.c
ar r parsers.a          binary.o hex.o
ar: creating parsers.a
make[1]: Leaving directory `/usr/local/stm32/stm32flash/parsers'
gcc -g -o stm32flash -I./ \
    main.c \
    utils.c \
    stm32.c \
    serial_common.c \
    serial_platform.c \
    parsers/parsers.a \
    stm32/stmreset_binary.c \
    -Wall
cp stm32flash /usr/local/bin
linux-nly0:/usr/local/stm32/stm32flash #
```


Tím je spustitelný program **stm32flash** vytvořen. Všimněme si jeho ikonky



Tento program můžeme spustit. Opět spustíme **Terminál** a napíšeme **./stm32flash**



The image shows a terminal window with a blue title bar labeled "Terminál". The menu bar contains "Soubor", "Upravit", "Zobrazit", "Terminál", and "Nápověda". The terminal output shows the current directory as "/usr/local/stm32/stm32flash", the date and time as "Pá říj 26 13:10:18 CEST 2012", and the command prompt "linux-nly0:/usr/local/stm32/stm32flash # ./stm32flash" with a cursor at the end. A mouse cursor is visible at the bottom right of the terminal area.

```
Terminál
Soubor Upravit Zobrazit Terminál Nápověda
Directory: /usr/local/stm32/stm32flash
Pá říj 26 13:10:18 CEST 2012
linux-nly0:/usr/local/stm32/stm32flash # ./stm32flash
```

Program spustíme, dostaneme

```
Terminál
Soubor  Upravit  Zobrazit  Terminál  Nápověda
Directory: /usr/local/stm32/stm32flash
Pá říj 26 13:10:18 CEST 2012
linux-nly0:/usr/local/stm32/stm32flash # ./stm32flash
stm32flash - http://stm32flash.googlecode.com/

ERROR: Device not specified
Usage: ./stm32flash [-bvngfhc] [-[rw] filename] /dev/ttyS0
    -b rate           Baud rate (default 57600)
    -r filename       Read flash to file
    -w filename       Write flash to file
    -u               Disable the flash write-protection
    -e n              Only erase n pages before writing the flash
    -v               Verify writes
    -n count          Retry failed writes up to count times (default 10)
    -g address        Start execution at specified address (0 = flash start)
    -f               Force binary parser
    -h               Show this help
    -c               Resume the connection (don't send initial INIT)
                    *Baud rate must be kept the same as the first init*
                    This is useful if the reset fails

Examples:
    Get device information:
        ./stm32flash /dev/ttyS0

    Write with verify and then start execution:
        ./stm32flash -w filename -v -g 0x0 /dev/ttyS0

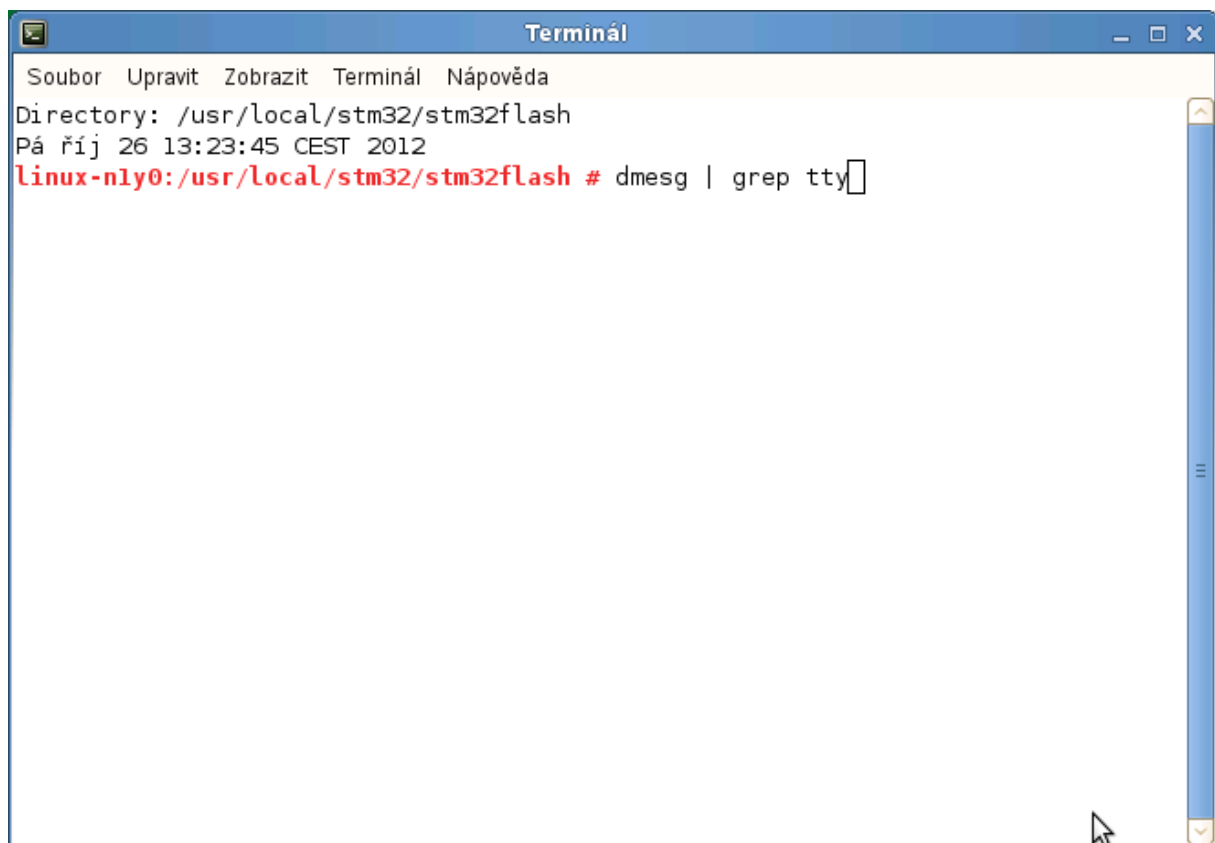
    Read flash to file:
        ./stm32flash -r filename /dev/ttyS0

