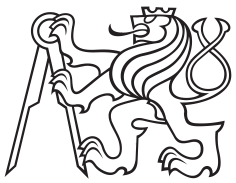
Dokumentácia



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra měření

Dodatkové materiály k bakalárskej práci

Lukáš Bielesch

Máj 2019
Vedúci práce: doc. Ing. Jan Fischer, CSc.

Obsah

Kapitola 1
Trieda Timeout prostredia mbed

Trieda Timeout slúži na vyvolanie jednorázového prerušenia po určitom časovom úseku.

1.1 Inicializácia

Konfiguráciu prevedieme vytvorením objektu triedy Timeout s nejakým názvom (napr. countdown). V prostredí Mbed je možné používať naraz neobmedzený počet objektov triedy Timeout.

```
Timeout countdown;
```

1.2 Priradenie funkcie obsluhy prerušenia

Funkcie *attach*() a *attach_us*() slúžia na spustenie odpočítavania požadovaného časového úseku, po ktorom sa vyvolá hardvérové prerušenie a na priradenie funkcie obsluhy prerušenia. Čas do vyvolania prerušenia sa odpočítava pomocou 32-bitového mikrosekundového čítača, teda je možné zadať hodnotu s rozlíšením na mikrosekundy až do

$$(2^{32} - 1)\mu s \approx 35\ min.$$

Funkcia obsluhy prerušenia nemá vstupné argumenty a nevracia žiadnu hodnotu. Nesmie obsahovať funkcie *wait_ms*(), *wait_us*(), *printf*() ani funkcie na alokáciu pamäti *new*() a *malloc*(), pretože by došlo k nedefinovanému chovaniu programu. Ak funkcia pristupuje ku globálnym premenným, tieto premenné by mali byť pri inicializácii označené kľúčovým slovom *volatile*, ktoré zabezpečí, že pri kompilácii programu nedôjde k optimalizácii použitia danej premennej a ku premennej bude možné pristupovať kedykoľvek počas behu programu.

1.3 attach

Prvým argumentom funkcie *attach*() je adresa funkcie, ktorá bude obsluhovať prerušenie. Druhým argumentom premená typu float, časový úsek v sekundách s rozlíšením na mikrosekundy, po ktorom sa prerušenie vyvolá.

```
void func(){
...
}
int main(){
...
countdown.attach(&func, 1.0);
...
}
```

1.4 attach us

Prvým argumentom funkcie *attach_us*() je adresa funkcie, ktorá bude obsluhovať prerušenie. Druhým argumentom premená typu uint32_t, časový úsek v mikrosekundách po ktorom sa prerušenie vyvolá.

```
void func(){
...
}
int main(){
    ...
    countdown.attach_us(&func, 1500);
    ...
}
```

1.5 detach

Funkcia *detach*() zruší aktuálne odpočítavanie časovača a tým zamedzí vyvolaniu hardvérového prerušenia.

```
countdown.detach();
```

Kapitola 2
Trieda Ticker prostredia mbed

Trieda Ticker slúži na previdelené vyvolávanie prerušenia v pravidelných intervaloch.

2.1 Inicializácia

Konfiguráciu prevedieme vytvorením objektu triedy Ticker s nejakým názvom (napr. flipper). V prostredí Mbed je možné používať naraz neobmedzený počet objektov triedy Ticker.

```
Ticker flipper;
```

2.2 Priradenie funkcie obsluhy prerušenia

Funkcie *attach*() a *attach_us*() slúžia na priradenie funkcie obsluhy prerušenia a na priradenie časového intervalu, v ktorom sa bude hardvérové prerušenie vyvolávať. Čas do vyvolania prerušenia sa odpočítava pomocou 32-bitového mikrosekundového čítača, teda je možné zadať hodnotu s rozlíšením na mikrosekundy až do

$$(2^{32} - 1)\mu s \approx 35\ min.$$

Funkcia obsluhy prerušenia nemá vstupné argumenty a nevracia žiadnu hodnotu. Nesmie obsahovať funkcie *wait_ms*(), *wait_us*(), *printf*() ani funkcie na alokáciu pamäti *new*() a *malloc*(), pretože by došlo k nedefinovanému chovaniu programu. Ak funkcia pristupuje ku globálnym premenným, tieto premenné by mali byť pri inicializácii označené kľúčovým slovom *volatile*, ktoré zabezpečí, že pri kompilácii programu nedôjde k optimalizácii použitia danej premennej a ku premennej bude možné pristupovať kedykoľvek počas behu programu.

2.3 attach

Prvým argumentom funkcie *attach*() je adresa funkcie, ktorá bude obsluhovať prerušenie. Druhým argumentom premená typu float, časový úsek v sekundách s rozlíšením na mikrosekundy, v ktorom sa prerušenie vyvoláva.

```
void func(){
    ...
}
int main(){
    ...
    flipper.attach(&func, 1.0);
    ...
}
```

2.4 attach us

Prvým argumentom funkcie *attach_us*() je adresa funkcie, ktorá bude obsluhovať prerušenie. Druhým argumentom premená typu uint32_t, časový úsek v mikrosekundách, v ktorom sa prerušenie vyvoláva.

```
void func(){
...
}
int main(){
...
flipper.attach_us(&func, 1500);
...
}
```

2.5 detach

Funkcia *detach*() zakáže hardvérové prerušenie.

