CanSat

poznamky 1

Praha 2011 **Vladimír Váňa**

my remarks: *CanSat Book for Students* – part.1 2011

1.Úvod (Introduction)

Poprve s myšlenkou CanSatu přišel profesor Robert Twiggs na konci 90. let. Tato myšlenka je založena na stavbě „minisatelitu“ využívající pro svou konstrukci plechovku od limonády. V plechovce o objemu např. 350ml je umístěn mikropočítač, vysílač a sensory teploty, tlaku. Dále např. GPS modul, kamerka, měřič zrychlení apod.

Cansat je vypuštěn např. pomocí meteorologické rakety či balonu ve výši cca 1 km a poté pomocí padášku pomalu sestupuje k Zemi a vysílá telemetrické údaje o naměřených hodnotách.

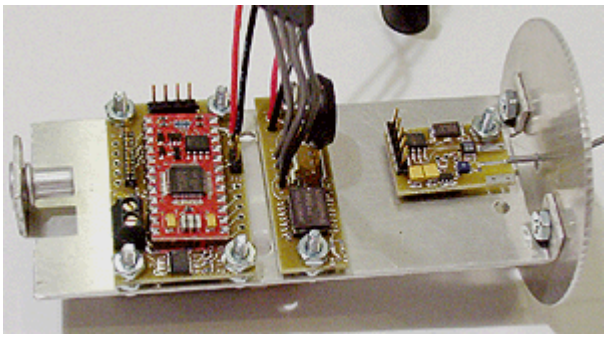
Je to cenově dosupný způsob seznámení studentů s stavbou satelitů, kosmickými technologiemi, moderními elektronickými prvky, technikou jednočipových počítačů, měřící technikou, programování, technologií GPS a zejména týmové práce studentů, zpracování technické dokumentace, vytváření reportů a prezentace výsledků své práce.

Stavba a provoz Cansatů je předmětem řady národních i mezinárodních soutěží zejména pro vysokoškolské studenty, ale i žáky sš. Pro ně je speciálně vyhlašováno „evropské mistrovství“ Evropskou kosmickou agenturou ESA. Je rovněž součástí magisterského studia Spacemaster na technických univerzitách mezi kterými je i ČVUT.

Předkládaná skripta jsou úvodem do této problematiky a mají studentům sloužit jako inspirace pro jejich další projekty.

2.Mechanická konstrukce (Mechanical Design)

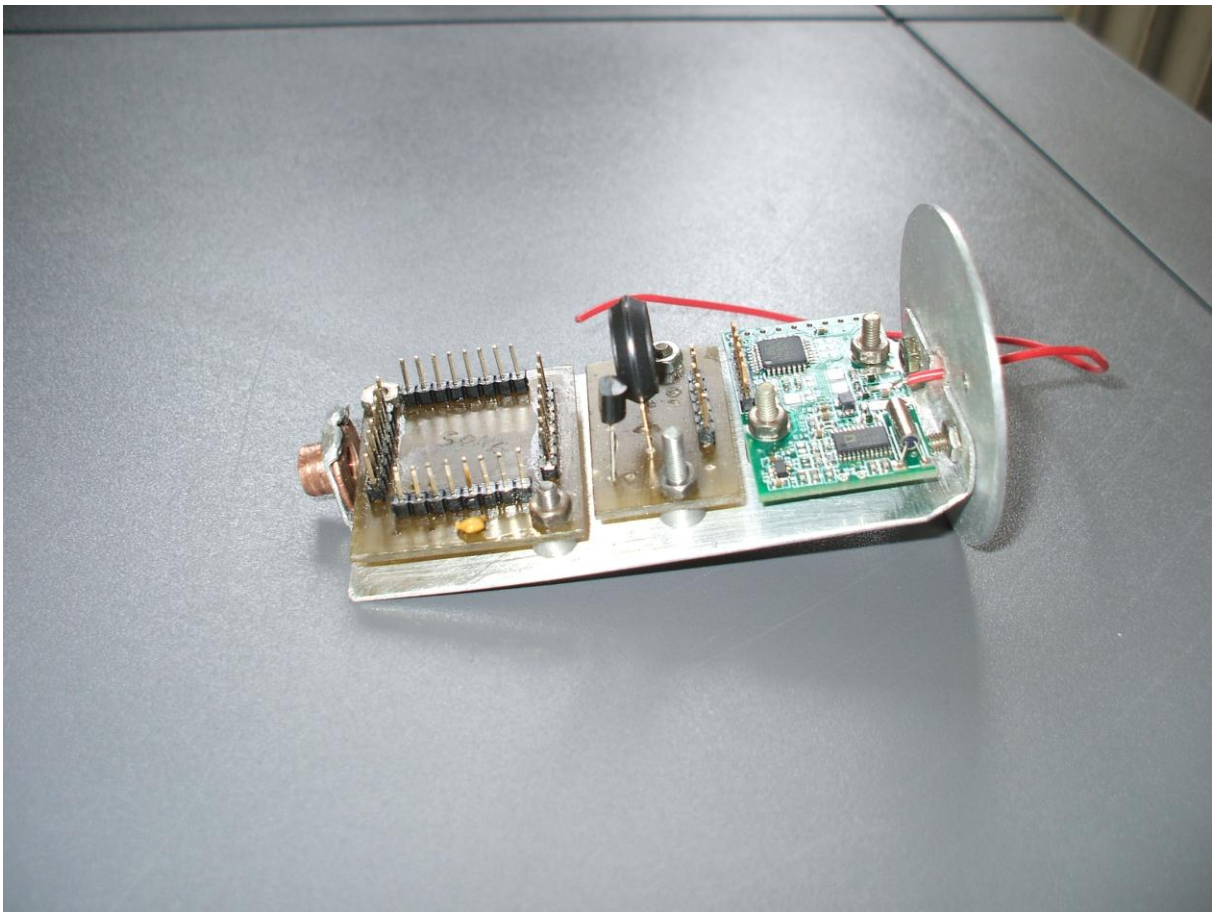
Při prvních pokusech s CanSaty si můžeme vyrobit jednoduchou mechanickou konstrukci, inspirovanou stavebnicí Pratthobies:



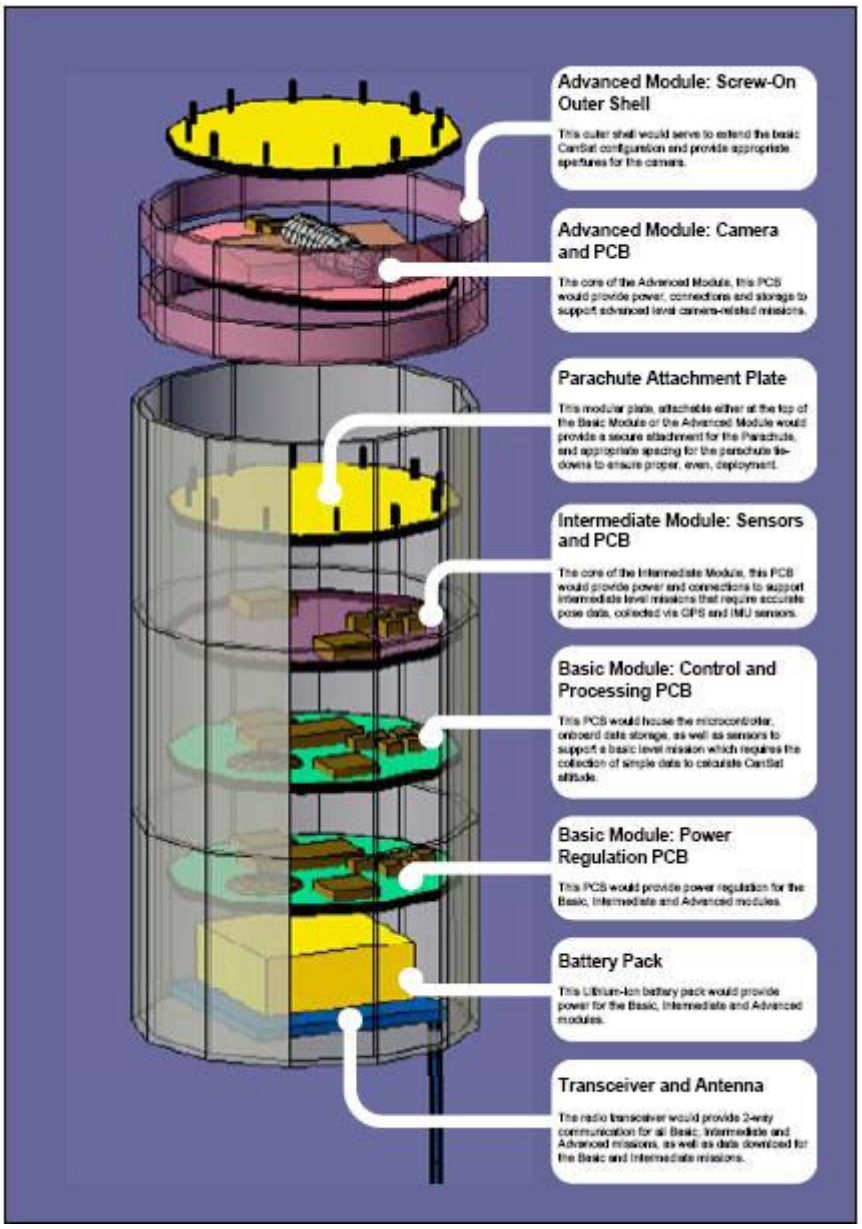
či

my remarks: *CanSat Book for Students* – part.1 2011

Výsledek naší práce je:



Lepší využití prostoru v plechovce však umožní odlišná konstrukce. Předpokládají ji např. autoři dokumentu **The ESA CanSat Kit. Preliminary Information Package**.



3.Padáček (Parachute)

Skutečné satelitní sondy obyčejně po čase skončí svůj život shořením ve vyšších vrstvách atmosféry, na Zemi se vracejí jen části s lidskou posádkou, vzorky apod. K snížení rychlosti přistání na zemi využívají mj. padáky.

Minisondy CanSat se však téměř výhradně pohybují v atmosféře a dosahují výšek max. několik km a poté se vracejí k Zemi pomocí padáku (padáčku). Takový padák je možné si buď zhotovit vlastními silami, nebo zakoupit hotový v prodejnách pro letecké a raketové modeláře. V obou případech však potřebujeme určit jeho velikost tak, aby vyhověl jak z hlediska velikosti zátěže , tak požadovaného zrychlení (zpomalení)

Např. pro soutěž ESA Cansat 2010 byla max. dovolená hmota sondy 350 g a zrychlení v rozsahu 8 m/s až 11 m/s.

Vzorec tahové síly padáku

my remarks: *CanSat Book for Students* – part.1 2011

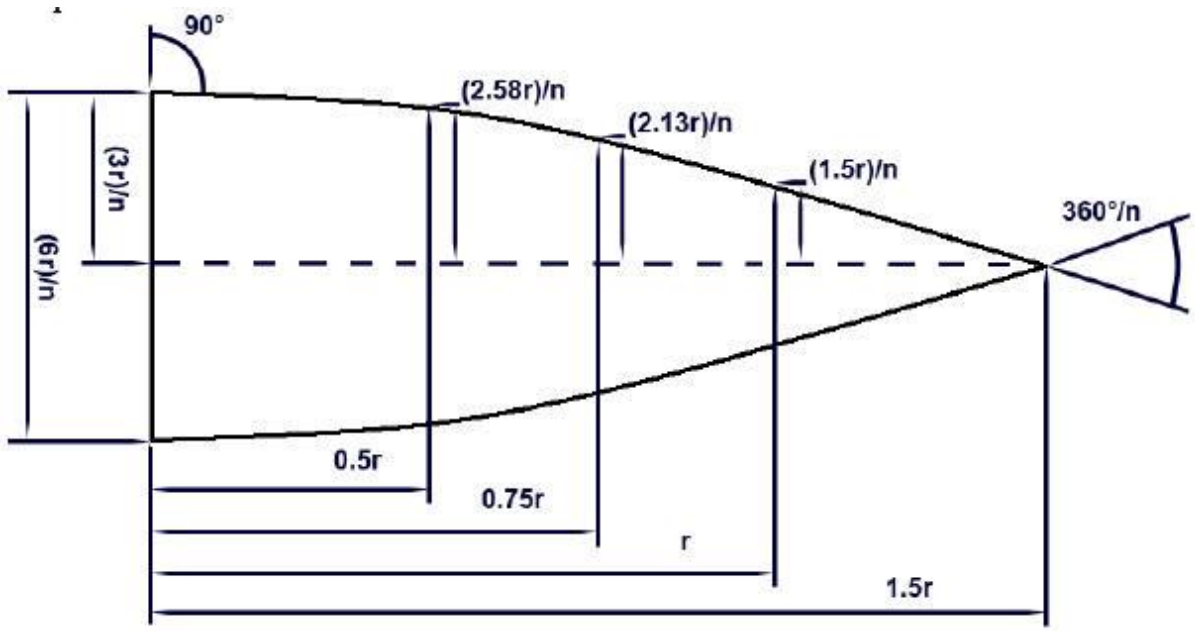
$$F_t = \frac{1}{2} C_D \rho A v^2$$

C_D – tahovy koeficient, zavisi na tvaru padaku, jeho hodnotu je 1,5 pro **semisférický** padák, 0,8 pro Cross Shapped (viz [14])
ρ – hustota vzduchu
A – celkova plocha padaku (ne pouze průřez). Z hodin fyziky a matematiky jsme zvyklí plochu označovat S. Budeme ale používat značení použité např. v [14], tj A …area)
v - rychlost

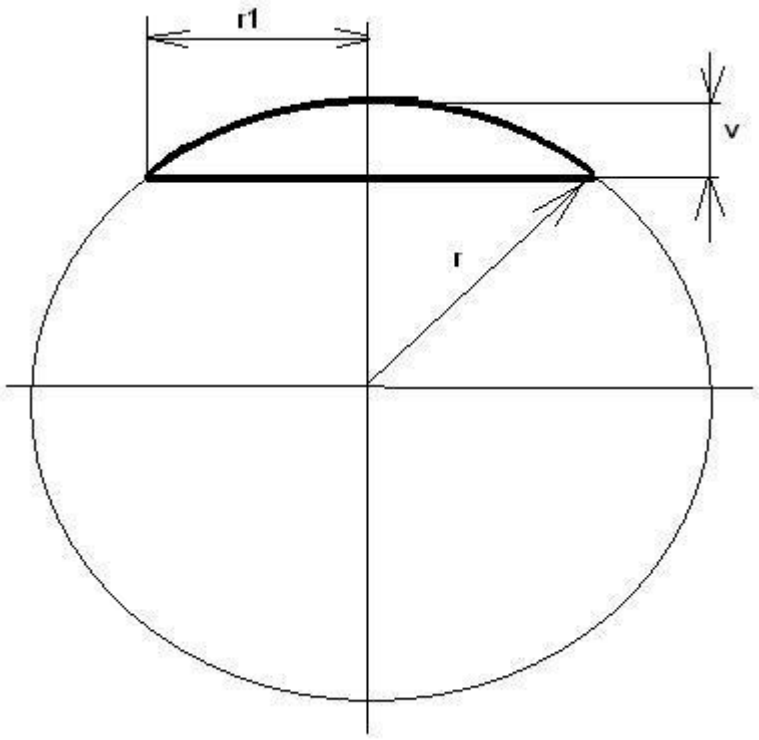
Pro rovnoměrny pohyb musi platit $F_t = F_g$ z čehož ziskame vyslednou plochu

$$A = \frac{2mg}{C_D \rho v^2}$$

Jako semisférický padák je v [14] označen padák sešitý z dílů dle obr.



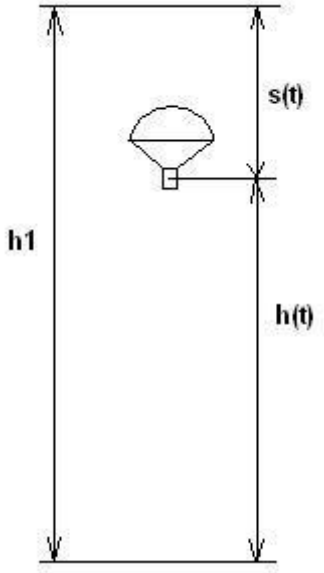
N je počet dílů, z nichž je padák sešit
r je poloměr
Při svém sestupu k Zemi CanSat obvykle vysílá hodnoty některých naměřených hodnot, snímá zemský povrch kamerou atd. Může nás proto zajímat i (alespoň přibližně) délka jeho sestupu proto, abychom mohli určit potřebnou kapacitu napájecího zdroje. Proto se pokusíme tuto dobu vypočítat.
Použijeme vzorec pro plochu vrchlíku znázorněného na následujícím obrázku:



Pro jeho plochu A platí vzorec

$$A = 2\,\pi \,.\, r \,.\, v \;+\; \tfrac{1}{2} \,.\, \pi \,.\, r \,.\, r1$$

Předpokládejme, že CanSat dosáhne (např. vystřelením signální raketou) do výše **h1** a odtud bude za pomoci padáku klesat k zemi. Čas budeme počítat od okamžiku dosažení výšky **h1**, tj, půjde o čas, po který CanSat klesá. V době **t** bude **s(t)** vzdálen od místa, kam byl vystřelen, tj. bude **h(t)** nad Zemí:



Provedeme odvození

$$F_z = F_u$$

$$m \,.\, g \;=\; \tfrac{1}{2} \,.\, C_D \,.\, \rho \,.\, v^2 \,.\, A$$

$$v^2 = \frac{2\,m\,g}{C_D \cdot \rho \cdot A}$$

$$v = \sqrt{\frac{2\,m\,g}{C_D \cdot \rho \cdot A}}$$

$$v = \sqrt{\frac{2\,m\,g}{C_D \cdot \rho \cdot A}}$$

$$s(t) = v.\,t$$
$$h(t) = h1 - s(t)$$
$$h(t) = h1 - v.t$$

takže nakonec dostaneme

$$h(t) = h1 - v = \sqrt{\frac{2\,m\,g}{C_D \cdot \rho \cdot A}} \cdot t$$

Pokud budeme padák vyrábět vlastními silami, můžeme využít některý z řady návodů na internetu, většinou na stránkách modelářů. Na výrobu je nejsložitější půlkulatý (semisférický) padák, o němž jako amaterští výrobci tvrdí, že jednotlivé díly je nutné sešít na stroji. Nejprve musíme přehnout a sešít okraje aby se nám látka netřepila, popřípadě oentlovat, a pak teprve sešít jednotlivé díly k sobě. Nakonec přišijeme vrchlík, nebo ho můžeme vynechat, vzniklý otvor nám pomůže stabilizovat padák. Šňůry použijeme syntetické motané z více vláken (podobné jako je horolezecké lano ale postačí nám nejmenší vyráběný průměr 1 mm). Přidělame je k padáku buď pomocí kroužků, ke kterým je navážeme, dírka v nákližku pak musí být obšitá pro její dostatečnou pevnost, nebo uděláme na šňůře smyčku, kterou k padáku přišijeme (pak nákližek vůbec nepotřebujeme).
Můžeme vyrobit i výrobně jednodužžší verzi padáku, např. krychlový. Pro padáky jiných tvarů musíme ovšem také vyhledat hodnotu příslušného tahového koeficientu.
Padák můžeme zhotovit např. z padákového hedvábí. Lze také použít hotový padák např. ze světlic. Na internetu jsem našel i návod, v němž autor jako nejlevnější materiál doporučujte tzv. banánové folie, což je pytel, do něhož se balí při přepravě banány. Jsou v něm sice větrací otvory (aby banány nehnily), autor příspěvku však tvrdí, že tyto otvory nesnižují funkčnost padáku.
Lze dokonce použít padák čínské výroby zhotovený z folie z umělé hmoty a prodávaný v potřebách pro modeláře. Jeho testování na SPŠE Ječná jsme publikovali na
http://www.youtube.com/watch?v=t30ZDRo1Iwk
http://www.youtube.com/watch?v=aJR7VOJLnFM

Ať již použijeme padák jakéhokoli původu, materiálu atd., musíme věnovat dostatečnou péči i jeho sbalení před vypuštěním Cansatu. Na internetu jsem dokonce našel radu doporučující před složením padák napudrovat dětským pudrem, aby se jeho složené části neslepovaly ☺

4.Čidla a měření (Sensors and Measurement)

my remarks: *CanSat Book for Students* – part.1 2011

4.1 Čidla teploty (Temperature Sensors)

Měření teploty patří mezi nejstarší měření vůbec, existuje celá řada čidel. My si ukážeme použití dvou z nich:

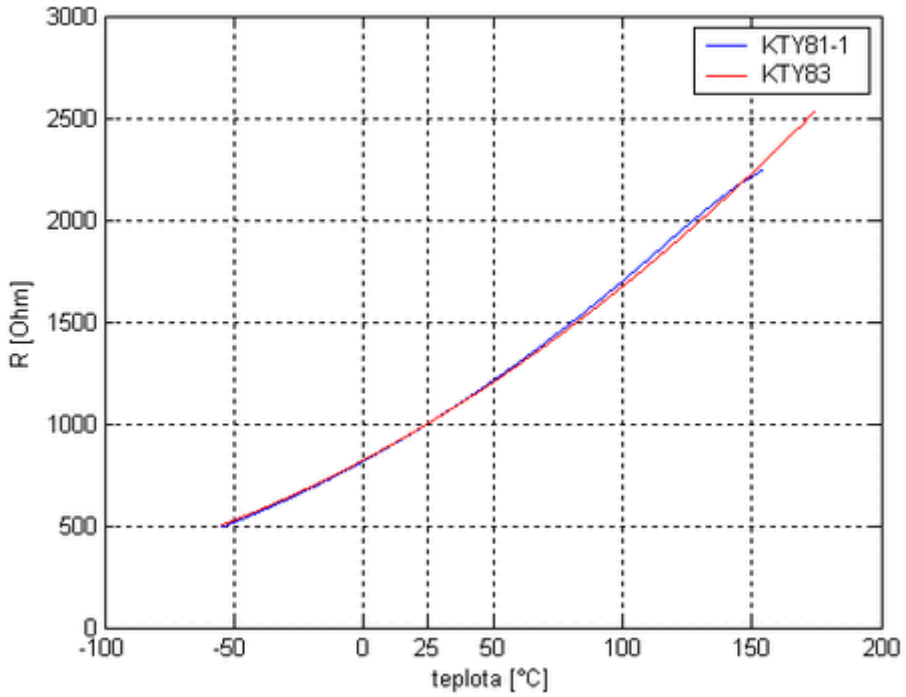
4.1.1 Monokrystalické Si senzory [dle HW server cz] (Monocrystal Si Sensors)

Pro výrobu křemíkových senzorů se používá nevlastního polovodiče typu N, tedy s dominantní elektronovou vodivostí. Pohyblivost volných nosičů náboje v krystalové mřížce křemíku závisí na teplotě a na počtu příměsí v jednotce objemu. S rostoucí teplotou dochází vlivem rozptylu nosičů náboje na mřížce polovodiče ke zmenšování pohyblivosti těchto nosičů, v důsledku čehož narůstá rezistivita, podobně jako je tomu u kovů. Monokrystalické Si senzory teploty tedy mají kladný teplotní součinitel odporu podobně jako PTC termistory, princip jejich vodivosti je však odlišný. Křemíkové senzory se obvykle používají pro rozsah teplot –50 až 150 °C.

Základní vlastnosti monokrystalických Si senzorů:

- **Teplotní součinitel odporu** je téměř konstantní v celém rozsahu teplot a jeho střední hodnota se pohybuje kolem 0,01 K^{-1} (platinové senzory: 0,004 K^{-1}, NTC: cca -0,03 až -0,06 K^{-1}).
- **Dlouhodobá stabilita.** Teplotní drift kolem 0,2 K po 10000 hodinách nepřetržitého provozu při maximální provozní teplotě.
- **Linearita** je lepší než u NTC termistorů, ale horší než u platinových senzorů, nelinearitu lze však vhodnými metodami úspěšně korigovat.
- **Teplotní rozsah** je obvykle -55 až 150 °C, k dostání jsou však běžně i senzory s horní teplotní hranicí 300 °C.
- **Referenční hodnota odporu** při teplotě 25 °C je obvykle 1000 nebo 2000 Ω.

Monokrystalické křemíkové senzory jsou běžně k dostání, a to za přijatelnou cenu, která se pohybuje kolem dvaceti až třiceti korun. V řadě aplikací mohou díky svým vlastnostem nahradit platinová čidla, je však nutno počítat s linearizačními obvody.Typickými představiteli křemíkových monokrystalických senzorů jsou čidla řad **KT** a **KTY**. Na následujícím obrázku je vyobrazena závislost odporu na teplotě senzorů KTY81-1 a KTY83 (KTY85).



Závislost odporu křemíkových monokrystalických senzorů KTY81- 1 a KTY83 na teplotě

Závislost odporu senzorů KTY83/85 na teplotě lze aproximovat vztahem :

$$R_T = R_{ref}\left[1 + A\cdot(T - T_{ref}) + B\cdot(T - T_{ref})^2\right],$$

kde A=7,635·10^{-3} K^{-1}, B=1,731·10^{-5} K^{-2}, T_{ref}=25 ° C. Senzor KTY85 se od senzoru KTY83 liší teplotním rozsahem a typem pouzdra.

Pro senzory KTY81/82/84 platí následující aproximační vztah:

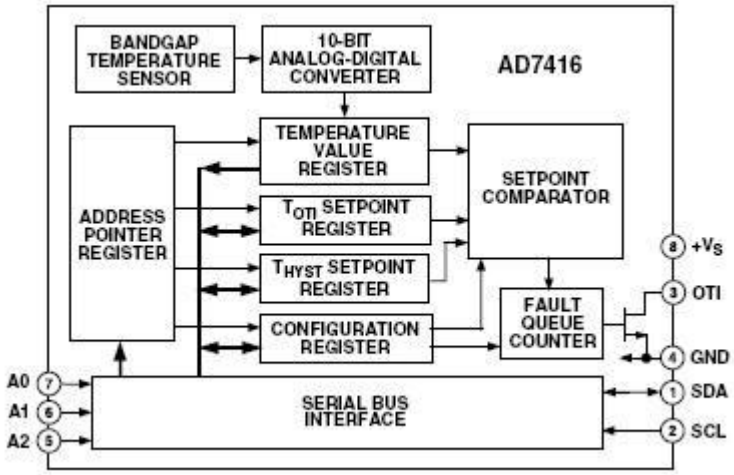
$$R_T = R_{ref}\left[1 + A\cdot(T - T_{ref}) + B\cdot(T - T_{ref})^2 - C\cdot(T - T_I)^D\right],$$

přičemž pro senzor KTY81-1 nabývají uvedené konstanty následujících hodnot: A=7,874·10^{-3} K^{-1}, B=1,874·10^{-5} K^{-2}, C=3,42·10^{-8} K^{-D} (pro $T < T_I$ je C=0), D=3,7, T_I=100 °C a T_{ref}=25 °C.

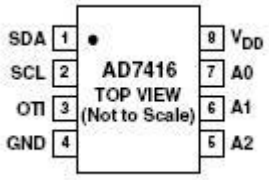
R_{ref} je referenční hodnota odporu při teplotě T_{ref} a u všech tří zmíněných senzorů (KTY81-1/83/85) je R_{ref}=1000 Ω . Více informací o výše uvedených senzorech můžete získat na webu výrobce. U vedené vztahy mohou být použity pro generování tabulky hodnot, kterou je možné nahrát do paměti ROM a v případě práce s mikrokontrolérem využít spolu s interpolačními algoritmy k přesnému stanovení teploty.
Pro naše experimenty jsme zakoupili KTY81 v GME.

4.1.2 Integrované čidlo teploty a 10-bitový A/D převodník AD7416 firmy Analog devices. (Integrated Temperature Sensor and 10 bits A/D Converter AD7416 from Analog Devices comp.)

Budeme-li naměřené údaje z čidla teploty dále zpracovávat jednočipovým počítačem, můžeme využít např. integrované čidlo od AnalogDevices AD7416. Jeho principielní zapojení ukazuje obr.:



Teplotní čidlo , 10-bitový A/D převodník i další elektronika jsou na jednom čipu



Rozsah měřených teplot je -55 ^{0} C až + 125 ^{0} C, rychlost převodu 10-ti bitovým A/D převodníkem je 15µs a 30 µs. Napájecí napětí je 2,7 až 5,5 V. Z hlediska uživatele tohoto obvodu nám stačí jen znát jeho komunikaci s okolím. Ta probíhá podle protokolu I2C. Kromě obecných znalostí tohoto protokolu budeme potřebovat jen několik dalších konkrétních údajů.
Použijeme obvod AD7416 s 8 piny. Jejich význam je následující:

1	SDA	I/O dat
2	SCL	Vstup pro hodinové pulzy
3	OTI	Over Temperature Indikator – je nastaven na 1, když výsledek A/D převodu je větší, než 8 bitové slovo v registru OTR
4	GND	Napájení, „nula“
5	A2	Nejvyšší bit pro nastavení adresy obvodu
6	A1	Prostřední bit pro nastavení adresy obvodu
7	A0	Nejnižší bit pro nastavení adresy obvodu
8	V_{DD}	Kladné napájecí napětí 2,7 až 5,5 V

Výrobce popisuje I2C komunikaci s tímto obvodem pomocí průběhů:

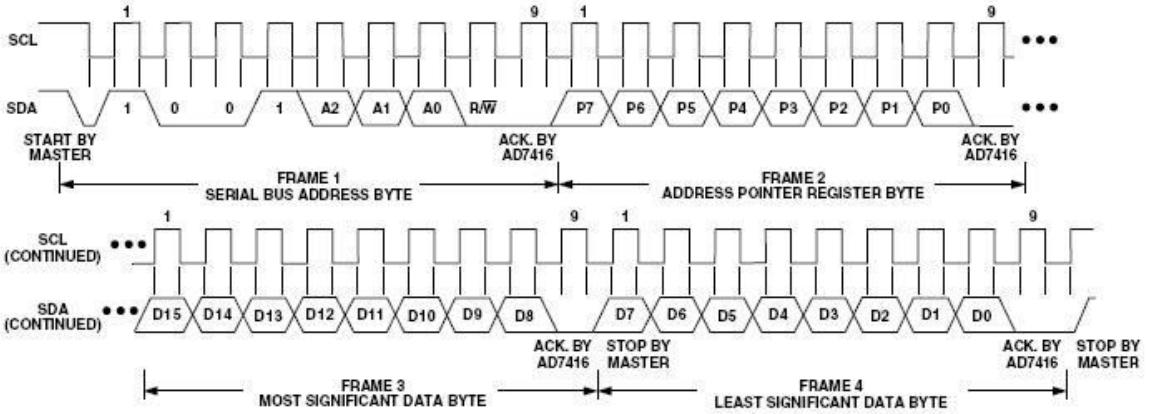


Figure 9. Writing to the Address Pointer Register Followed by a Two Bytes of Data to the T_{OTI} or T_{HYST} Register

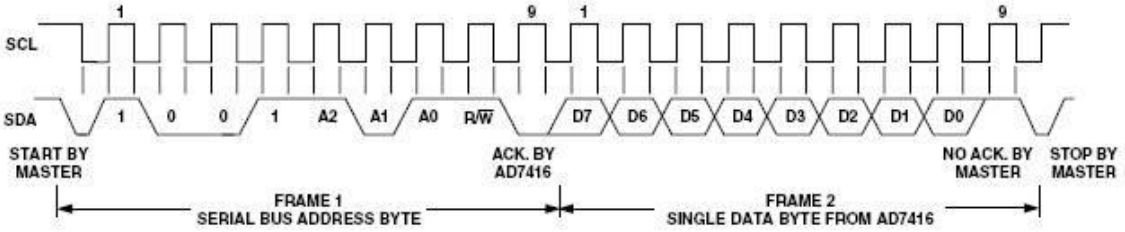


Figure 10. Reading a Single Byte of Data from the Configuration Register

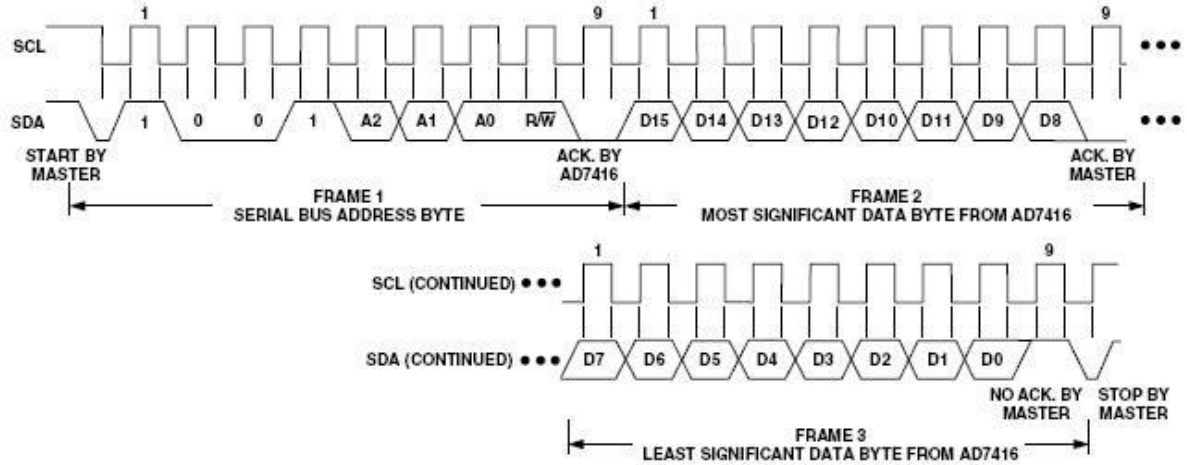
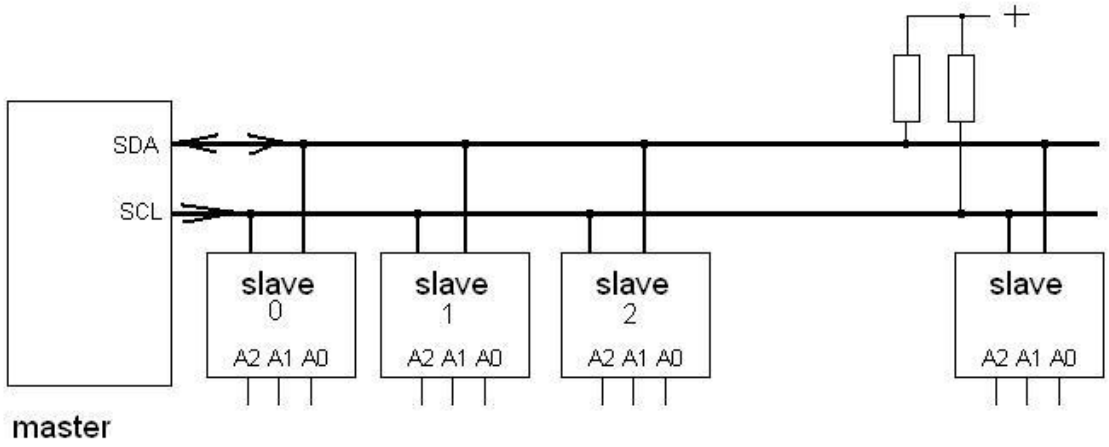


Figure 11. Reading Two Bytes of Data from T_{OTI} or T_{HYST} Register

Výše uvedené průběhy jsem převzal z firemní dokumentace AD. Ta ještě obsahuje i slovní popis této komunikace. Pro snadnější pochopení výše uvedených průběhů i slovního popisu v dokumentaci však potřebujeme znát principy přenosu dat podle protokolu sběrnice I2C. Proto si je nyní uvedeme:

Protokolem sběrnice I2C se řídí komunikace na dvouvodičové sběrnicí I2C. Ta používá datový vodič SDA a vodič pro hodinové impulzy SCL. Každý z těchto vodičů je přes pull-up odpory (cca 4k7) připojen k napětí o úrovni logické 1 (prakticky to znamená k napájecímu napětí cca 3.3 až 5V). V klidovém stavu jsou tedy na obou vodičích signály o logické úrovni . Ke sběrnici se připojují (paralelně) zařízení dvojího druhu – **master** (což je většinou nějaký jednočipový počítač, to bude i náš případ) a **slave** (v našem případě obvod AD7416, ale může to být celá řada dalších zařízení)



Master ovládá sběrnici. Pouze **master** může vysílat hodinové impulzy a zahajuje přenos vygenerováním signálu START. Master může vysílat data na sběrnici (do slave zařízení) nebo přijímat data ze zařízení slave. Master také generuje signál STOP.

Slave na základě výzvy od zařízení master může přijímat nebo vysílat data v časových okamžicích daných hodinovými pulzy SCL.

Pro přenos dat sběrnicí I2C platí tato pravidla:

- Přenos dat může být zahájen jen pokud na sběrnici neprobíhá jiný přenos dat
- Po dobu přenosu dat SDA nesmí měnit svůj stav, pokud je SCL na vyšší úrovni (tj. logická 1)
- Změna SDA v době, kdy SCL ja na vyšší úlovnu (tj. ve stavu log. 1) je považována za řídící signál

Protokol sběrnice I2C definuje tyto situace:

- **Klidový stav** – SDA a SCL jsou ve stavu vyšší úrovně, jsou neaktivní
- **Zahájení přenosu (START):** master stáhne SDA na nízkou úroveň (logická 0), SCL zůstává na úrovni 1
- **Přenos dat**: Příslušný zdroj signálu (vysílač) postupně vysílá na datový vodič osm datových bitů , které se posouvají s *hodinovými impulzy* na vodiči SCL *vysílanými blokem master*. Přenos začíná bitem s nejvyšší váhou.
- **Potvrzování (Acknowledge)**: Příslušný impulz potvrdí příjem bytu (8 bitů) stažením SDA na logickou 0, ve které drží SDA do doby než master vyšle devátý hodinový impulz. Toto potvrzení znamená, že se má přijímat další byte. Požadovaný konec přenosu oznámí tím, že nevyšle potvrzení. Vlastní ukončení přenosu se dosáhne signálem STOP.
- **Signál ukončení přenosu (STOP)** – SDA z nízké úrovně (0) přejde na vyšší úroveň (1). SCL zůstává na úrovni 1.