    Start execution:
        ./stm32flash -g 0x0 /dev/ttyS0

linux-nly0:/usr/local/stm32/stm32flash #
```

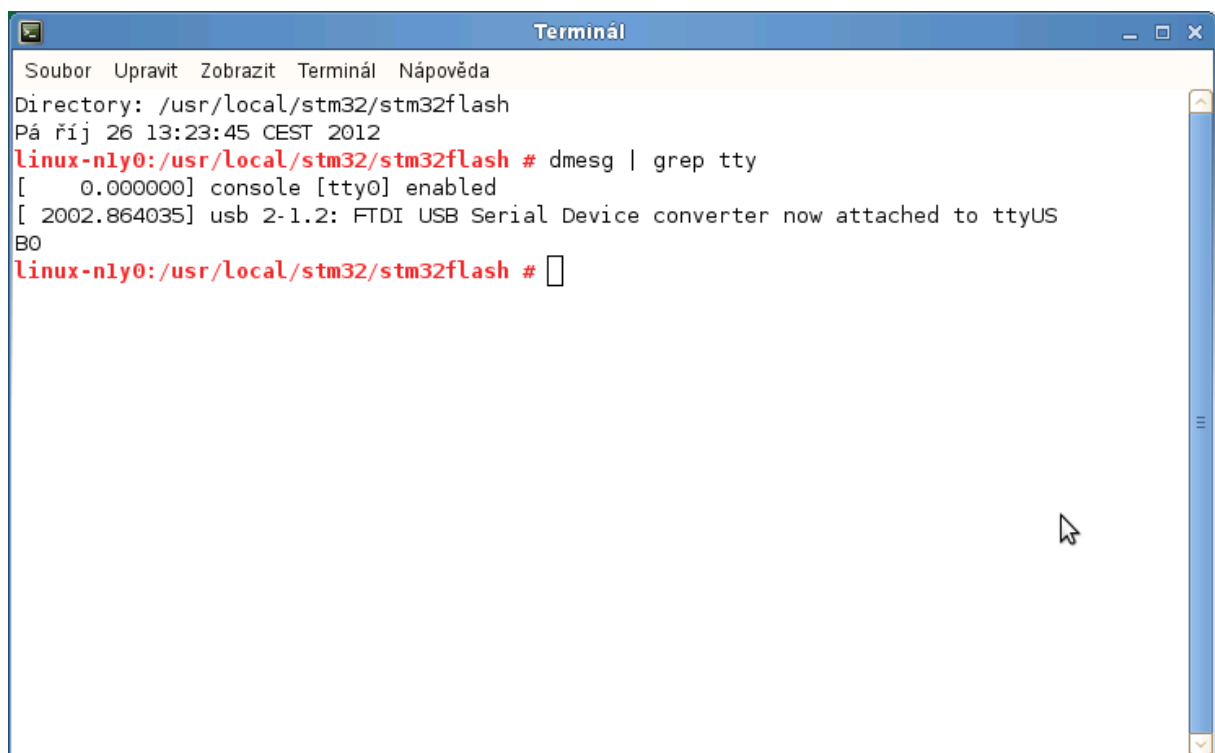
Vypsal se nám help ukazující používání programu **stm32flash**

K usb PC připojíme převodník FT232R. Převědčíme se, že ho náš linux zná. V terminálu napíšeme



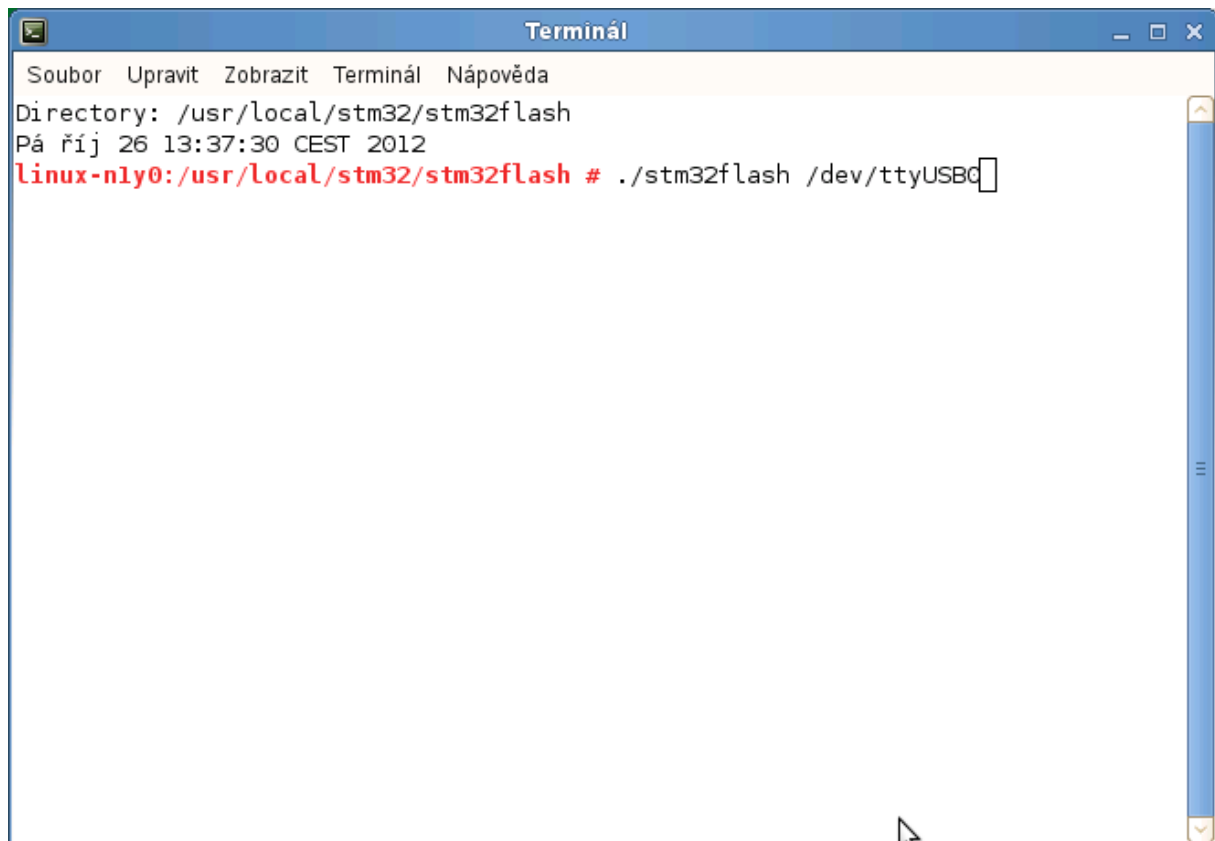
```
Terminál
Soubor Upravit Zobrazit Terminál Nápověda
Directory: /usr/local/stm32/stm32flash
Pá říj 26 13:23:45 CEST 2012
linux-nly0:/usr/local/stm32/stm32flash # dmesg | grep tty
```

Dostaneme



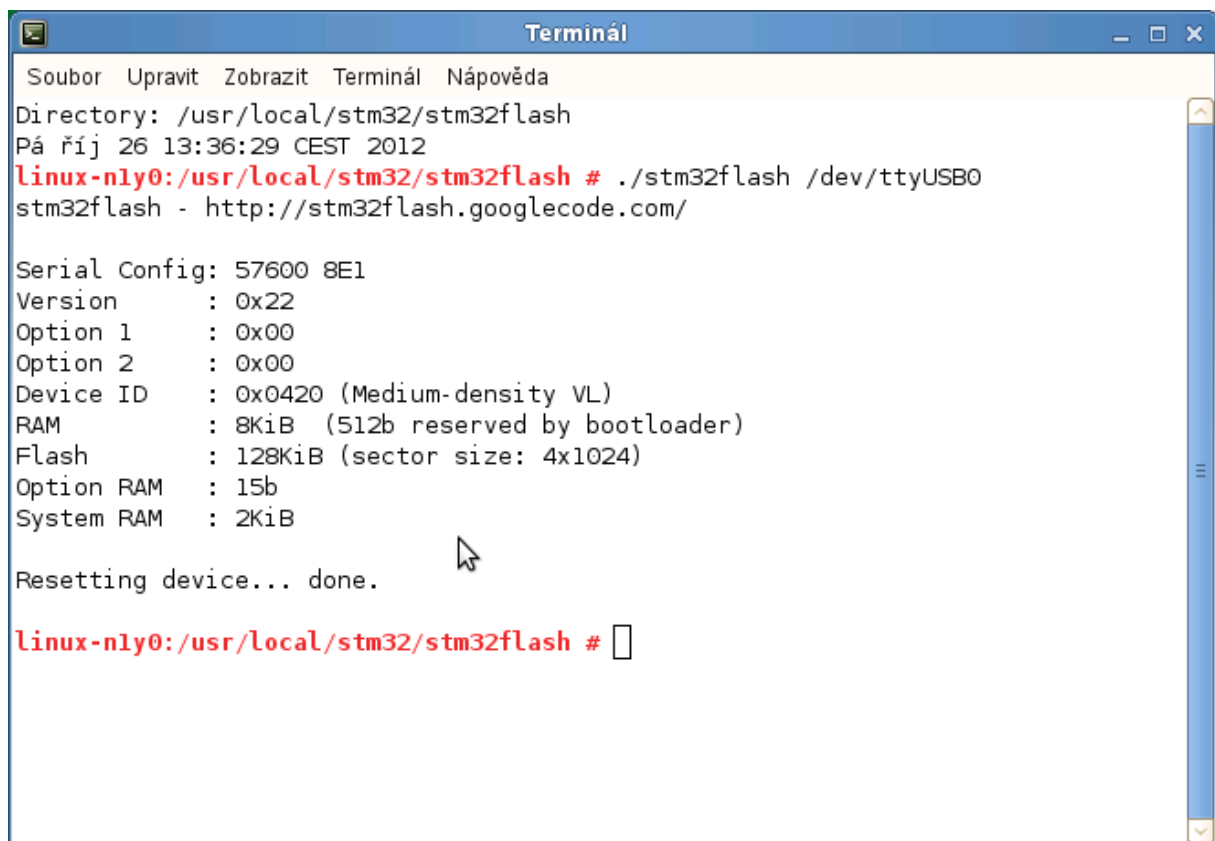
```
Terminál
Soubor Upravit Zobrazit Terminál Nápověda
Directory: /usr/local/stm32/stm32flash
Pá říj 26 13:23:45 CEST 2012
linux-nly0:/usr/local/stm32/stm32flash # dmesg | grep tty
[ 0.000000] console [tty0] enabled
[ 2002.864035] usb 2-1.2: FTDI USB Serial Device converter now attached to ttyUSB0
linux-nly0:/usr/local/stm32/stm32flash #
```

Vidíme, že je označen jako **ttyUSB0**. K tomuto převodníku připojíme onboard počítač s **STM32F100RBT6**. V terminálu napíšeme