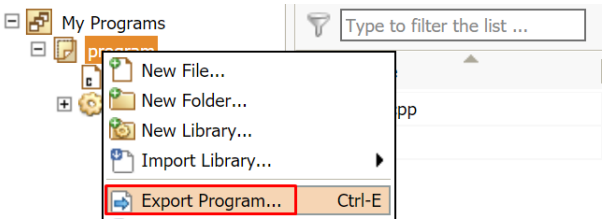
```
flipper.detach();
```

Kapitola 3
Export a debugovanie programu v prostredí µVision Keil 5 pre STM32F042F6P6

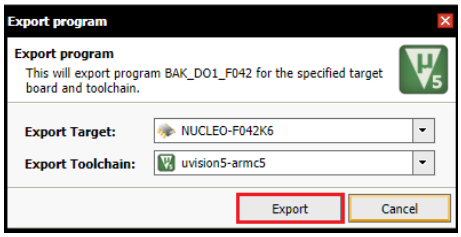
V prostredí uVision Keil 5 je možné bezplatne vytvárať programy iba do veľkosti 32kB, čo je postačujúce iba pre menšie procesory (napr. STM32F042F6P60). Na debugovanie mikrokontrolérov STM32 a STM8 sa používa ST-Link, ktorý sa nachádza priamo na vývojových doskách (napr. NUCLEO-F303RE), alebo je možné použiť samostatný ST-LINK/V2.

3.1 Export programu z mbedu

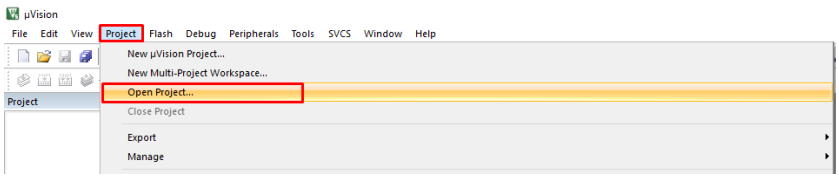
V prostredí Mbed exportujeme funkčný program pomocou klávesovej skratky *Ctrl-E* alebo po kliknutí pravím tlačidlom myši na zložku s programom zvolíme *Export Program...*



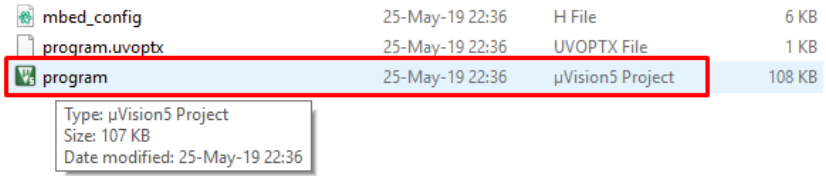
Platformu ponecháme NUCLEO-F042K6, ako cieľ zvolíme uVision5 a stiahneme ako *.zip* súbor.



Spustíme program uVision5 a stiahnutý projekt otvoríme tak, že v paneli nástrojov klikneme na *Project->Open Project*

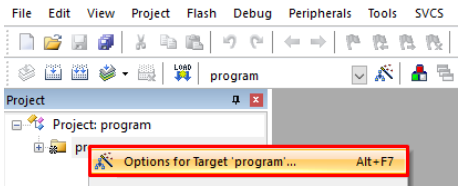


a zo zložky, ktorú sme stiahli, vyberieme súbor, ktorý je označený ako uVision5 Project.



3.2 Zmena nastavení v projekte

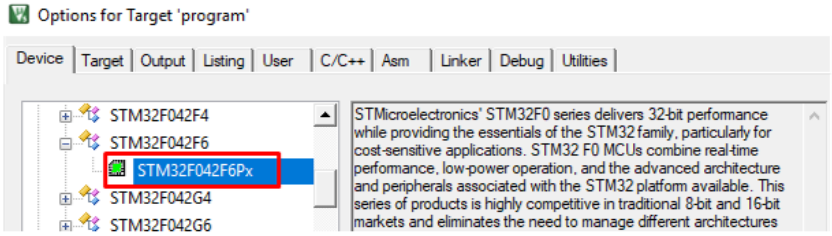
Nastavenia projektu je možné meniť v sekcii *Options for Target*, do ktorej sa dostaneme pomocou klávesovej skratky *Alt-F7* alebo po kliknutí pravým tlačidlom na náš projekt prejdeme na *Options for Target*.



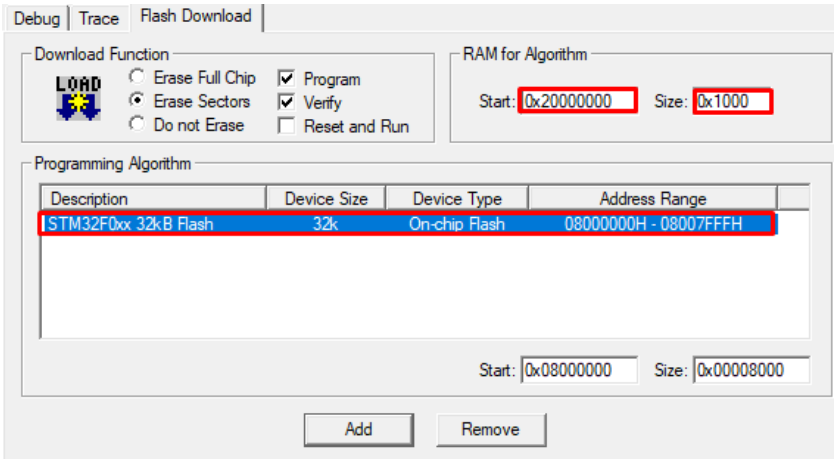
V prostredí Mbed, na rozdiel od uVision 5, nie je procesor STM32F042F6P6 podorovaný, preto sa program vytvára pre platformu STM32F042K6, ktorá je nášmu mikroprocesoru najbližšia. V nastaveniach projektu je potrebné zmeniť platformu na STM32F042F6Px a ďalšie nastavenia súvisiace s touto zmenou.