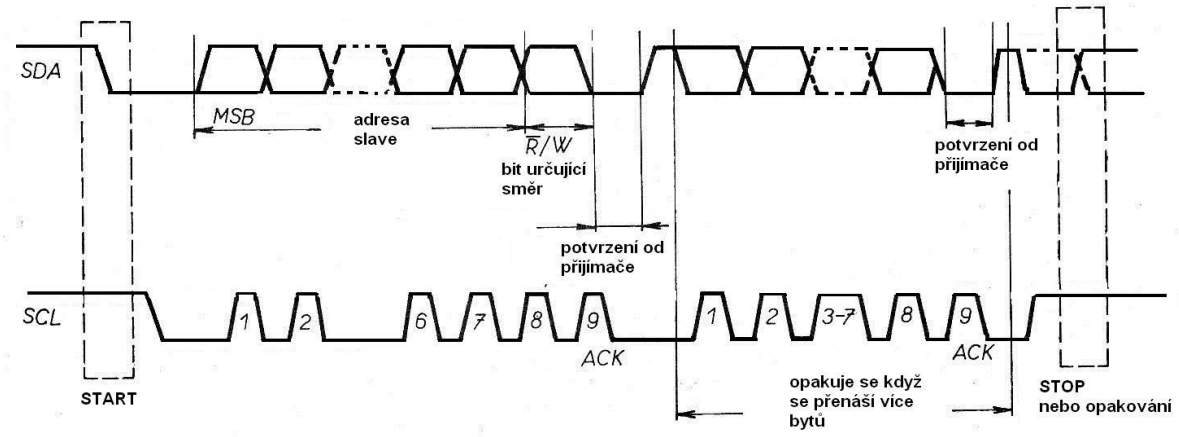
Na sběrnici I2C mohou nastat dva druhy přenosu dat:

- **Přenos dat z vysílače Master do přijímače slave** – První byte vysílaný masterem je *adresa slave*. Poté následuje několik byte dat. Slave vrací acknowledge (potvzovací) bit po každém přijatém byte.
- **Přenos dat z vysílače slave do přijímače master** – První byte – *adresa slave* zařízení je vyslané master zařízením. Slave vrátí acknowledge (potvrzovací bit). Poté následuje několik byte vyslaných zařízením slave do zařízení master. Master za každým přijatým byte vrací potvrzovací bit acknowledge až na poslední byte. Zařízení master generuje všechny hodinové impulzy SCL, signály START a STOP. Přenos končí signálem STOP nebo signálem START začínajícím další přenos.

Každé slave zařízení I2C má adresu, jejíž strukturu ukazuje obrázek:

A6	A5	A4	A3	A2	A1	A0	R/W

Skládá zez několika částí. První část A6 A5 A4 A3 charakterizuje o jaké jde zařízení. Je stejná pro celou třídu zařízení. Např všechny A/D převodníky mají tuto část adresy stejnou, obdobně čidla teploty atd. další část A2 A1 A0 je dána nastavením logiských signálů na takto značené piny obvodu slave. Většinou se jedná o přímé propojení s úrovní logické 0 či logické 1. Poslední bit R/W určuje, zda se budou data posílat do zařízení nebo ze zařízení. Dvě různá zařízení nesmí mít stejnou adresu, takže zařízeních stejného charakteru může být připojeno nanejvýš 8 (neboť je možné mít až 8 různých kombinací A2 A1 A0). Časový průběh komunikace na sběrnici I2C popisuje obrázek:



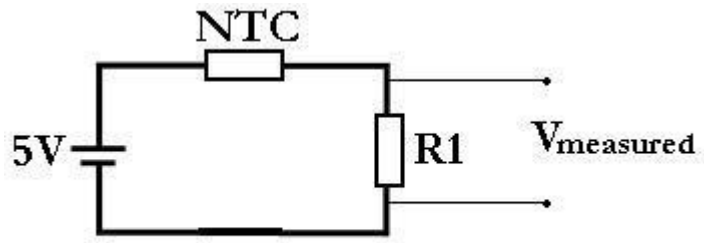
Komunikace na sběrnici I2C se Vám možná bude zdát složitá, nicméně opak je pravdou. O komunikaci zařízení slave se postaravi liž výrobci příslušných obvodů. V případě (i našem), kdy funkci řídícího zařízení master vykonává jednočipový počítač, můžeme při vytváření firmware tohoto počítače použít již hotové knihovny a wizardy vývojových prostředí, nejčastěji v jazyce C. Konkrétní ukázky pro jazyk C a mikropočítače ATMEL AVR jsem popsal v knížce [1]. Pro STM32VL Discovery najdete ukázky kódu v 2.díle tohoto výukového material.

4.1.3 Čidla teploty s termistorem (podle ESA Cansat Book 2010) (Temperature Sensors with Thermistor)

Pro poměrně velké množství materiálů (např. kovové vodiče) lze jejich teplotní závislost jejich odporu na teplotě popsat (**přibližným**) vzorcem vám jistě dobře známým z učebnic fyziky

$$R = R_0 . (1 + \alpha \Delta T)$$

Kde odpor R_0 je odpor při nějaké vztažné teplotě, např. 20 ^{0}C. α je teplotní koeficient odporu a ΔT je rozdíl skutečné teploty (kdy vodič má odpor R) a teplotou, kdy má hodnotu R_0 . Hodnota teplotního koeficientu α bývá kladná a tedy odpor vodiče se s zvětšující se teplotou zvyšuje. Vytvoříme –li odporový dělič složený ze dvou odporů – odporu s malým teplotním koeficientem α (tj odporem na teplotě málo závislým) a z druhého odporu s velkým α (termistoru, odporu na teplotě velmi závislém), a napájíme-li tento odporový dělič konstantním napětím, bude napětí na odporovém děliči dosti závislé na teplotě. Této skutečnosti lze pro měření teploty využít.
Autoři ESA Cansat Book 2010 pro měření teploty využili termistor *NTCLE203E3103GB0* vyráběný firmou Vishay/BC components. Materiál, z něhož je tento termistor vyroben má však záporný teplotní koeficient a proto je nazýván NTC, nebo Negative Temperature Coefficient termosistorem.
V této knížce je také uvedeno jaeo zapojení sloužící k měření teploty. Je to odporový dělič, o němž jsem se již zmínil v předchozím odstavci:



Odpor R1 má konstantní hodnotu R1 = 10 kΩ. Celkový odpor obou do série zapojených odporů je

$$R_T = R_1 + R_{NTC}$$

Proud protékající tímto děličem určíme z ohmova zákona

$$I = U/R = 5V/(R_1 + R_{NTC})$$

Protože stejný proud i prochází oběma odpory, bude platit

$$I = U/R = V_{messure} / R_1$$

Sloučením obou předchozích výrazů dostaneme :

$$5V/(R_1 + R_{NTC}) = V_{messure} / R_1$$

Dalšími úpravami tohoto vzorce dostaneme

$$V_{messure} = 5V * R_1/(R_1 + R_{NTC}), \text{ or}$$

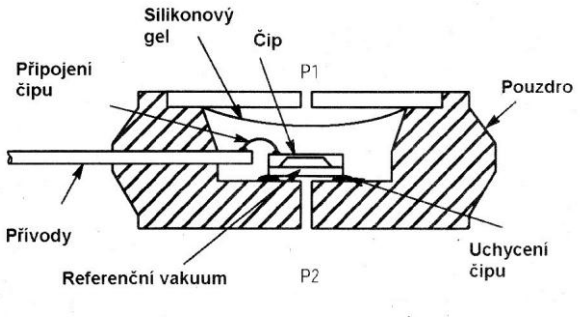
$$R_{NTC} = (V_{messure} /(R_1 + R_{NTC})) - R_1$$

Abychom získali převodní funkci potřebujeme znát závislost RNTC na teplotě. Tuto závislost uvádí výrobce termistoru v datasheetu. Pro termistor NTCLE203E3103GB0 firmy Vishay uvádí tento výrobce zmiňovanou závislost ve formě rozsáhlé tabulky. Tuto tabulku lze naprogramovat do vyhodnocovacího firmware přičemž pro hodnoty v tabulce neuvedené je možné je získat např. lineární interpolací.

my remarks: *CanSat Book for Students* – part.1 2011

4.2 Čidla tlaku (Pressure Sensors)

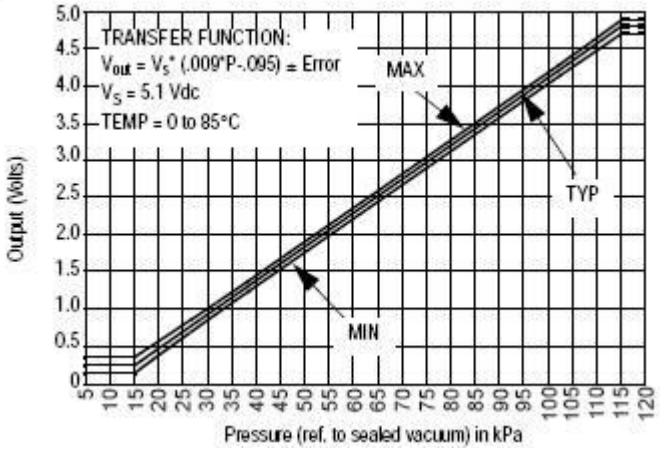
Jednou ze základních meteorologických veličin je atmosférický tlak. Pro jeho měření se na trhu dají zakoupit různá čidla. Vybral jsem čidlo od firmy Motorola (přesněji řečeno Freescale Semiconductors) s označením MPX4115A. jde o součástku pro měření absolutního tlaku. Obvod v sobě integruje vlastní čidlo, výstupní zesilovač a obvody pro teplotní kompenzaci. Může být používán v rozsahu teplot -40 až +125 ^{0}C. Vnitřní uspořádání obvodu je na obrázku:



Čidlo tvoří odporová síť vyrobená na polovodičovém čipu , která se rozvažuje podle mechanického prohnutí čipu. Díky přítomnosti komůrky s referenčním tlakem je možné přímo určit absolutní hodnotu tlaku. Na rozdíl od jiných čidel je v tomto typu zaintegrován teplotně kompenzovaný zesilovač. Výstup se tak dá přímo připojit na vstup převodníku A/D (např. vestavěného v jednočipovém mikropočítači).

Napájecí napětí čipu je 4,85 až 5,35 V
Odběr proudu 7 mA
Časová odezva 1 ms
Rozsah použití 15 až 115 kPa

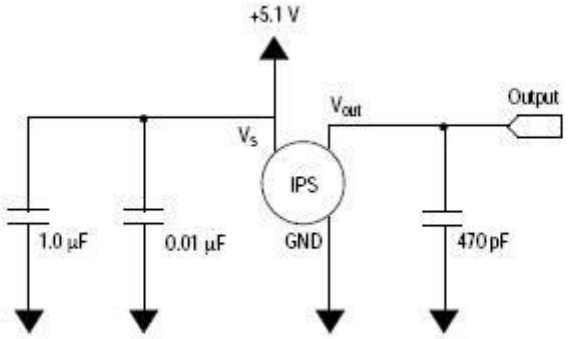
Závislost výstupního napětí pak ukazuje obrázek:



Výstupní napětí je popsáno vztahem

$$U_{out} = U_S\,(0{,}009\,P - 0{,}095)$$

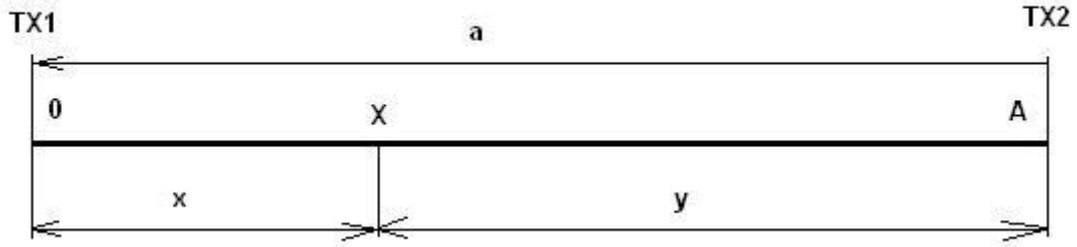
Kde U_S je napájecí napětí, P je tlak v kPa.
Výše uvedený graf obsahuje i toleranční pásmo, v němž se může pohybovat graf skutečného čidla.
Doporučené zapojení čidla podle firemní dokumentace je



4.3 GPS a Galileo.

4.3.1 Základní informace (Basic Informations)

GPS (Global Positioning System) je projekt, který umožňuje komukoli na povrchu planety Země zjistit své zeměpisné souřadnice. Ke své funkci využívá několika specializovaných družic, které ze svých oběžných drah vysílají směrem k Zemi signály v podobě elektromagnetických vln. Signál se (ve vakuu) šíří rychlostí cca 300 000 km/s. Družice jsou seřízeny tak, že všechny vyšlou signál v přesně definovaný okamžik. Přijímač umístěný na Zemi vypočítá svou pozici na základě toho, s jakým zpožděním přijme signál z jednotlivých družic. Když přijmeme signál, tak nevíme, jak dlouho mu trvalo, než k nám dorazil. Známe pouze časové rozdíly. Tato koncepce se často označuje zkratkou TDOA (Time Difference of Arrival). Princip si lze snadno představit na situaci jednorozměrného prostoru. V jednorozměrném případě postačí k určení polohy dva vysílače TX1 a TX2:



Určit polohu v tomto jednorozměrném případě znamená vypočítat vzdálenost x. Oba vysílače ve stejný okamžik vyšlou signál. Ten se rychlostí světla c šíří, a v čase t1 dorazí do bodu X signál z vysílače TX1, v čase t2 signál z vysílače TX2. Použijeme známé vzorečky z fyziky zš že „dráha se rovná rychlost krát čas“

$$x = c \cdot t1 \quad \text{a} \quad y = c \cdot t2$$

Vzdálenost vysílačů a je známá a dále platí

$$a = x + y$$

Doby t1 a t2 sice neznáme, můžeme však změřit jejich rozdíl **t2 – t1**. K tomu stačí změřit dobu Δt mezi okamžikem, kdy do bodu X dorazí signál z prvního vysílače a okamžikem, kdy dorazí signál z druhého vysílače

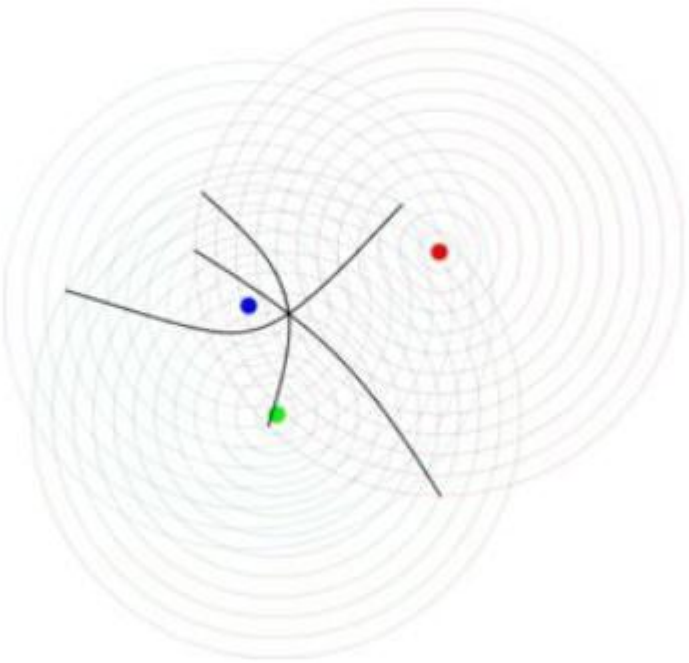
$$\Delta t = t2 - t1 = (y - x) / c$$

Máme tedy dvě rovnice o dvou neznýmých **x** a **y** (konstanty **a** a **c** známe, stejně jako změřený rozdíl časů **Δt**)

$$a = x + y$$
$$\Delta t = (y - x) / c$$

Rovnice si vyřešte sami v rámci Vaší přípravy na státní maturitní zkoušku z matematiky.

Uživatel drží v rukou přijímač, který zaznamená signály ze dvou zdrojů s časovým rozdílem Δt=t2 - t1. Rychlost šíření signálu známe, a tak můžeme snadno vypočítat pozici přijímače vzhledem k vysílačům. Předpokládali jsme však, že vysílače i přijímač se nacházejí v přímce. Složitější situace nastane, když uvážíme, že se přijímač může nacházet kdekoli v rovině. V takovém případě již nelze jednoznačně určit pozici. Jediné, co můžeme s jistotou tvrdit, je to, že je přijímač umístěn v kterémkoli bodě hyperboly. Právě hyperbola má totiž tu vlastnost, že všechny body na ní ležící mají stejný rozdíl vzdáleností od obou ohnisek hyperboly. V ohniscích oné hyperboly se nacházejí vysílače.
Ke stanovení polohy potřebujeme ještě jeden vysílač. Jsou-li vysílače tři, pak získáme tři časové rozdíly (t2 - t1, t3 - t2 a t3 - t1) z nichž pouze dva jsou nezávislé, tedy z libovolných dvou lze vypočítat zbývající třetí. Dva časové rozdíly určují dvě hyperboly a my víme, že se vysílač současně nachází na obou hyperbolách. Průsečík těchto hyperbol tedy jednoznačně určuje polohu přijímače. Na následujícím obrázku jsou znázorněny tři vysílače a tři hyperboly, z nichž libovolné dvě stačí k určení polohy.



Ve dvojrozměrném případě jsou k určení pozice zapotřebí tři vysílače.
Uvážili jsme jednorozměrný příklad, kdy k určení polohy postačily dva vysílače. Následoval případ dvojrozměrný, kdy počet vysílačů musel být zvýšen na tři. Zbývá poslední případ - trojrozměrný. Nebudeme již zabíhat do podrobností, a tak shrňme pouze výsledky. Z časového rozdílu mezi dvěma signály můžeme stanovit, že se přijímač nachází někde na povrchu rotačního hyperboloidu. Máme-li k dispozici čtyři vysílače, pak získáme šest časových rozdílů (t2 - t1, t3 - t2, t4 - t3, t3 - t1, t4 - t2 a t4 - t1), z nichž pouze tři jsou nezávislé - zbývající lze dopočítat. Můžeme zkonstruovat šest rotačních hyperboloidů protínajících se v jednom bodě v prostoru. Postačí však libovolné tři. Nejdůležitější závěr plynoucí z celého rozboru je to, že v trojrozměrném prostoru potřebujeme čtyři družice. Díky nim lze stanovit všechny tři souřadnice bodu v prostoru, tedy zeměpisnou délku, zeměpisnou šířku a nadmořskou výšku. Pro úplnost je třeba podotknout, že existuje i dvojrozměrný mód GPS přijímačů,

který se aktivuje ve chvíli, kdy je k dispozici signál pouze ze tří družic. V takovém případě nelze určit zbývající třetí hyberboloid a místo něj se při výpočtu použije Zemský povrch. Jedná se o východisko z nouze - výsledkem je pouze odhad dvou zeměpisných souřadnic, přičemž výšková souřadnice zcela chybí.
Dosud jsme uvažovali pouze počet družic, ale je též nutno vzít v úvahu i jejich rozmístění vzhledem k poloze přijímače. Kdyby byly všechny čtyři družice umístěné v jednom bodě, jsou k určení polohy zcela bezcenné. Kdyby spolu s přijímačem tvořily přímku, můžeme určit pouze jednu souřadnici. Kdyby ležely v právě jedné rovině, můžeme vypočítat pouze dvě souřadnice. Družice a přijímač se tedy nikdy nesmí dostat do jedné roviny. Může se zdát, že tato podmínka není příliš svazující, a že je téměř vyloučeno, aby se vše nacházelo přesně v rovině. Reálná situace je však komplikovanější. Čím plošší je uspořádání, tím větší chyba nastává při určování polohy. Měření časových rozdílů je totiž vždy zatíženo chybami. Tyto chyby se promítnou do celkové chyby výsledku v závislosti na tom, jaké je rozestavění družic. Úhel družice-přijímač-družice by měl být co největší. Byl zaveden koeficient označovaný PDOP (Position Dilution Of Precision), který reprezentuje rozestavění družic. Čím lépe jsou družice rozmístěny vzhledem k uživateli, tím je tento koeficient vyšší. Z výše uvedeného a z řady další důvodů je potřebný počet satelitů ještě větší.
Je také zřejmé, že výpočty k určení polohy jsou poměrně složité, přičemž nejde jen pouhé dosazování do matematických vzorců, ale musí se vzít v úvahu i další skutečnosti. Naštěstí nic takové nemusíme programovat. O tyto výpočty se stará firmware GPS modulů. Ty s námi pak komunikují některým z několika protokolů. Pro naše účely bude najvýhodnější použít některý z GPS modulů komunikující sériovým signálem protokolem NMEA 0183 (vytvořeným National Marine Electronics Association)
Poznatky týkající se tohoto protokolu najdeme na internetu na celé řadě stránek. Pro naše účely si dovoluji uvést několik poznatků publikovaných v roce 2006 Janem Martínkem v sériálu GPS a komunikační protokol NMEA:

Asociace NMEA na své stránce http://www.nmea.org/pub/0183/index.html uvádí, že komunikační standard je dokument podléhající copyrightu a lze jej pouze zakoupit od asociace NMEA. Cena představuje řádově stovky dolarů. Ostatní zdroje na internetu prý nejsou autorizované a mohou představovat porušení copyrightu. NMEA dále uvádí, že obsah rozličných stránek s touto tématikou mnohdy obsahuje zastaralé informace. Pod klíčovým slovem NMEA 0183 lze na internetu skutečně nalézt řadu odkazů a jejich obsah se jeví býti konzistentním.
Protokol, kterým komunikuje většina GPS přijímačů je sériový „dálnopisný“ signál s přenosovou rychlostí (baud rate) např. 4800. Počet datových bitů je 8, přičemž sedmý bit (MSB) je vždy nulový. Počet stop bitů je jeden nebo více, parita není žádná. Navzájem spolu komunikuje vždy jeden mluvčí (talker) a jeden nebo více posluchačů (listeners). Veškerá data jsou posílána ve formě vět (sentences). Jsou dovoleny pouze tisknutelné ASCII znaky plus znaky konce řádku, tedy <CR> a <LF> (0x0d, 0x0a hexadecimálně). Každá věta začíná znakem $ (dolar) a končí sekvencí <CR><LF>.

Existují tři základní druhy vět:
- věty ze strany mluvčího (talker sentences)
- proprietární věty (proprietary sentences)
- dotazovací věty (query sentences)

Obecný formát vět ze strany mluvčího je

První dvě písmena, která následují po znaku dolar, jsou označená tt a představují identifikátor mluvčího (talker identifier). Další tři písmena (sss) jsou identifikátor věty (sentence identifier). Následují datové položky oddělené čárkami (znak ","). Po nich následuje nepovinný kontrolní součet. Věta je ukončena znaky <CR><LF>. Význam jednotlivých datových položek je jednoznačně definován pro konkrétní typ věty (ten je určen identifikátorem sss). Jestliže určitá datová položka není k

dispozici, zůstane datové pole prázdné, ale čárky oddělující datová pole zůstávají (bez mezery). Kontrolní součet začíná znakem hvězdička ("*") a za ní jsou dvě hexadecimální číslice představující logickou operaci XOR (exclusive OR) ze všech znaků mezi "$" a "*". Samotný dolar a hvězdička se do kontrolního součtu nezapočítávají. Každá věta může obsahovat nejvýše 80 znaků plus "$" a <CR><LF>, celkem tedy 83 bajtů.
Věty proprietární umožňují výrobcům nadefinovat vlastní větu. Tyto věty začínají sekvencí "$P", pak následuje třípísmenný identifikátor výrobce, a dále následují jednotlivé datové položky v souladu s přáním výrobce. Obecný formát věty musí být zachován.
Dotazovací věty představují způsob, kterým může posluchač zažádat mluvčího o zaslání konkrétní věty. Obecný formát je

$ttllQ,sss<CR><LF>

První dva znaky (tt) za znakem dolar jsou identifikátorem toho, kdo podává žádost. Následující dva znaky (ll) označují dotazovaného - tedy toho, komu je žádost posílána. Pátým znakem je vždy písmeno "Q", které označuje, že se jedná o dotazovací typ věty. Následuje třípísmenná datová položka (sss) určující, o jaký typ věty se žádá. Příkladem dotazovací věty může být následující sekvence:

$CCGPQ,GGA<CR><LF>

V této větě písmena CC označují počítač, který žádá přístroj GP (tedy GPS přijímač), aby zasílal věty typu GGA. Po této dotazovací větě by měl GPS přijímač zasílat každou sekundu větu typu GGA, dokud nedostane povel k zasílání jiného typu věty. Dvoupísmenných identifikátorů existuje mnoho, pro nás je však v této chvíli nejdůležitější, že pro GPS přijímače se používá identifikátor GP. Existuje nepřeberné množství různých vět, avšak moduly GPS nejčastěji používají pouze ***čtyři,*** které jsou uvedeny v následujících tabulkách.

1. GSA, aktivní satelity a DOP (Dilution Of Precision)
Příklad:

$GPGSA,A,3,29,26,22,09,07,05,04,,,,,1.7,1.0,1.4*30

#	formát	příklad	komentář
1	c	A	Přepínání mezi N-rozměrnými módy (A=automatické, M=manuální)
2	d	3	Počet dimenzí N (1=?, 2=2D, 3=3D)
3	dd	29	ID prvního satelitu použitelného pro výpočet
4	dd	26	ID druhého satelitu použitelného pro výpočet
5	dd	22	ID třetího satelitu použitelného pro výpočet
6	dd	09	ID čtvrtého satelitu použitelného pro výpočet
7	dd	07	ID pátého satelitu použitelného pro výpočet
8	dd	05	ID šestého satelitu použitelného pro výpočet
9	dd	04	ID sedmého satelitu použitelného pro výpočet

10	dd	N.A.	ID osmého satelitu použitelného pro výpočet
11	dd	N.A.	ID devátého satelitu použitelného pro výpočet
12	dd	N.A.	ID desátého satelitu použitelného pro výpočet
13	dd	N.A.	ID jedenáctého satelitu použitelného pro výpočet
14	dd	N.A.	ID dvanáctého satelitu použitelného pro výpočet
15	d.d	1.7	PDOP (Position Dilution Of Precision) v metrech
16	d.d	1.0	HDOP (Horizontal Dilution Of Precision) v metrech
17	d.d	1.4	VDOP (Vertical Dilution Of Precision) v metrech
18	*xx	30	Kontrolní součet

2. RMC (Recommended Minimum Navigation Information) Minimální doporučená informace pro navigaci
Příklad:

$GPRMC,170138.615,A,4912.2525,N,01635.0378,E,0.04,16.43,280705,,*32

#	formát	příklad	komentář
1	hhmmss.sss	170138.615	Čas (UTC)
2	c	A	Status (A=OK, V=varování)
3	ddmm.mmmm	4912.2525	Zeměpisná šířka
4	c	N	Indikátor sever/jih (N=sever, S=jih)
5	ddmm.mmmm	01635.0378	Zeměpisná délka
6	c	E	Indikátor východ/západ (E=východ, W=západ)
7	d.d	0.04	Vodorovná rychlost (Speed Over Ground, v uzlech)
8	d.d	16.43	Kurz pohybu ve stupních
9	ddmmyy	280705	Datum ddmmyy
10	d.d	N.A.	Magnetická deklinace ve stupních
11	c	N.A.	Indikátor východ/západ (E=východ, W=západ)
12	*xx	32	Kontrolní součet

3. GSV (Satellites in View) Informace o družicích
Množství údajů závisí na počtu viditelných družic. Jedna věta může obsahovat nejvýše 80 znaků, což vystačí pouze k uložení dat týkajících se nejvýše čtyř družic. Informace proto bývá rozdělena do několika dílčích vět.
Příklad (trojice vět):

$GPGSV,3,1,11,09,84,297,41,05,48,256,45,07,38,059,41,26,22,178,41*74
$GPGSV,3,2,11,24,13,063,00,14,12,324,00,30,12,251,00,22,12,286,38*78
$GPGSV,3,3,11,29,10,173,35,04,09,105,30,18,06,254,00*46

Poznámka: Příklad v tabulce se vztahuje pouze k první větě.

my remarks: *CanSat Book for Students* – part.1 2011

#	formát	příklad	komentář
1	d	3	Celkový počet vět (číslují se od 1)
2	d	1	Číslo aktuální věty (taktéž se čísluje od 1)
3	dd	11	Počet viditelných družic
4	dd	09	Identifikační číslo družice
5	dd	84	Úhlová výška, kde se daná družice nachází
6	ddd	297	Azimut, kde se daná družice nachází
7	dd	41	Odstup signálu od šumu (SNR - Signal to Noise Ratio). Je-li tento údaj roven nule, nelze daný satelit využít k výpočtu polohy. Nejčastěji proto, že je zastíněn.
…	…	…	Podle počtu viditelných družic mohou následovat další čtveřice údajů (4-7)
n	*xx	74	Kontrolní součet

4. GGA - zeměpisná délka a šířka, geodetická výška, čas určení souřadnic (Geographical Longitude and Latitude, Geodetic Head,Geographical Coordinates Determination Time)

Příklad:

$GPGGA,170139.615,4912.2526,N,01635.0378,E,1,07,1.0,357.5,M,43.5,M,0.0,0000*7D

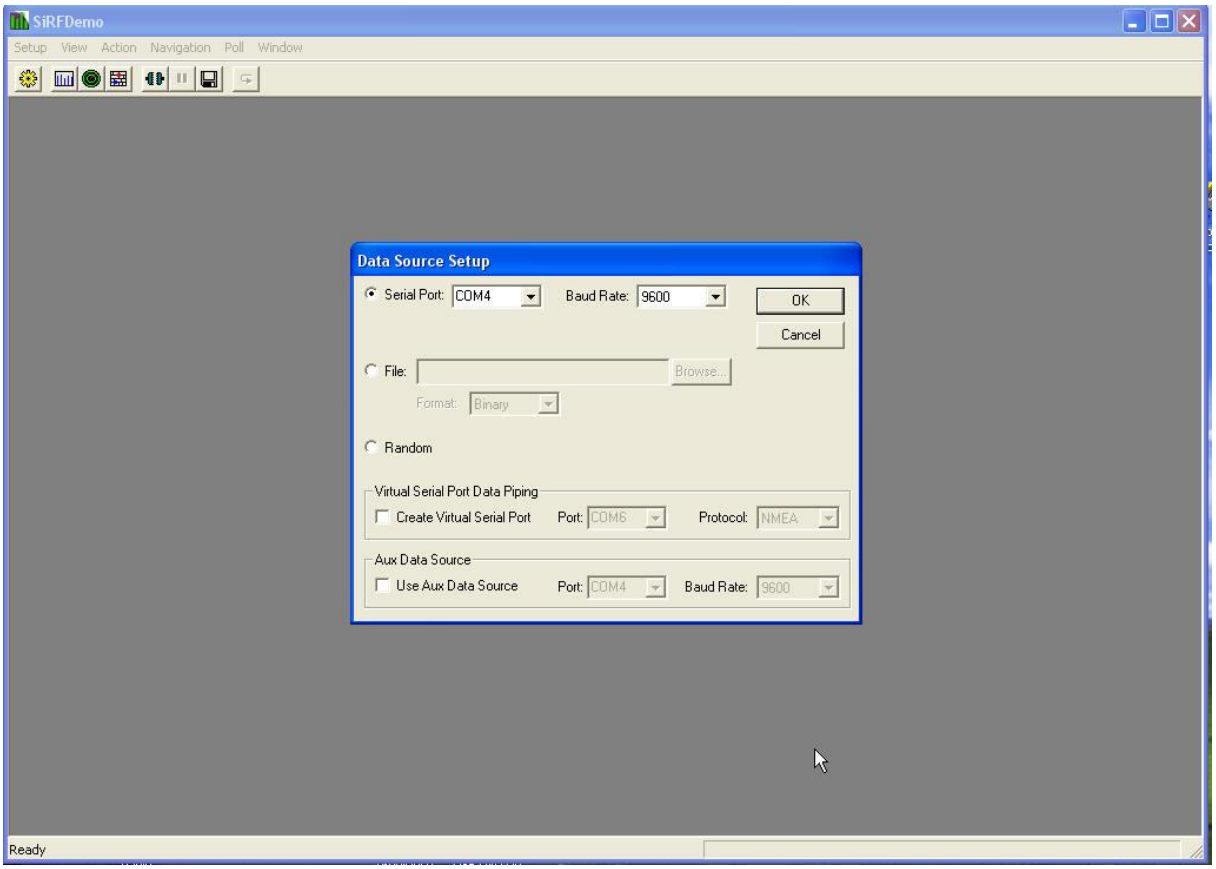
#	formát	příklad	komentář
1	hhmmss.sss	170139.615	Čas (UTC), pro který platí údaje o vypočtené pozici
2	ddmm.mmmm	4912.2526	Zeměpisná šířka
3	c	N	Indikátor severní/jižní šířka (N=sever, S=jih)
4	dddmm.mmmm	01635.0378	Zeměpisná délka
5	c	E	Indikátor východní/západní délky (E=východ, W=západ)
6	d	1	Indikátor kvality: 0 — nebylo možno určit pozici 1 — pozice úspěšně určena 2 — pozice úspěšně určena (diferenční GPS)
7	dd	07	Počet viditelných satelitů 00 — 12
8	d.d	1.0	Vliv rozestavění družic na určení polohy HDOP (*Horizontal Dilution of precision*)
9	d.d	357.5	Výška antény nad geoidem
10	c	M	Jednotka pro předchozí údaj (č.9) (M=metr)
11	d.d	43.5	Geoidal separation, rozdíl mezi WGS-84 zemským elipsoidem a střední úrovní moře (geoid). Znaménko mínus znamená, že střední úroveň země je pod elipsoidem.
12	c	M	Jednotka vzdálenosti pro předchozí položku (č.11) (M=metr)
13	d.d	0.0	Stáří poslední aktualizace DGPS. Údaj je uváděn v sekundách. Jestliže údaj chybí, nepoužívá se DGPS.
14	dddd	0000	Identifikační číslo referenční stanice pro DGPS (0000 — 1023)
15	*xx	7D	Kontrolní součet

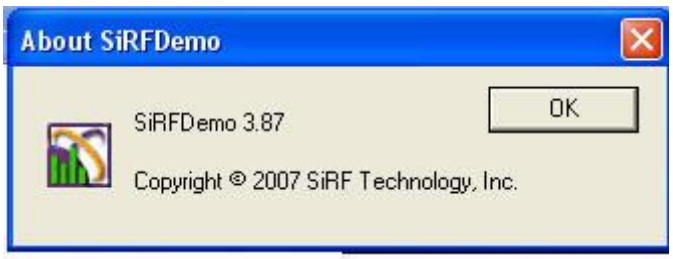
Další informace lze nalést v zmiňovaném seriálu i v řadě další internetových zdrojů, wikipedii apod.

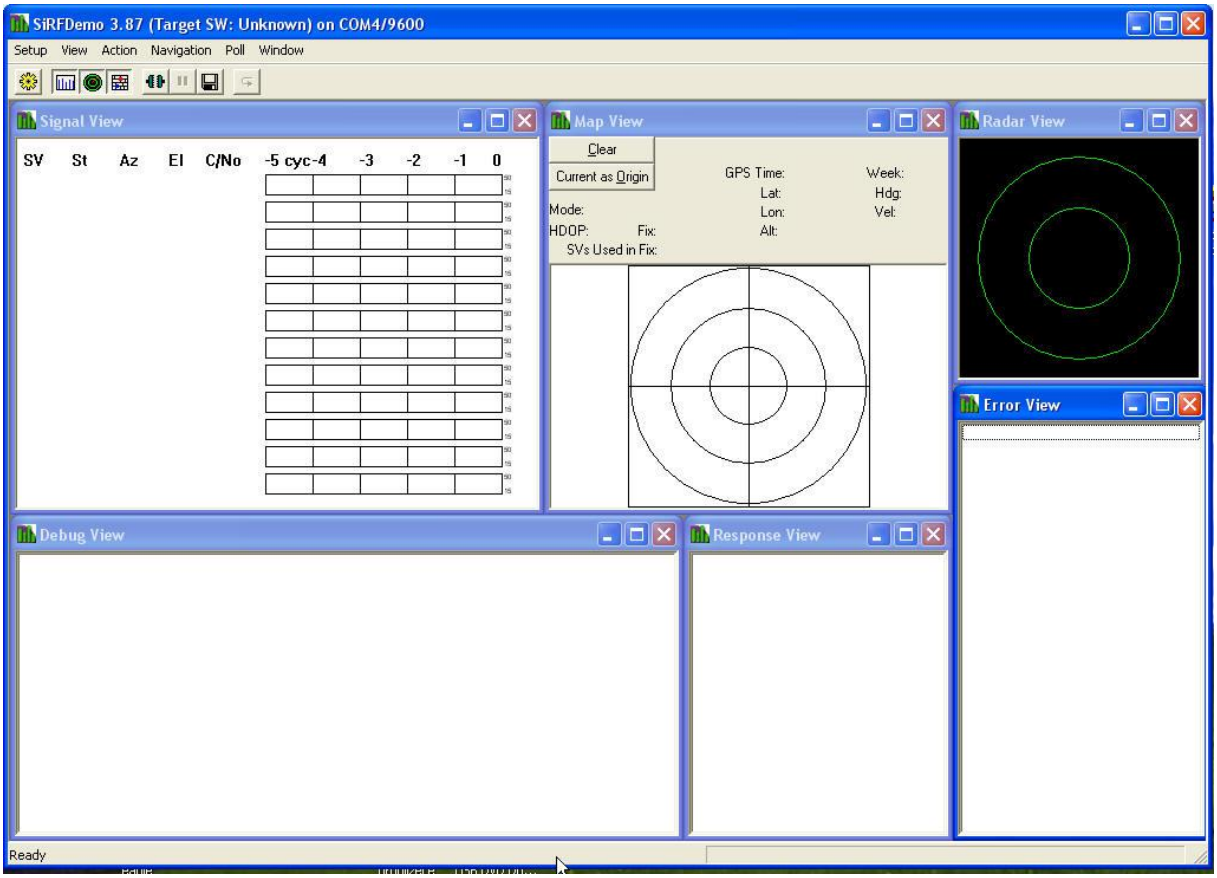
NMEA není jediným protokolem, který můžeme používat při komunikaci s GPS modulem. Další možností je např. GPX, IGC či SiRF protokol .