```
Terminál
Soubor Upravit Zobrazit Terminál Nápověda
Directory: /usr/local/stm32/stm32flash
Pá říj 26 13:37:30 CEST 2012
linux-nly0:/usr/local/stm32/stm32flash # ./stm32flash /dev/ttyUSB0
```

Na **STM32F100RBT6** připojíme logickou 1 na **boot0** a zresetujeme. Poté již příkaz **./stm32flash /dev/ttyUSB0** spustíme. Dostaneme



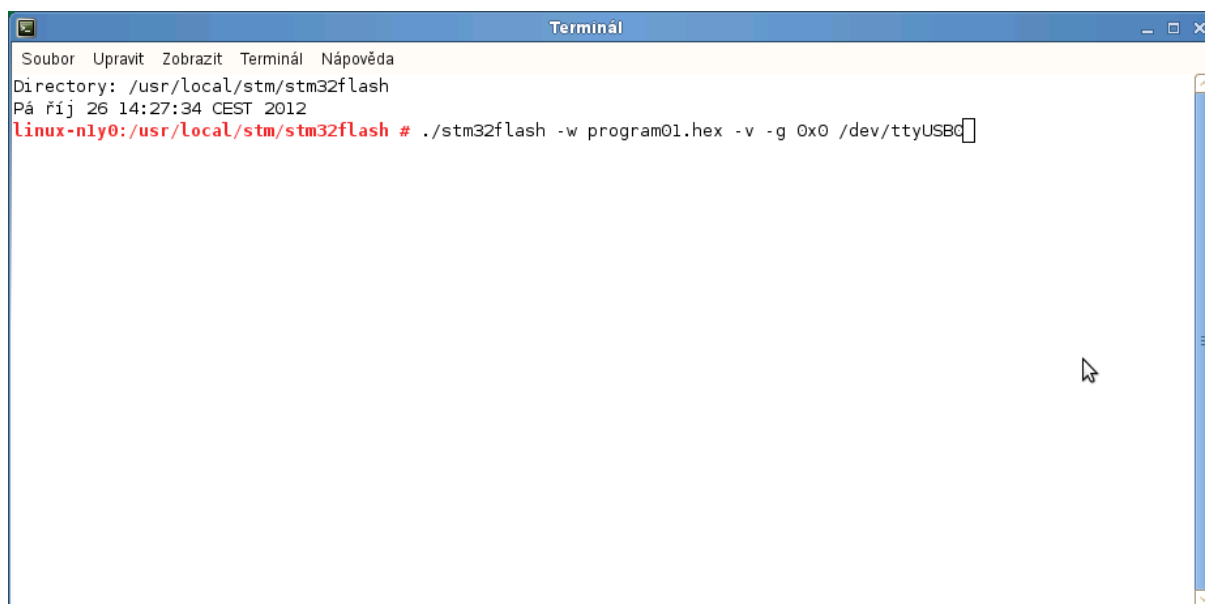
```
Terminál
Soubor Upravit Zobrazit Terminál Nápověda
Directory: /usr/local/stm32/stm32flash
Pá říj 26 13:36:29 CEST 2012
linux-nly0:/usr/local/stm32/stm32flash # ./stm32flash /dev/ttyUSB0
stm32flash - http://stm32flash.googlecode.com/