3.3 Zmena zariadenia

V nastaveniach projektu *Options for Target* v sekcii *Device* je potrebné zmeniť zariadenie na STM32F042F6Px.



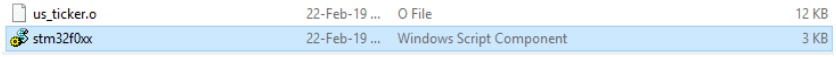
Ďalej v sekcii *Utilities->settings* zmeníme začiatok RAM for algorithm na $0x20000000$ a veľkosť na $0x1000$. Ak ako programovací algoritmus nie je vybratý *STM32F0xx 32kB Flash*, aktuálny algoritmus kliknutím na *Remove* vymažeme a pomocou *Add* pridáme algoritmus *STM32F0xx 32kB Flash*.



3.4 Zmena linkovacieho súbora

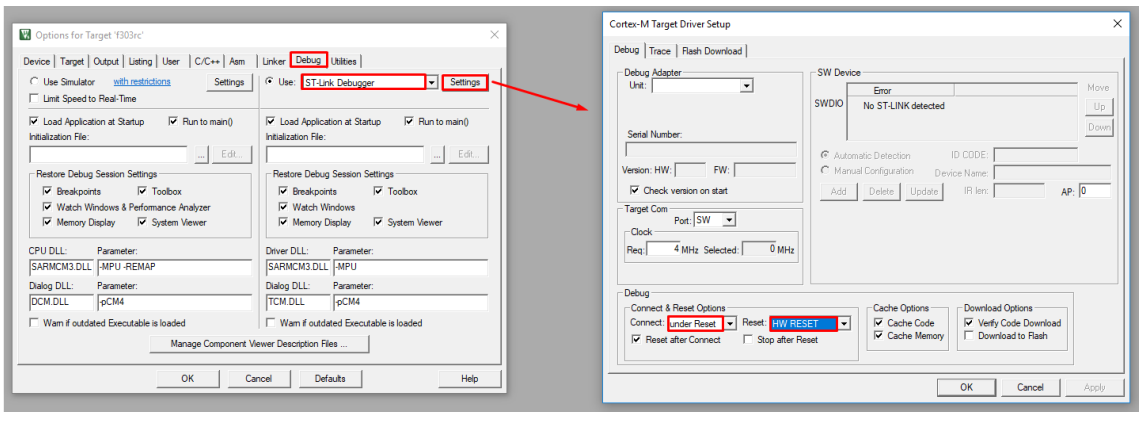
Procesor STM32F042F6P6 patrí do skupiny STM32F0. Každá skupina mikrokontrolérov má iný linkovací súbor, preto ak sme v prostredí uVision5 pracovali s mikrokontrolérom patriacim do inej skupiny (NUCLEO-F303RE patrí do STM32F3), je potrebné v sekcii Linker zmeniť Scatter file na *STM32F0xx.sct*

Cesta k tomuto súboru je nasledovná:
zložka_programu->mbed->TARGET_NUCLEO_F042K->
->TOOLCHAIN_ARM_MICRO->stm32f0xx.sct



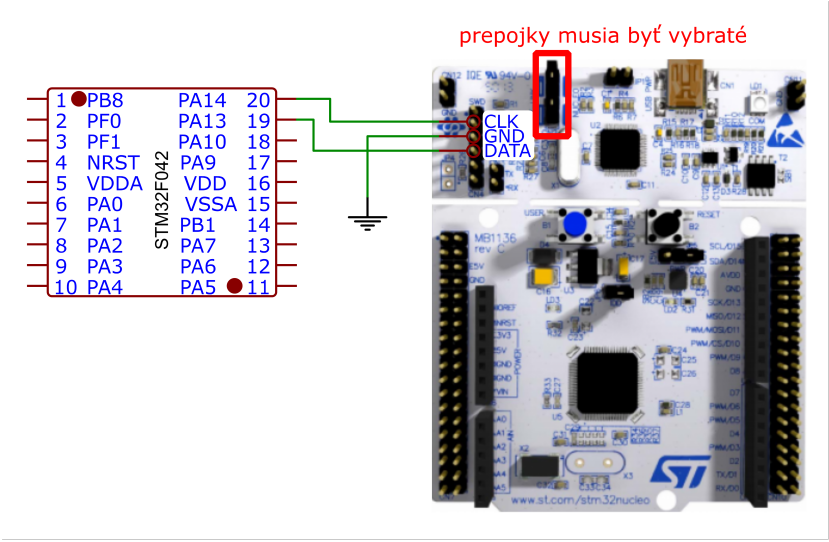
3.5 Zmena nastavení debuggera

Ak budeme program do dosky nahrávať a debugovať pomocou ST-Linku , musíme zmeniť aj nastavenia v sekcii Debugger. Ako použitý debugger vyberieme *ST-Link Debugger* a prejdeme do jeho nastavení. Tam zmeníme nastavenie pripojenia na *Under Reset* a nastavenie resetu na *HW RESET*.