4.3.2 Testování GPS modulů pomocí PC (GPS modules testing with help of PC)

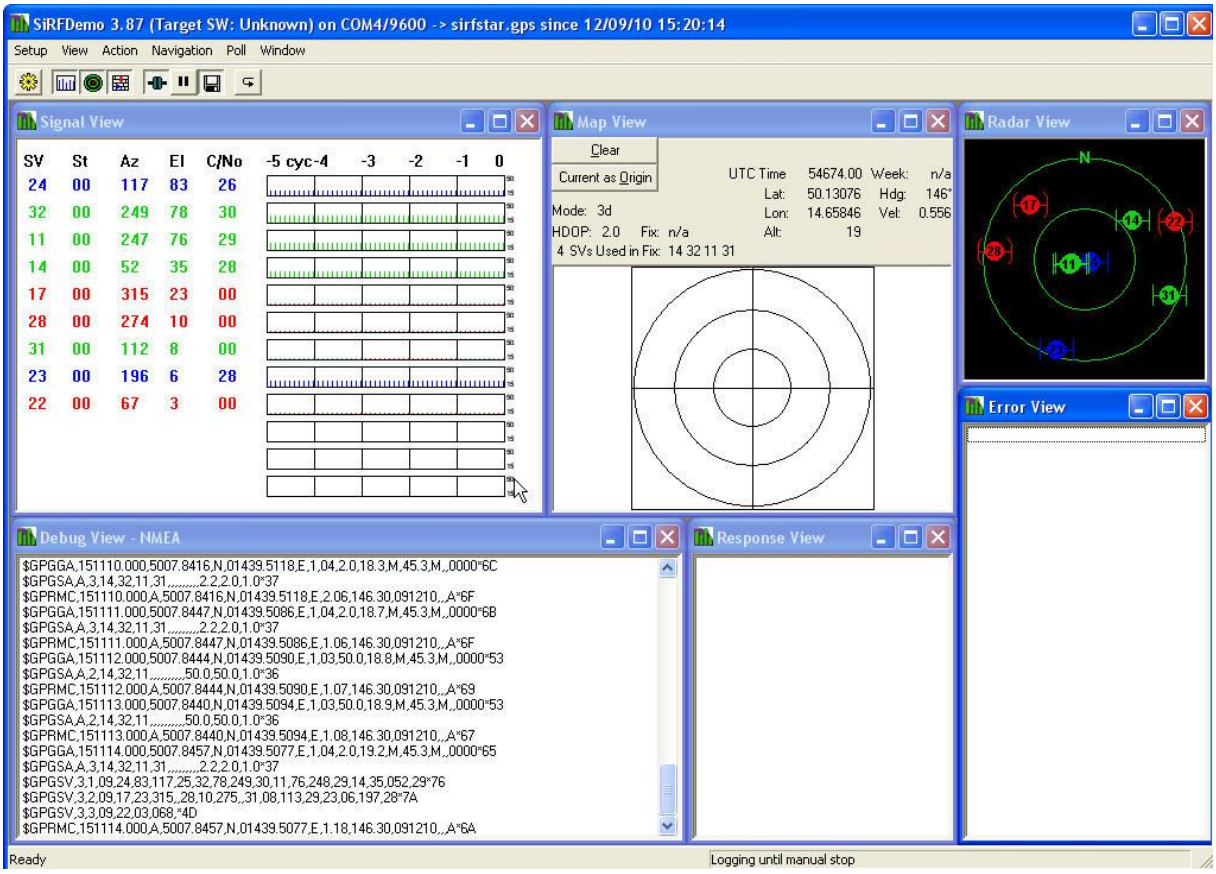
Předtím, než začneme ve firmware palubního počítače programovat komunikaci s modulem GPS, bude dobré si někde odzkoušet funkční komunikaci s GPS moduly. Nejsnažší je k tomuto účelu použít PC a některý free program. Jako příklad použijeme GPS modul Navibe GM720 komunikující NMEA nebo SiRF protokolem. Je SiRF Star III kompatibilní. Software pak SiRF Demo:

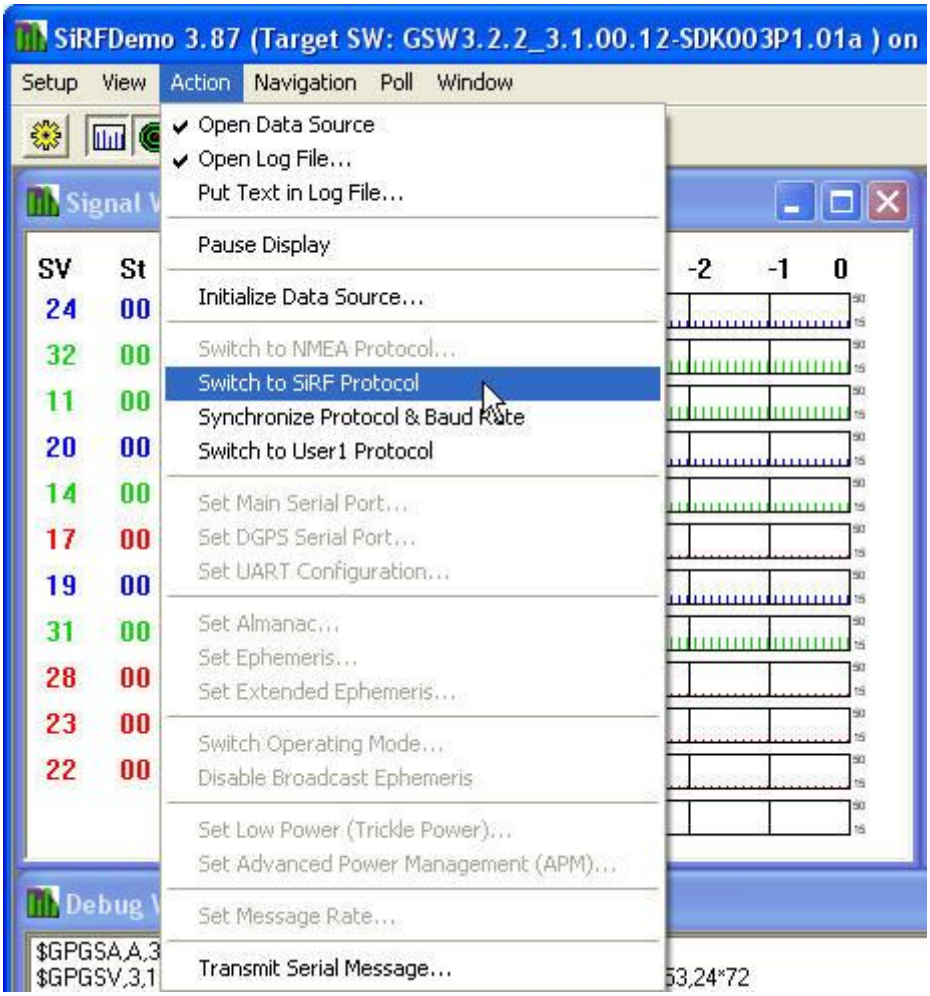


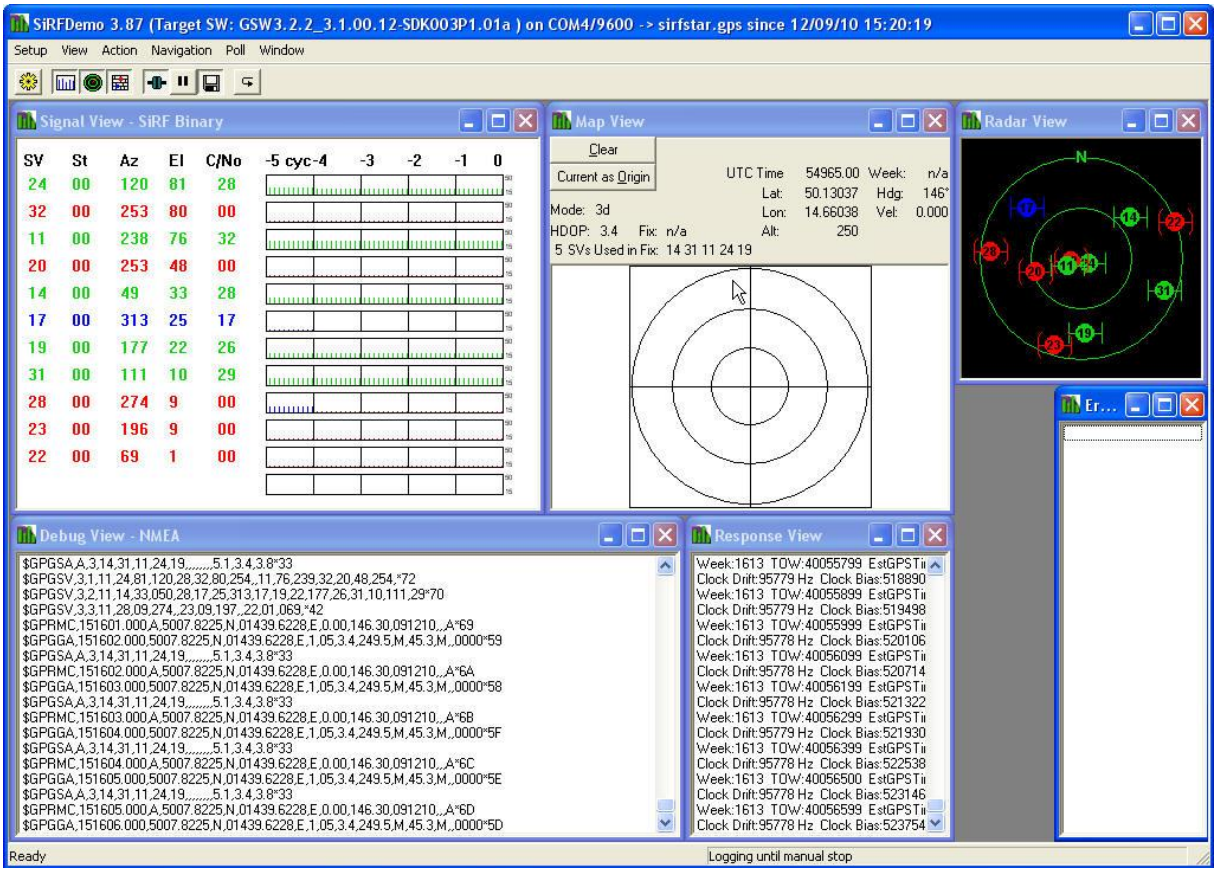




Klikneme na ikonku **Action** v horním menu, po chvíli se objeví např.:


SiRFDemo 3.87 (Target SW: Unknown) on COM4/9600 -> sirfstar.gps since 12/09/10 15:20:14
Setup View Action Navigation Poll Window
Signal View
SV St Az El C/No -5 cyc-4 -3 -2 -1 0
24 00 117 83 26
32 00 249 78 30
11 00 247 76 29
14 00 52 35 28
17 00 315 23 00
28 00 274 10 00
31 00 112 8 00
23 00 196 6 28
22 00 67 3 00
Map View
Clear
Current as Origin
Mode: 3d
HDOP: 2.0 Fix: n/a
4 SVs Used in Fix: 14 32 11 31
UTC Time 54674.00 Week: n/a
Lat: 50.13076 Hdg: 146°
Lon: 14.65846 Vel: 0.556
Alt: 19
Radar View
N
Error View
Debug View - NMEA
$GPGGA,151110.000,5007.8416,N,01439.5118,E,1,04,2.0,18.3,M,45.3,M,,0000*6C
$GPGSA,A,3,14,32,11,31,,,,,,,,,2.2,2.0,1.0*37
$GPRMC,151110.000,A,5007.8416,N,01439.5118,E,2.06,146.30,091210,,,A*6F
$GPGGA,151111.000,5007.8447,N,01439.5086,E,1,04,2.0,18.7,M,45.3,M,,0000*6B
$GPGSA,A,3,14,32,11,31,,,,,,,,,2.2,2.0,1.0*37
$GPRMC,151111.000,A,5007.8447,N,01439.5086,E,1.06,146.30,091210,,,A*6F
$GPGGA,151112.000,5007.8444,N,01439.5090,E,1,03,50.0,18.8,M,45.3,M,,0000*53
$GPGSA,A,2,14,32,11,,,,,,,,,,50.0,50.0,1.0*36
$GPRMC,151112.000,A,5007.8444,N,01439.5090,E,1.07,146.30,091210,,,A*69
$GPGGA,151113.000,5007.8440,N,01439.5094,E,1,03,50.0,18.9,M,45.3,M,,0000*53
$GPGSA,A,2,14,32,11,,,,,,,,,,50.0,50.0,1.0*36
$GPRMC,151113.000,A,5007.8440,N,01439.5094,E,1.08,146.30,091210,,,A*67
$GPGGA,151114.000,5007.8457,N,01439.5077,E,1,04,2.0,19.2,M,45.3,M,,0000*65
$GPGSA,A,3,14,32,11,31,,,,,,,,,2.2,2.0,1.0*37
$GPGSV,3,1,09,24,83,117,25,32,78,249,30,11,76,248,29,14,35,052,29*76
$GPGSV,3,2,09,17,23,315,,28,10,275,,31,08,113,29,23,06,197,28*7A
$GPGSV,3,3,09,22,03,068,*4D
$GPRMC,151114.000,A,5007.8457,N,01439.5077,E,1.18,146.30,091210,,,A*6A
Response View
Ready
Logging until manual stop


SiRFDemo 3.87 (Target SW: GSW3.2.2_3.1.00.12-SDK003P1.01a) on
Setup
View
Action
Navigation
Poll
Window
Open Data Source
Open Log File...
Put Text in Log File...
Pause Display
Initialize Data Source...
Switch to NMEA Protocol...
Switch to SiRF Protocol
Synchronize Protocol & Baud Rate
Switch to User1 Protocol
Set Main Serial Port...
Set DGPS Serial Port...
Set UART Configuration...
Set Almanac...
Set Ephemeris...
Set Extended Ephemeris...
Switch Operating Mode...
Disable Broadcast Ephemeris
Set Low Power (Trickle Power)...
Set Advanced Power Management (APM)...
Set Message Rate...
Transmit Serial Message...
SV
St
-2
-1
0
24
32
11
20
14
17
19
31
28
23
22
Debug
$GPGSA,A,3
$GPGSV,3,1
53,24*72



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

zemepisná šírka: 5007.8020 severní šírky => 50°7,8020'N = 50°7'48,12"N
zemepisná 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

SiRFDemo Version 3.87 log file opened 12/09/10 15:20:19
SiRFDemo Build Date: June 20, 2007

$GPGGA,151055.000,5007.8340,N,01439.5185,E,1,03,50.0,16.6,M,45.3,M,,0000*57
$GPGSA,A,2,14,32,11,,,,,,,,,,50.0,50.0,1.0*36
$GPRMC,151055.000,A,5007.8340,N,01439.5185,E,5.98,146.31,091210,,,A*6E
$GPGGA,151056.000,5007.8323,N,01439.5202,E,1,03,50.0,16.6,M,45.3,M,,0000*5D
$GPGSA,A,2,14,32,11,,,,,,,,,,50.0,50.0,1.0*36
$GPGSV,3,1,09,24,83,117,25,32,78,249,29,11,76,248,28,14,35,052,28*7E
$GPGSV,3,2,09,17,23,315,25,28,10,275,,31,08,113,,23,06,197,*7C
$GPGSV,3,3,09,22,03,068,*4D
$GPRMC,151056.000,A,5007.8323,N,01439.5202,E,5.99,146.58,091210,,,A*6A
$GPGGA,151057.000,5007.8306,N,01439.5219,E,1,03,50.0,16.6,M,45.3,M,,0000*51

my remarks: *CanSat Book for Students* – part.1 2011

```
$GPGSA,A,2,14,32,11,,,,,,,,,50.0,50.0,1.0*36
$GPRMC,151057.000,A,5007.8306,N,01439.5219,E,6.00,146.40,091210,,,A*6C
$GPGGA,151058.000,5007.8296,N,01439.5229,E,1,03,50.0,16.6,M,45.3,M,,0000*55
$GPGSA,A,2,14,32,11,,,,,,,,,50.0,50.0,1.0*36
$GPRMC,151058.000,A,5007.8296,N,01439.5229,E,4.96,146.55,091210,,,A*61
$GPGGA,151059.000,5007.8282,N,01439.5244,E,1,03,50.0,16.5,M,45.3,M,,0000*59
$GPGSA,A,2,14,32,11,,,,,,,,,50.0,50.0,1.0*36
$GPRMC,151059.000,A,5007.8282,N,01439.5244,E,4.97,146.10,091210,,,A*6E
$GPGGA,151100.000,5007.8402,N,01439.5124,E,1,04,2.0,17.5,M,45.3,M,,0000*6E
$GPGSA,A,3,14,32,11,31,,,,,,,,2.2,2.0,1.0*37
```

Popř. v SiRF protokolu:

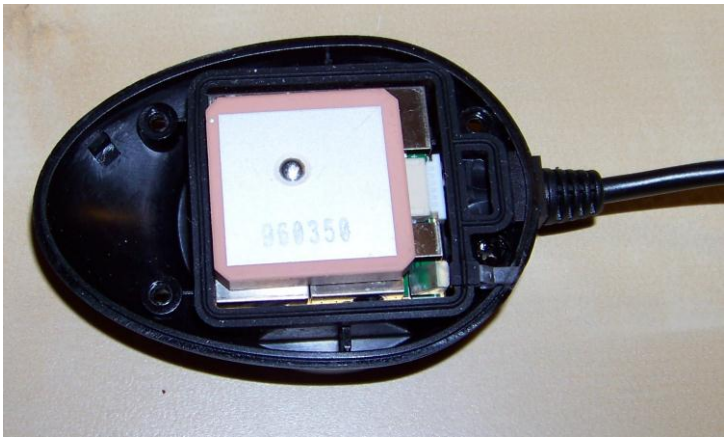
```
$PSRF100... sent to switch to SiRF protocol at 57600
GeoNav PC Time=1291904506 12/09/2010 15:21:46
41,0,532,1613,400354999,2010,12,9,15,12,19999,-
1073732608,501308890,146583594,6917,2386,21,0,28892,0,0,0,632,808,0,0,1185562192,0,182261
4,0,0,0,0,4,9,0
Tx PC Time=1291904506.000
Tx: 0xA0A2000284000084B0B3
4,589,40035499,12,28,274,9,0,0,0,0,0,0,0,0,0,0,0,24,117,83,45,24,24,24,24,23,23,23,23,23,23,14,51,
34,191,27,27,27,27,26,26,26,26,26,26,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,32,250,79,191,32,32,32,31,31,31,31,
31,31,31,11,246,76,191,31,30,30,30,30,30,30,30,30,30,31,112,9,191,27,27,27,27,27,27,27,27,26,26,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,19,177,23,63,27,27,27,26,26,26,26,26,26,26,20,252,47,63,27,26,26,26,26,26
,26,26,26,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
2,3963404,1036701,4872188,0.000,0.000,0.000,20,1.8,2,589,40035500,4,14,31,32,11,0,0,0,0,0,0,0,0
ThrPut(186=1ms): Latency:24576 SegStatMax:31324 AveTrkTime:014 Nav Complete:0000 ms
Week:1613  TOW:40035499  EstGPSTime:400354999 ms  SVCnt:4  Clock Drift:95779 Hz  Clock
Bias:39546098 ns
27,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1613,2,1613,3,1613,4,1613,5,1613,6,1613,7,1613,8,1613,9,1613,10,161
3,11,1613,12,1613
SW Version: GSW3.2.2_3.1.00.12-SDK003P1.01a
Ack: MID_PollSWVersion
```

4.3.3 Návrh a realizace připojení GPS modulu k řídícímu počítači CANSATu
(Design and realizing GPS module connection to CANSAT controll computer)

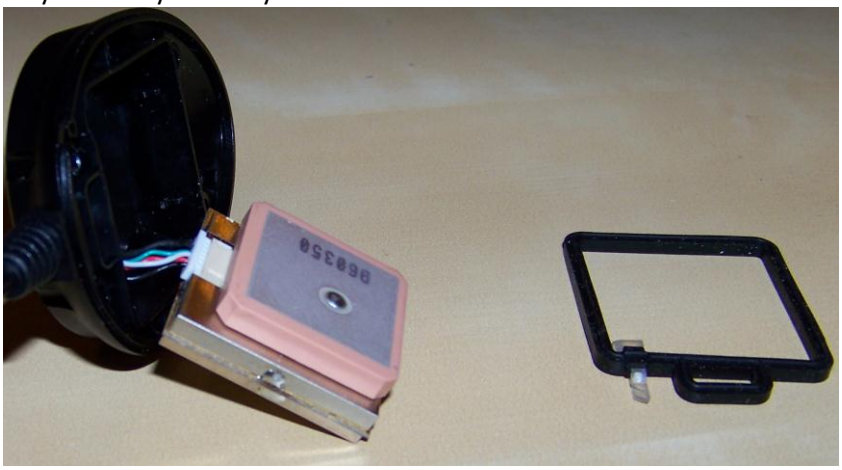
GPS modul Navibe GM720 stejně jako většina externích modulů pro počítače PC se k počítačům PC v současné době (září 2011) připojuje buď pomocí USB konektoru, nebo i prostřednictvím bluetooth. Před několika lety byly k dispozici pro spolupráci s PC i GPS moduly s konektorem PS2. Jeho zapojení se od klasického zapojení konektoru PS2 poněkud lišilo. Pro přenos signálů do/z modulu GPS se používal signály TxD a RxD o úrovni RS232, popř. i signály RxD a TxD s úrovní TTL. Pro naše účely by takový modul byl velice vhodný, neboť většina jednočipových počítačů umožňuje komunikaci pomocí sériových signálů TxD a RxD (s úrovní TTL, popř. nižší, např. 3,3V logikou).

Naštěstí je možné v ceně pod 1000 Kč zakoupit GPS modul Navilock NL-303P GPS PDA Reciever SiRF III.

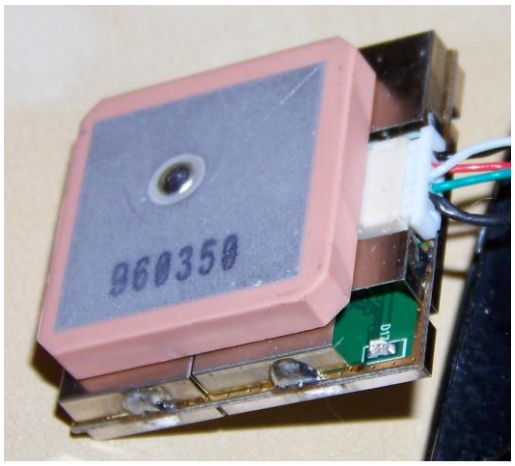
Po vyšroubování 4 šroubků máme přístup k vlastnímu modulu:



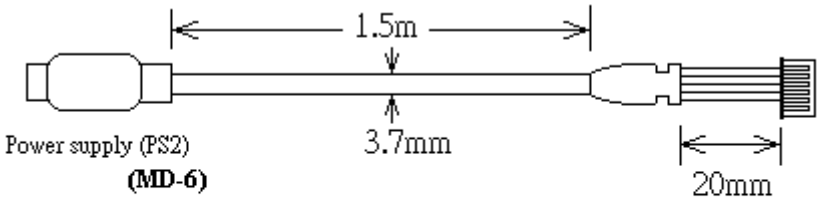
Ještě vyndáme „těsnění“ obsahující i světlovod sloužící k tomu, aby svit zelené LED diody indikující stav GPS modulu byl viditelný i vně krytu.

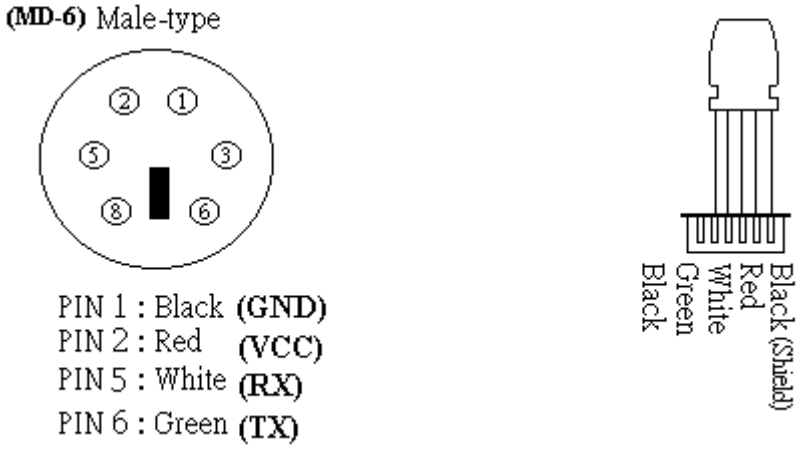


Kryt obsahuje i magnet sloužící k přichycení GPS modulu. Pro použití v CanSatu ovšem nebudeme potřebovat ani kryt, ani těžký magnet.



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





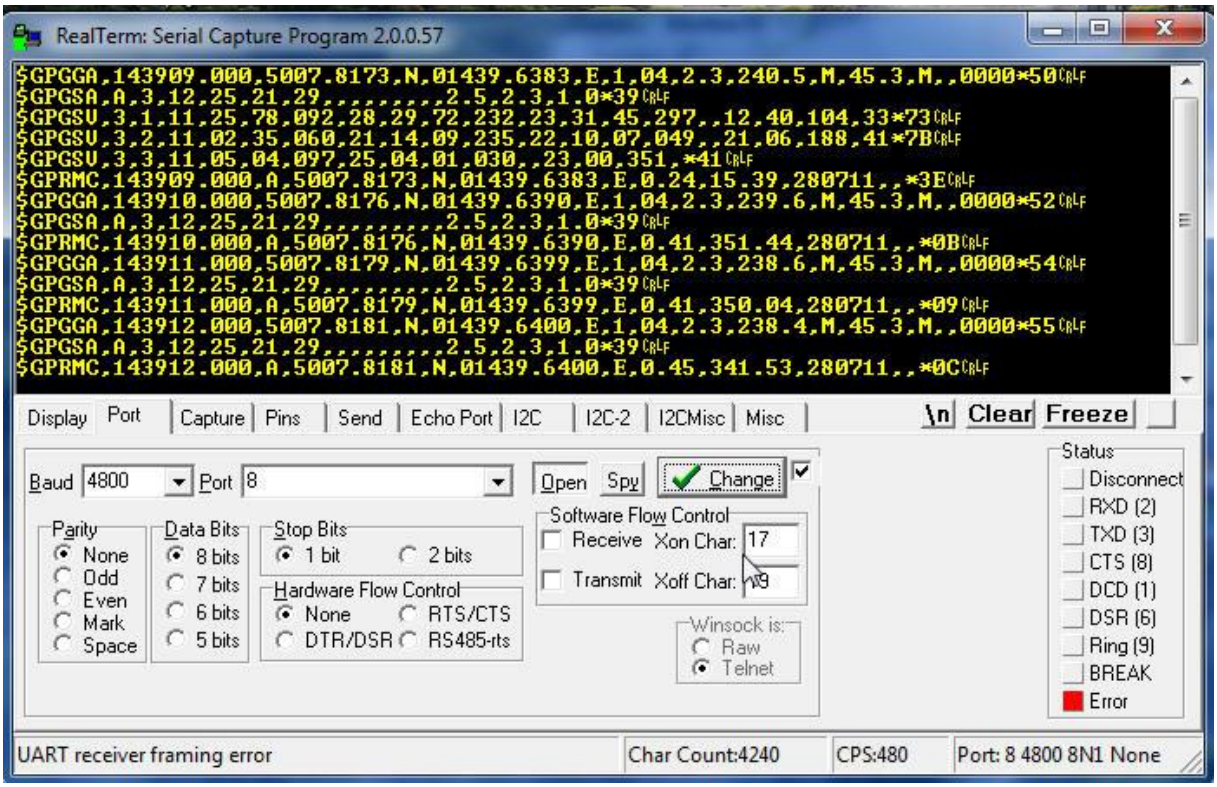
Pozn. 1

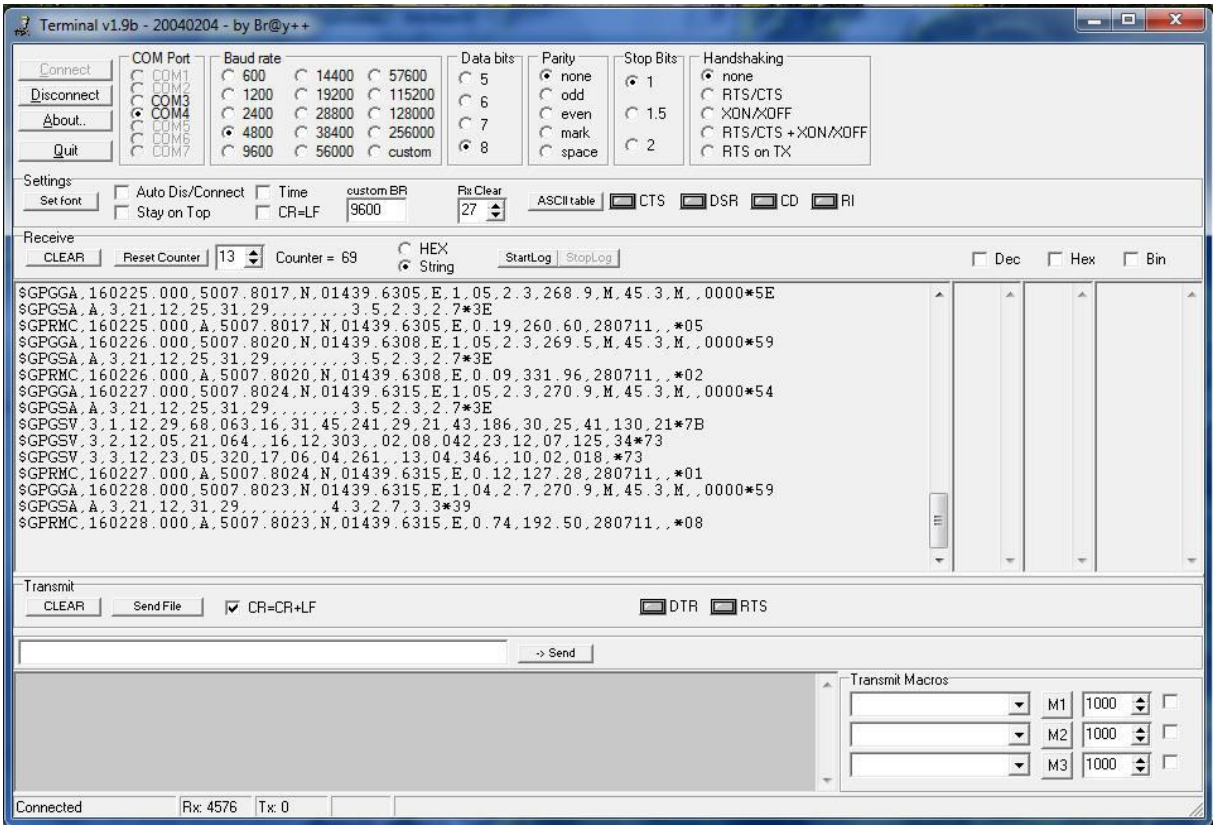
Na fotografii modulu jsou vidět i barvy vodičů přívodního kabelu a skutečně odpovídají výše uvedenému obrázku, tj je mezi nimi i vodič se zelenou barvou (na forografii se zdá, že jde o modrou barvu). Měřením jsme ověřili, že tento modul generuje signál TxD skutečně s úrovní RS232, konkrétně -5V pro logickou jedničku a +5V pro logickou nulu. Komunikace modulu je v defaultním stavu jednosměrná, tj není nutné u tohoto modulu používat jeho vstup RxD.

Po připojení modulu k napájení +5V bliká zelená LED dioda (pokud nesvítí vůbec, je přijímač vypnut). Pokud zachytí signál ze satelitů a vypočítá polohu, bude zelená LED dioda již svítit trvale, ztratí-li signál, bude opět blikat. Po celou dobu vysílá rychlostí 4800 Bd (ASCII, 8 znaků, bez parity) textové řetězce dle NMEA 0183. K otestování této činnosti můžeme připojit modul k sériovému portu COM počítače PC a zachycené textové řetězce zobrazovat některým komunikačním programem. Pro

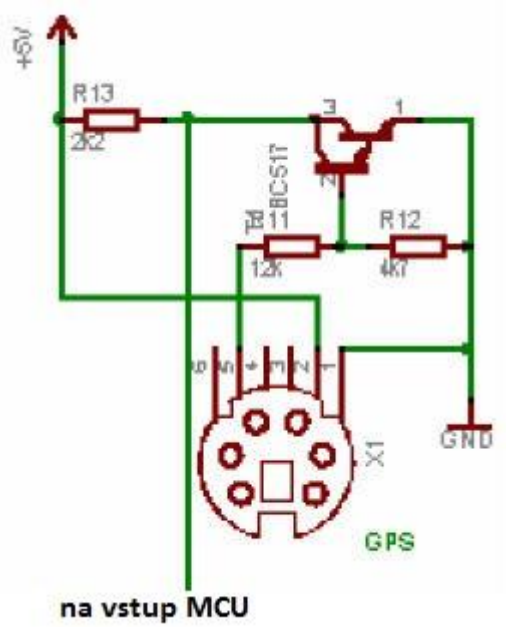
ilustraci jsme použili program Hyperterminál (je součástí instalace windows) a dále dva free programy. Všimněme si nastavení parametrů komunikace:

Pozn. 2
Při realizaci výše uvedených propojení však nic nefungovalo správně. 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 portu jednočipového počítače.

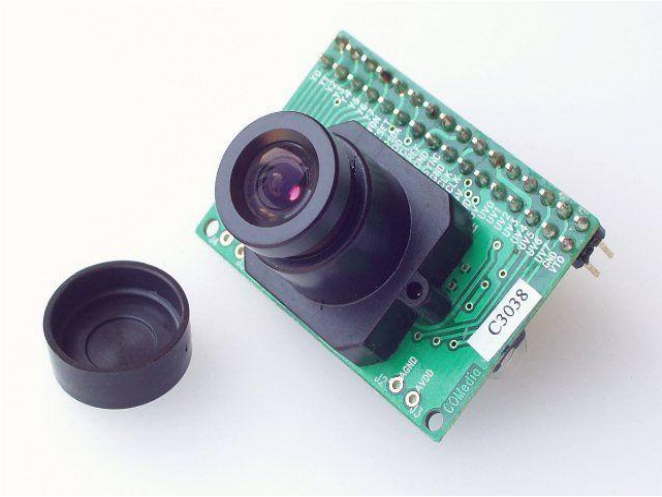




Pro připojení tohoto GPS modulu k palubnímu počítači CanSatu je potřeba provést převod úrovní RS232 na TTL (resp. 3.3 V logiku). V případě obousměrné komunikace se k tomuto účelu běžně používá obvod MAX3232 (nejlépe v SMD provedení). Pro jednosměrnou komunikaci můžeme použít obvod uvedený v bakalářské práci O. Talandy [23].