Serial Config: 57600 8E1
Version       : 0x22
Option 1      : 0x00
Option 2      : 0x00
Device ID     : 0x0420 (Medium-density VL)
RAM           : 8KiB (512b reserved by bootloader)
Flash         : 128KiB (sector size: 4x1024)
Option RAM    : 15b
System RAM    : 2KiB

Resetting device... done.

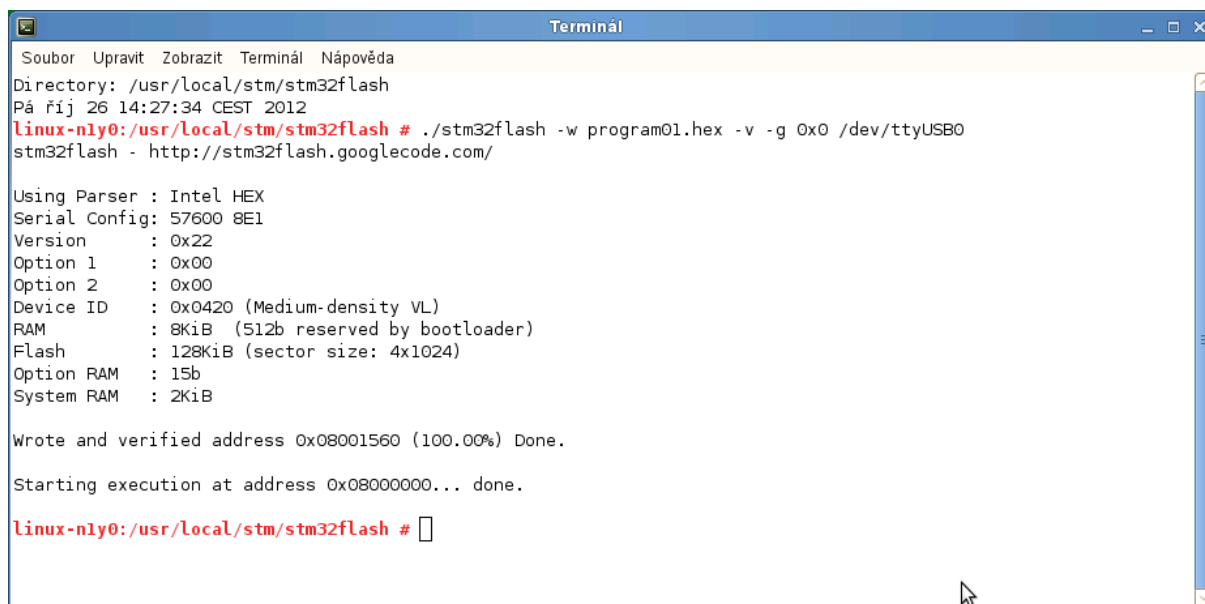
linux-nly0:/usr/local/stm32/stm32flash #
```

Pozn. Device ID 0x0420 značí STM32F100RBT6. Do adresáře se stm32flash programem umístíme soubor program01.hex který jsme získali pomocí vývojového prostředí, např. **Keil uVision4**. V terminálu pak napíšeme



```
Terminál
Soubor Upravit Zobrazit Terminál Nápověda
Directory: /usr/local/stm/stm32flash
Pá říj 26 14:27:34 CEST 2012
linux-nly0:/usr/local/stm/stm32flash # ./stm32flash -w program01.hex -v -g 0x0 /dev/ttyUSB0
```

Než příkaz **./stm32flash -w program01.hex -v -g 0x0 /dev/ttzUSB0** spustíme, nezapomeneme na **boot0** dát úroveň 1 a MCU zresetovat. Po provedení tohoto příkazu máme

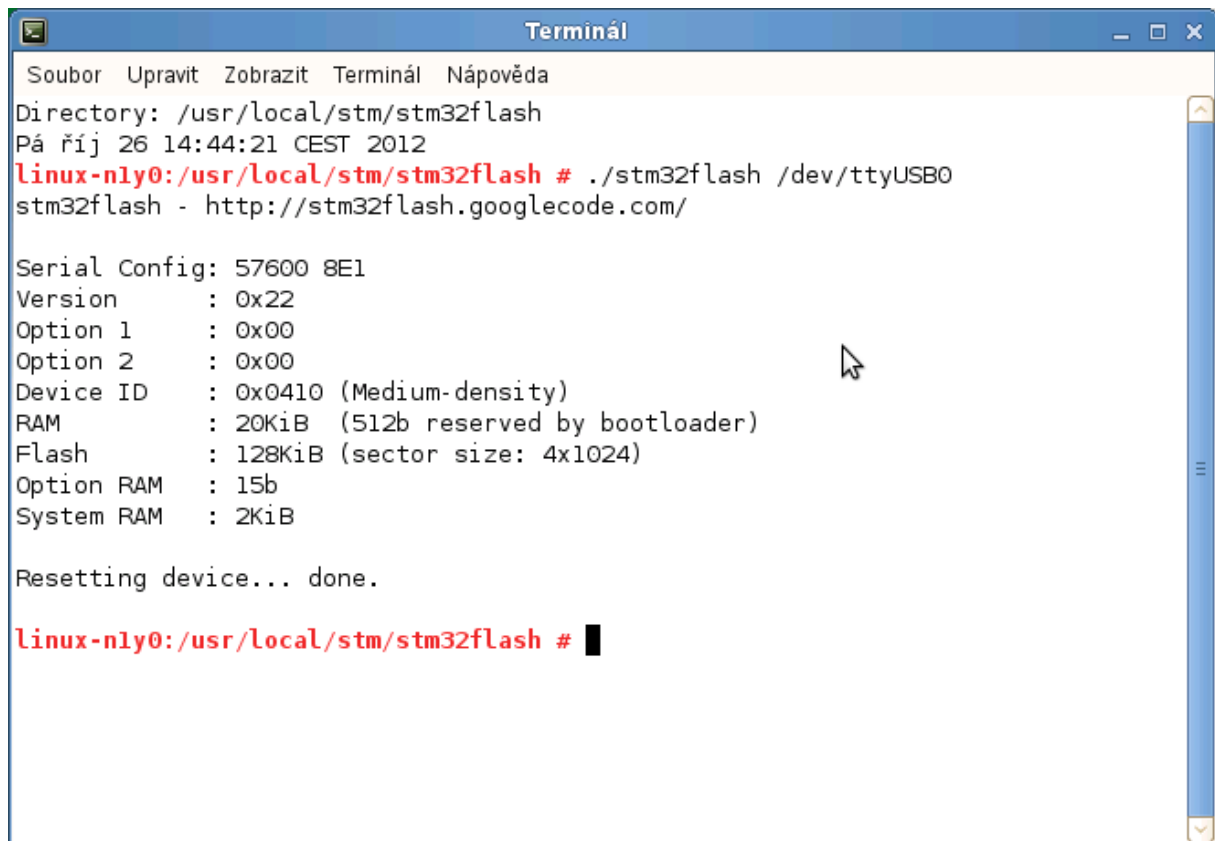