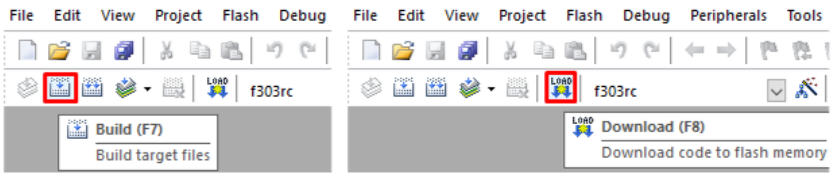


3.6 Nahratie programu cez ST-Link na doske NUCLEO-F303RE

Ak budeme program sťahovať do dosky a debugovať pomocou ST-Linku, musíme ST-Link prepojiť s mikrokontrolérom podľa nasledujúcej schémy. Najskôr prepojíme zem nášho mikroprocesoru so zemou ST-Linku, ktorá sa nachádza na vývode CN4, na treťom pine zhora. Potom prepojíme aj zvyšné 2 piny. Okrem toho je potrebné vybrať prepojky, ktoré slúžia na prepojenie ST-Linku s procesorom STM32F303RE.

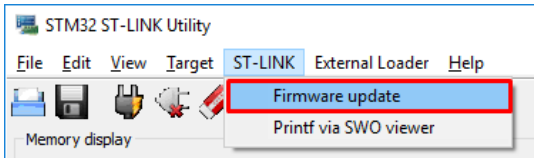


Hotový program skompilujeme pomocou klávesy *F7* alebo pomocou ikony *Build* v paneli nástrojov. V prípade chyby kompilácie, ktorá nesúvisí s chybami v kóde, je potrebné skontrolovať, či zmeny v nastaveniach projektu boli uložené. Po úspešnej kompilácii program nahráme pomocou klávesovej skratky F8 alebo pomocou ikony *Load* v paneli nástrojov. V tomto prípade nie je potrebné pomocou prepínača meniť režim procesoru na bootovací.

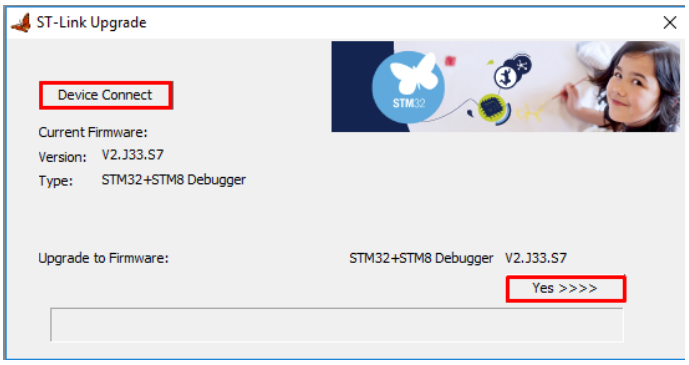


3.7 Firmware update pre ST-Link V2

Ak používame samostatný ST-Link V2, musíme mu pred použitím aktualizovať firmware pomocou programu ST-Link Utility, ktorý je dostupný na stránke
`https://www.st.com/en/development-tools/stsw-link004.html`
Po stiahnutí a inštalácii otvoríme aplikáciu STM32 ST-LINK UTILITY a v sekcii *ST-LINK* klikneme na *Firmware update*

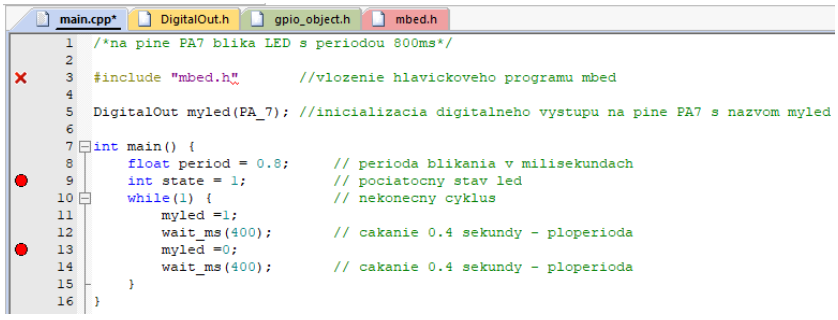


ST-Link pripojíme k počítaču, po kliknutí na *Device Connect* a následne na *Yes* sa do ST-Linku nahrá najnovšia veria firmware.

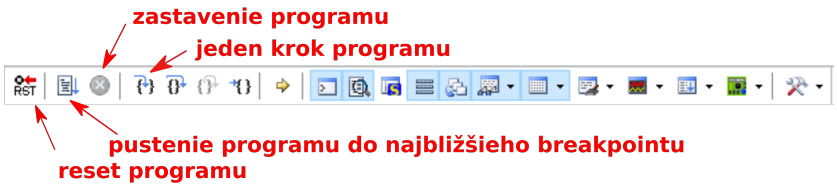


3.8 Debugovanie programu

Možnosť debugovať program je hlavný dôvod, prečo sme si program z Mbedu exportovali do prostredia uVision5. Do debugovacieho režimu sa dostaneme klávesovou skratkou *Ctrl+F5*. Kliknutím myši na ľavo od čísla riadku vyznačíme breakpointy(červený krúžok).