4.4 Kamera CCD (CCD Camera)
Pro naše první kroky s kamerkou jsme zvolili barevnou (RGB) kamerku C3038

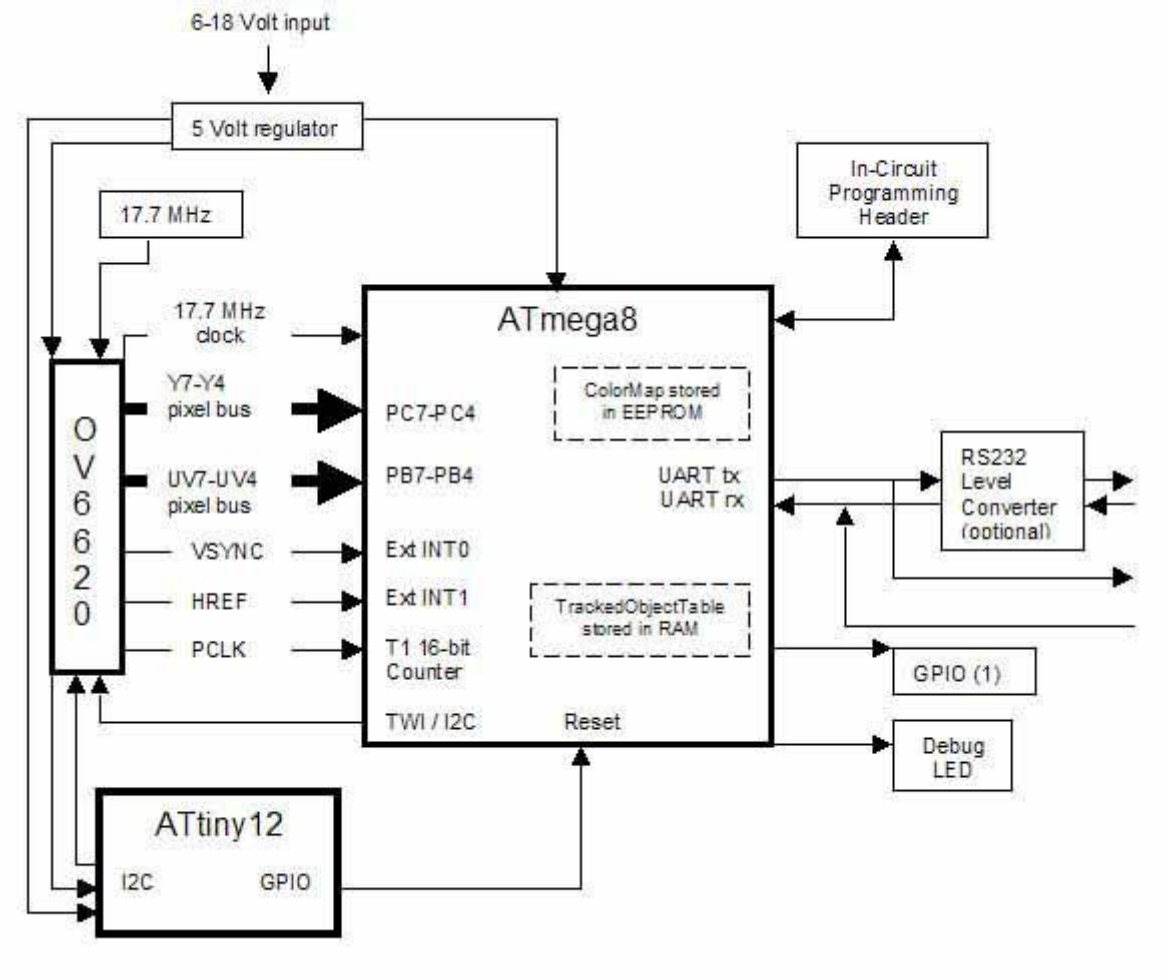


Základní paremetry C3038 :
Čip OmniVision OV6630
Rozlišení 356x292
Napájení 3.3V/16mA
Objektiv f3.6mm, F2.0 s IR filtrem
Zorné pole 43.7° x 25.8°
Rozměry 40x28mm
(pořizovací cena byla 1380 Kč na internetovém obchodu Snail Instruments z Berouna http://shop.snailinstruments.com/index.php)

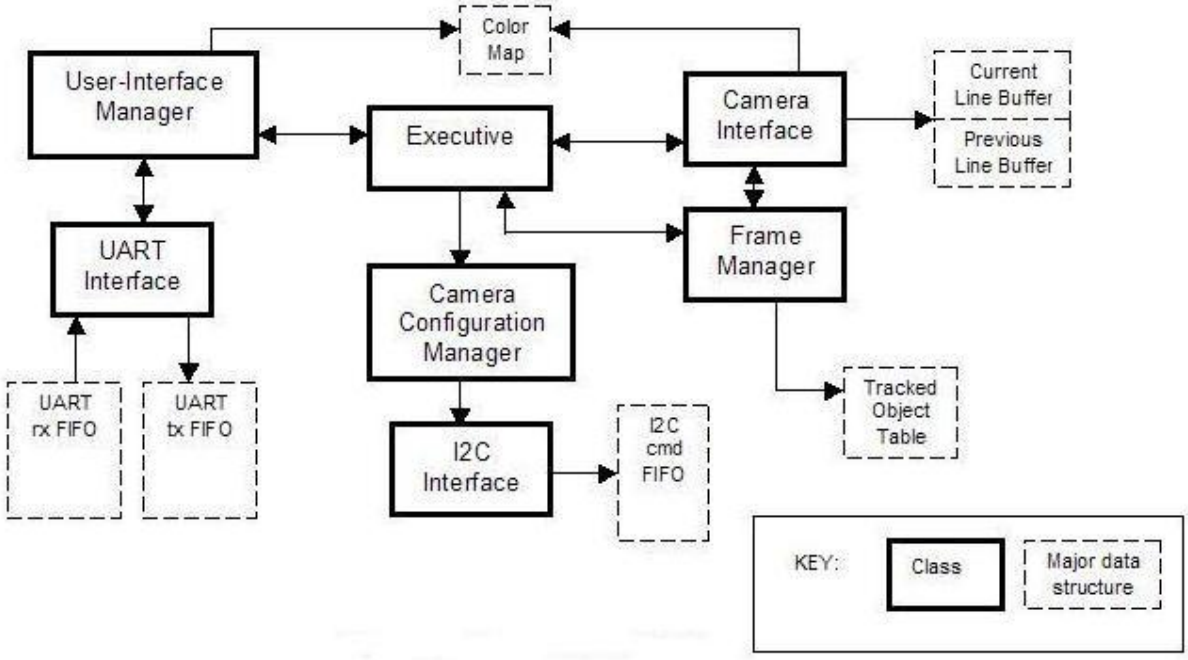
Inspirací k jejímu zapojení a tvorbě hw i sw nám může posloužit projekt AVRcam z JROBOT
http://www.jrobot.net/Projects/AVRcam.html

Pro ilustraci uvádí obrázky AVRcam v.1.1

Blokové schema hardware:

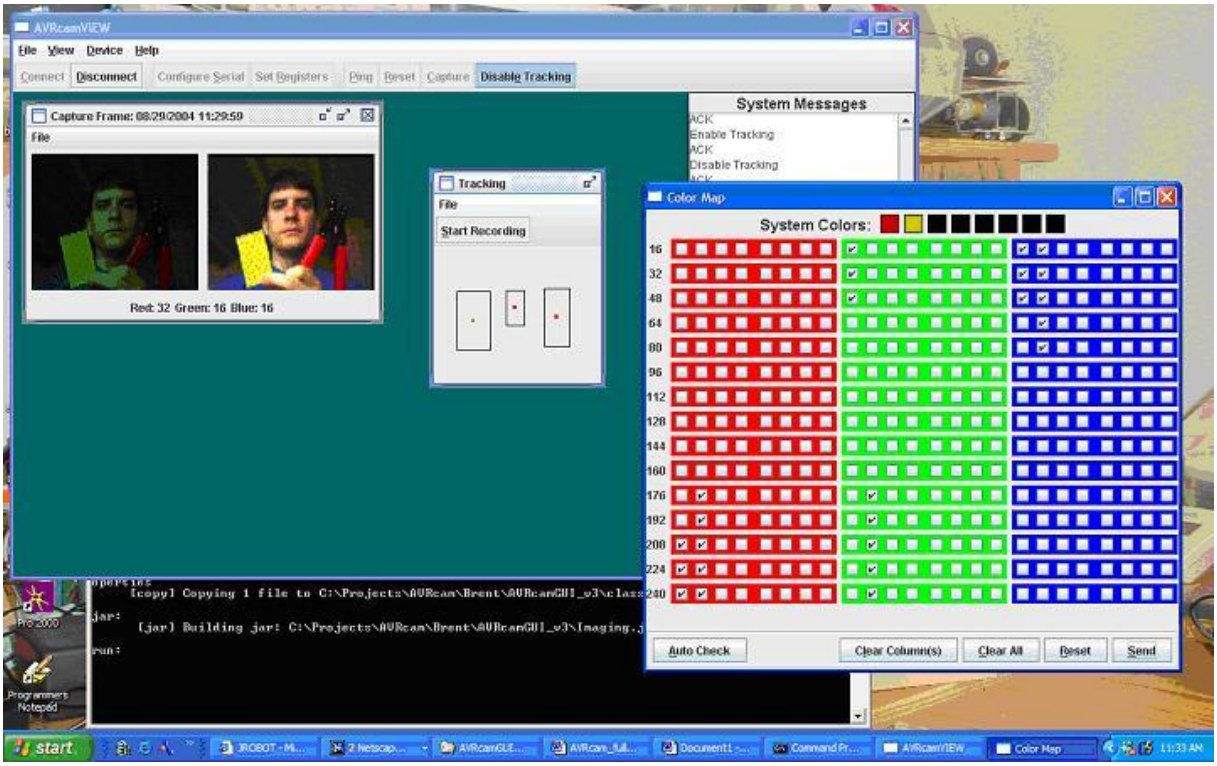


Blokové schema software:



AVRcam VIEW PC Software:

my remarks: *CanSat Book for Students* – part.1 2011



Z http://www.jrobot.net/Download.html si můžeme stáhnou zdrojové kódy aktuální v.1.4 i další konstrukční podklady

5.Palubní počítače (Onboard computers)

Na Internetu můžeme najít řadu reportů studentů pracujících na vývoji Cansatů. Jako řídící počítač obvykle používají ATMega firmy ATMEL. Jistě k tomu přispěl i projekt Open hardware Arduino. ATMega byla použita i ve stavebnicích Pratt Hobies použitých v prvním ročníku Cansat soutěže ESA. Konstrukcí a programováním jednočipových 8bitových ATMEL AVR se zabývá dosti rozsáhlá literatura včetně knih a článků v češtině. Mezi nimi najdeme i knihy [1] až [5] autora těchto skript. Proto je zbytečné se touto problematikou dále zabývat v těchto skriptech.

V současné době jsou k dispozici i výkonné 32 bitové počítače s jádrem ARM. Cenové 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 . Tato firma je i sponzorem SPŠE Ječná. Hardware s STM32 i jeho programováním se zabývá 2.díl tohoto výukového materiálu.

6.Komunikační systém (Communication System)

6.1 Pratt Hobbies vysílač (PrattHobbies Transmitter)

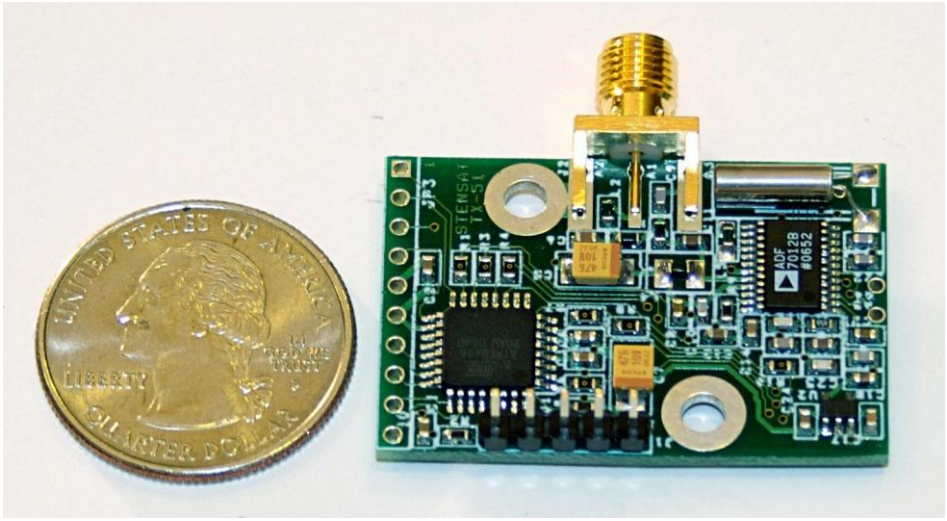
ESA Education Office vyhlásila v lednu 2010 *první evropskou studentskou středoškolskou soutěž* ve stavbě funkčního „minisatelitu“. Soutěž byla určena pro přibližně čtyřčlenné týmy středoškolských studentů z členských zemi ESA a jejich učitele. Pouze na základě přihlášky bylo vybráno 11 týmů, aby si vyzkoušely vyslání svého „minisatelitu“, sestaveného ze speciální sady, sondážní raketou do výše 1 kilometru. Po dosažení dané výšky byly CanSaty z rakety uvolněny a během jejich sestupu na padáku budou provádět měření (teplota, tlak, GPS poloha), které budou studenti v reálném čase zachycovat a vyhodnocovat. CanSaty byly navrženy tak, aby se jejich veškeré vybavení, vč. napájení, vešlo do plechovky (350ml) od nápoje**.** http://www.czechspace.cz/cs/vzdelavani/cansat a http://www.esa.int/SPECIALS/Education/SEMYYTBV34G_0.html .

Úvodní seminář pro učitele vybraných týmů se konal v nizozemském středisku ESA ESTEC ve dnech 12.-13.2.2010. Zde také dostali učitelé stavebnice CanSat Kit od firmy Pratt Hobbies v ceně $259,- http://www.pratthobbies.com/products.asp?cat=10 . Starttkit obsahoval mechanickou konstrukci CanSatu (samostatně prodávaná za $25,- jako *Cansat Hardware*) , desku vysílače (samostatně prodávaný za $89,95 jako *CANSAT transmitter*), desku palubního počítače a desku s čidlem tlaku a teploty. Soutěžící týmy měly složit a zprovoznit tuto stavebnici a výsledný CanSat doplnit o vlastní třetí *misi*. První dvě mise byly *měření teploty* a *měření tlaku*. Právě v návrhu a realizaci této třetí mise se týmy značně lišily.

Jeden tým měl jako třetí misi použití padáčku z ekologického material. Uvážím-li, že padáky se zhotovují z “padákového” hedvábí t.j. produktu bource morušového tak nevím, co je na nich neekologické. Naopak vítězný tým Eclipse z britské školy jako třetí misi měl GPS modul a akcelerometr http://arctanb.wordpress.com/2010/08/21/esa-cansat-competition-2010 .

Sestavit stavebnici CanSat Kit od Pratt Hobbies je směšně jednoduché zvláště pro studenty odborných škol. Uvážíme-li (na naše poměry) ještě značnou cenu startkitu, může být vlastní vývoj a realizace CanSatu pro studenty přínosnější. Pro účast v soutěžích pak bude dobré, bude-li funkčně ekvivalentní s CanSatem od Pratt Hobbies. Toho lze docílit tím, že požijeme hotový vysílač od Pratt Hobbies nebo vyrobíme s ním funkčně ekvivalentní vysílač vlastní. Jako palubní počítač můžeme použít prakticky jakýkoli jednočipový počítač k němuž napíšeme program zpracovávající signály z čidel a komunikující s vysílačem pomocí příkazů tohoto vysílače. Tyto příkazy jsou popsány v datasheetu vysílače Pratt Hobbies a bude se jimi muset řídit i funkčně ekvivalentní vlastní vysílač. Proto si nejprve popíšeme hardware i software originálního vysílače. Vysílač použitý v první evropské středoškolské soutěži vysílal v radioamatérském pásmu 70 cm (433 MHz) úzkopásmovou FM rychlosti 1200baud a protokolem AX25.

6.1.1 Hardware

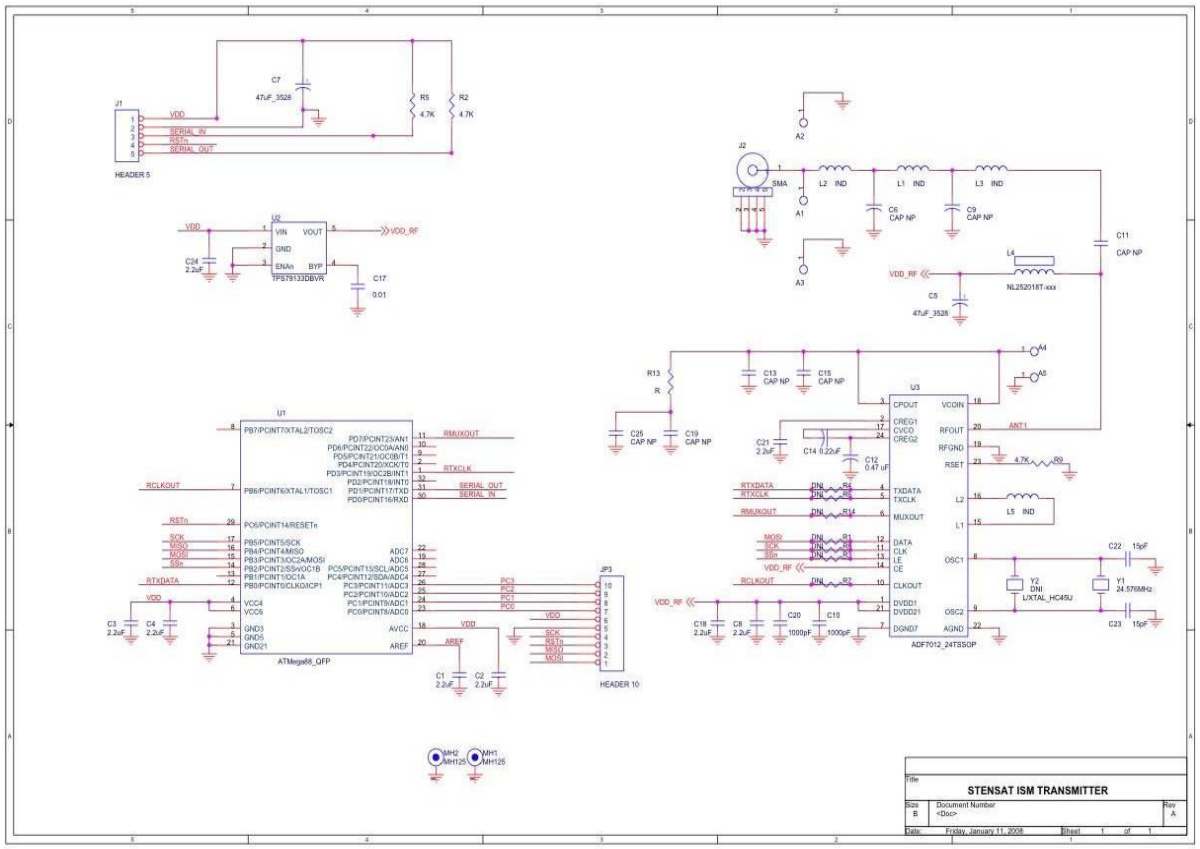


Vysílač vyráběný firmou *Pratt Hobies* byl navržen organizaci **The Stensat Group** http://www.stensat.org . V této organizaci byl rovněž vytvořen firmware.
Deska obsahuje dva integrované obvody. Jednak je to vlastní vysílač od firmy *Analog Devices ADF7012*. Jde o jednočipový UHF vysílač s možností FSK/GFSK/OOK/GOOK/ASK modulace a pracující na kmitočtech 75MHz až 1GHz. Jeho typické užití je pro vysílání v pásmech **ISM**. Jako příklad jsou uváděna pásma 315MHz, 433MHz (což je radioamatérské pásmo 70cm), 868 MHz a 915 MHz. Napájení obvodu je 2,3 až 3,6V při odběru 10 až 21mA. Výstupní výkon je programově nastavitelný na −16 dBm až +14 dBm, v 0.4 dB krocích ( ). Rychlost datového přenosu je max. 179,2 kb/s. Obvod *ADF7012* stojí cca $2, ale

lze ho získat zdarma na http://www.analog.com jako tzv. free samples.

Programové řízení obvodu *ADF7012* stejně jako vytváření paketů dle AX25 a jejich posílání do obvodu *ADF7012* a komunikaci desky s okolím zajišťuje jednočipový 8 bitový počítač ATMEL *Atmega88*. Firmware pro tento obvodu najdete na DVD příloze. Jednočipový počítač *Atmega88* je naprogramován tak, že komunikuje s okolím sériovým signálem (na TTL úrovni) 38,4Kbaud.Výkon vysílače je naprogramován na 6 až 10dBm. (4 až 10mW) (pozn 0dBm÷1mW, 1dBm÷1,2589mW, 10dBm÷10mW.)

V datasheetu vysílače je publikované **schema**:



Zapojení konektorů je:

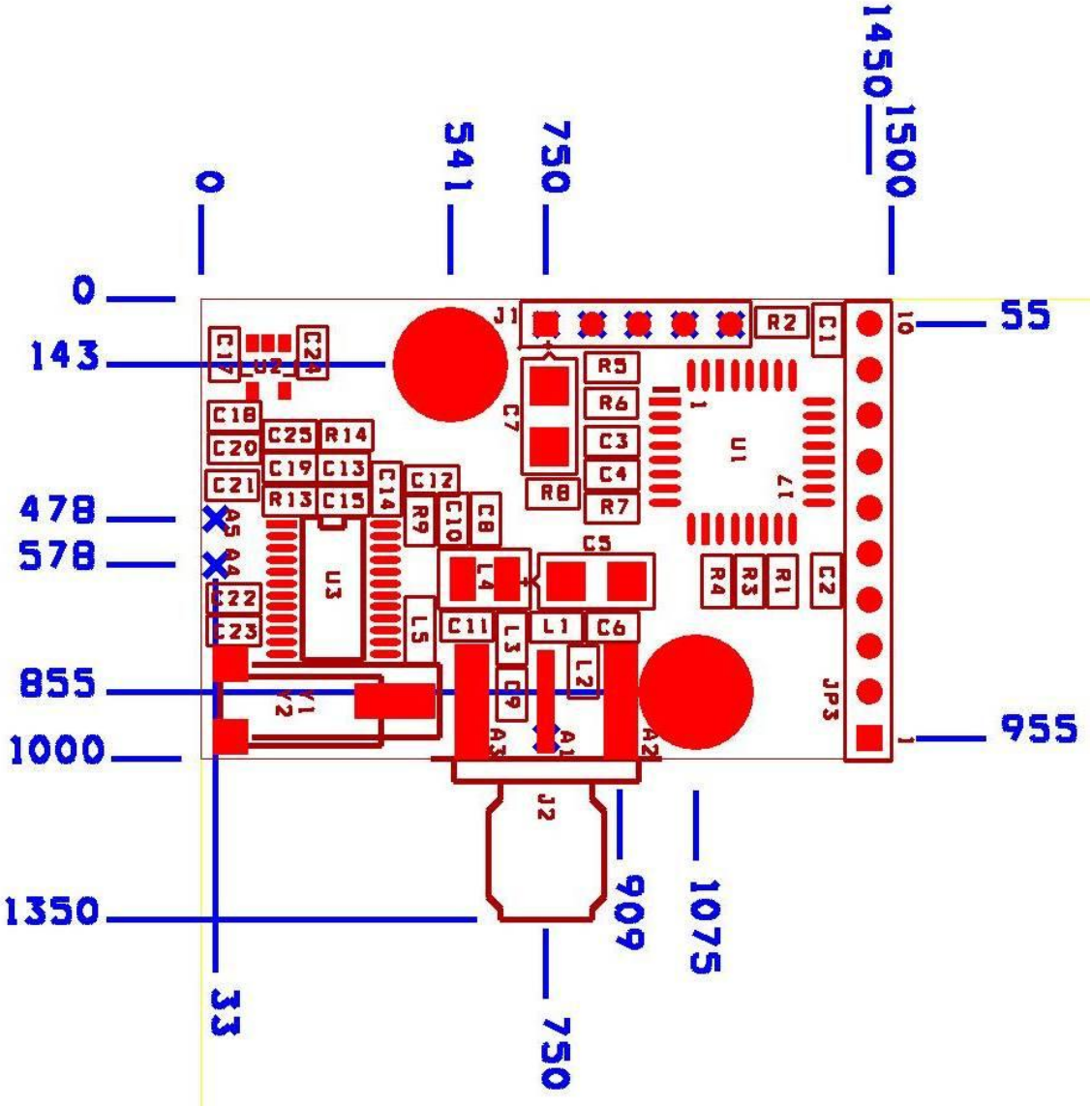
5 Pin Header (J1)

Pin	Description
1	VDD, Positive supply, 3.3 to 5 volts
2	Ground
3	Serial Data In, 38.4Kbaud, 8 bit, no parity, one stop bit
4	Reset (active low)
5	Serial Data Out, 38.4Kbaud, 8 bit, no parity, one stop bit

10 Pin Header (JP3)

Pin	Description
1	MOSI, Digital IO or SPI Data output
2	MISO, Digital IO or SPI Data input
3	RST, reset (active low)
4	SCK, Digital IO or SPI clock
5	Ground
6	VDD, positive supply, 3.3 to 5 volts, same as pin1 on 5 pin header
7	PC0, Digital IO or ADC input 0
8	PC1, Digital IO or ADC input 1
9	PC2, Digital IO or ADC input 2
10	PC3, Digital IO or ADC input 3

Dále je tam uvedeno i **rozložení součástí** :



Vysílač je vyráběn ve dvou provedeních

- s konektorem
- s drátovou anténou λ/4 tj. cca 17cm (f= 433MHz tj. λ = 69,2 cm, λ/4=17,3cm)

6.1.2 Software

6.1.2.1 Komunikace s vysílačem *Pratt Hobbies* (PrattHobbies Transmitter Communication)

V datasheetu vysílače je uvedena následující tabulka příkazů:

Command Set

Command	Description	Format
C	Set source Call sign	Ccccccc<CR>
D	Set Destination call sign	Ddddddd<CR>
V	Set Via (relay) call sign	Vvvvvvv<CR>
S	Send ASCII String	Sssssssssss…sssss<CR>
F	Set the Frequency	Fffff<CR>
M	Set the bit-rate Mode	M1200<CR> or M9600<CR>

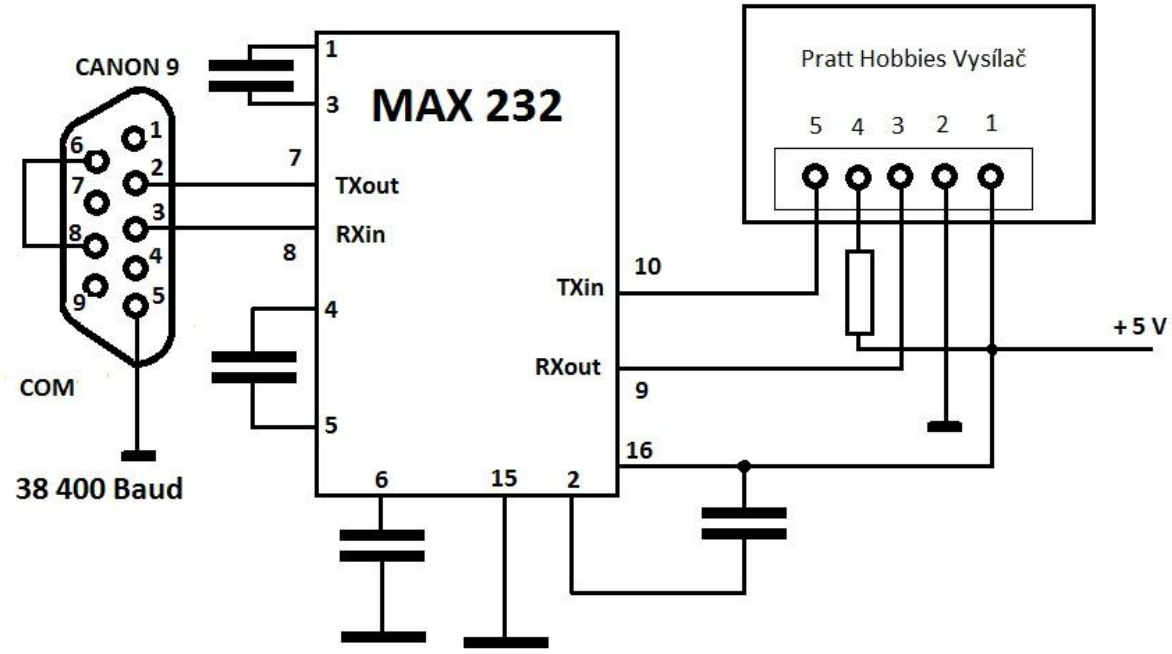
Pro nastavení frekvence (příkaz **F**) ještě využijeme tabulku:

Appendix B: Transmitter Frequencies

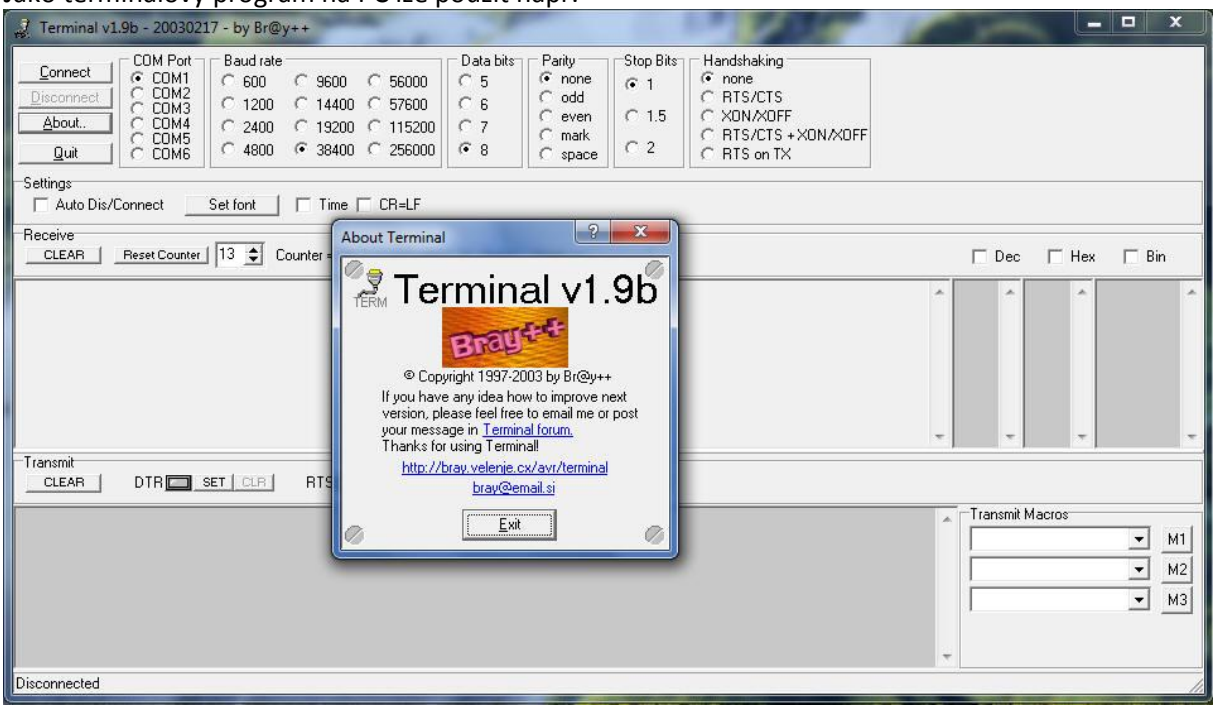
Frequentie (MHz)	Code	Frequentie (MHz)	Code
433,050	F8CF79	433,950	F8D429
433,100	F8CFBD	434,000	F8D46D
433,150	F8CFFD	434,050	F8D4AD
433,200	F8D03D	434,100	F8D4F1
433,250	F8D07D	434,150	F8D535
433,300	F8D0CD	434,200	F8D575
433,350	F8D10D	434,250	F8D5B9
433,400	F8D14D	434,300	F8D5FD
433,450	F8D18D	434,350	F8D63D
433,500	F8D1D1	434,400	F8D681
433,550	F8D215	434,450	F8D6C5
433,600	F8D255	434,500	F8D705
433,650	F8D299	434,550	F8D749
433,700	F8D2D7	434,600	F8D78D
433,750	F8D31D	434,650	F8D7CD
433,800	F8D361	434,700	F8D811
433,850	F8D3A5	434,750	F8D855
433,900	F8D3E5	434,800	F8D895

Pozn.: v této tabulce je pro kmitočet 433.700 MHz *chybná hodnota* (tabulka převzata z [14]).

Tyto údaje stačí pro základní pokusy s vysílačem. (ovšem k tomu, abychom mohli pozorovat výsledek již potřebujeme nějaký FM přijímač pro pásmo 433MHz). Do vysílače budeme posílat příkazy v sériovém tvaru, které vytvoříme v PC. Mezi sériový výstup počítače PC (COM1, COM2 nebo USB s převodníkem USB/COM) a vstup vysílače *Pratt Hobbies* musíme zapojit převodník úrovní RS232 / TTL např. realizovaný obvodem *MAX232*.

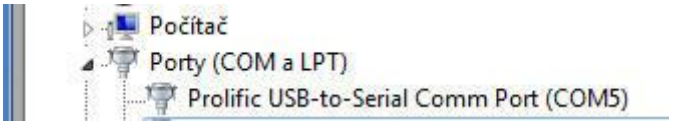


Jako terminálový program na PC lze použít např.

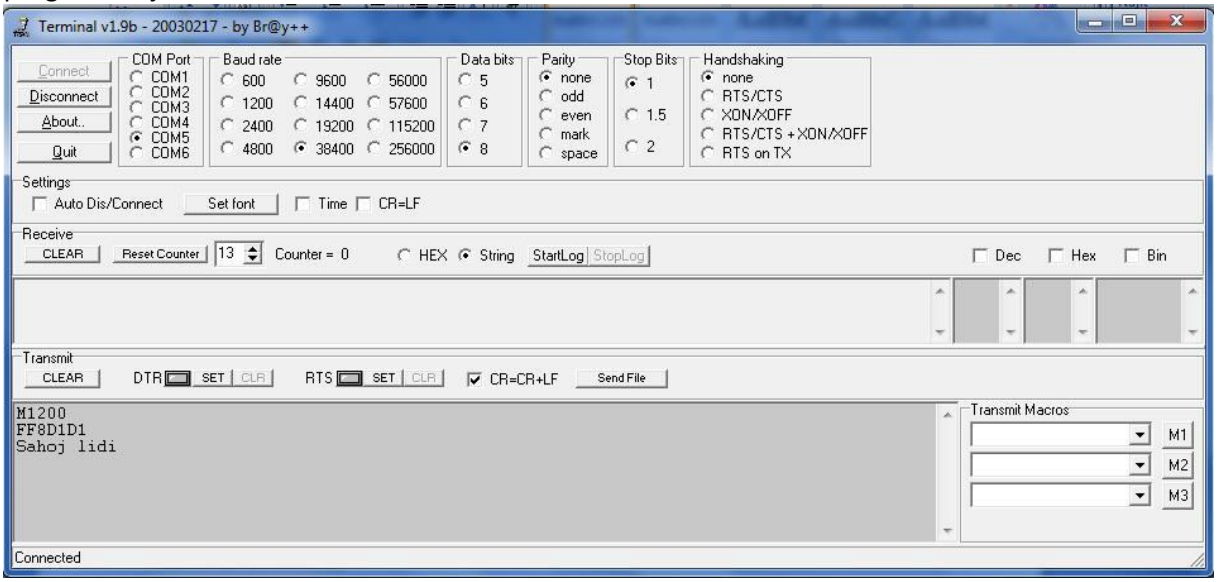


U tohoto programu nastavíme **Baud rate** na 38400, **COM Port** nastavíme na ten port, ke kterému máme připojen přes *MAX232* vysílač *Pratt Hobbies*, popř. virtuální sériový port v případě připojení

přes USB a příslušný převodník USB na sériový port. Poté klikneme na tlačítko **Connect** a následně můžeme posílat příkazy do vysílače *Pratt Hobbies*:
Odešleme **M**1200 **[CR]** čímž nastavíme rychlost 1200baud
Poté pošleme **F**8D1D1 **[CR]** tím nastavíme kmitočet *ADF7012* na 433,500 MHz
Dále pošleme **S**ahoj lidi **[CR]** tím odešleme řetězec znaků *ahoj lidi* , k odvysílání v datové části paketu AX25
Protože můj notebook nemá sériový port, použil jsem USB a převodník USB na sériový port



Virtuální port mám nastaven jako COM5, takže odeslání tří výše uvedených příkazů terminálovým programem je:

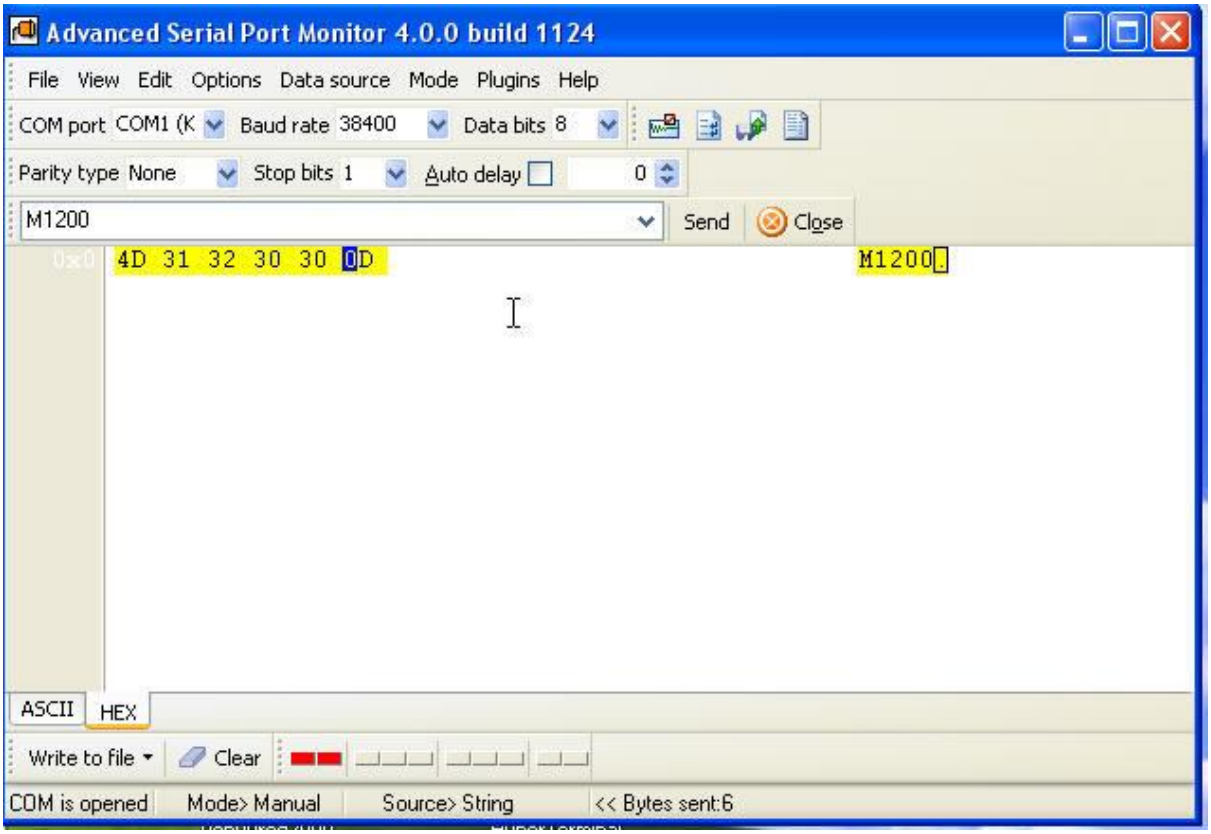


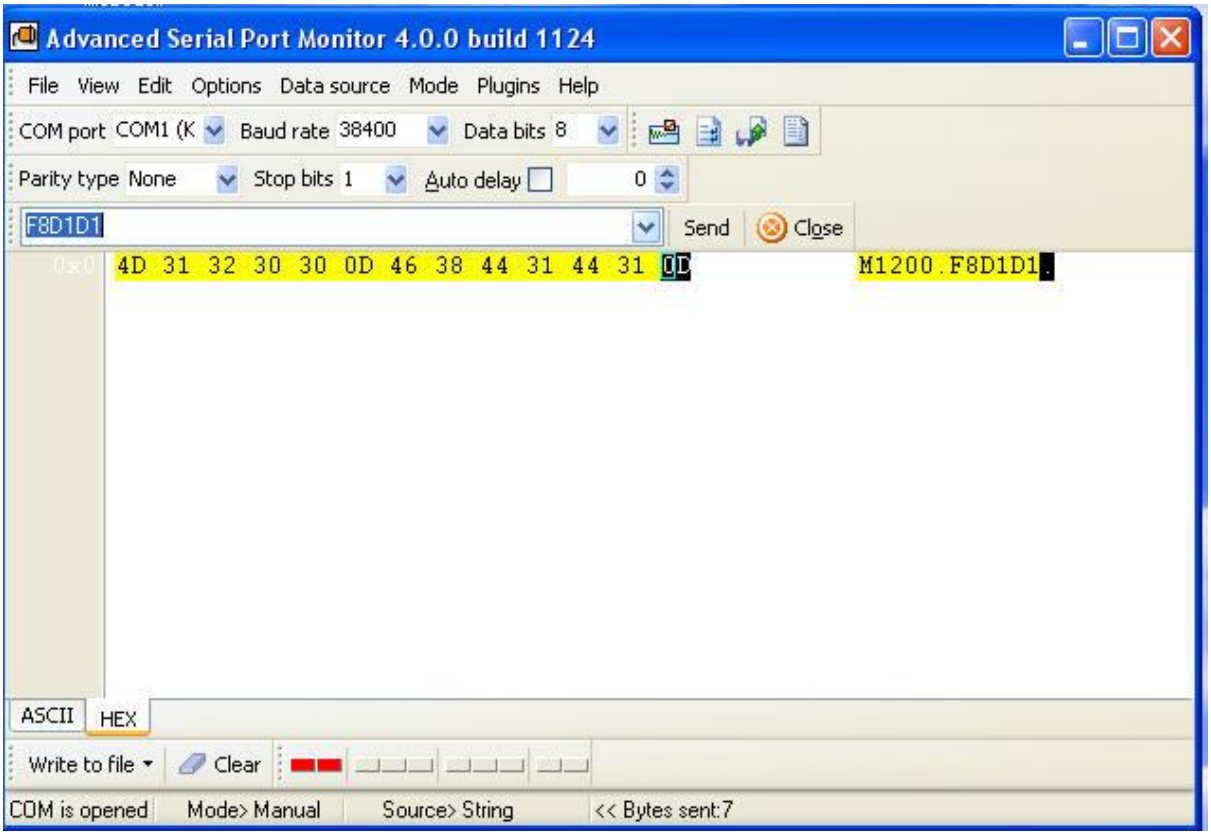
.