```
Terminál
Soubor Upravit Zobrazit Terminál Nápověda
Directory: /usr/local/stm/stm32flash
Pá říj 26 14:27:34 CEST 2012
linux-nly0:/usr/local/stm/stm32flash # ./stm32flash -w program01.hex -v -g 0x0 /dev/ttyUSB0
stm32flash - http://stm32flash.googlecode.com/

Using Parser : Intel HEX
Serial Config: 57600 8E1
Version      : 0x22
Option 1     : 0x00
Option 2     : 0x00
Device ID    : 0x0420 (Medium-density VL)
RAM          : 8KiB (512b reserved by bootloader)
Flash       : 128KiB (sector size: 4x1024)
Option RAM   : 15b
System RAM   : 2KiB

Wrote and verified address 0x08001560 (100.00%) Done.
Starting execution at address 0x08000000... done.
linux-nly0:/usr/local/stm/stm32flash #
```

Obdobně bychom postupovali i u MCU STM32F103C8T6. Vidíme, že jeho ID je 0x0410



A terminal window titled "Terminál" with a menu bar containing "Soubor", "Upravit", "Zobrazit", "Terminál", and "Nápověda". The directory is `/usr/local/stm/stm32flash` and the date is "Pá říj 26 14:44:21 CEST 2012". The user `linux-nly0` runs the command `./stm32flash /dev/ttyUSB0`. The output shows the STM32Flash version 0x22, device ID 0x0410 (Medium-density), 20KiB RAM, and 128KiB Flash. The device is successfully reset.

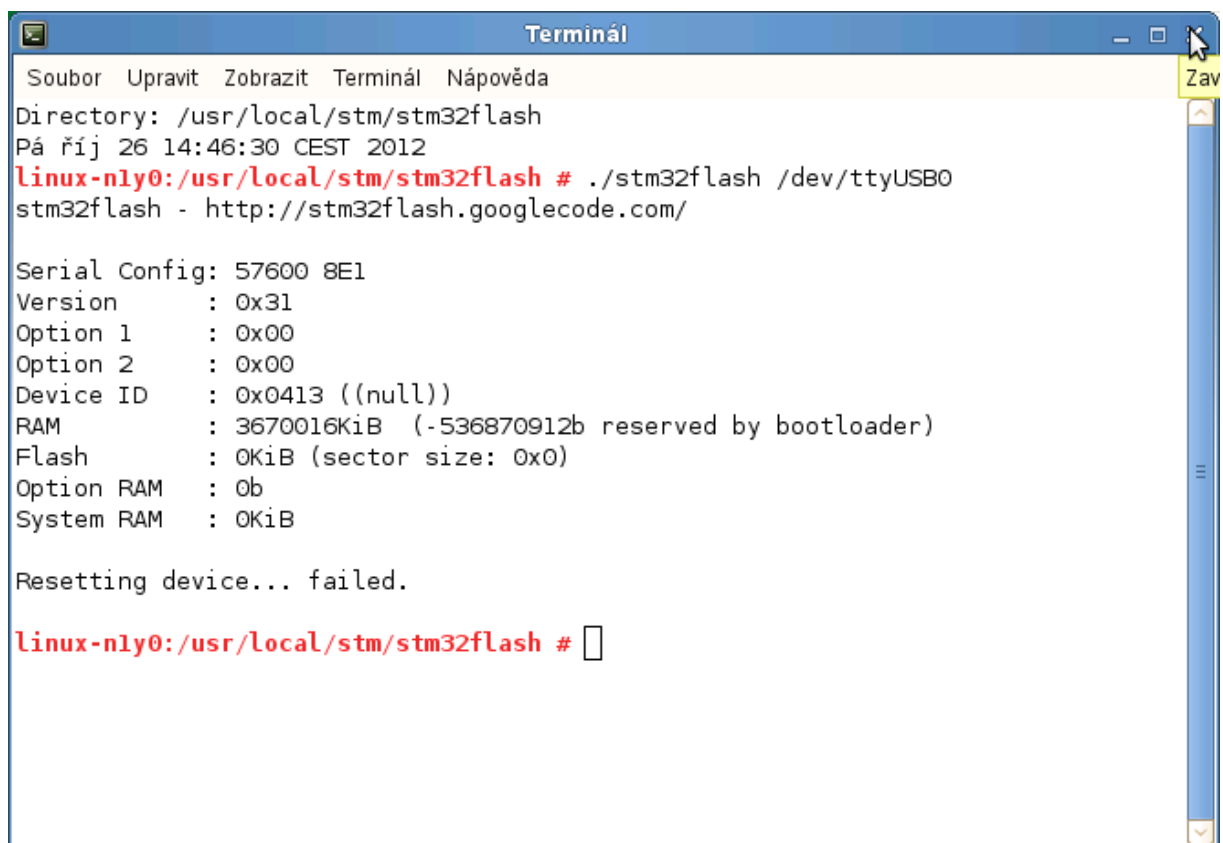
```
Soubor Upravit Zobrazit Terminál Nápověda
Directory: /usr/local/stm/stm32flash
Pá říj 26 14:44:21 CEST 2012
linux-nly0:/usr/local/stm/stm32flash # ./stm32flash /dev/ttyUSB0
stm32flash - http://stm32flash.googlecode.com/