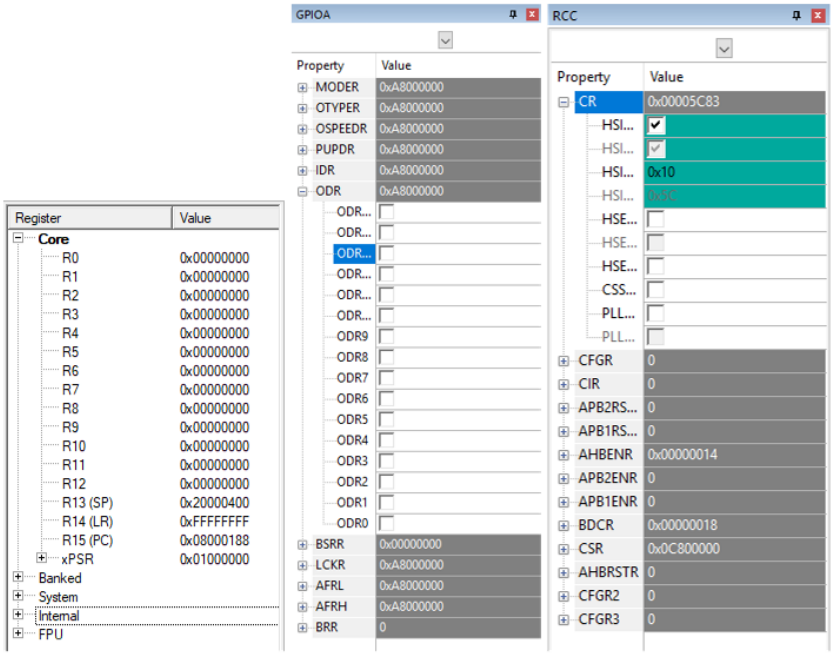


Následne je možné pomocou program krokovať, pustiť, zastaviť a resetovať klikaním na jednotlivé ikony na debugovacom paneli.



Ďalej je možné sledovať hodnoty v jednotlivých registroch procesora, hodnoty SP, LR a PC, ktoré sú zobrazené v paneli na ľavej strane okna. Nakoniec je možné sledovať aj konfigurácie jednotlivých periférii (napr GPIOA) po kliknutí na *Peripherals->System viewer->GPIOA*. Na pravej strane okna sa zobrazí panel s konfiguračnými registrami.

Pri niektorých registroch (napr. ODR) je možné priamo meniť hodnotu jednotlivých bitov vrámci registra.

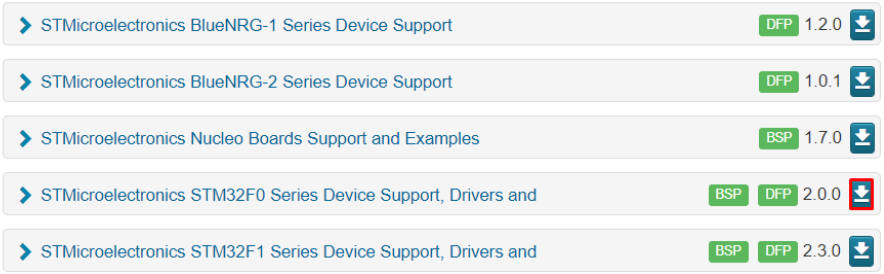


Kapitola 4
Založenie projektu v μVision5 pre procesor zo série STM32F0

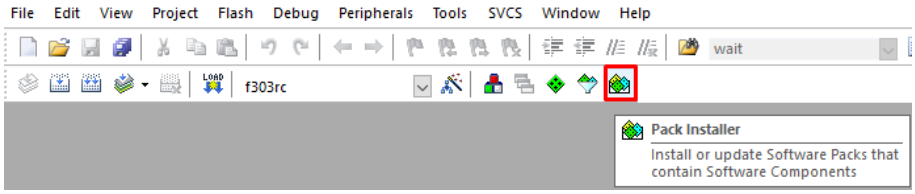
V nasledujúcej kapitole je popísaný podrobný na vytvorenie projektu v prostredí μVision 5 pre mikrokontrolér zo série STM32F0, konkrétne STM32F042F6P6.

4.1 Inštalácia potrebných modulov

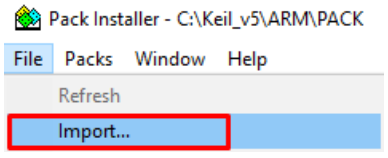
Aby bolo možné pracovať v prostredí s nejakým procesorom, najskôr je potrebné stiahnuť balík podpory, v našom prípade zo stránky
`https://www.keil.com/dd2/pack/`
stiahneme balík pre všetky mikroprocesory zo série STM32F0.



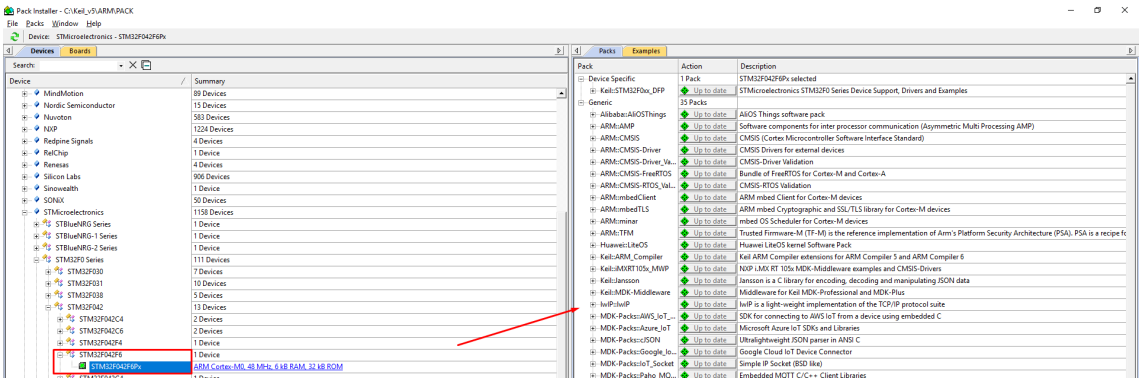
Po stiahnutí balíka otvoríme program uVision5 a prejdeme do sekcie *Pack Installer*.