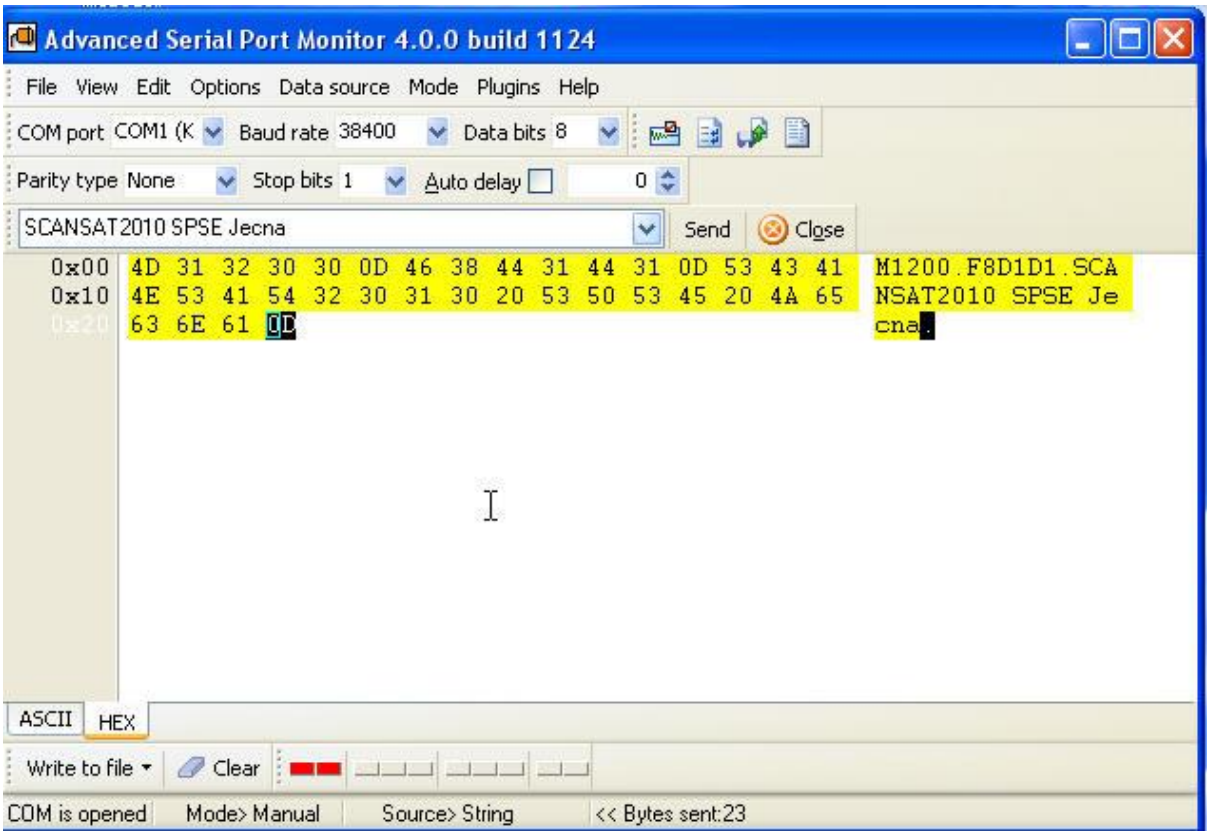
K tomu, aby vysílač *Pratt Hobbies* chápal přijatý řetězec znaků jako příkaz **je nutné**, aby za ním byly ještě poslán znak CR, tj. **0x0D** v hexadecimálním kódu. Ideální je si příkazy i znaky CR předem připravit do nějakého souboru a jeho obsah odvysílat kliknutím na tlačítko **SendFile** a následným výběrem tohoto souboru.

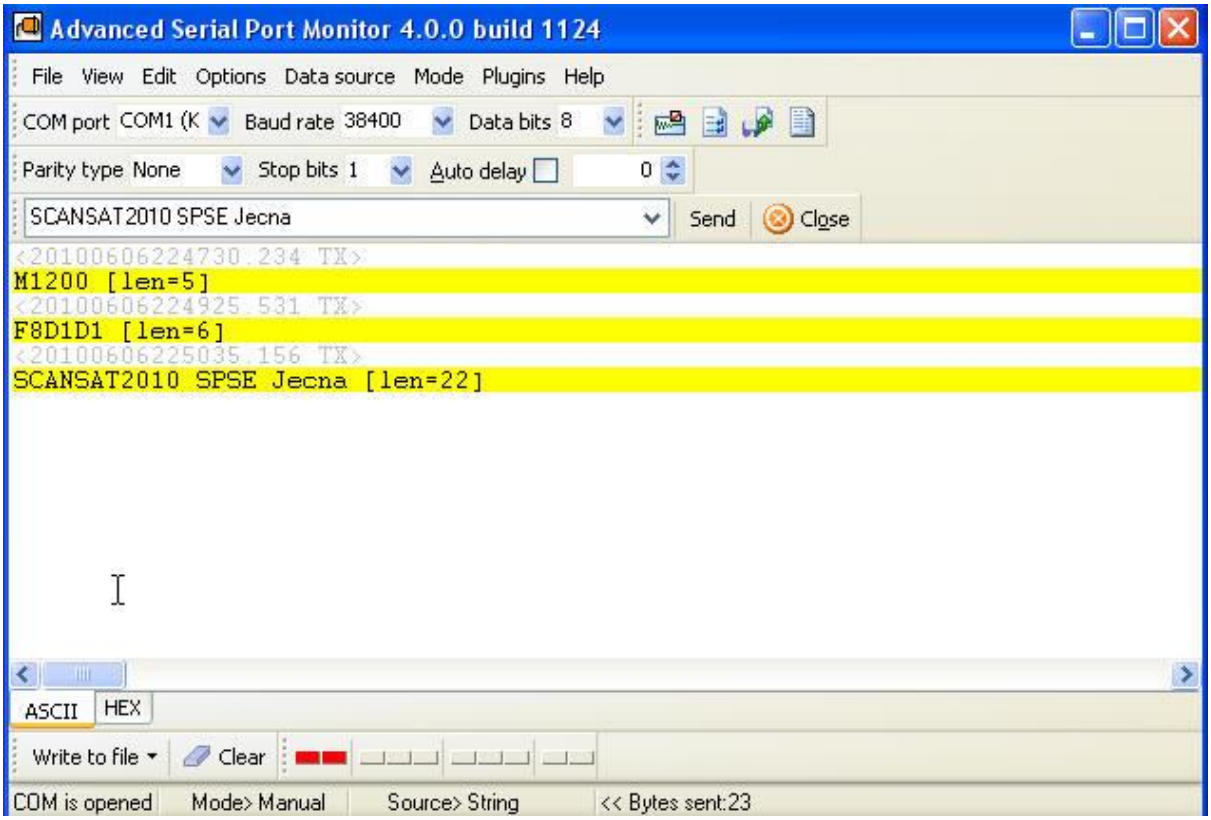
Lze použít i jiný sw, např **Advanced Serial Port Monitor** z http://www.aggsoft.com/serial-port-monitor/download.htm


UNREGISTERED
Advanced
Serial Port
Monitor
Watch It!
Version 4.0.0 build 1124
© Copyright 1999-2010, AGG Software
Home page
Information about registration on our site
Send your comments and questions to our e-mail
OK


Advanced Serial Port Monitor 4.0.0 build 1124
File View Edit Options Data source Mode Plugins Help
COM port COM1 (K Baud rate 38400 Data bits 8
Parity type None Stop bits 1 Auto delay 0
M1200
Send Close
4D 31 32 30 30 0D
M1200
ASCII HEX
Write to file Clear
COM is opened Mode> Manual Source> String << Bytes sent:6


Advanced Serial Port Monitor 4.0.0 build 1124
File View Edit Options Data source Mode Plugins Help
COM port COM1 (K Baud rate 38400 Data bits 8
Parity type None Stop bits 1 Auto delay 0
F8D1D1
Send Close
0x0 4D 31 32 30 30 0D 46 38 44 31 44 31 0D
M1200.F8D1D1.
ASCII HEX
Write to file Clear
COM is opened Mode> Manual Source> String << Bytes sent:7


Advanced Serial Port Monitor 4.0.0 build 1124
File View Edit Options Data source Mode Plugins Help
COM port COM1 (K Baud rate 38400 Data bits 8
Parity type None Stop bits 1 Auto delay 0
SCANSAT2010 SPSE Jecna
Send Close
0x00 4D 31 32 30 30 0D 46 38 44 31 44 31 0D 53 43 41
M1200.F8D1D1.SCA
0x10 4E 53 41 54 32 30 31 30 20 53 50 53 45 20 4A 65
NSAT2010 SPSE Je
0x20 63 6E 61 0D
cna.
ASCII HEX
Write to file Clear
COM is opened Mode> Manual Source> String << Bytes sent:23



Práce s tímto programem (free demo Advanced Serial Port Monitor) je z výše uvedených obrázků zřejmá. Po nastavení sériového portu klikneme na tlačítko **Open** (napravo od tlačítka **Send**). Tlačítko **Send** se změní na tlačítko **Close**. Příkazy píšeme do texboxu a odesíláme tlačítkem **Send**. Tím se automaticky kromě napsaného textu odešle i 0x0D. Některé terminálové programy znak "návrat vozíku" neposílají a spolupráce s blokem vysílače tak nenastane.

V **CanSat**u s vysílačem *Pratt Hobbies* komunikuje tímto způsobem palubní počítač. Ukažme si, jak bude např. vypadat program v jazyce *Code Vision AVR C* [1], jimž je vytvořen program pro ATMEL AVR jednočipový počítač *AT90S2313* :

```
#include <90s2313.h>
#include <stdio.h>
#include <Delay.h>

void main(void)
{
PORTB=0x00;
DDRB=0x00;
PORTD=0x00;
DDRD=0x00;
TCCR0=0x00;
TCNT0=0x00;
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
OCR1H=0x00;
OCR1L=0x00;
GIMSK=0x00;
```

```
MCUCR=0x00;
TIMSK=0x00;
UCR=0x08;     // 38400 baud rate
UBRR=0x05;
ACSR=0x80;
delay_ms(200);
while(1)
      {
          printf("M1200\r");
          delay_ms(2);
          printf("F8D1D1\r");
          delay_ms(2);
          printf("ahoj lidi\r");
          delay_ms(2);
      }
}
```

V reálném firmware palubního počítače ovšem místo řetězce ahoj lidi bude naprogramováno posílání naměřených hodnot teploty, tlaku, GPS souřadnic apod. Rovněž bude naprogramována volací značka, jiný kmitočet vysílače apod. Pravděpodobně bude použit i jiný jednočipový počítač např. nějaká Atmega firmy ATMEL či v našem případě počítač s jádrem ARM Cortex od STMicroelectronics.

6.1.2.2 Komunikace Packet Rádio AX25 (Packet Radio AX25 Communication)

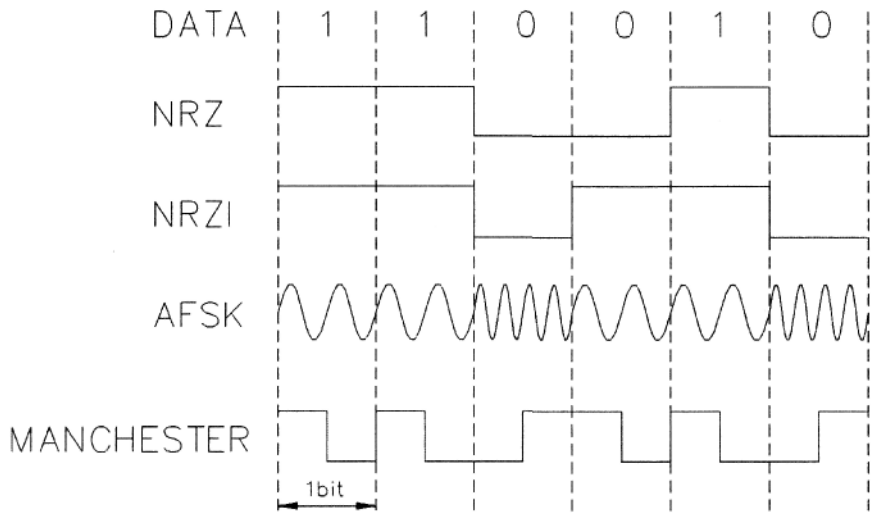
V předchozí části jsme několikrát zmínili protokol **AX25** či **Packet Rádio**. V této kapitole si tyto pojmy vysvětlíme. *Jejich podrobná znalost však není nezbytně nutná a tak je tato kapitola spíše jen informativni.*

Od doby počátků rozhlasového vysílání před téměř 100 lety, kdy tato činnost byla především experimentální, docházelo k pokusům o vzájemnou radiovou komunikaci mezi soukromými osobami. Postupem času (především zásluhou radioamatérů v USA) se tato činnost stále více rozšiřovala, stala se legální a je od té doby uznávána Telekomunikačními úřady jednotlivých zemí i Mezinárodní telekomunikační unií. Radioamatéři mají pro svou činnost přiděleny určité kmitočtové úseky především v pásmu krátkých a ultrakrátkých vln jako tzv. radioamatérská pásma. Pro nás je důležité to, že některá z těchto pásem (433MHz, 2400 MHz) jsou použitelná i používaná pro vysílání dat z CanSatů.

Radioamatéři pro svá radiová spojení používali (a dodnes i používají) telegrafní přenos Morseovou abecedou i fonický přenos (modulací AM, nyní spíše USB a FM) pro přenos zvukových signálů – lidského hlasu komunikujících radioamatérů. Později rozšířili svůj provoz i o přenos textových zpráv prostřednictvím radiodálnopisu.

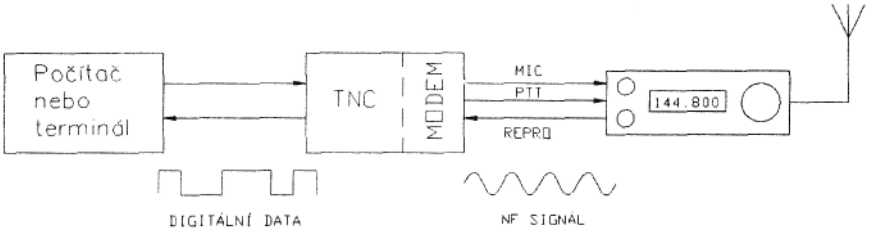
S rozvojem přenosu datových signálů a vznikem počítačových sítí se tyto technologie staly předmětem zájmu i radioamatérů a tak se od počátku 90.let rozšiřuje i o digitální druhy provozu. Především je to tzv. **packet rádio**.

Podstatu PR tvoří přenos digitální informace od jednoho uživatele k druhému prostřednictvím sítě. Základním zdrojem informace není lidský hlas, ale data (text) z počítače nebo terminálu. Text nesoucí informaci se určitým způsobem zpracuje (zakóduje) a prostřednictvím transceiveru (TRX) je vyslán do nejbližšího nódu (uzlu) sítě PR, v níž se dále šíří až na místo určení, kde se na straně přijímací stanice zpětně dekóduje. Zjednodušeně lze říci, že při každém stisknutí klávesy vyšle počítač na rozhraní (např. sériový port PC) odpovídající binárně kódovanou hodnotu. Jednotlivé znaky jsou vyjádřeny posloupností logických jedniček a nul podle standardu nazývaného ASCII. Jeden znak se skládá z 8 bitů (bit je jednotka informace), 8 bitů se označuje jako bajt. Jednotlivé bajty jsou přivedeny do modemu, kde jsou dále upraveny k přenosu. Data jsou kódována NRZI (NON-RETURN TO ZERO INVERTED), viz následující obrázek),



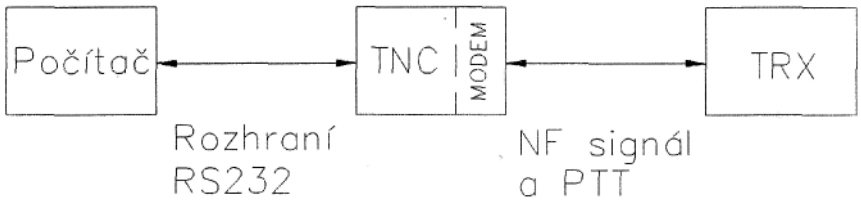
změně logické úrovně se potom přiřadí signál o kmitočtu 1200 Hz a stavu beze změny odpovídá signál o kmitočtu 2200 Hz (norma BELL 202). Informace je tedy přenášena změnou kmitočtu proti předcházejícímu stavu. Tím, že se používají signály spadající do oblasti nízkých akustických kmitočtů, lze použít pro přenos běžného transcieveru. Vzniká tak diskrétní modulace **AFSK** (Audio Frequency Shift Keying - klíčování změnou kmitočtu, tedy diskrétní obdoba spojité modulace FM). Vztah mezi jednotlivými signály je patrný z obrázku. Pro přenos dat na rychlých uživatelských vstupech a na linkách se používá **modulace FSK** (Frequency Shift Keying).

V prvním řádku jsou uvedena základní data. Ve druhém pak kódování NRZ, kdy „1" odpovídá jedna logická úroveň a „0" pak druhá. V praxi to mohou být úrovně logiky TTL (+5V a 0V), či RS232 (-12 a +12V). Ve třetím řádku jsou původní data kódována pomocí NRZI. „0" na vstupu NRZI kodéru způsobí na výstupu změnu proti předchozímu stavu. Při „1" na vstupu se výstup NRZI kodéru nemění. Výhodou NRZI kódování je přenos informace změnou stavu. Případné negování signálu nezpůsobí ztrátu nebe zkreslení informace. Na dalším řádku je zobrazena AFSK modulace, kdy jsou jednotlivým úrovním NRZI signálu přiřazeny dva kmitočty. Tento signál je potom přiváděn do mikrofonního vstupu TRXu. Pro zajímavost je na pátém řádku uvedeno kódování Manchester, jehož výhodou je nulová střední hodnota signálu (stejnosměrná složka). Zjednodušené blokové schéma pracoviště pro PR je na *obr.*



Digitální data z počítače (v tomto případě data úrovní RS232 sériového portu PC) jsou zpracována v tzv. TNC (bude vysvětleno podrobně dále) a převedena do úrovní TTL. V modemu se potom převedou na akustický signál a ten je přiveden na mikrofon-ní vstup běžného TRXu a vyslán anténou. Při příjmu signálů z nódu nebo od uživatele je situace podobná, demodulovaný nízkofrekvenční signál se v modemu převede na digitální, je zpracován v TNC a odeslán do počítače. Signál vysílaný uživatelem přijat nejbližším nódem sítě PR a šířen až ke konečnému uživateli. Komunikace uživatele s nódem probíhá podle určitého pořádku, který se označuje jako protokol AX25. Uživatel nemusí protokol

ovládat, protože vlastní komunikaci řídí buď počítač nebo modem. Historicky nejstarší je provoz PR s řadičem TNC (viz *obr).*



TNC (Terminál Node Controller) je technické zařízení, které se skládá z modemu a mikroprocesorové části. Zapojuje se mezi počítač a TRX a zajišťuje tvorbu protokolu AX25 a komunikaci uživatele s uživatelem, nódem či BBS. Tento způsob provozování PR patří k nejspolehlivějším a v současnosti existuje nepřeberné množství různých TNC. Rostoucí výkon PC a snaha o zjednodušení hw vedla radioamatéry k vytvoření jednodušších konstrukcí jako např. BayCom modemu obsahujícího pouze obvod TMC3106 . Dalšího zjednodušení hw bylo dosaženo s příchodem multimediálních PC, kdy již výkonnost zvukových karet a výkonnost počítačů dovoluje zpracovávat analogový signál přivedený z TRXu do zvukové karty, v reálném čase, stejně tak se generuje i signál pro vysílací trakt.
Provoz "paket rádio" je řízen protokolem AX.25 vydaným IAR. Tento protokol je modifikací profesionálního protokolu X.25. Protokol používá sedm úrovní architektury OSI, vlastní výměna informace probíhá na druhé vrstvě. Je prováděna mezi terminály -počítači připojenými do sítě- nebo v jednodušším případě mezi dvěma stanicemi. Řízena je speciálním řadičem, nebo je naopak realizována poměrně jednoduchými adaptéry počítačů a zprostředkována hlavně důmyslným programovým vybavením.
V rámci tohoto protokolu mají všechny stanice stejnou prioritu pro přístup ke spojení. Spojení je navázáno pouze tehdy, není-li protistanice zaměstnána spojením s jinou stanicí. Při vlastním spojení je informace vysílána v blocích označovaných jako rámce (obr.).

Flag	Adresní pole	Řídící pole	Informační pole	FCS	Flag

V informaci uvnitř rámce lze použít jakýkoliv kód, z tohoto důvodu označujeme takovýto přenos informace jako transparentní. Rámec je rozdělen na několik polí. Začátek a konce rámce je vyznačen speciálním znakem, nazývaným křídelní značkou , která je obsahem pole "flag". Hexadecimální hodnota křídelní značky je 7E, binárně vyjádřeno 01111110. Kombinace šesti jedniček za sebou se nesmí v rámci mimo jeho konec již vyskytnout.
Poněvadž v rámci může být umístěna jakákoliv binární nebo textová informace, ve které by se tato kombinace bitů mohla náhodně vyskytovat, je umístění šesti jedniček za sebou vyloučeno na vysílací straně. Jakmile je zjištěno v informaci uvnitř rámce pět binárních jedniček za sebou, je za nimi automaticky vložena nula. Ta je opět na přijímací straně z informace vyjmuta. To znamená, že přijímací strana sleduje, zda se po pěti binárních jedničkách objevila nula, nebo šestá jednička. Při výskytu nuly dojde k jejímu odstranění a příjem rámce dále pokračuje, při šesté jedničce v pořadí za sebou je příjem rámce ukončen.
Adresní pole může obsahovat od dvou do deseti adres - volacích značek, komunikujících stanic. Nejprve je uvedena *značka* volaného, následuje *značka* volajícího a značky retranslátorů, kterých může být až osm. Podle počtu použitých adres je délka adresního pole od čtrnácti do sedmdesáti slabik. Každá volací značka může sestávat až ze šesti symbolů, jestliže je kratší, je doplněna mezerami. V sedmé slabice u každé adresy je uveden **SSID** - sekundární identifikátor stanice. V rámci každé slabiky volací značky je informace posunuta o jeden bit vlevo, pro adresu je předepsáno používat hlavně z tohoto důvodu pouze velká písmena v kódu ASCII. V slabice **SSID** je v nejnižším bitu poslední adresy umístěna jednička, ta označuje konec adresního pole.

Obvyklý sekundární identifikátor má hodnotu nula nebo jedna, při práci s poštovní schránkou a u uzlových stanic se přiřazují **SSID** vyšší čísla, až do patnácti. U adresy retranslátorů je řádově nejvyšší bit vyhrazen pro indikaci, zda rámec retranslátorem prošel.
V jednoslabikovém řídícím poli je uveden příkaz, t j. informace o druhu rámce. Rozeznáváme tři typy rámců: **I** – informační **S** – servisní **U** - nečíslované
Informační rámce obsahuji číslicovou, abecední, nebo libovolnou informaci (transparentní text). V informačním poli tohoto rámce nemusí být nutně obsažen pouze zakódovaný text, rámec může přenášet například i obraz paměti, program ve strojovém kódu nebo obrazovou informaci.
Servisní rámce přenášejí služební informaci, obsahující potvrzeni přijatého rámce, nebo požadavek na vyslání dalšího informačního rámce. Nečíslovanými rámci je obvykle předáván požadavek na spojení.
Řídící pole obsahuje i číslo vysílaného rámce a číslo *rámce,* jehož příjem je od protistanice očekáván. To umožňuje přijímajícímu vyžádat opětné vysílání rámce, který byl přijat s chybou.

Řídicí pole rámce:

Typ rámce	Bity řídícího pole 7 6 5 4 3 2 1 0				
Rámec I	N(R)	P	N(S)		0
Rámec S	N(R)	P/F	S S	0	1
Rámec U	M M M	P/F	M M	1	1

Vysvětlivky:
N(S) - číslo vysílaného rámce
N(R) - číslo přijímaného (očekávaného) rámce
S - bity supervizoru
M - modifikační bity pro nečíslované rámce
P/F - pool/final bit pro příkaz / odpověď

Příkazy rámce S (řídícího):

Typ příkazu	Bity řídícího pole 7 6 5 4 3 2 1 0		
RR (Receive Ready)	N(R)	P/F	0 0 0 1
RNR (Receive Not Ready)	N(R)	P/F	0 1 0 1
REJ (Reject)	N(R)	P/F	1 0 0 1

Vysvětlivky:
RR - připravenost na příjem
RNR - stanice nemůže přijmout další rámec
REJ - požadavek na opětné vyslání rámců od N(R) výše

Příkazy rámce U (nečíslovaného):

Typ příkazu	Druh	Bity řídícího pole 7	6	5	4	3	2	1	0
SABM	příkaz	0	0	1	P	1	1	1	1
DISC	příkaz	0	1	0	P	0	0	1	1
DM	odpov.	0	0	0	F	1	1	1	1
UA	odpov.	P	1	1	F	0	0	1	1
FRMR	odpov.	1	0	0	F	0	1	1	1
UI	obojí	0	0	0	P/F	0	0	1	1

Vysvětlivky:
SABM - Set Asynchronous Balanced Mode - nastavení asynchronního režimu práce stanic (tj. stanice do spojení vstupují asynchronně)
DISC - Disconnect - ukončení spojení (odpojení)
DM - Disconnected Mode - režim ukončeného spojení
UA - Unnumered Acknowledge - nečíslované potvrzení
FRMR - Frame Reject - odmítnutí rámce
UI - Unnumered Information - nečíslovaný rámec s informací (právě tyto rámce se často používají pro přenos dat z CanSatů)
Pozn.: Rámec FRMR nese tříslabikové informační pole
U některých rámců (nesoucích informaci - informačních nebo nečíslovaných) následuje informační pole. V jeho prvé slabice bývá uveden PID - identifikátor protokolu. Zde je uvedena použitá síťová úroveň, tj. zda je použita třetí vrstva. V informačním poli dále následuje informace, která je ve formě textové zprávy zobrazena na obrazovce, nebo binární informace. Tou je přenášen například program nebo část obrazu. V posledních dvou slabikách před koncovou křídělní značkou je umístěno FCS-cyklické zabezpečení rámce, někdy označované též jako CRC. (FCS - Frame Check Sequence, CRC - Cyclic Redundance Check). Při vysílání rámce jsou postupně jednotlivé bity celého rámce (kromě křídelních značek a samozřejmě i kromě kontrolního součtu) přičítány ke kontrolnímu součtu. K tomu je použit algoritmus $x^{16} + x^{12} + x^5 + 1$. Výsledný kontrolní součet je vyslán jako dvouslabíkové zabezpečovací pole před koncovou křídelní značkou. Na přijímací straně je obdobným postupem provedeno odečítání uvedeného mnohočlenu od zde vytvářeného kontrolního součtu. Prakticky je odečítání nahrazeno přičítáním mnohočlenu do invertované dvojice osmibitových, nebo jednoho šestnáctibitového registru. Výchozí stav registru, do kterého je prováděno načítáni při příjmu je FFh. Do této operace jsou při příjmu zahrnuty i všechny bity obou slabik kontrolního součtu. Po skončení operace musí být, bez ohledu na obsah rámce, výsledkem vždy stejná hodnota, t j . 1D0Fh. Jestliže je výsledek odlišný, rámec byl přijat chybně a přijímající stanice musí vyžádat opakované vyslání

stejného rámce. Zabezpečovací pole rámce se liší od ostatních polí. Zatímco u jiných polí je vždy v každé slabice vysílán nejprve váhově nejnižší bit, u tohoto pole jsou odeslány nejdříve nejvyšší bity slabik.

```
1. vysílaný bit
↓
01111110|01111001|01101001|00100110|00110101|00011001|01101001
 Flag       O        K        2        V        L        K
00001100|01111001|01011001|01100110|00100101|01001101|01100101
 SSID       O        M        3        R        Y        S
10000111|10000110|10001100|11010101|01111110 ← poslední bit
 SSID    RR-3.bl.   FCS      FCS      Flag
```

Příklad rámce typu S s příkazem RR
Pozn.: Jednotlivé slabiky jsou odděleny svislými oddělovači, očekáván je příjem třetího bloku
Rámec je ukončen opět polem "flag", které obsahuje křídelní značku 01111110 (binárně) .
Pro vysílání a příjem je informace překódována způsobem NRZI (Non Return to Zero Inverted). Tento způsob znamená, že ke změně vysílaného kmitočtu dojde při nulovém bitu, při jedničkovém bitu nedochází ke změně kmitočtu.
V současné době se protokol AX.25 používa i pro další typ komunikace nazývaný **APRS** (**Automatic Position Reporting Systém**), kdy radioamatéři předávají své souřadnice z GPS, zprávy o počasí, DX zpávy a jiné krátké množiny dat. Při komunikaci CanSaty používají právě APRS formát. [18]

The AX.25 Frame All APRS transmissions use AX.25 UI-frames, with 9 fields of data:

	AX.25 UI-FRAME FORMAT								
	Flag	***Destination Address***	***Source Address***	***Digipeater Addresses (0-8)***	***Control Field (UI)***	***Protocol ID***	***INFORMATION FIELD***	***FCS***	***Flag***
Bytes:	1	7	7	0–56	1	1	1–256	2	1

APRS Data Type Identifiers

Ident	Data Type
0x1c	Current Mic-E Data (Rev 0 beta)
0x1d	Old Mic-E Data (Rev 0 beta)
!	Position without timestamp (no APRS messaging), or Ultimeter 2000 WX Station
"	*[Unused]*
#	Peet Bros U-II Weather Station
$	Raw GPS data or Ultimeter 2000
%	Agrelo DFJr / MicroFinder
&	*[Reserved — Map Feature]*
'	Old Mic-E Data (but *Current* data for TM-D700)
(	*[Unused]*
)	Item
*	Peet Bros U-II Weather Station
+	*[Reserved — Shelter data with time]*
,	Invalid data or test data
-	*[Unused]*
.	*[Reserved — Space weather]*
/	Position with timestamp (no APRS messaging)
0–9	*[Do not use]*
:	Message
;	Object

Ident	Data Type
<	Station Capabilities
=	Position without timestamp (with APRS messaging)
>	Status
?	Query
@	Position with timestamp (with APRS messaging)
A–S	*[Do not use]*
T	Telemetry data
U–Z	*[Do not use]*
[	Maidenhead grid locator beacon (obsolete)
\	*[Unused]*
]	*[Unused]*
^	*[Unused]*
_	Weather Report (without position)
`	Current Mic-E Data (*not used* in TM-D700)
a–z	*[Do not use]*
{	User-Defined APRS packet format
|	*[Do not use — TNC stream switch character]*
}	Third-party traffic
~	*[Do not use — TNC stream switch character]*

PRIMARY SYMBOL TABLE			
/$	GPS xyz	GPS Cnn	Icon
/D	PD␣	36	
/E	PE␣	37	Eyeball (eye catcher)
/F	PF␣	38	
/G	PG␣	39	Grid Square (6-character)
/H	PH␣	40	Hotel (blue bed icon)
/I	PI␣	41	TCP/IP
/J	PJ␣	42	
/K	PK␣	43	School
/L	PL␣	44	
/M	PM␣	45	MacAPRS
/N	PN␣	46	NTS Station
/O	PO␣	47	Balloon (SSID –11)
/P	PP␣	48	Police

ALTERNATE SYMBOL TABLE			
\$	GPS xyz	GPS Enn	Icon
\D	AD␣	36	Drizzle
\E	AE␣	37	Smoke
\F	AF␣	38	Freezing Rain
\G	AG␣	39	Snow Shower
\H	AH␣	40	Haze
\I	AI␣	41	Rain Shower
\J	AJ␣	42	Lightning
\K	AK␣	43	Kenwood
\L	AL␣	44	Lighthouse
\M	AM␣	45	
\N	AN␣	46	Navigation Buoy
\O	AO␣	47	
\P	AP␣	48	Parking

Další popis APRS včetně vysílačů, vhodných i pro Cansaty najdeme v diplomových pracech VUT Brno [26], [27] a [28].

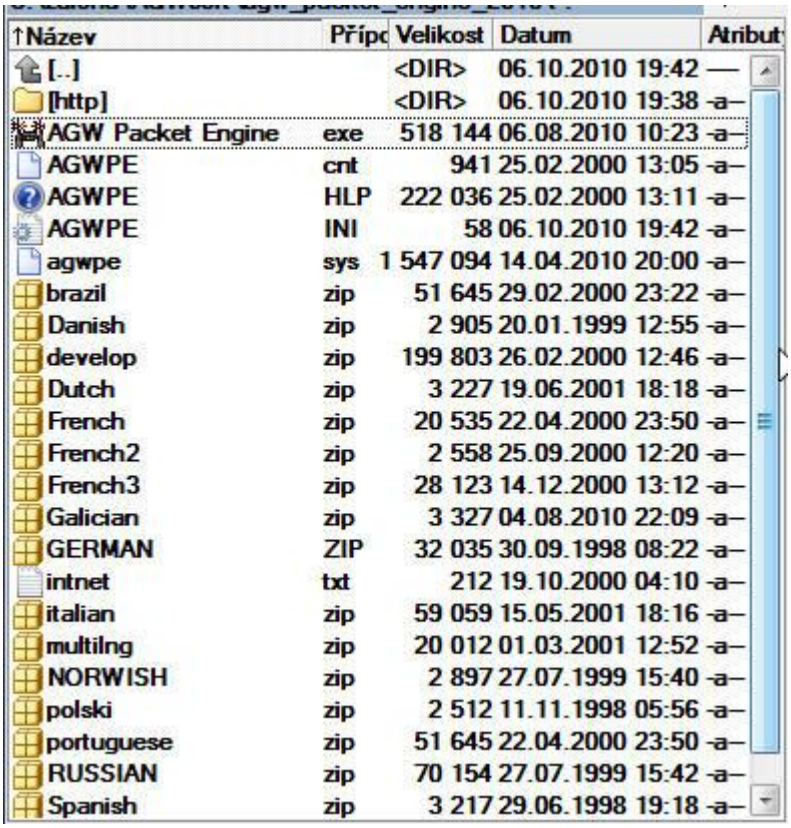
Informace uvedené v této kapitole budeme potřebovat, pokud bychom vytvářeli firmware jednočipového počítače, který je součástí vysílače či pokud bychom psali veškerý software pro PC pozemní stanice.

Pokud budeme vyvíjet vlastní vysílač kompatibilní s vysílačem *Pratt Hobbies* (*ADF7012* a *ATMEGA88*), můžeme pro *ATMEGA88* použít firmware STENSAT uvedený na DVD příloze (výpis v Dodatku 1). Pokud jde o počítač PC je výhodné použít jeho zvukovou kartou a využívají jako základní sw **AGW**

Packet Engine [19] a nějakou jeho nadstavbu, např. **AGW monitor**. Tato dvojici programů byla používána i při 1.evropské soutěži středoškoláků pořádané ESA ESTEC Education v roce 2010 [14], [15].