Serial Config: 57600 8E1
Version      : 0x22
Option 1     : 0x00
Option 2     : 0x00
Device ID    : 0x0410 (Medium-density)
RAM          : 20KiB (512b reserved by bootloader)
Flash       : 128KiB (sector size: 4x1024)
Option RAM   : 15b
System RAM   : 2KiB

Resetting device... done.

linux-nly0:/usr/local/stm/stm32flash #
```

V případě STM32F405RGT6 však dostaneme:



A terminal window titled "Terminál" with a menu bar containing "Soubor", "Upravit", "Zobrazit", "Terminál", and "Nápověda". The directory is `/usr/local/stm/stm32flash` and the date is "Pá říj 26 14:46:30 CEST 2012". The user `linux-nly0` runs the command `./stm32flash /dev/ttyUSB0`. The output shows the STM32Flash version 0x31, device ID 0x0413 ((null)), 3670016KiB RAM, and 0KiB Flash. The device reset fails.

```
Soubor Upravit Zobrazit Terminál Nápověda
Directory: /usr/local/stm/stm32flash
Pá říj 26 14:46:30 CEST 2012
linux-nly0:/usr/local/stm/stm32flash # ./stm32flash /dev/ttyUSB0
stm32flash - http://stm32flash.googlecode.com/

Serial Config: 57600 8E1
Version      : 0x31
Option 1     : 0x00
Option 2     : 0x00
Device ID    : 0x0413 ((null))
RAM          : 3670016KiB (-536870912b reserved by bootloader)
Flash       : 0KiB (sector size: 0x0)
Option RAM   : 0b
System RAM   : 0KiB

Resetting device... failed.