V paneli nástrojov klikneme na *File->Import* a importujeme stiahnutý súbor.

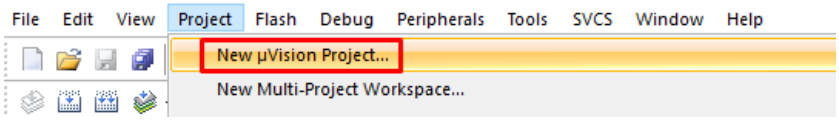


Následne sa preklikáme až k mikroprocesoru, ktorý budeme používať:
STMicroelectronics->STM32F0 Series->STM32F042->STM32F042F6->
->STM32F042F6Px a podľa potreby doinštalujeme ďalšie dostupné balíky, ktoré sú zobrazené na pravej strane okna.

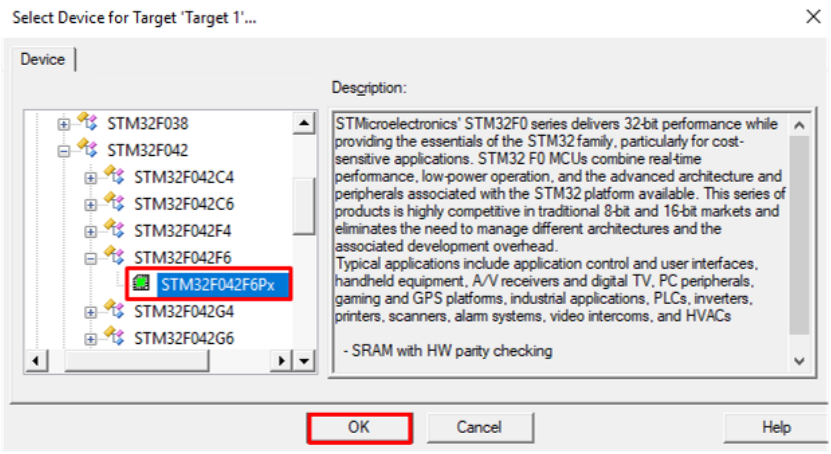


4.2 Vytvorenie projektu

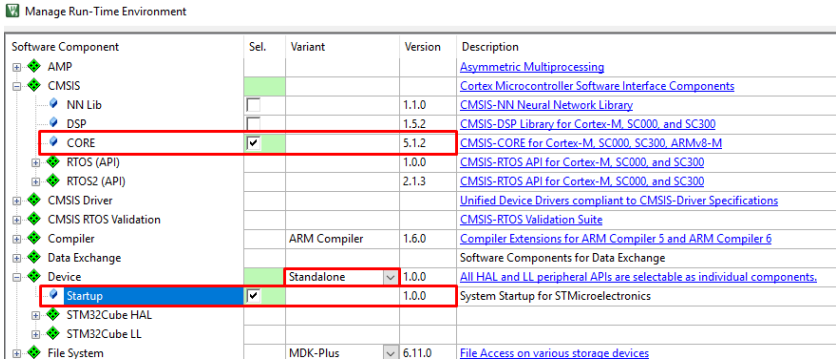
Nový projekt v prostredí uVision5 pre mikroprocesor STM32F042F6P6 si vytvoríme nasledovne: V paneli nástrojov klikneme na *Project->New uVision Project*,



Projekt pomenujeme a uložíme do nami vybranej zložky. Ďalej vyberieme platformu, s ktorou budeme pracovať, v našom si vyberieme platformu STM32F042F6Px: *STMicroelectronics->STM32F0 Series->STM32F042->STM32F042F6->* *->STM32F042F6Px*.



V sekcii *Device* zmeníme položku na *Standalone* a zaškrtneme položky *Core* a *Startup*.

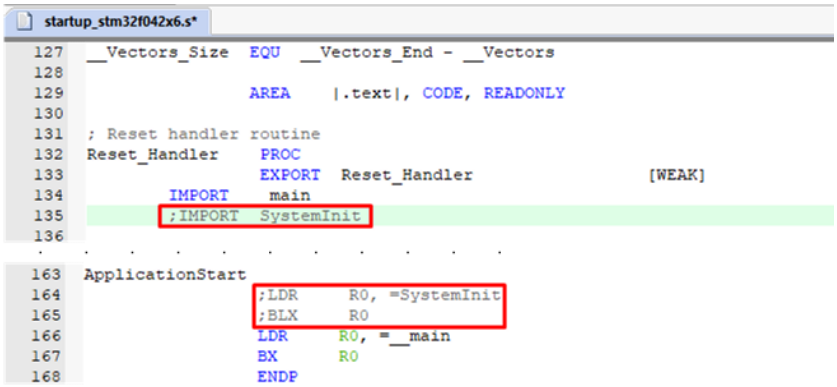


Po vytvorení projektu skontrolujeme, že položka Device obsahuje 2 startup súbory, jeden s príponou *.s* a druhý s príponou *.c*.