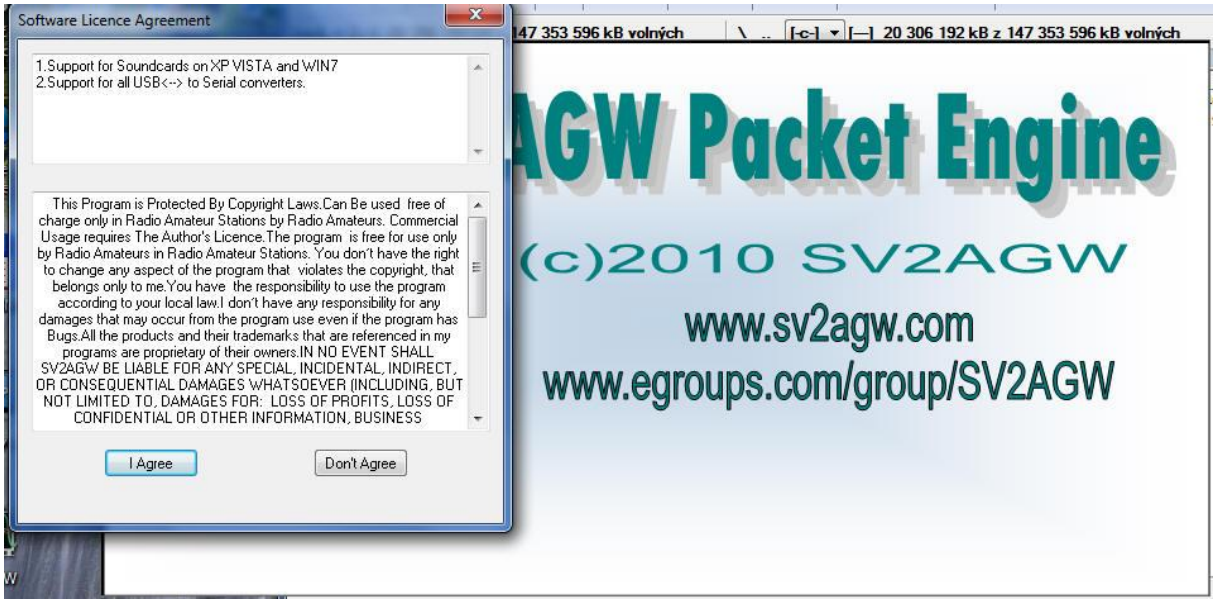
6.1.2.3 AGW engine

AGW engine a další sw pro packet rádio napsal řecký radioamatér **George Rossopoulos.** Jeho radioamatérská volací značka je SV2AGW a z jejího sufixu AGW je také odvozen název software. Domovská stránka tohoto sw je http://www.sv2agw.com/default.htm .
Z ní můžeme stáhnout „AGW Packet Engine ver 2010.414 win95/98/NT/2k/ME/XP“ obsaženou v souboru AGWPE.zip. Tento sw je funkční i pod Windows 7. Nevyžaduje žádnou instalaci. Pouze vytvoříme nějaký adresář, do kterého umístíme soubory obsažené v AGWPE.zip (tj. rozbalíme neboli „rozzipujeme“ soubor agwpe.zip). Obsah tohoto adresáře bude:



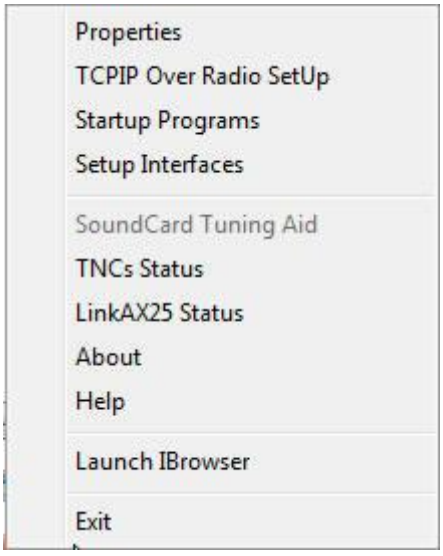
↑Název	Přípc	Velikost	Datum	Atribut
[..]		<DIR>	06.10.2010 19:42	—
[http]		<DIR>	06.10.2010 19:38	-a-
AGW Packet Engine	exe	518 144	06.08.2010 10:23	-a-
AGWPE	cnt	941	25.02.2000 13:05	-a-
AGWPE	HLP	222 036	25.02.2000 13:11	-a-
AGWPE	INI	58	06.10.2010 19:42	-a-
agwpe	sys	1 547 094	14.04.2010 20:00	-a-
brazil	zip	51 645	29.02.2000 23:22	-a-
Danish	zip	2 905	20.01.1999 12:55	-a-
develop	zip	199 803	26.02.2000 12:46	-a-
Dutch	zip	3 227	19.06.2001 18:18	-a-
French	zip	20 535	22.04.2000 23:50	-a-
French2	zip	2 558	25.09.2000 12:20	-a-
French3	zip	28 123	14.12.2000 13:12	-a-
Galician	zip	3 327	04.08.2010 22:09	-a-
GERMAN	ZIP	32 035	30.09.1998 08:22	-a-
intnet	txt	212	19.10.2000 04:10	-a-
italian	zip	59 059	15.05.2001 18:16	-a-
multilng	zip	20 012	01.03.2001 12:52	-a-
NORWISH	zip	2 897	27.07.1999 15:40	-a-
polski	zip	2 512	11.11.1998 05:56	-a-
portuguese	zip	51 645	22.04.2000 23:50	-a-
RUSSIAN	zip	70 154	27.07.1999 15:42	-a-
Spanish	zip	3 217	29.06.1998 19:18	-a-

Spustíme **AGW Packet Engine.exe**. Při prvním spuštění se objeví

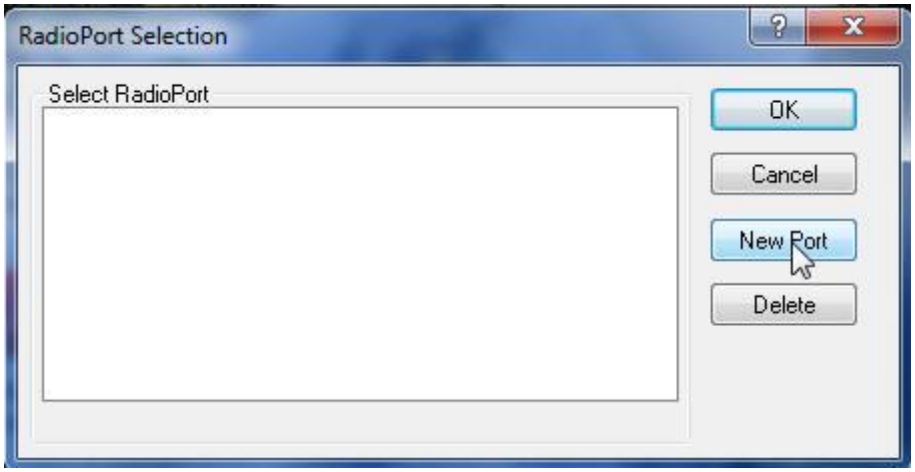


Kliknutím na tlačítko **I Agree** odsouhlasíme licenční podmínky. Poté již **AGW Packet Engine** běží jako rezidentní program navenek se projevující jen jeho ikonkou

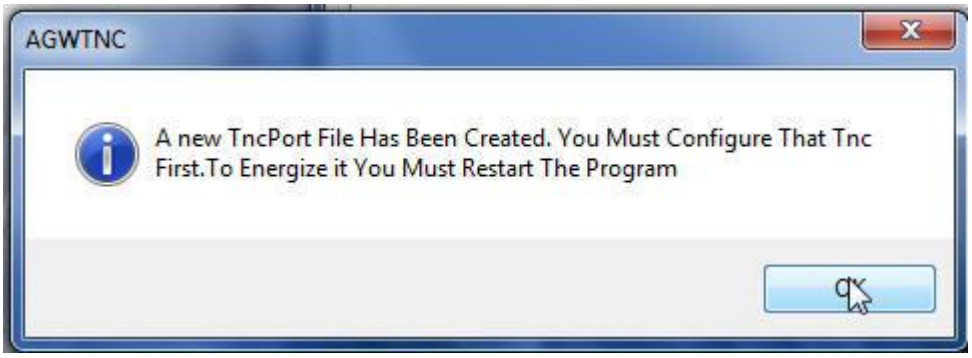
dole vpravo na dolním pásu OS Windows. Kliknutím na pravé tlačítko myši nastavené na tuto ikonku se zobrazí místní menu



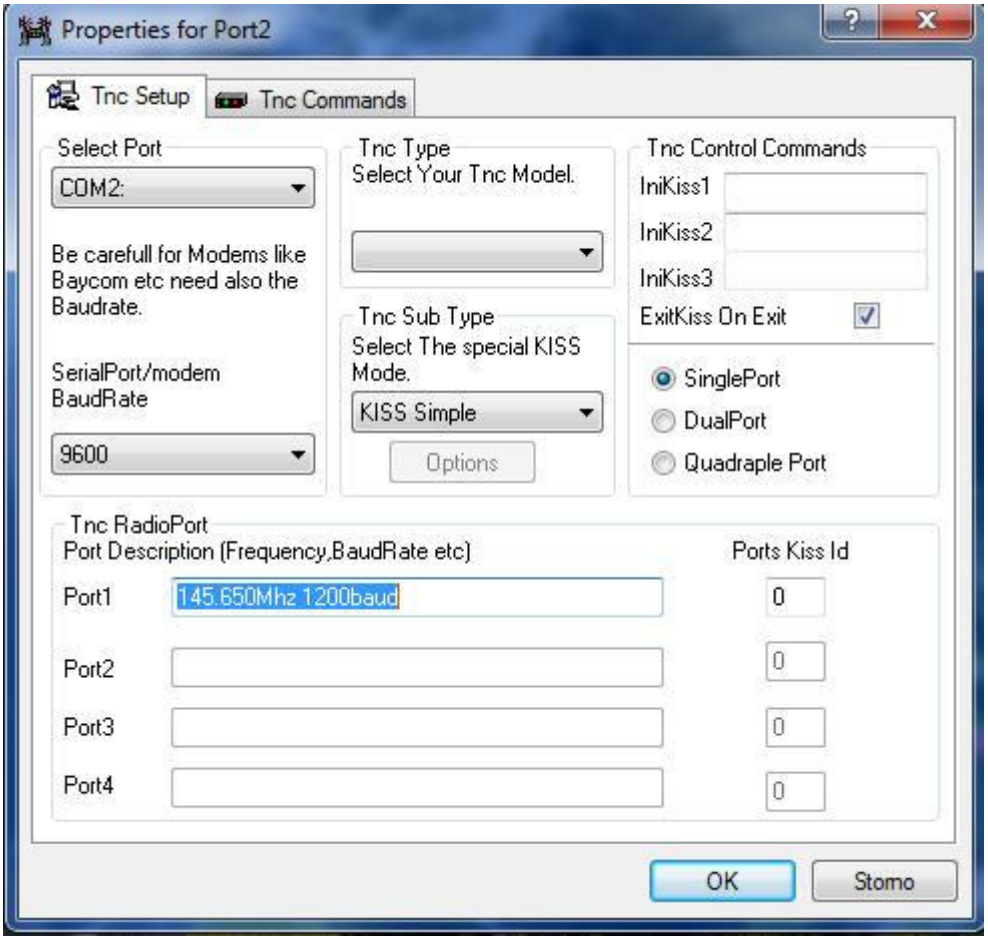
Nejprve musíme **AGW engine** sdělit, jaký TNC nebo jeho náhradu (což je v našem případě zvuková karta) budeme používat. Vybereme tedy první položku **Properties**. Objeví se:



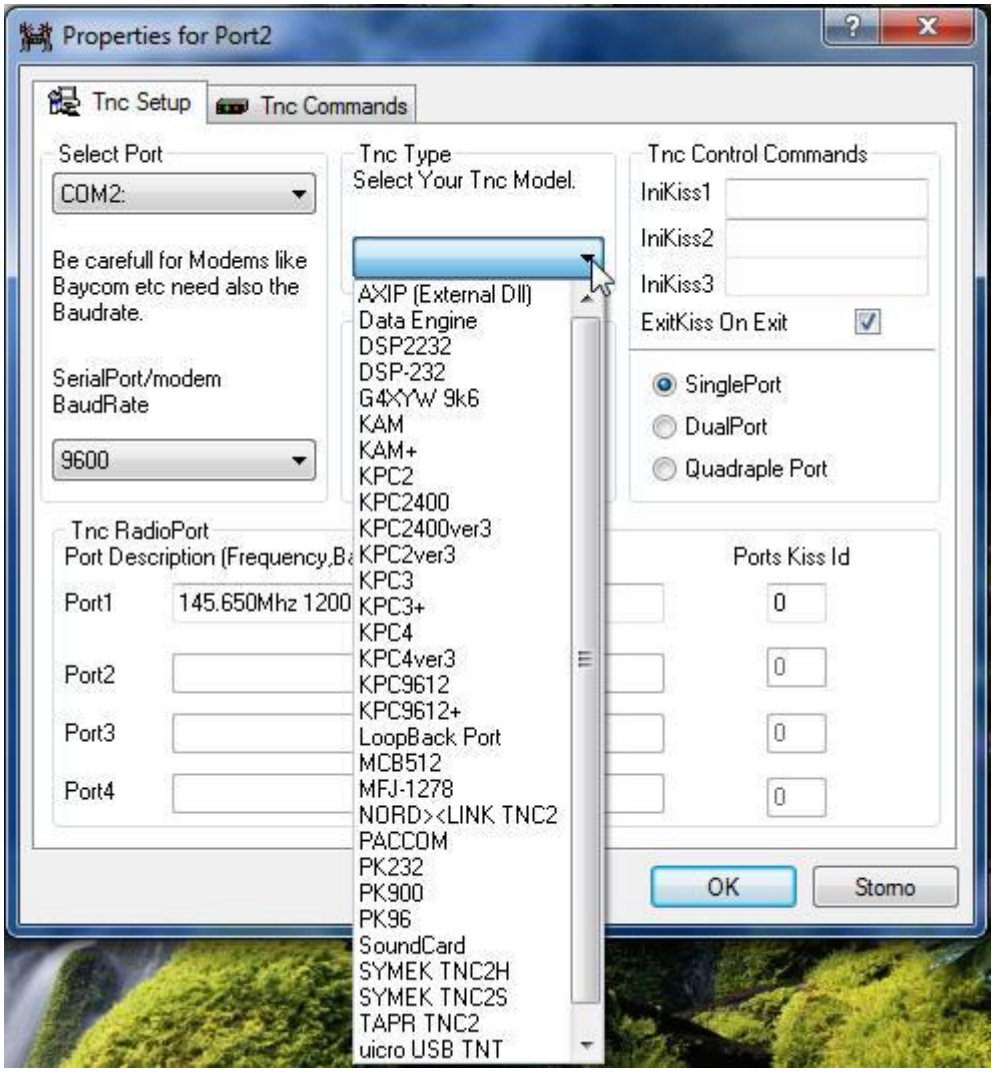
Nyní klikneme na tlačítko **New Port**. Dostaneme



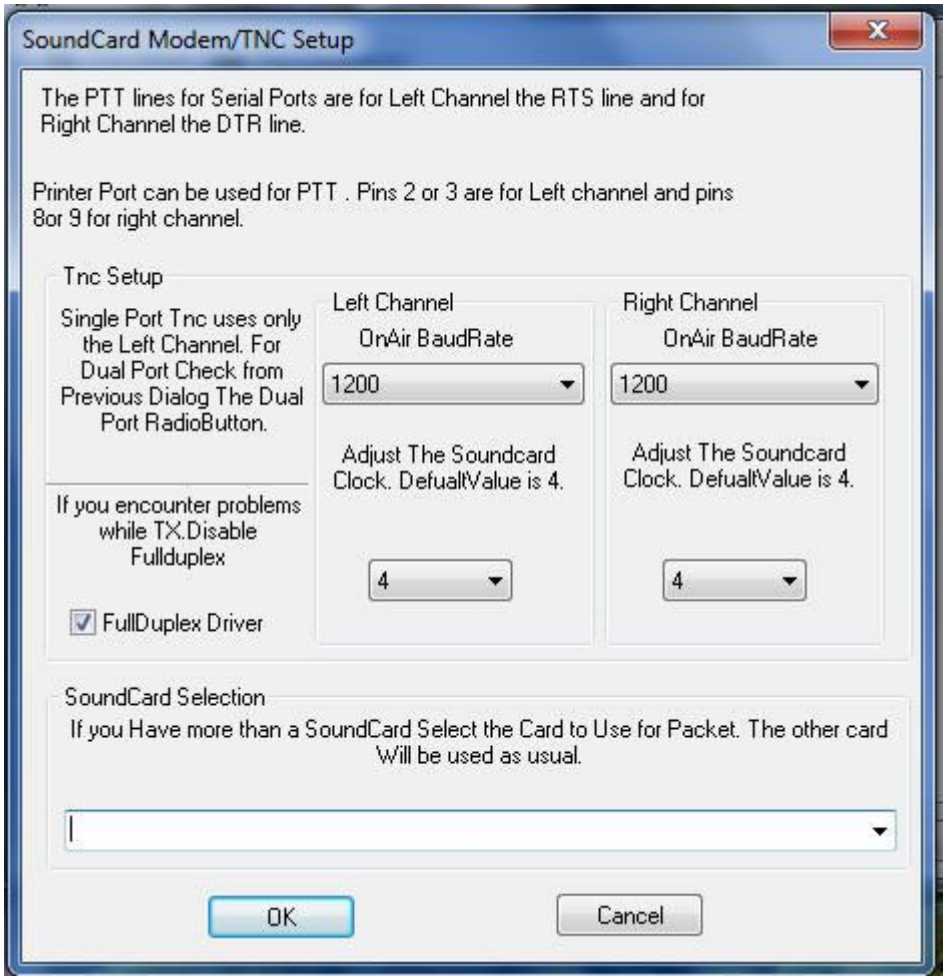
Potvrdíme **OK**. Nyní máme



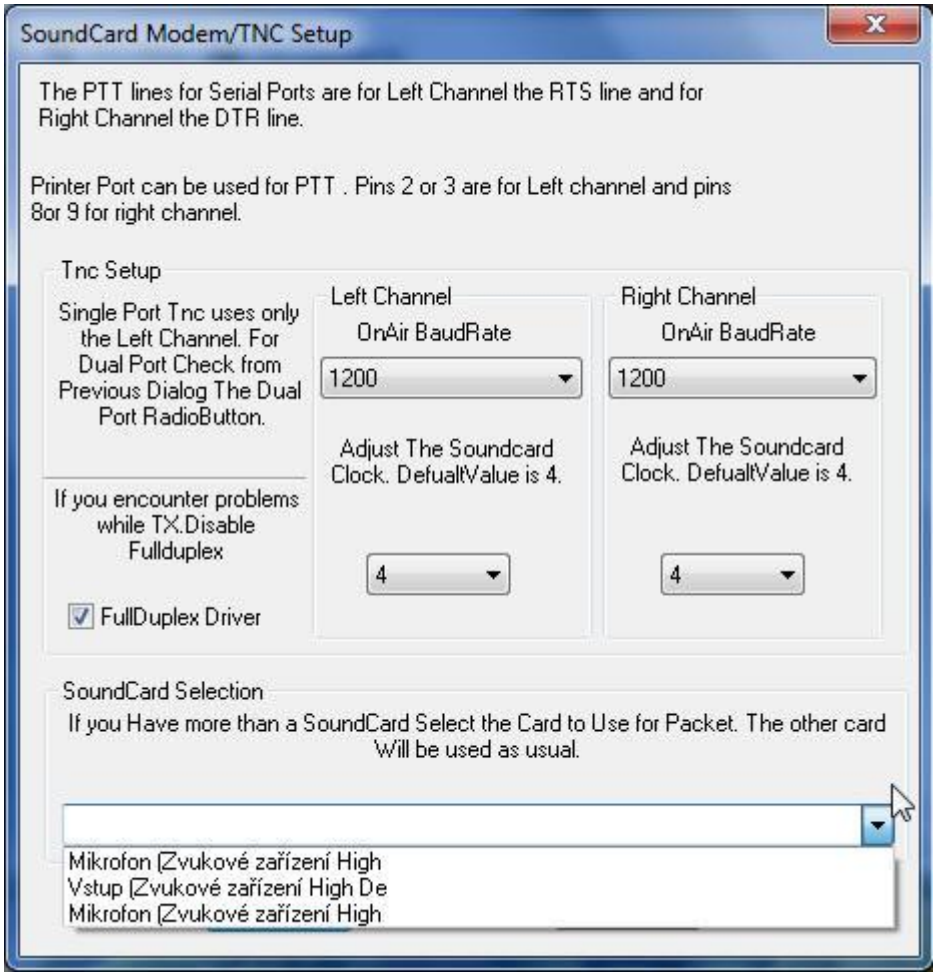
Vidíme, že v okně **Properties for PORT2** jsou některé volby již nabízeny. Ponecháme *Tnc Sub Type* **KISS Simple**. *SerialPort/modem BaudRate* však přenastavíme na **1200** baud. Musíme dále zvolit **Tnc Type**



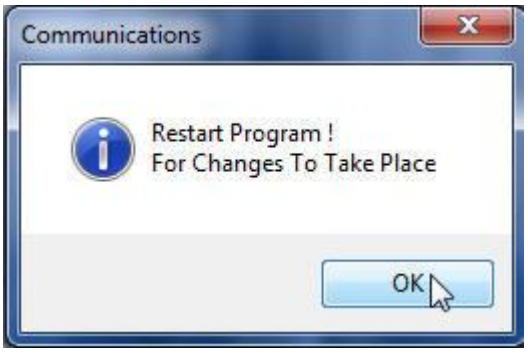
Vybereme **SoundCard** a potvrdíme **OK**. Objeví se



Rozklikneme **SoundCard Selection**

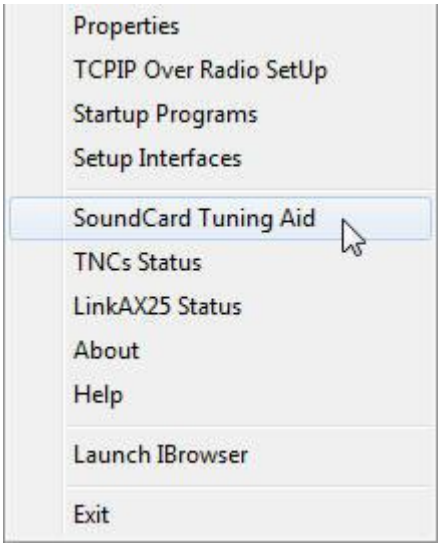


Vybereme **Vstup (Zvukové zařízení High De)** (K tomuto vstupu, konektoru, máme kabelem přiveden nf signál z výstupu FM přijímače naladěného na kmitočet vysílače vysílajícího data prostřednictvím Packet Radio podle protokolu AX25). Poté jsme vyzváni k restartování počítače.

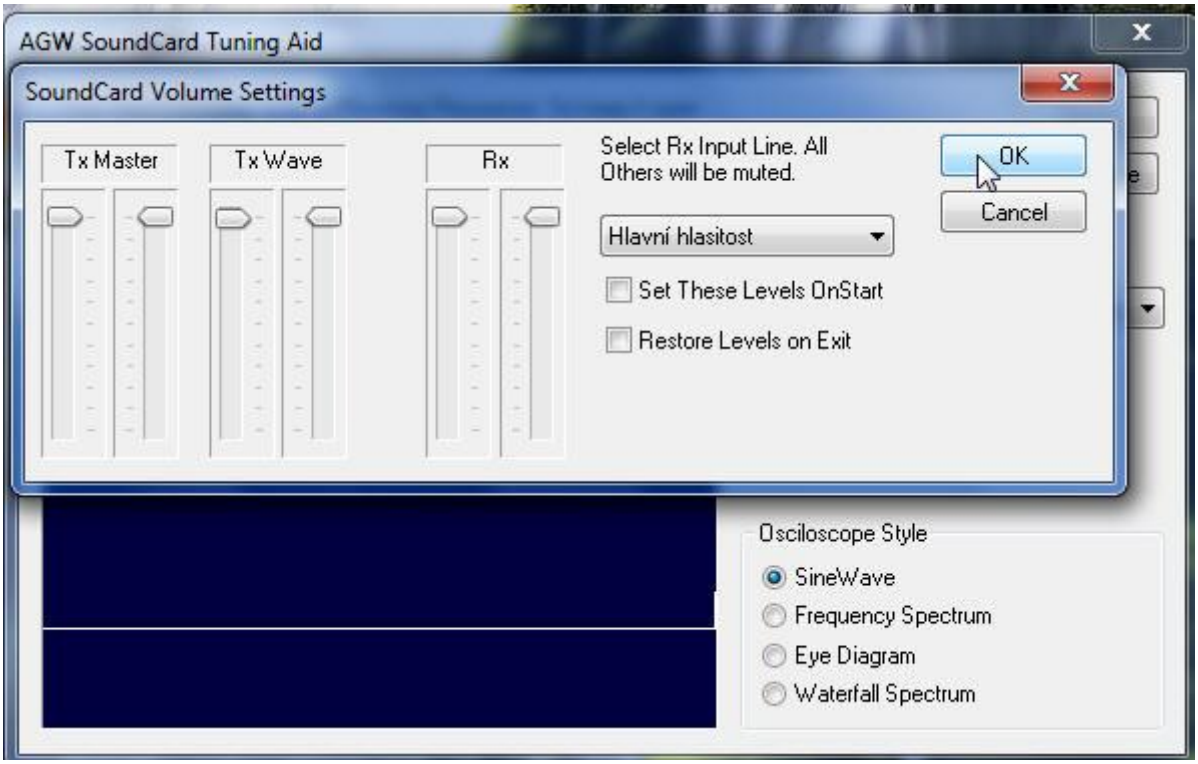


Potvrdíme **OK**. Po restartování počítače spustíme *Packet Radio Engine*.

Poté na ikonce **AGW Packet Engine** pravým tlačítkem rozvineme místní menu.



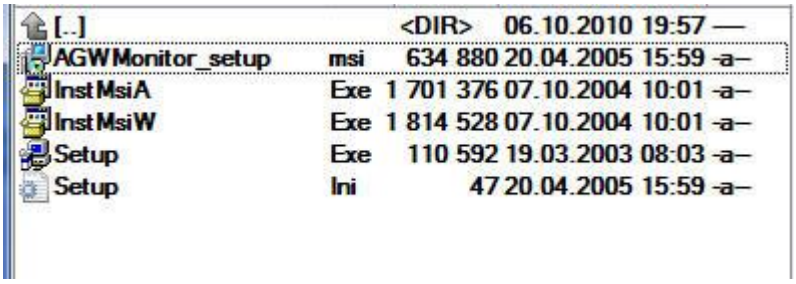
Vybereme položku **SoundCard Tuning Aid**.



Nastavíme úroveň zesílení vstupních zesilovačů zvukové karty. Pokud máme na vstup zvukové karty přiveden signál z přijímače, měl by se zobrazovat na modré obrazovce „osciloskopu“. Třebaže nyní na pozadí běží **AGW Packet Engine**, žádný užitek z něho zatím nemáme. Musíme totiž nainstalovat nějaký další program mající již uživatelské rozhraní a pracující nad **AGW Packet Engine**. Nejjednodužším takovým programem je **AGW monitor**.

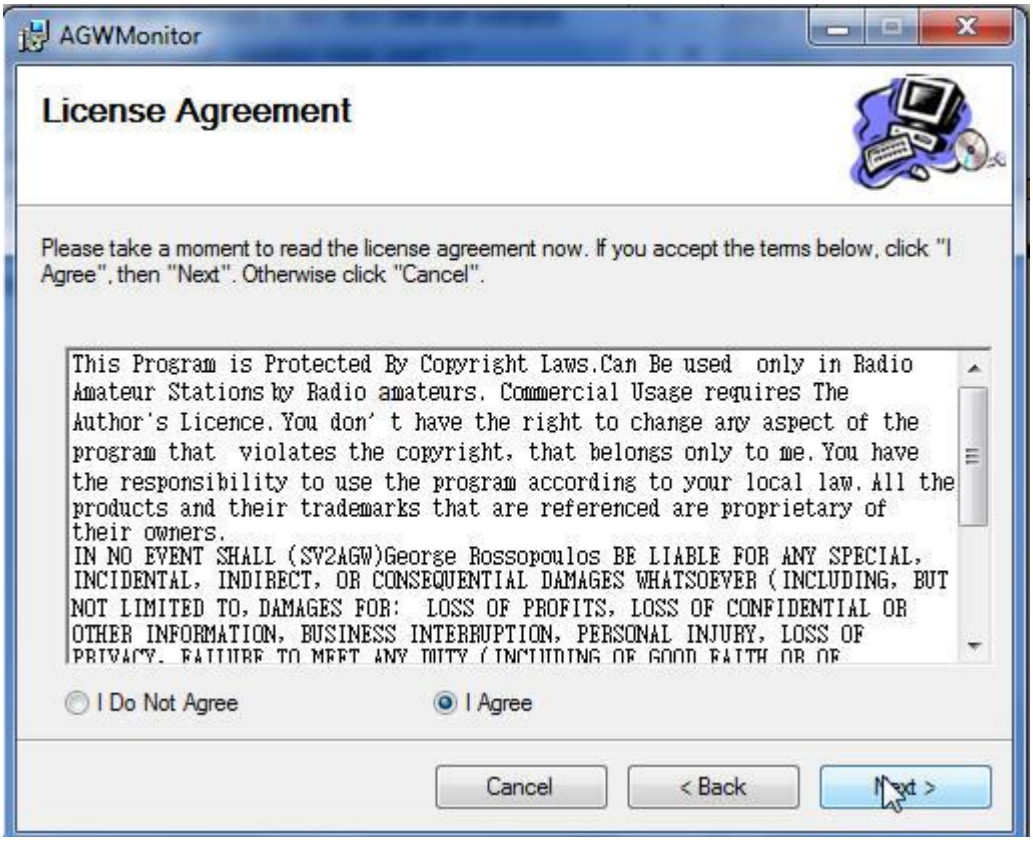
6.1.2.4 AGW monitor a další AGW programy (AGW monitor and another programs)

Na rozdíl od **AGW Packet Engine**, která se vlastně ani neinstaluje, **AGW Monitor** je nutné nainstalovat. Instalačním souborem je AGW Monitor_setup.exe.



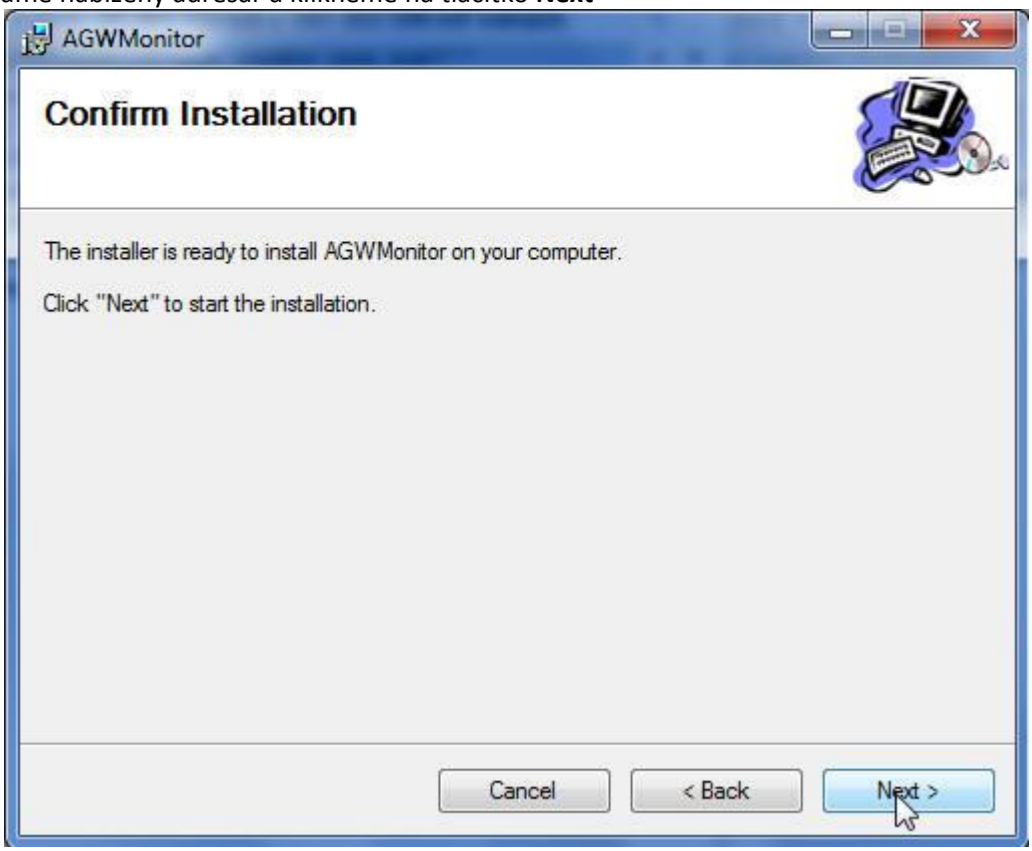
Jeho spuštěním se zahájí **setup wizard**:

Klikneme na **Next**. Dostaneme

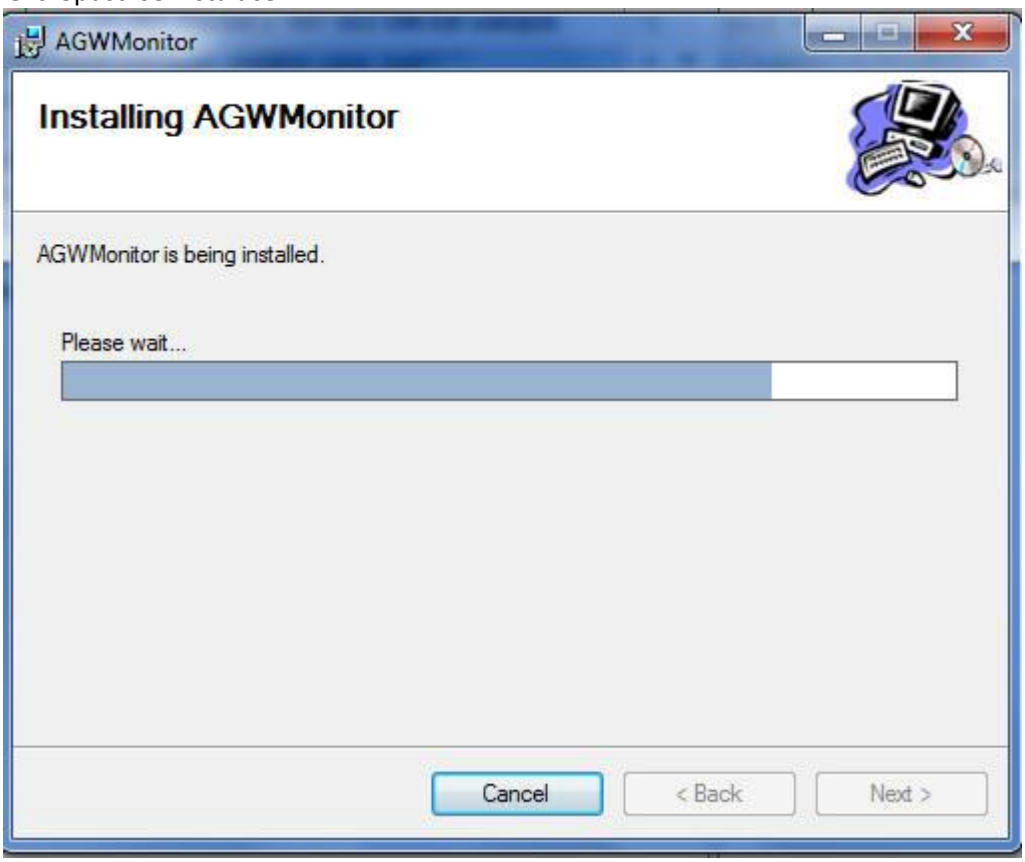


Potvrdíme souhlas s licenčními podmínkami a klikneme na **Next**

Ponecháme nabízený adresář a klikneme na tlačítko **Next**

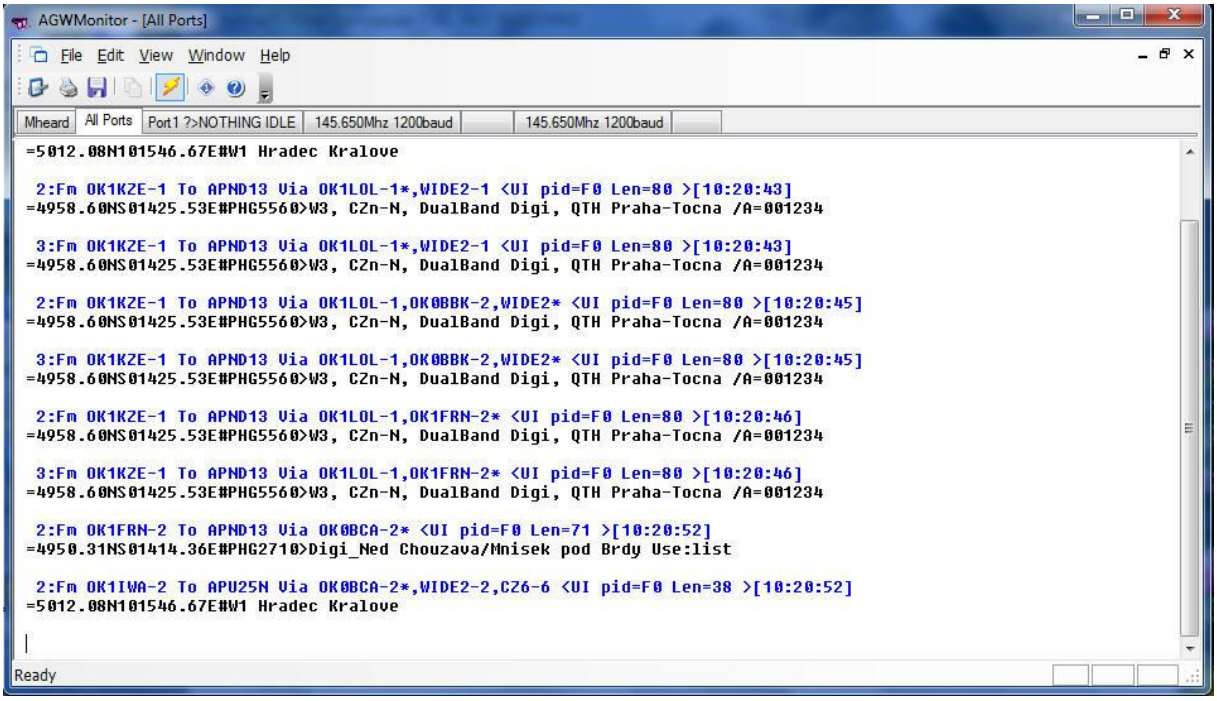


Znovu **Next.** Spustí se instalace

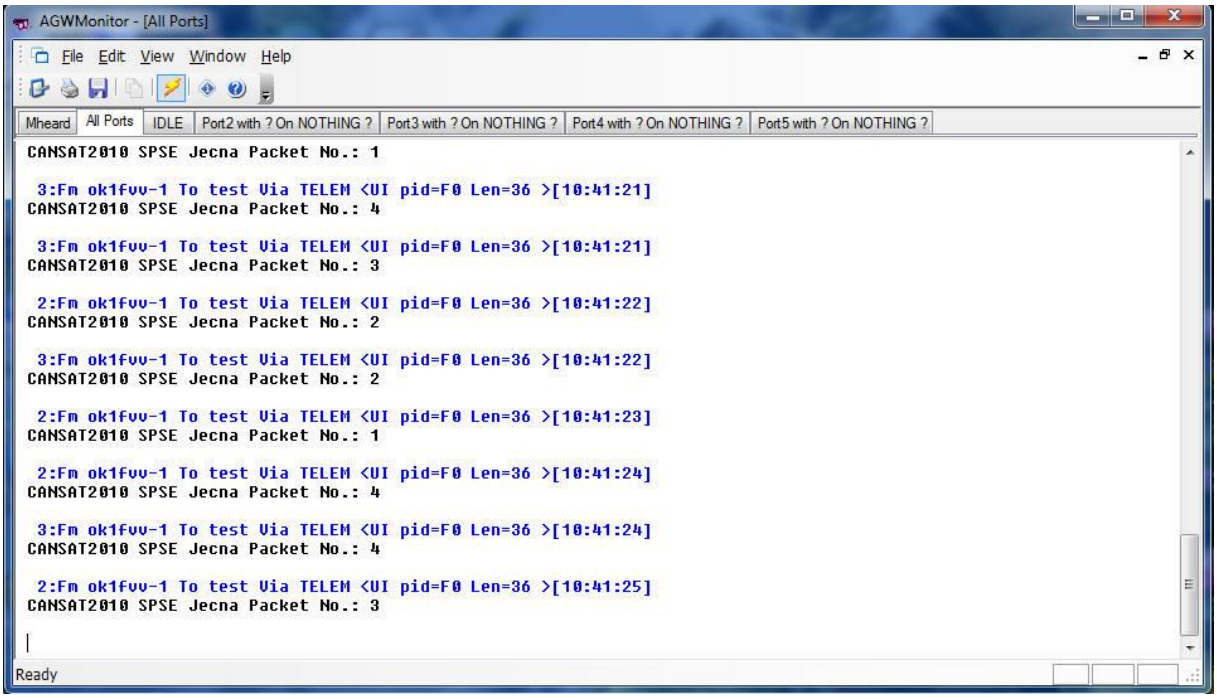


Poté se spustí **AGW Monitor**.