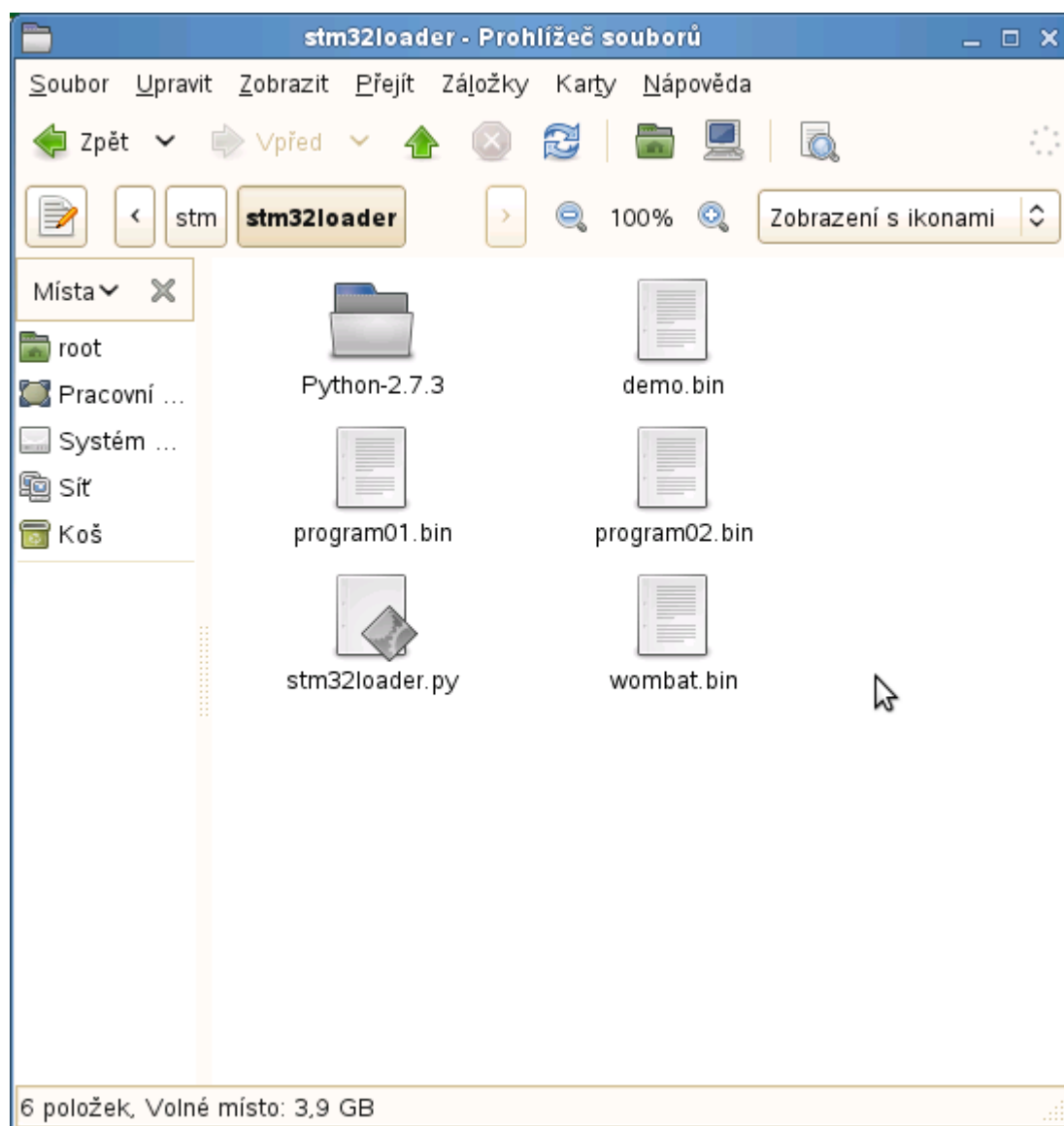
linux-nly0:/usr/local/stm/stm32flash #
```

Vidíme žeID tohoto obvodu je 0x0413. Kromě toho je zřejmé, že pro STM32F405RGT6 se **stm32flash** použít nedá. Tento obvod totiž ještě nepodporuje. Vzhledem k tomu, že k programu **stm32flash** máme zdrojové kódy, snadno najdeme v souboru **stm32.c** i tento kus kódu

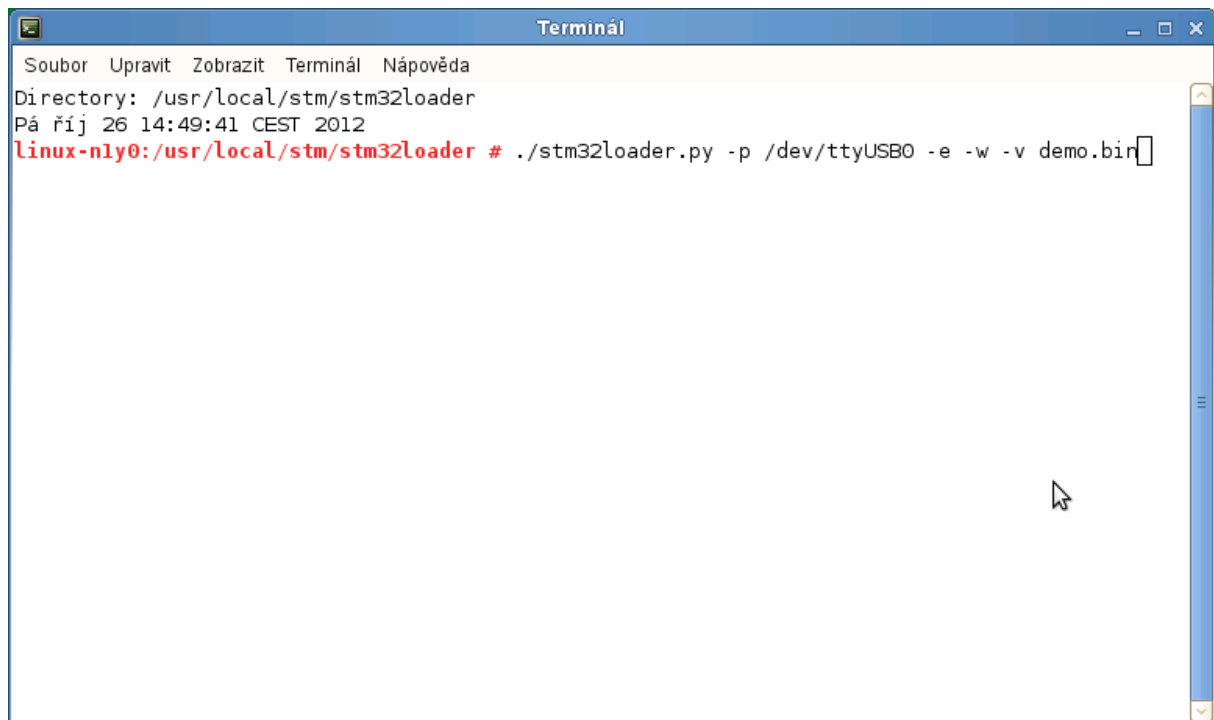
```
/* device table */
const stm32_dev_t devices[] = {
    {0x412, "Low-density", 0x20000200, 0x20002800, 0x08000000,
0x08008000, 4, 1024, 0x1FFFF800, 0x1FFFF80F, 0x1FFFF000, 0x1FFFF800},
    {0x410, "Medium-density", 0x20000200, 0x20005000, 0x08000000,
0x08020000, 4, 1024, 0x1FFFF800, 0x1FFFF80F, 0x1FFFF000, 0x1FFFF800},
    {0x414, "High-density", 0x20000200, 0x20010000, 0x08000000,
0x08080000, 2, 2048, 0x1FFFF800, 0x1FFFF80F, 0x1FFFF000, 0x1FFFF800},
    {0x418, "Connectivity line", 0x20001000, 0x20010000, 0x08000000,
0x08040000, 2, 2048, 0x1FFFF800, 0x1FFFF80F, 0x1FFFB000, 0x1FFFF800},
    {0x420, "Medium-density VL", 0x20000200, 0x20002000, 0x08000000,
0x08020000, 4, 1024, 0x1FFFF800, 0x1FFFF80F, 0x1FFFF000, 0x1FFFF800},
    {0x430, "XL-density", 0x20000800, 0x20018000, 0x08000000,
0x08100000, 2, 2048, 0x1FFFF800, 0x1FFFF80F, 0x1FFFE000, 0x1FFFF800},
    {0x0}
};
```

Vidíme, že podporuje jen MCU s ID 0x0412, 0x0410, 0x0414, 0x0418, 0x0420 a 0x0430 . Můžeme si tuto tabulku doplnit a program stm32flash znovu přeložit. Místo stm32flash můžeme použít jiný program, který i STM32F405RGT6 podporuje. Takovým programem je např. stm32loader. Rovněž tento program je free. Protože je napsán v jazyce Python, musíme mít Python v Linuxu nainstalovaný.

Tento jazyk je často používán pro tvorbu sw pro linux pro práci s jednočipovými počítači. Proto ho mám na svém PC s Linuxem již nainstalovaný. Zbývá proto jen z <https://github.com/jsnyder/stm32loader> stáhnout soubor **stm32loader.py** . Tento soubor umístíme do nějakého adresáře spolu s binárním(i) souborem (soubory), které chceme naprogramovat do paměti flash. (Protože **stm32flash.py** bude komunikovat přes usb, musíme mít ovšem nainstalovanou i knihovnu PySerial.)

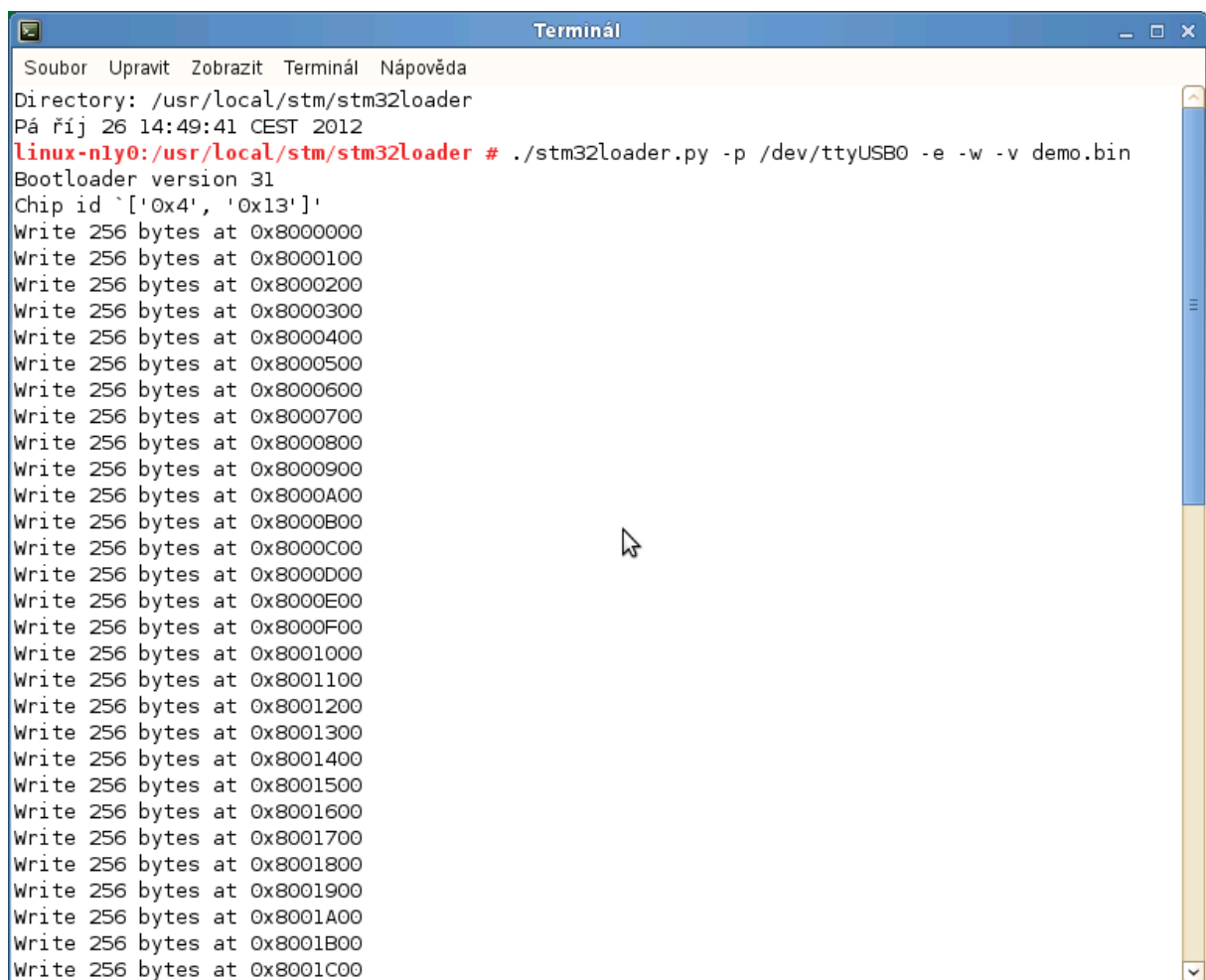


V Terminálu napíšeme