4.3 Úprava startup súboru s príponou *.s*

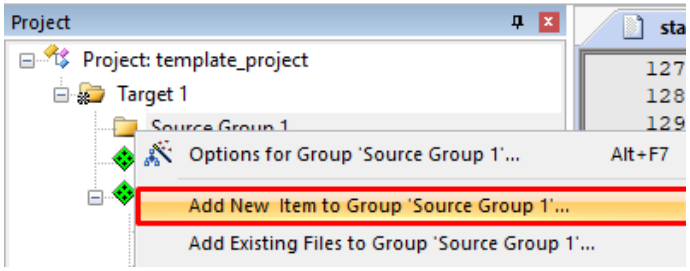
V projekte prejdeme do položky *Device* a otvoríme startup súbor s príponou *.s*. V programe pomocou znaku ";" zakomentujeme 3 riadky. Konkrétne zakomentujeme riadky:

```
IMPORT  SystemInit          (v mojom prípade na riadku 135)
LDR     R0, =SystemInit     (v mojom prípade na riadku 164)
BLX     R0                  (v mojom prípade na riadku 165)
```

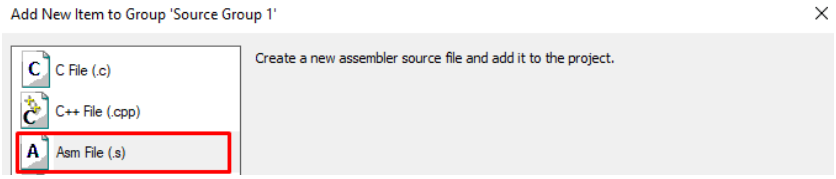


4.4 Vytvorenie *.s* súboru

V projekte sa automaticky vytvorila zložka *Source Group 1*, ktorú si môžeme prípadne premenovať a v ktorej budeme mať uložený program. Ten si vytvoríme tak, že pravým tlačidlom klikneme na zložku a pridáme do nej nový prvok: *Add New Item to Group ...*.



Následne si vyberieme *Asm File (.s)* súbor, pomenujeme ho, vyberieme miesto uloženia súboru (ideálne v jednom priečinku s projektom) a vytvoríme ho.



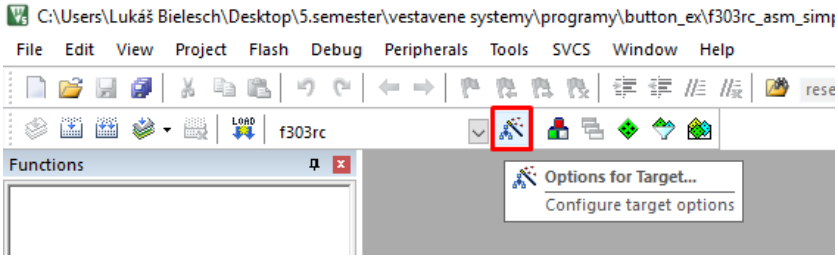
4.5 Nahratie programu do mikrokontroléra

4.5.1 Nahratie s použitím ST-Linku

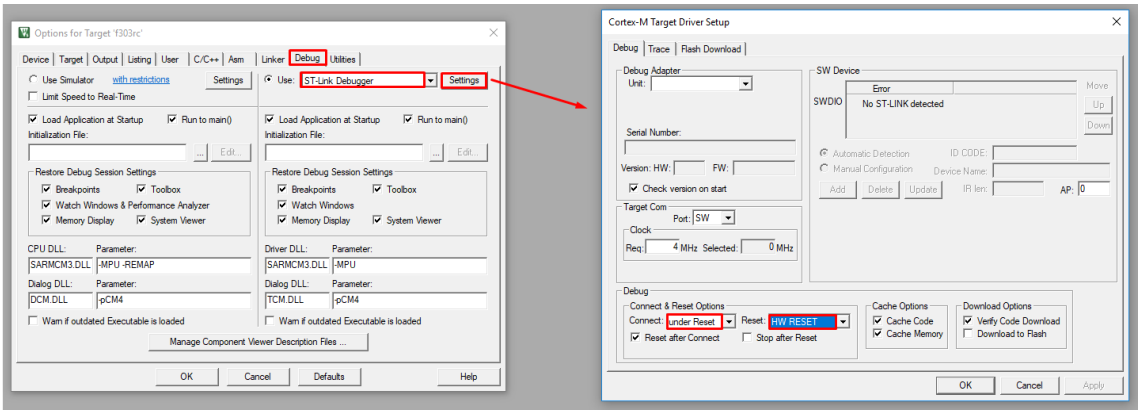
Pomocou ST-Linku je možné nahrať alebo debugovať bežiaci program v mikroprocesore. Dá sa kúpiť samostatne alebo je súčasťou na vývojovej doske, napríklad na NUCLEO-F303RE.

Zmena nastavení projektu pri použití ST-Linku

Ak budeme používať ST-Link, musíme ho v nastaveniach zvoliť ako predvolený debugovací nástroj.



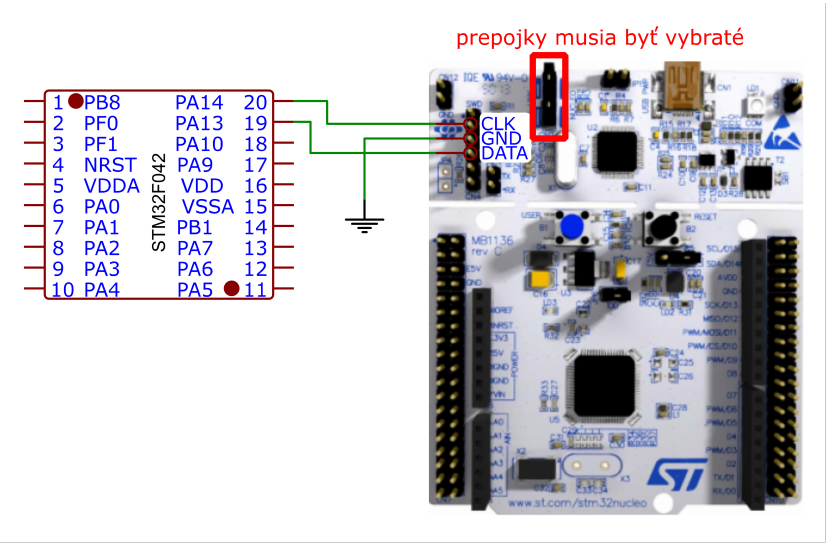
Prejdeme do *Options for Target* a v sekcii *Debug* zvolíme ako predvolený ST-Link Debugger. Ďalej prejdeme do nastavení debuggera a zmeníme nastavenie pripojenia na *under Reset* a nastavenie resetu na *HW RESET*.