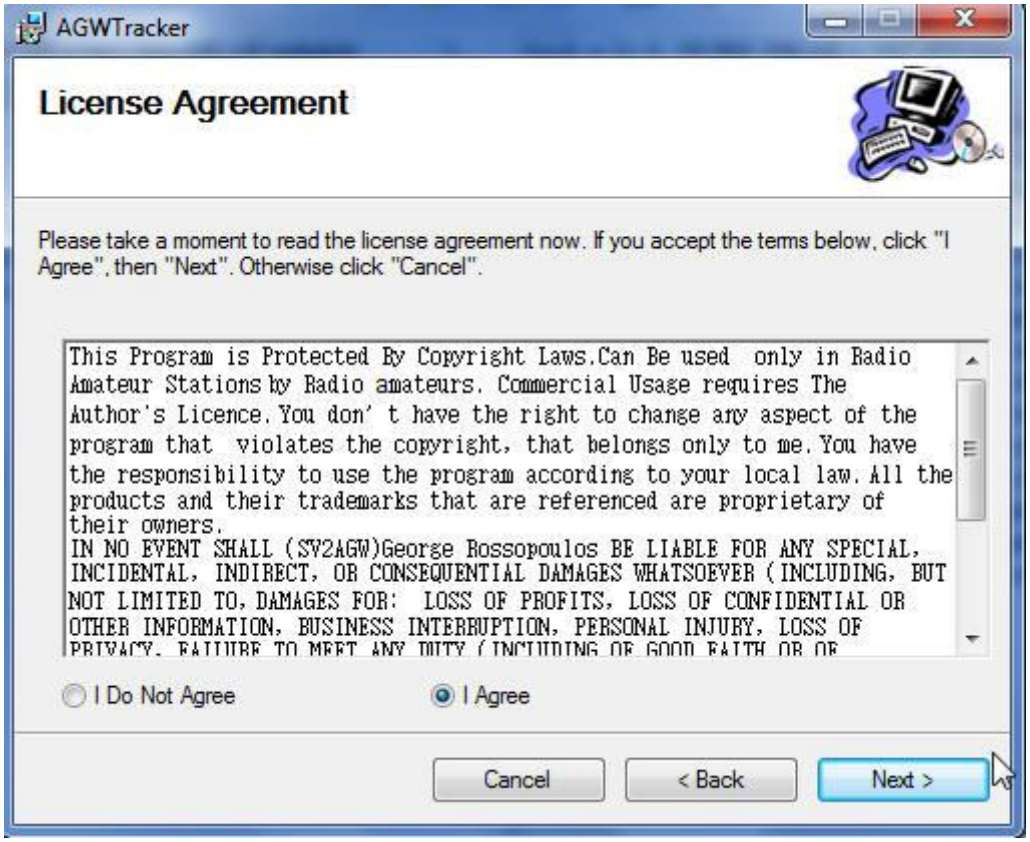
Pokud přijímač naladíme na kmitočet, na kterém probíhá PR komunikace, např. na 144,800 MHz, uvidíme něco takovéto



AGW Monitor zobrazuje obsah přijímaných paketů. Modře vypisuje služební údaje, jako jsou adresy (volací značky), typ paketu, jeho délka, čas apod. Černě se pak zobrazují vlastní přenášená data. Na výše uvedeném obrázku jde o data APRS. Na dalším obrázku jsou data přijímaná na 433,500 MHz z CanSatu jehož vysílač *Pratt Hobbies* dostává data z jednočipového počítače ATMEL AVR viz kap. **6.1.2.1**

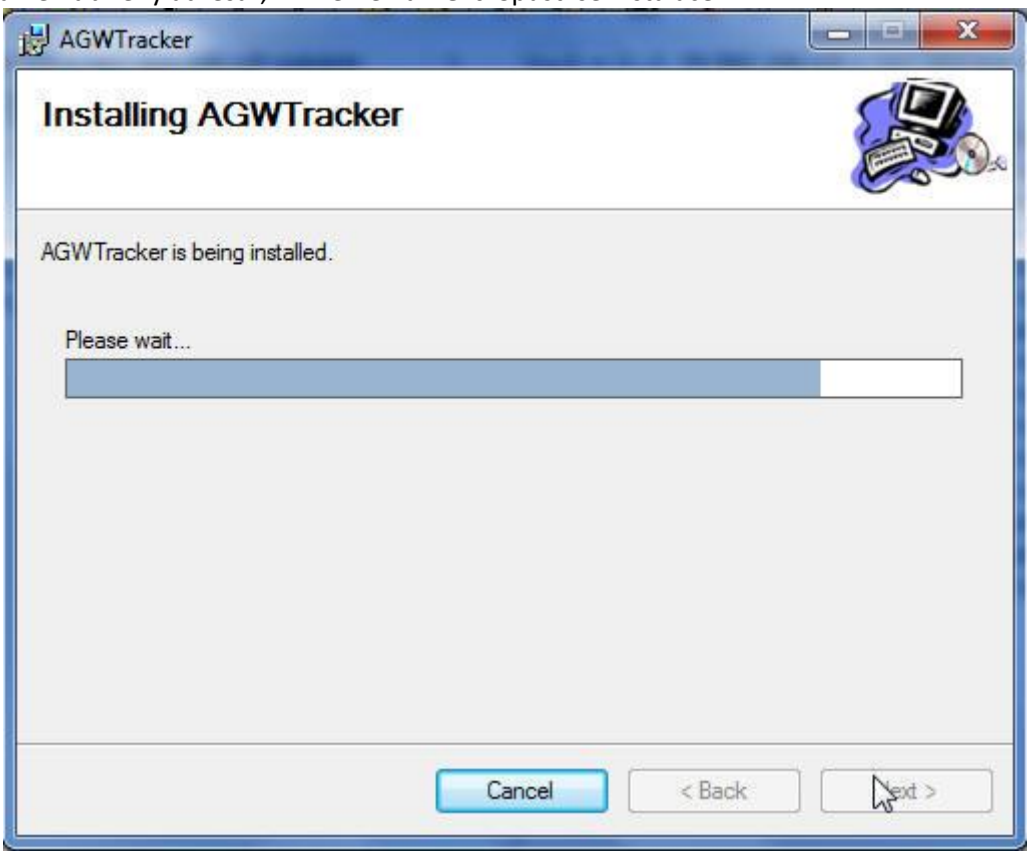


Místo **AGW monitoru** můžeme používat i nějaký jiný sw, např. **AGW Tracker**. Jeho instalace se provede obvyklým způsobem, tj. spustíme instalační soubor. Dostaneme:

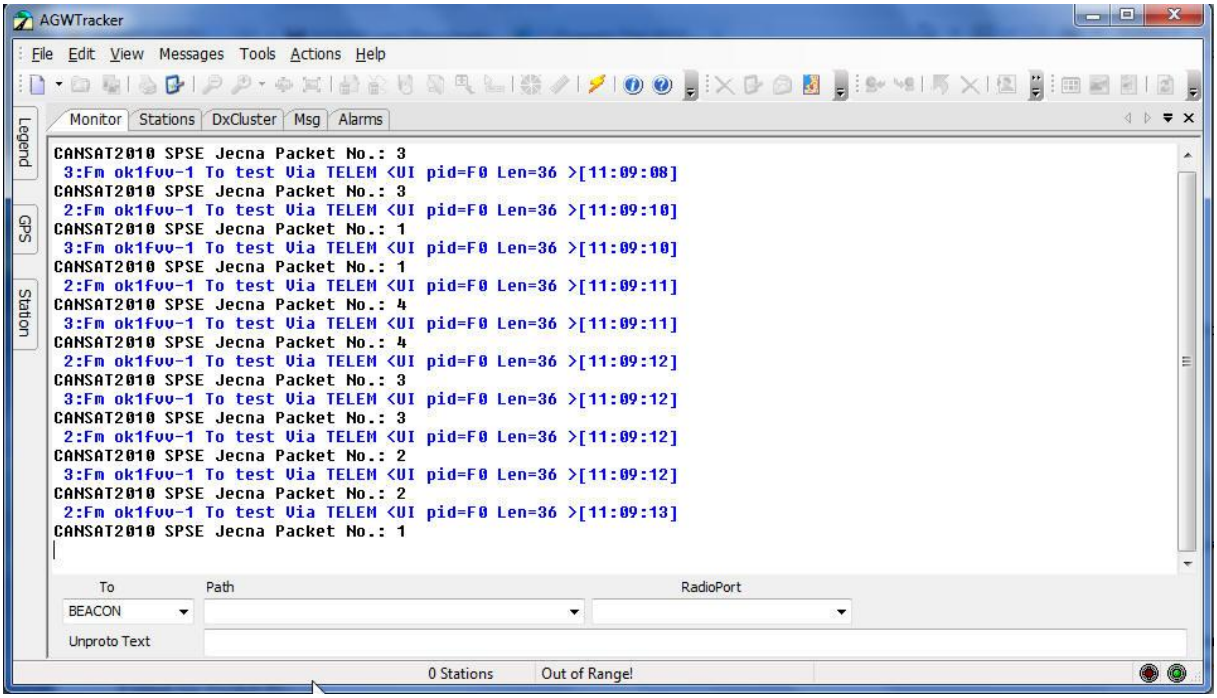


Odsouhlasíme licenci. Klikneme na **Next**

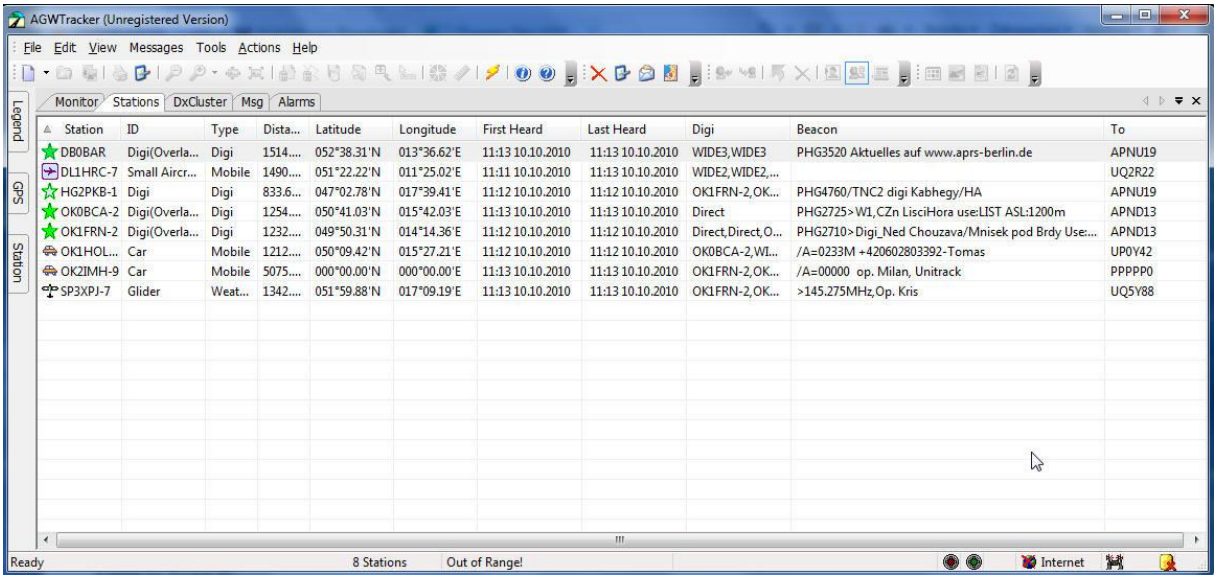
Ponecháme nabízený adresář, klikneme na **Next.** Spustí se instalace



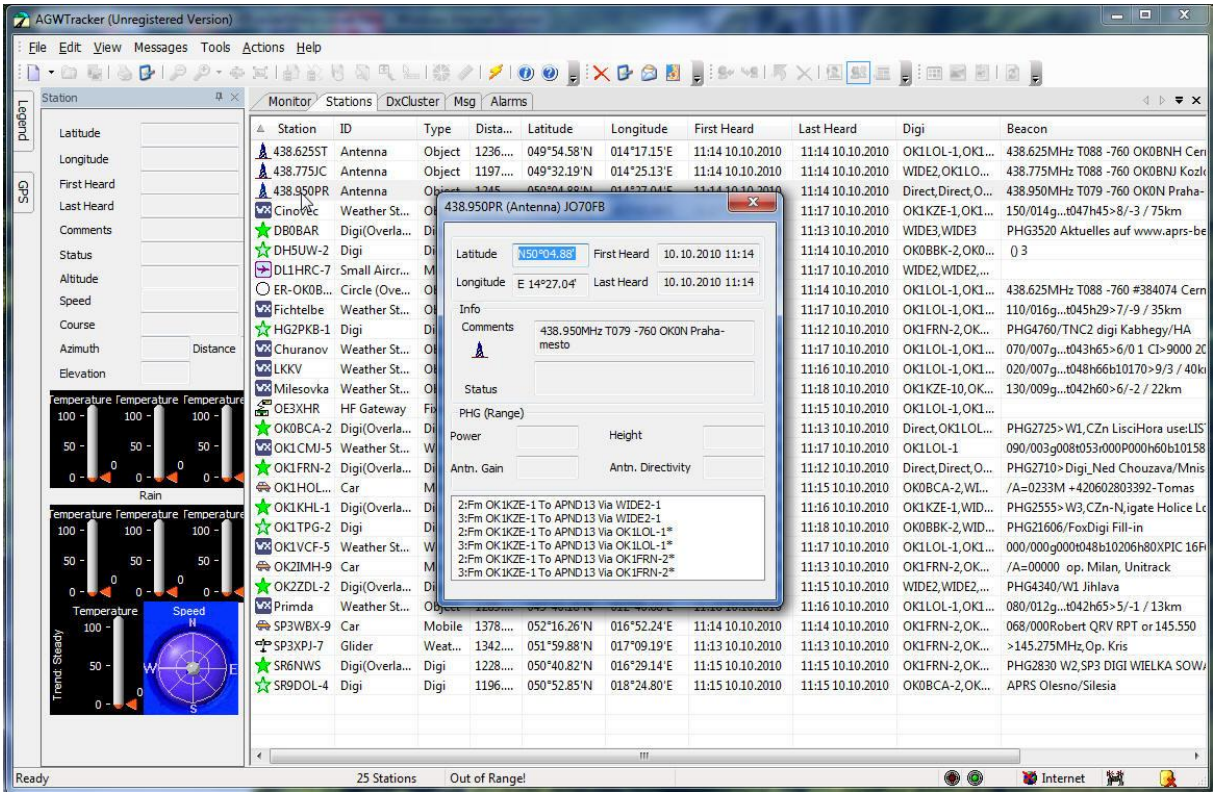
Po skončení instalace již můžeme provozovat tento program. Zahrnuje v sobě jednak funkce **AGW Monitor**u,



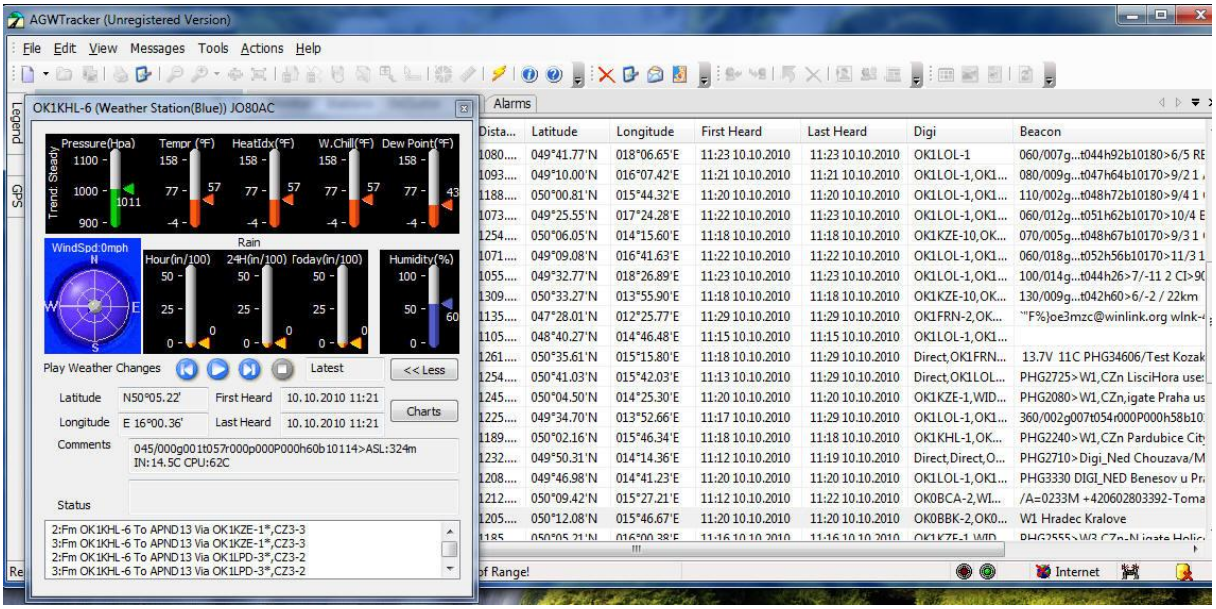
A dále i další funkce. Tento program totiž „rozumí“ obsahu přenášených dat APRS.



Je to umožněno tím, že pro formát přenášených dat existují „de facto“ normy.

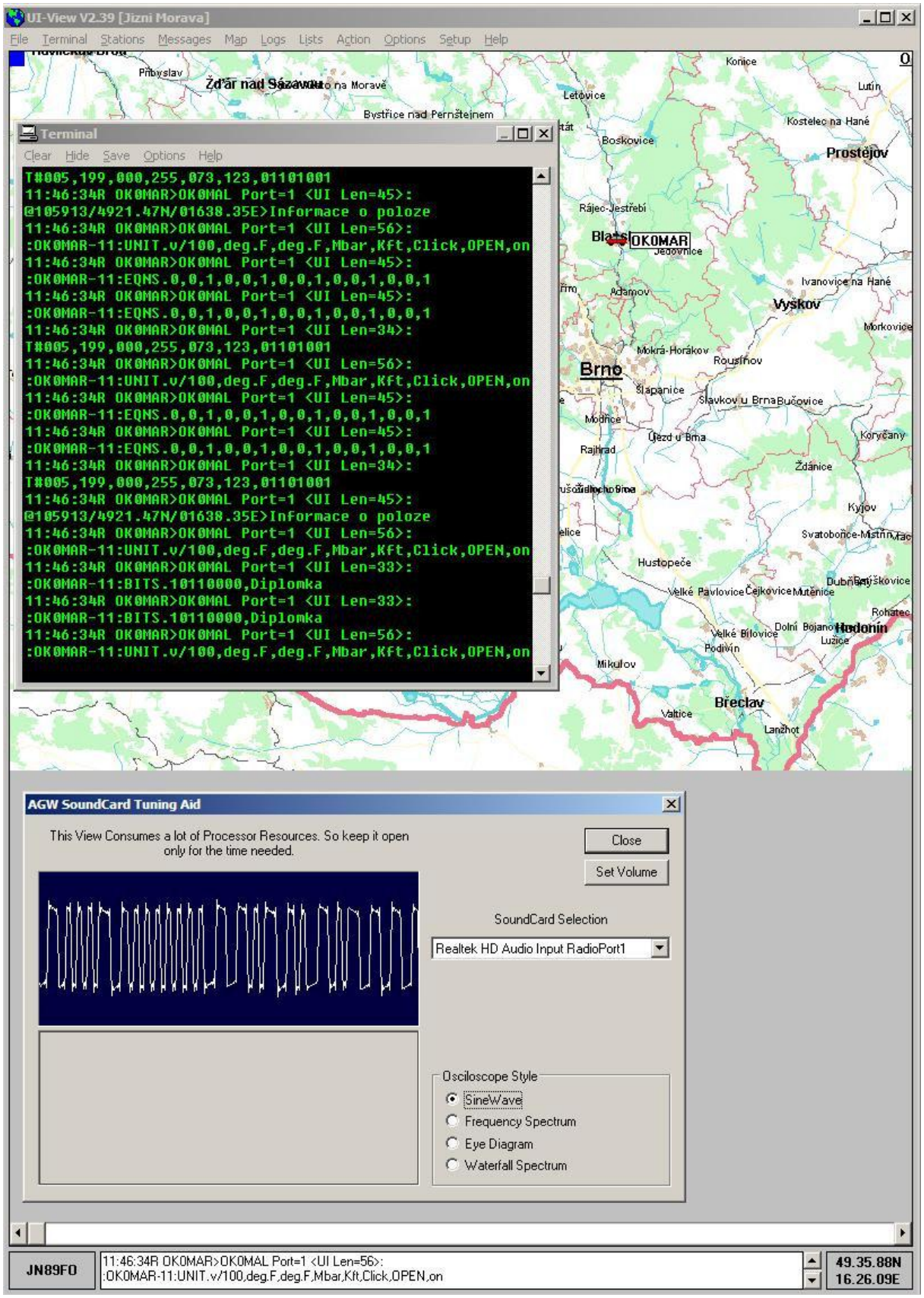


Takže lze např. zobrazit zeměpisné souřadnice příslušné stanice získané z její GPS a odvysílané prostřednictvím APRS.



Obdobně se pomocí APRS někdy přenášejí údaje o počasí naměřené amatérskou meteorologickou stanicí.

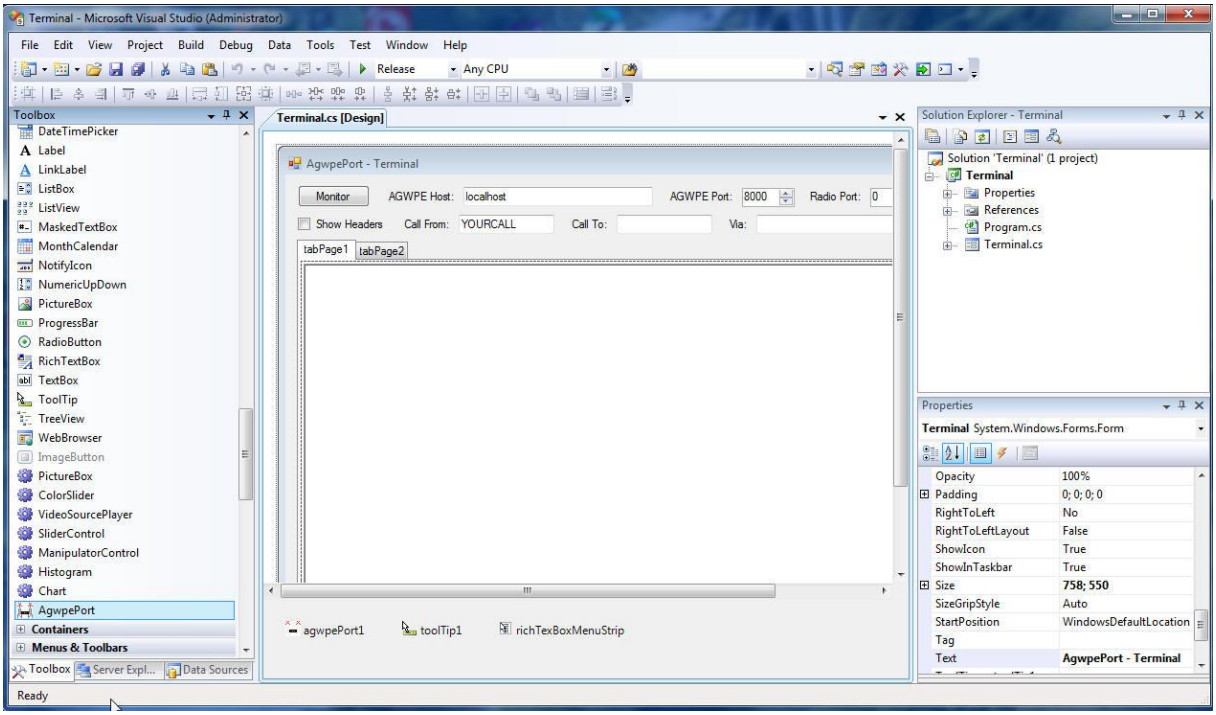
Dále je v prípade pořeby možné použít program zobrazující přijímaná GPS data přímo na mapě. Volne dostupná je starší 16-bitová verze programu UI-View ve verzi 2.39 . Lze nakonfigurovat pro komunikaci s aplikací AGW Packet Engine, což umožnuje přímo zobrazit polohu na mapě. Ve formě zpráv jsou přijímaná i telemetrická data. Přednastavené mapy nejsou moc podrobné, nicméně na internetu lze najít a stáhnout i velmi podrobné mapy určené právě pro tento program. Následující obr. Je z diplomové práce [26] M.Sabola.



6.1.2.5 Vlastní software na platformě .NET napsané v C# (Software Design on .NET platform written in C#)

my remarks: *CanSat Book for Students* – part.1 2011

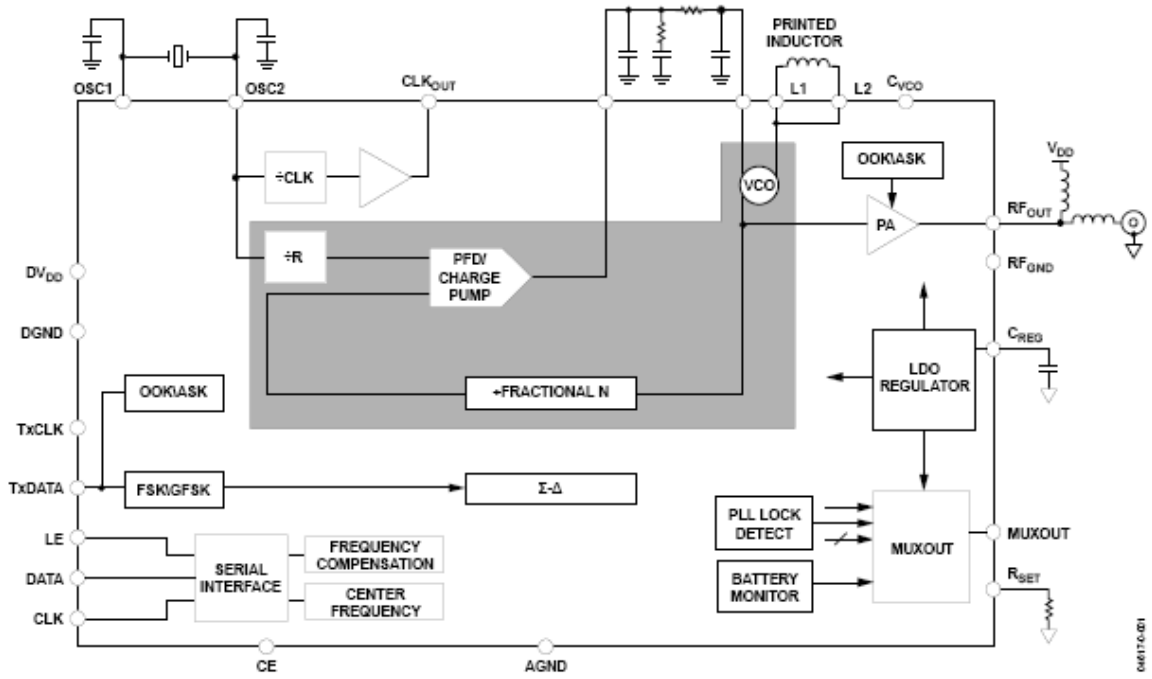
Při popisu činnosti programu **AGW Tracker** v předchozí kapitole si možná řekneme, že by se tento program mohl hodit i pro získání dat vysílaných z našeho CanSatu. Skutečně je tomu tak, ovšem za předpokladu, že palubní počítač CanSatu naprogramujeme tak, aby do vysílače posílal naměřené hodnoty teploty, tlaku, GPS atd. ve formátu APRS. Může se ovšem stát, že z nějakého důvodu tento postup nechceme nebo nemůžeme použít. Např. proto, že pro naše data žádný takový APRS formát neexistuje. Tak tomu bude např. pokud budeme měřit nějaké záření, radioaktivitu, přenášet obraz z kamery apod.
V tom případě si vhodný formát přenášených dat nadefinujeme sami a naprogramujeme palubní počítač. Na PC základnové stanice můžeme použít AGW Monitor a obsah přijímaných paketů pak uložit do textového souboru. Obsah textového souboru můžeme posléze ručně vyhodnotit, popř. převést např. do excelu apod.
Můžeme se ale napsat i vlastní program obdobný **AGW Tracker**u. Není to nic obtížné díky tomu, že máme k dispozici [16] free .NET komponentu **AgwpePort**. Program napíšeme např. ve *VisualStudio 2008*



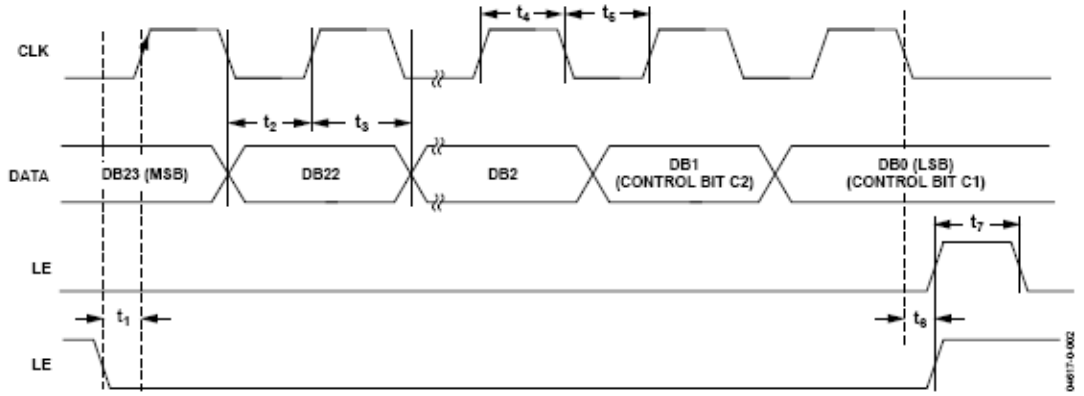
Ze stránek projektu [16] můžeme stáhnout i zdrojový kód komponenty a zdrojové kódy příkladů (Monitor a Terminál). Bude-li např. náš CanSat na Zem mj. vysílat i svou okamžitou polohu (z GPS), můžeme napsat program, který bude na mapě zobrazovat trajektorii pohybu CanSatu a z naměřené hodnoty tlaku průběžně počítat i výšku CanSatu nad zemí a průběžně predikovat místo přistání, což umožní snadnější nalezení Cansatu po dopadu na zem i v případě ztráty radiového signálu.

6.1.3 Vlastní konstrukce vysílače – obdoba Pratt Hobbies vysílače (Our transmitter design –Pratt Hobbies like transmitter)

Vysílač obsahuje dva integrované obvody a to jednočipový počítač *ATMega88* firmy ATMEL a jednočipový vysílač pro kmitočty 50MHz až 1GHz *ADF7012* firmy *AnalogDevices*. Jeho vnitřní struktura je:



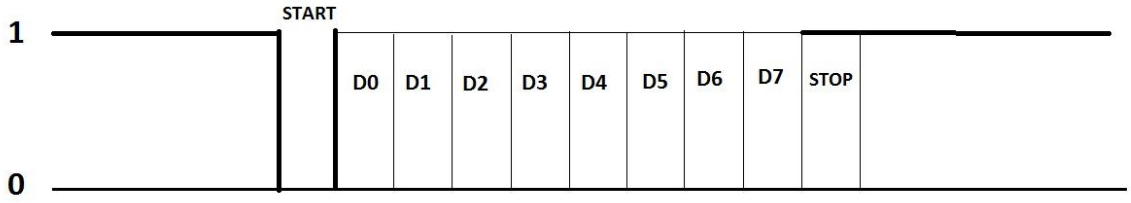
Vidíme, že jde o obvod poměrně složitý a univerzální. To, jak bude pracovat se musí na začátku jeho činnosti nastavit, provede se jeho **inicializace**. Ta se provádí tak, že prostřednictvím vstupů **LE**, **DATA** a **CLK** do něj pošleme sériově čtyři 32 bitová slova. Výrobce označuje jednotlivé bity těchto slov DB31, DB30 …….. DB1 a DB0. V tomto pořadí se také do obvodu *ADF7012* při jeho inicializaci posílají. TJ. posílají se sériově počínajíc nejvíce významným bitem. Význam jednotlivých bitů si ukážeme v následujících podkapitolách. Zde jen uvedu, že dva nejméně významné bity DB1 a DB0 tvoří tzv. adresu. Tyto adresy jsou binárně 00, 01, 10 a 11 což je dekadicky 0, 1, 2 a 3. Adresa 0,1,2 či 3 určuje, do jakého z registrů Registr 0, Registr 1, Registr 2 či Registr 3 se bude provádět zápis. Co určuje obsah těchto registrů si rovněž později popíšeme. Tyto registry mají i další pojmenování dané jejich významem. Sériovou komunikaci pomocí vstupů **LE**, **DATA** a **CLK** popisuje výrobce následujícím obrázkem:



K obvodu *ADF7012* (piny OSC1 a OSC2) je připojen externí krystal 24,576 MHZ. Obvod je v inicializační části (viz výše) m.j. nastaven tak že na výstupu $\mathbf{CLK_{out}}$ generuje pulzy o polovičním kmitočtu, než je kmitočet vnitřního oscilátoru 24,576 tj 12.288MHz. Tyto pulzy jsou přivedeny na hodinový vstup jednočipového počítače *ATMega 88*. Ten je mj. naprogramován tak, že na začátku vygeneruje signály pro počáteční inicializaci obvodu *ADF7012* kdy nastavuje defaultní kmitočet tohoto vysílače na 433,92 MHZ, vypnutý koncový stupeň PA a modulaci FSK. Obvod *ATMega88* dále na svém vstupu RxD přijímá (*na úrovni TTL*) dálnopisným 8bitovým kódem povely např **M**1200 nebo **F**8D1D1 či **S**ahoj.

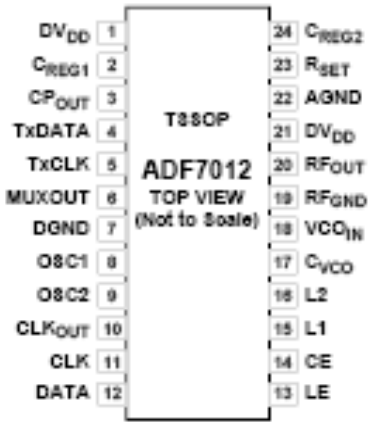
my remarks: *CanSat Book for Students* – part.1 2011

Použitý sériový přenos je znakový, tj přenášíme postupně jeden znak za druhým. Znaky jsou přenášeny jako 8bitů. Přenos znaků naznačuje následující obrázek:

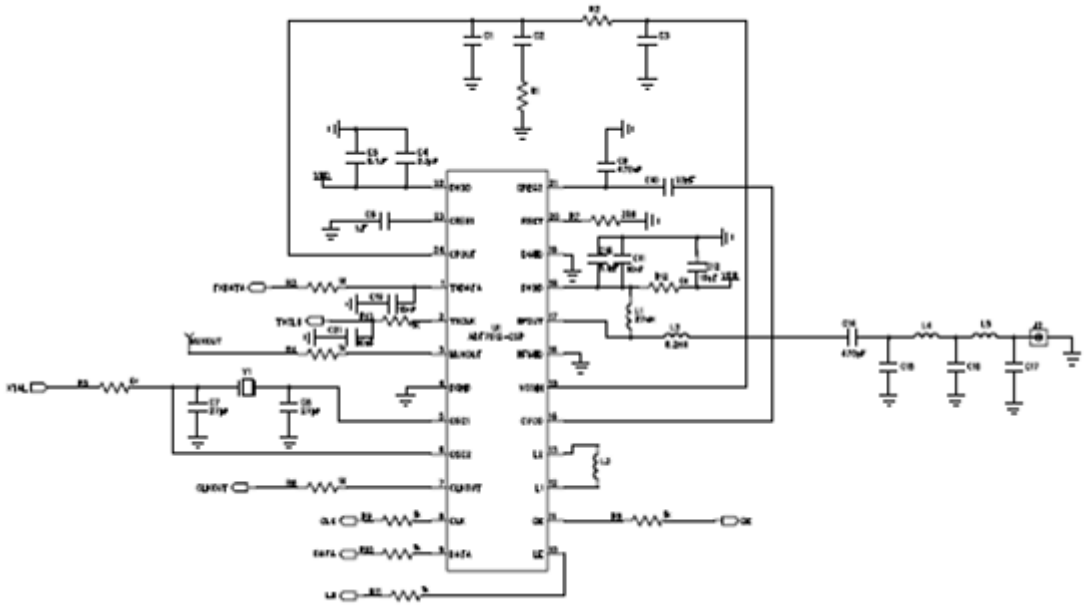


V klidovém stavu je úroveň logická 1. Přenos znaku je zahájen odesláním START pulzu o úrovni logické 0. Potom je postupně přeneseno 8 bitů D0, D1 až D7 a přenos znaku je zakončen odesláním STOP pulzu o úrovni logické 1. Rychlost tohoto přenosu ke 38400 Baudů (při hodinovém kmitočtu *ATMega88* 12.288 MHz). Za příkazy se musí poslat znak CR (hexadecimálně OD) .