```
Terminál
Soubor Upravit Zobrazit Terminál Nápověda
Directory: /usr/local/stm/stm32loader
Pá říj 26 14:49:41 CEST 2012
linux-nly0:/usr/local/stm/stm32loader # ./stm32loader.py -p /dev/ttyUSB0 -e -w -v demo.bin
```

A poté, co na **boot0** dáme úroveň logické 1 a MCU zresetujeme, spustíme. Dostaneme



```
Terminál
Soubor Upravit Zobrazit Terminál Nápověda
Directory: /usr/local/stm/stm32loader
Pá říj 26 14:49:41 CEST 2012
linux-nly0:/usr/local/stm/stm32loader # ./stm32loader.py -p /dev/ttyUSB0 -e -w -v demo.bin
Bootloader version 31
Chip id '['0x4', '0x13']'
Write 256 bytes at 0x8000000
Write 256 bytes at 0x8000100
Write 256 bytes at 0x8000200
Write 256 bytes at 0x8000300
Write 256 bytes at 0x8000400
Write 256 bytes at 0x8000500
Write 256 bytes at 0x8000600
Write 256 bytes at 0x8000700
Write 256 bytes at 0x8000800
Write 256 bytes at 0x8000900
Write 256 bytes at 0x8000A00
Write 256 bytes at 0x8000B00
Write 256 bytes at 0x8000C00
Write 256 bytes at 0x8000D00
Write 256 bytes at 0x8000E00
Write 256 bytes at 0x8000F00
Write 256 bytes at 0x8001000
Write 256 bytes at 0x8001100
Write 256 bytes at 0x8001200
Write 256 bytes at 0x8001300
Write 256 bytes at 0x8001400
Write 256 bytes at 0x8001500
Write 256 bytes at 0x8001600
Write 256 bytes at 0x8001700
Write 256 bytes at 0x8001800
Write 256 bytes at 0x8001900
Write 256 bytes at 0x8001A00
Write 256 bytes at 0x8001B00
Write 256 bytes at 0x8001C00
```

```
Terminál
Soubor  Upravit  Zobrazit  Terminál  Nápověda
Write 256 bytes at 0x8001E00
Write 256 bytes at 0x8001F00
Write 256 bytes at 0x8002000
Read 256 bytes at 0x8000000
Read 256 bytes at 0x8000100
Read 256 bytes at 0x8000200
Read 256 bytes at 0x8000300
Read 256 bytes at 0x8000400
Read 256 bytes at 0x8000500
Read 256 bytes at 0x8000600
Read 256 bytes at 0x8000700
Read 256 bytes at 0x8000800
Read 256 bytes at 0x8000900
Read 256 bytes at 0x8000A00
Read 256 bytes at 0x8000B00
Read 256 bytes at 0x8000C00
Read 256 bytes at 0x8000D00
Read 256 bytes at 0x8000E00
Read 256 bytes at 0x8000F00
Read 256 bytes at 0x8001000
Read 256 bytes at 0x8001100
Read 256 bytes at 0x8001200
Read 256 bytes at 0x8001300
Read 256 bytes at 0x8001400
Read 256 bytes at 0x8001500
Read 256 bytes at 0x8001600
Read 256 bytes at 0x8001700
Read 256 bytes at 0x8001800
Read 256 bytes at 0x8001900
Read 256 bytes at 0x8001A00
Read 256 bytes at 0x8001B00
Read 256 bytes at 0x8001C00
Read 256 bytes at 0x8001D00
Read 256 bytes at 0x8001E00
Read 256 bytes at 0x8001F00
Read 256 bytes at 0x8002000
Verification OK
linux-nly0:/usr/local/stm/stm32loader #
```

Vidíme, že naprogramování se podařilo