Prepojenie mikroprocesoru s ST-Linkom

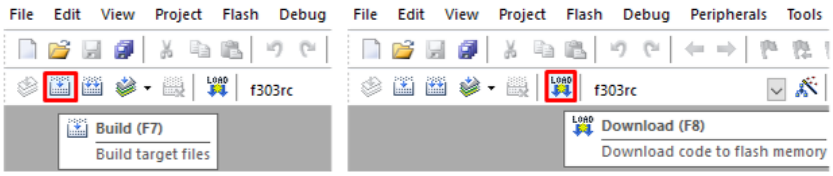
Program je možné sťahovať do dosky a debugovať pomocou ST-Linku, ktorý sa nachádza na vrchnej časti vývojovej dosky NUCLEO-F303RE. Najskôr prepojíme zem nášho mikroprocesoru so zemou ST-Linku, ktorá sa nachádza na vývode CN4, na treťom pine zhora. Potom prepojíme aj zvyšné 2 piny podľa nasledujúcej schémy. Okerm toho je potrebné vybrať prepojky, ktoré slúžia na prepojenie ST-Linku s procesom

STM32F303RE. Pri použití ST-Linku nemusíme na našom mikroprocesore pri nahrávaní meniť režim na bootovací pomocou prepínača.



Kompilácia a nahratie programu

Hotový program skompilujeme pomocou klávesy F7 alebo pomocou ikony *Build* v paneli nástrojov. Po úspešnej kompilácii program nahráme do dosky pomocou klávesovej skratky *F8* alebo pomocou ikony *Load* v paneli nástrojov.



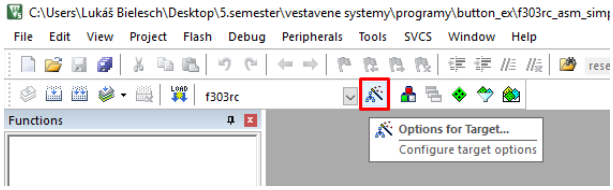
Po stlačení resetovacieho tlačidla bude na mikroprocesore bežať nahratý program.

4.5.2 Nahratie pomocou programu STM32CubeProgrammer

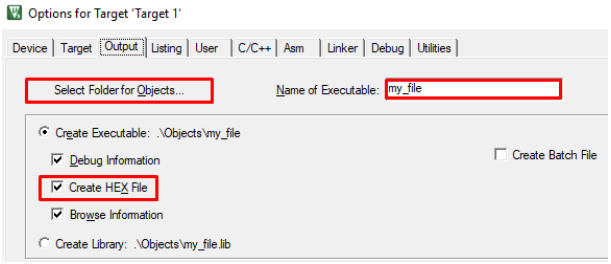
Ak nemáme k dispozícii ST-Link, program môžeme do mikroprocesoru nahrať manuálne použitím STM32CubeProgrammer.

Zmena nastavení projektu pri použití programu STM32CubeProgrammer

V nastaveniach musíme povoliť vytvorenie .hex súboru pri kompilácii projektu, ktorý budeme následne pomocou programu STM32CubeProgrammer nahrávať do dosky.

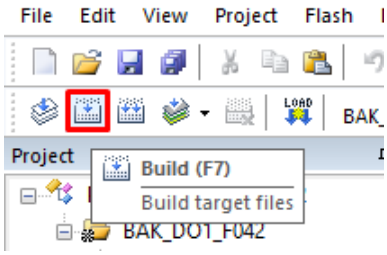


Prejdeme do *Options for Target* a v sekcii Output povolíme vytvorenie *.hex* súboru, pomenujeme si ho a vyberieme priečinok, do ktorého sa *.hex* súbor uloží.



Kompilácia a nahratie programu

Hotový program skompilujeme pomocou klávesy F7 alebo pomocou ikony *Build* v paneli nástrojov.



Tým sa vygeneruje a uloží .hex file do zložky zadanej v nastaveniach. Ďalej otvoríme program STM32CubeProgrammer a v pravom hornom rohu zvolíme ako druh komunikácie s procesorom *USB*. Pomocou prepínača na mikroprocesore zmeníme režim na bootovací a stlačíme resetovacie tlačidlo. V programe STM32CubeProgrammer sa nám objaví nové zariadenie na ktoré sa pripojíme.

Po pripojení prejdeme do sekcie Erasing Programming kliknutím na prostrednú ikonu na paneli na ľavej strane okna. Potom vyberieme .hex súbor, ktorý sme stiahli a program nahráme do mikroprocesoru kliknutím na Start Programming. Nakoniec pomocou prepínača na mikroprocesore vypneme bootovací režim a po stlačení resetovacieho tlačidla bude na mikroprocesore bežať nahratý program.