Po přijmu povelu počítač *ATMega88* v případě povelu **M** nastaví druh modulace a rychlost přenosu dat, což provede posláním příslušných 32bitových slov do odpovídajících registrů (později to probereme podrobně) vysíláním řídících signálů **LE**, **DATA** a **CLK**. Na příkaz **S** reaguje počítač tak, že následný řetězec doplní o další bity před a za ním tak, aby to byl úplný AX25 packet a ten rychlostí uvedeném v předchozím příkazu **M** tj. 1200 Bd pošle do vstupu **TxData** obvodu *ADF7012*.
Konkrétní zapojení pinů tohoto obvodu je na obr

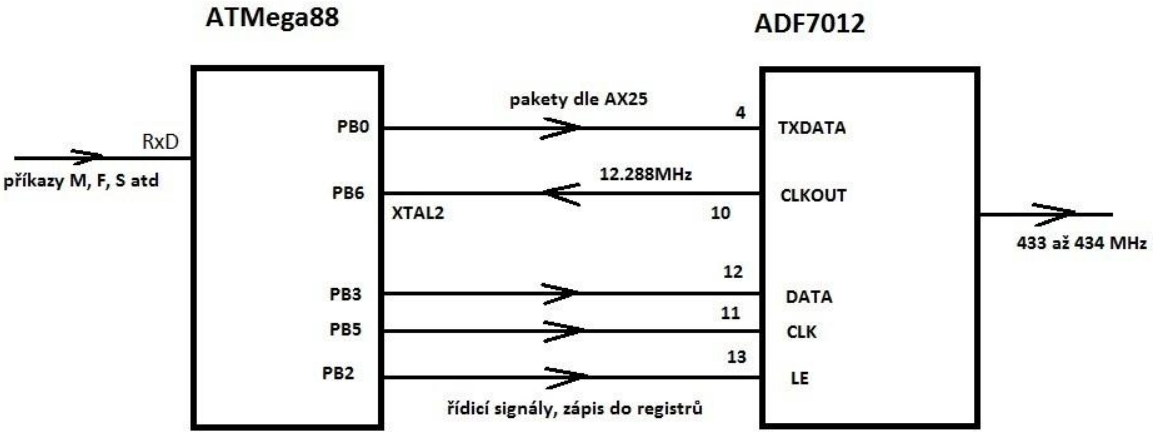


A jeho doporučené zapojení (aplikace) je na obr.



6.1.3.1 Analýza vysílače *Pratt Hobbies* (Pratt Hobbies Transmmiter Analyse)

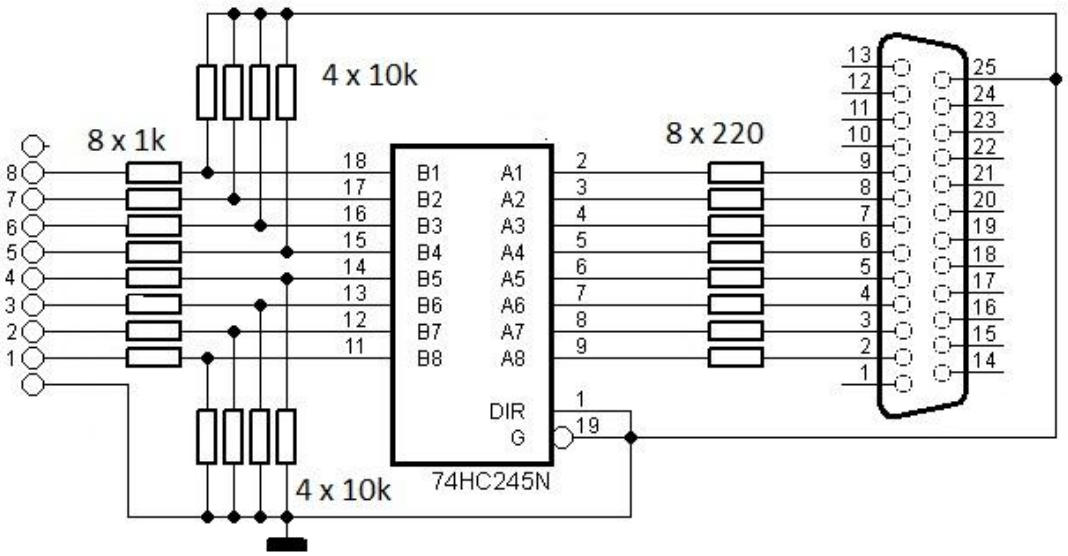
Zjednodušené zapojení vysílače je:



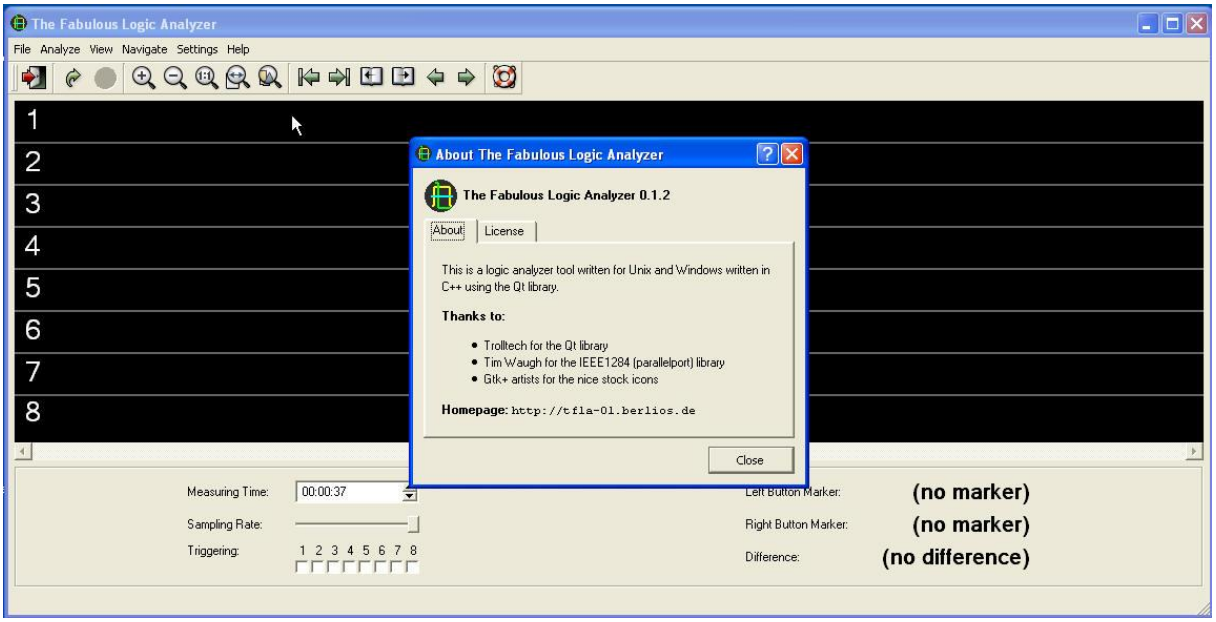
Funkce obvodu *ADF7012* je dostatečně popsána v dokumentaci výrobce obvodu, činnost obvodu *ATMega88* je dána tím, jak je naprogramován, tj činností firmware. Nepodařilo se mi sice získat jeho zdrojové kódy, nicméně binární kód lze z jeho programové flash paměti zkopírován. Je uveden v příloze. Existuje několik disassemblerů pro počítače ATMEL AVR, ty ale umožňují získat zpětným překladem jen kód v assembleru AVR, navíc bez komentářů a mnemotechnického označení proměnných. Kód ve vyšším jazyce (např. v C) tak stejně nezískáme a analýza kódu v assembleru, či simulace činnosti obvodu není jednoduchá.

Jednodužší je experimentálně sledovat odezvu *ATMega88* na řídící povely **M**, **F** a **S** připojením logického analyzátoru odpovídající výstupní piny portu PB obvodu *ATmega88*. Drobným problémem je, že zřejmě nemáme logický analyzátor. Můžeme si však jednoduchý analyzátor postavit sami. Potřebujeme k tomu počítač s výstupem LPT (25 pinový CANON konektor). Použijeme starší počítač,

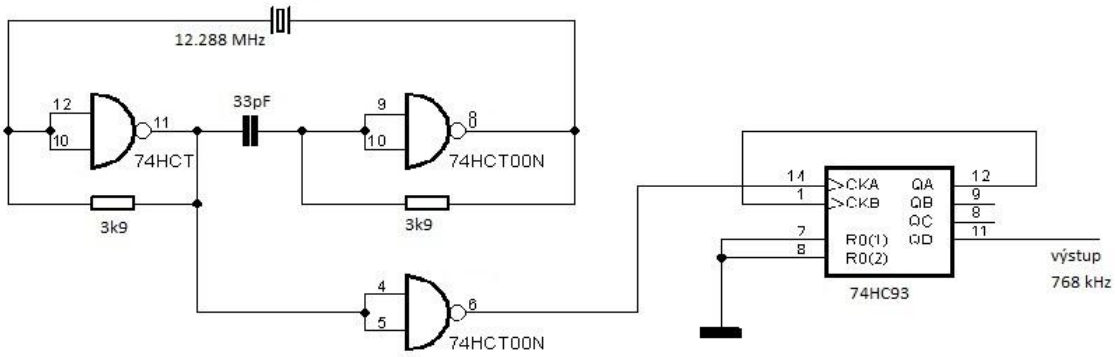
novější mají většinou jen USB. Jako operační systém na takovém starém PC se pro tyto účely osvědčil WIN XP. Zapojení takového „logického analyzátoru“ je:



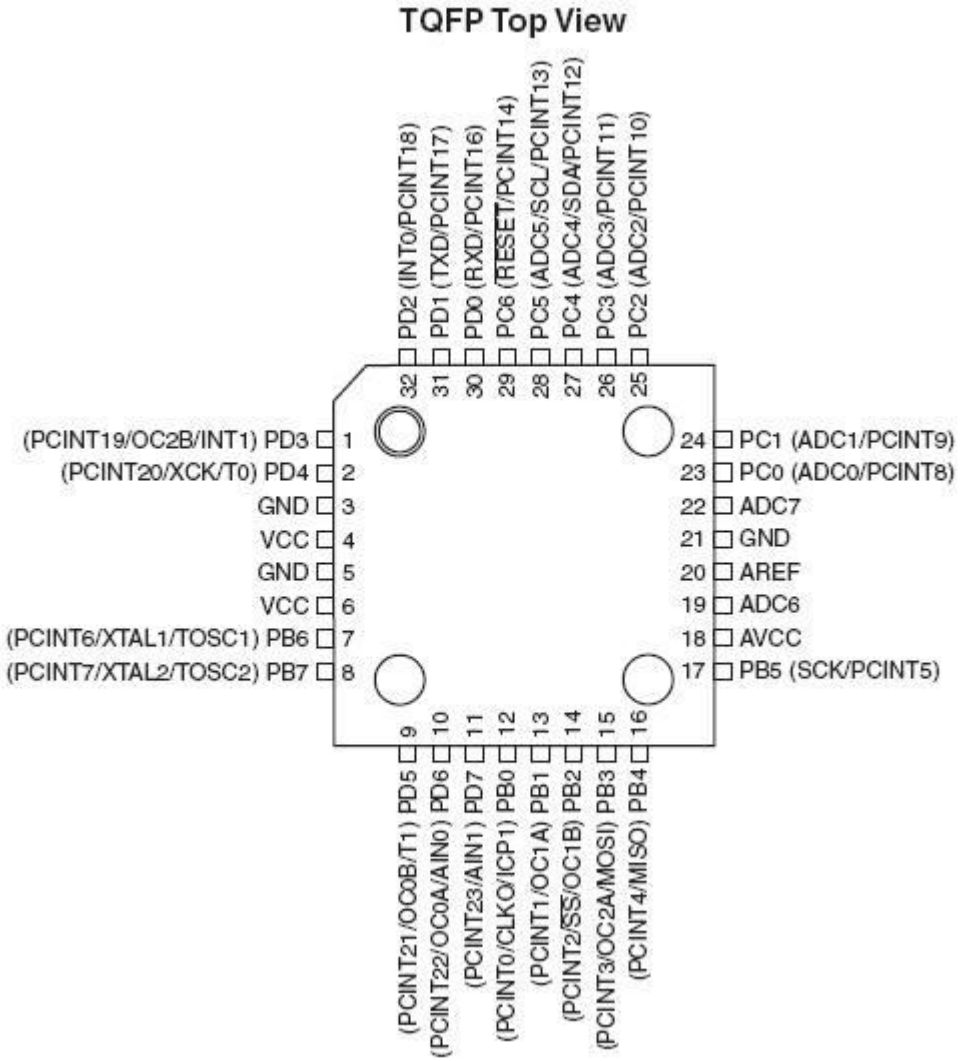
Tento logický analyzátor ke své činnosti potřebuje software. Používám free **The Fabulous Logic Analyzer 0.1.2**

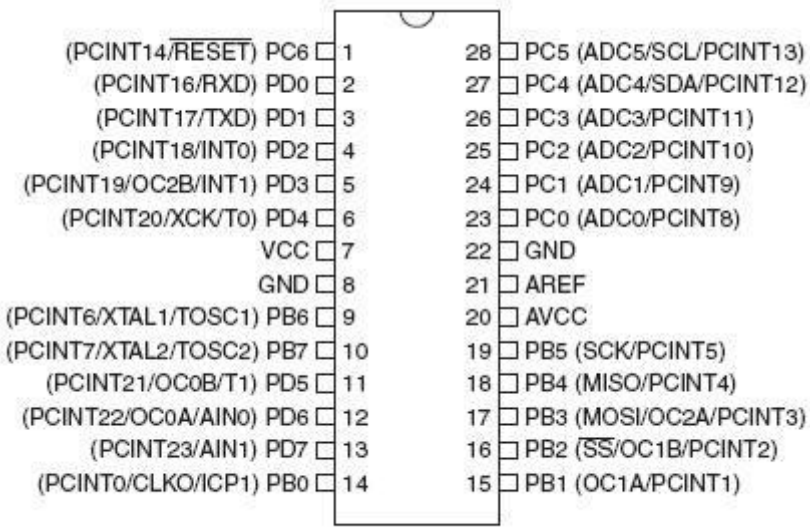


Mezní kmitočet tohoto logického analyzátoru je cca 500kHz, což je pro naše účely téměř nepoužitelné, ovšem za předpokladu, že hodinový kmitočet *ATMega88* je 12.288MHz. Pro účely analýzy však není nutné mít tento hodinový kmitočet. Dokonce ani není nutné mít počítač *ATMega88* propojený s *ADF7012*. Výhodnější je mít počítač *ATMega88* samostatný a používající hodinový kmitočet např. 16 x nižší z nějakého externího generátoru. Postavil jsem pro účely analýzy činnosti firmware *ATMega88* generátor hodinového kmitočtu viz následující obrázek.

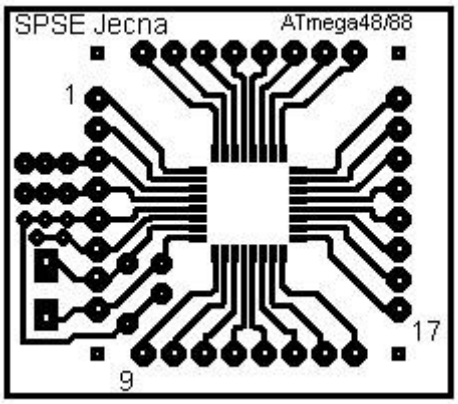


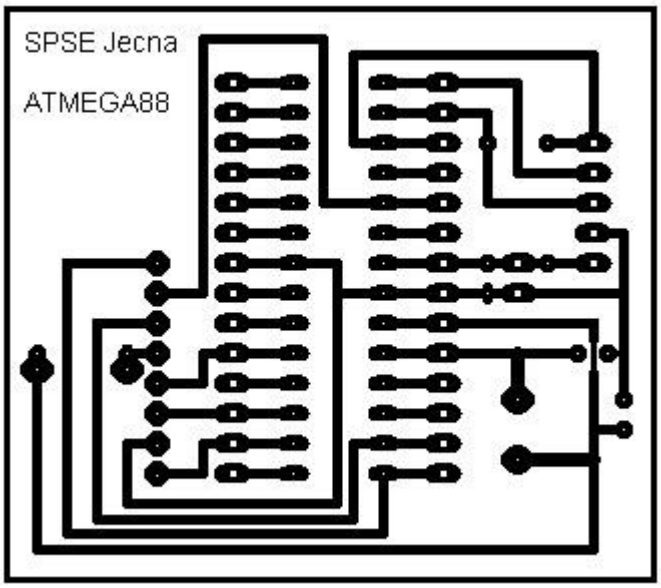
Obvod *ATMega88* je pak zapojen takto



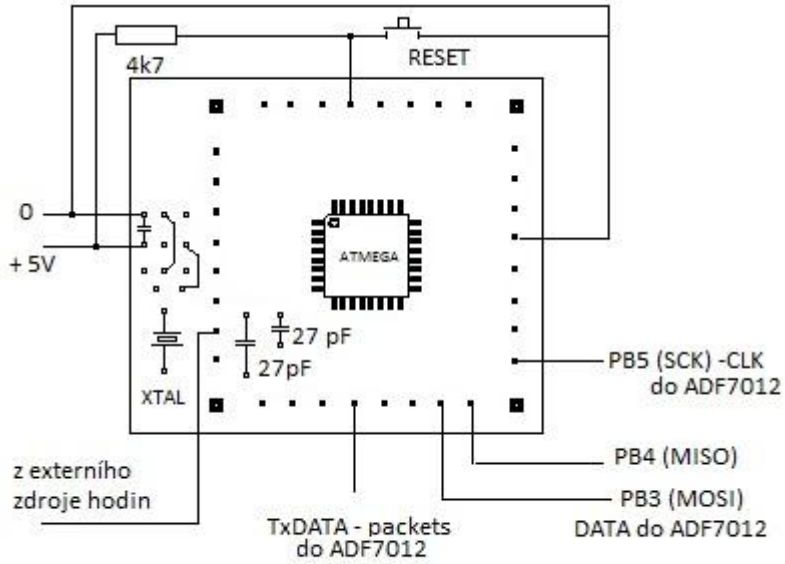


Řídící počítače vysílače s ATMega může kromě tohoto čipu obsahovat již jen vnější krastal a konektory (popř. pájecí plošky pro propojovací vodiče):

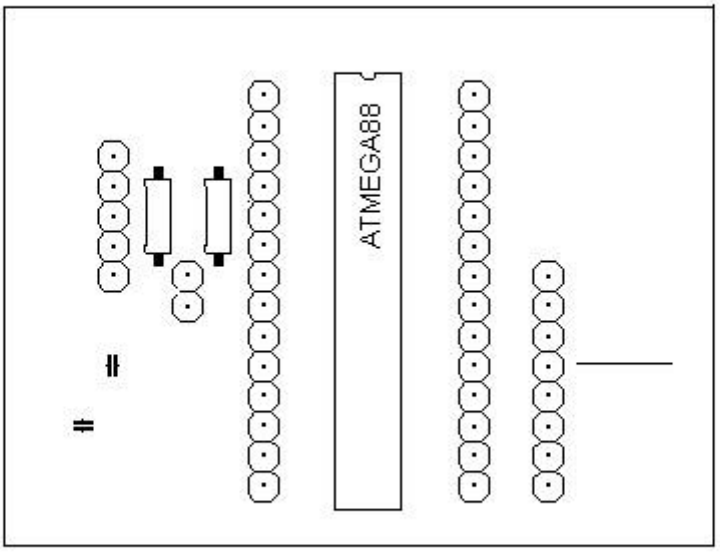




Rozložení součástí:

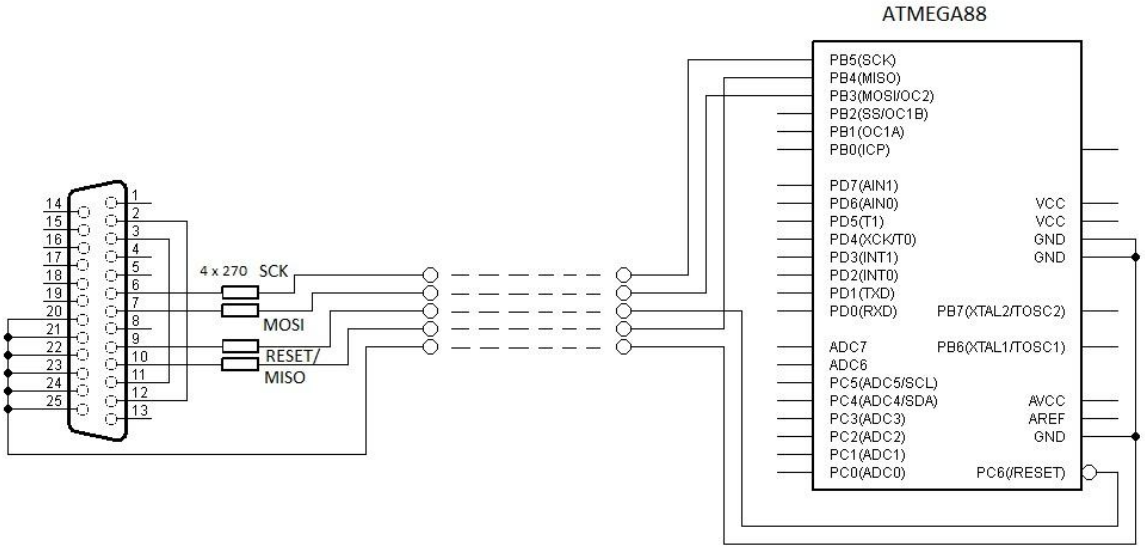


krystal 12.288 MHz a kondenzátory 27 pF osadit jen v případě vhodného nastavení fuses

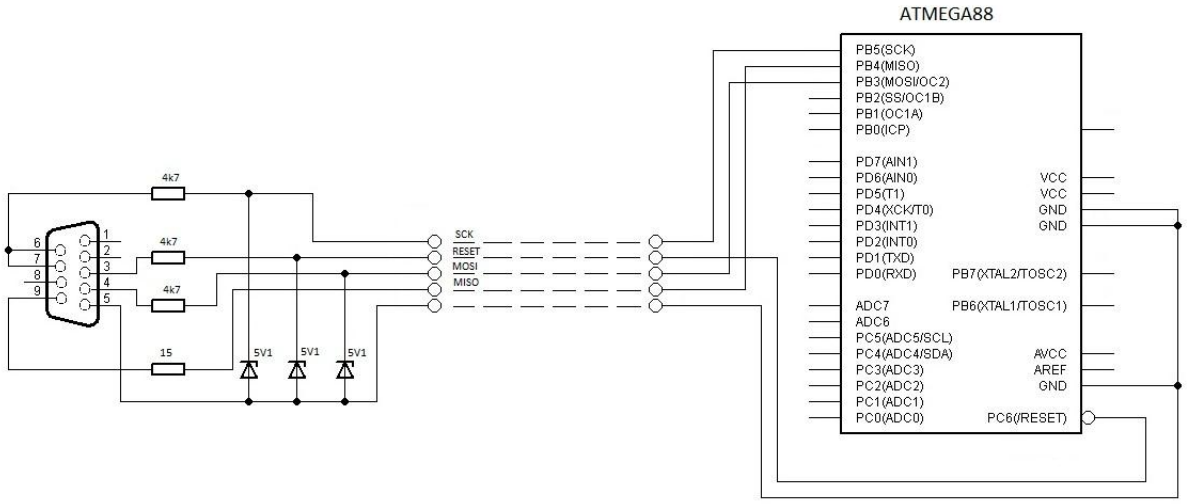


K naprogramování *ATMega88* lze použít některý z profesionálních či amatérských programátorů. Ukážeme si použití jednoho z nejjednoduššího programátoru spolu s free obslužným programem *PonyProg2000*. Do *ATMega88* naprogramujeme stejný binární soubor, který jsem získal z originálního vysílače *PrattHobbies*, jeho výpis viz. příloha. Domovská stránka *PonyProg* je http://www.lancos.com a najdeme na ní jak zapojení programátorů, tak program *PonyProg2000*. Ten mj. umožňuje ISP (In Systém Programming) procesorů ATMEL AVR včetně *ATMega88*. Programátor komunikuje s těmito

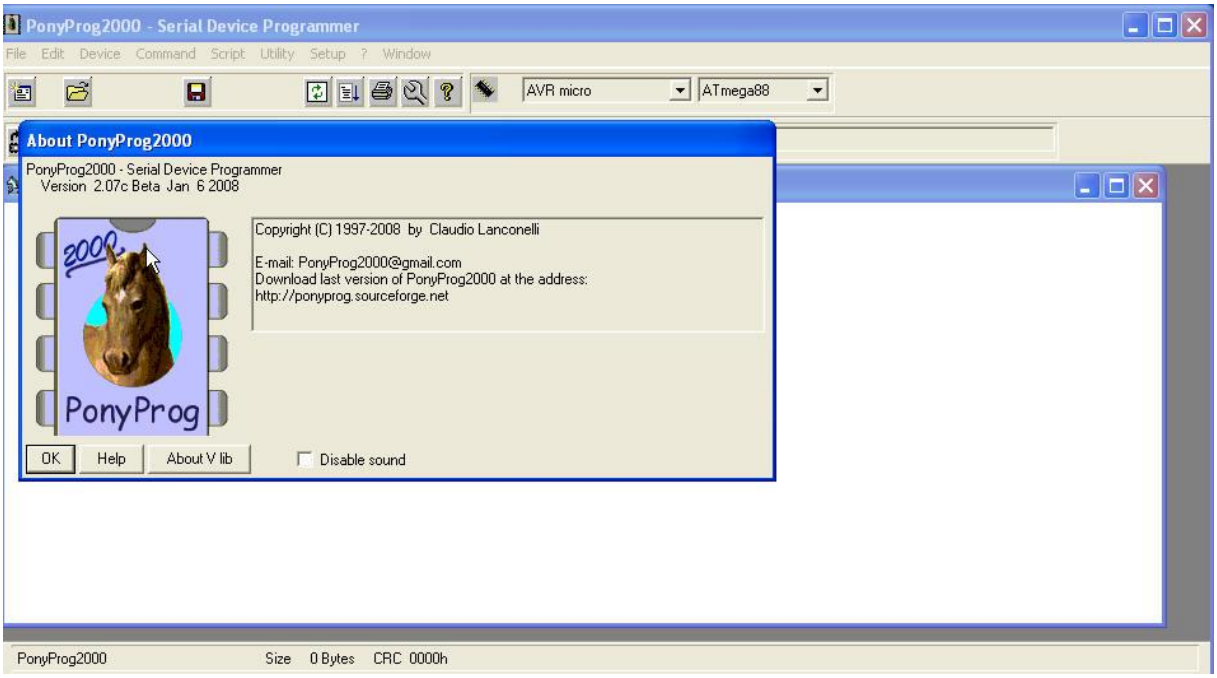
obvody pomocí jejich 4 pinů: **MOSI**, **MISO**, **SCK** a **Reset/**. Programátor, se kterým komunikuje PC prostřednictvím paralelního portu je zapojen podle následujícího obrázku.



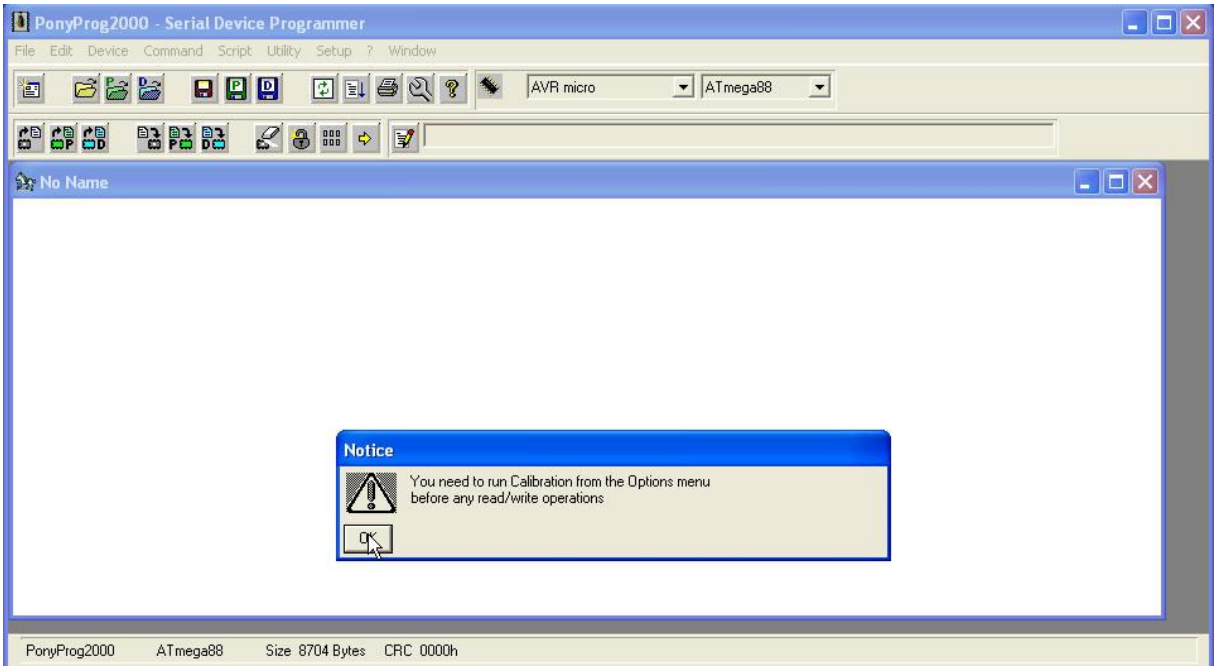
Programátor, s nímž PC komunikuje prostřednictvím sériového portu má zapojení:



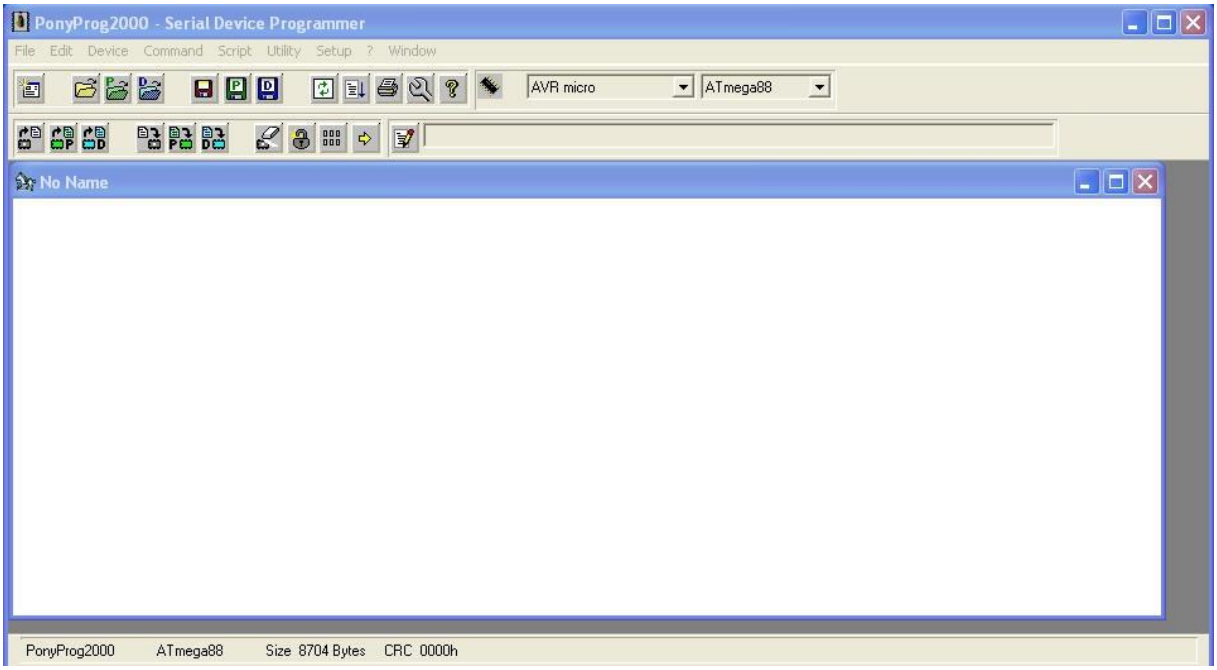
Jeho výhodou je, že pracuje i s počítačem majícím jen USB a s převodníkem USB-COM. Převodník se dá běžně zakoupit a jeho cena je několik set korun. Nyní si již můžeme popsat práci s *PonyProg2000*. Použil jsem přitom sériový programátor a programoval jsem jím nově zakoupenou *ATMega88*. Po spuštění tohoto programu se objeví:



Klikneme na tlačítko **OK**. Dostaneme

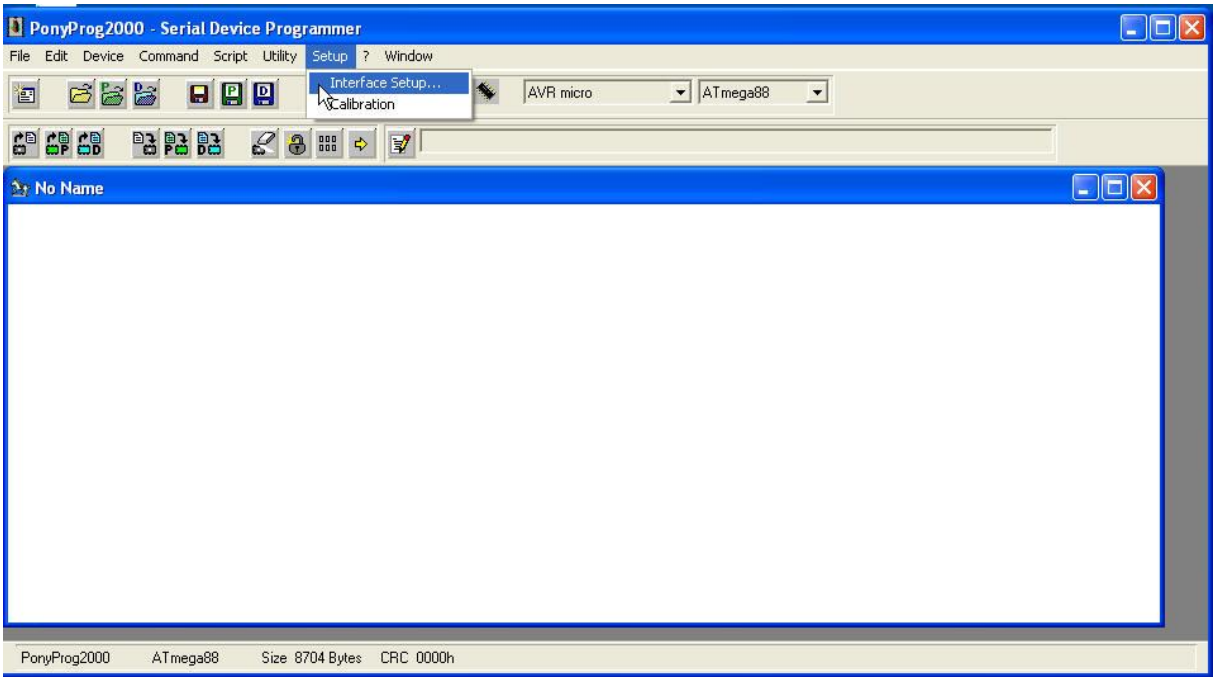


Nyní klikneme na tlačítko **OK** v okně **Notice**. Nyní již můžeme s programem pracovat. Nejprve v jeho horní části zvolíme obvod *ATMega88* :

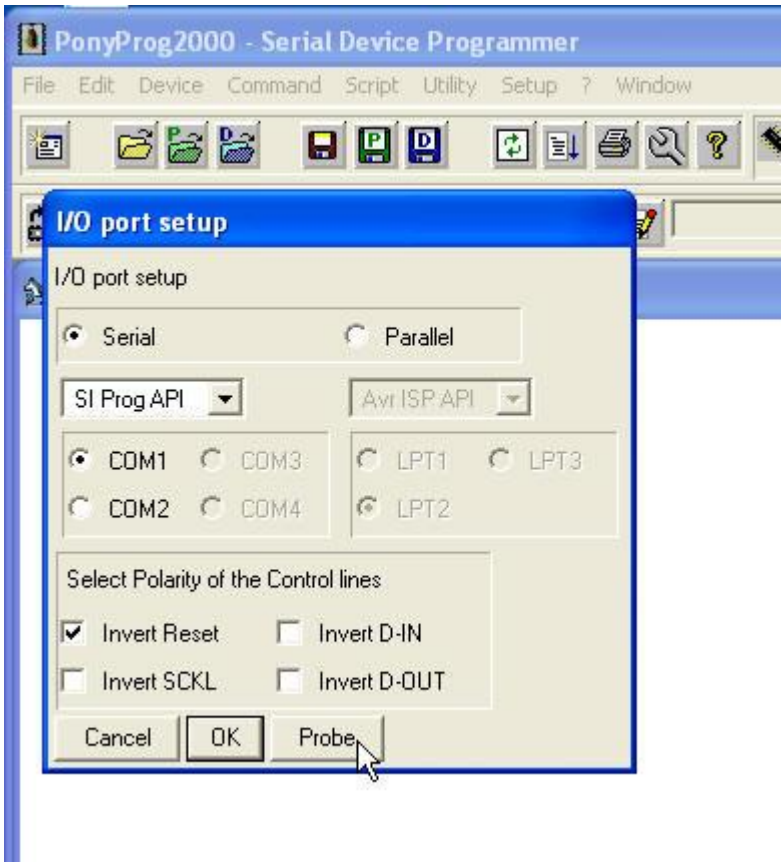


Všimněme si nastavení

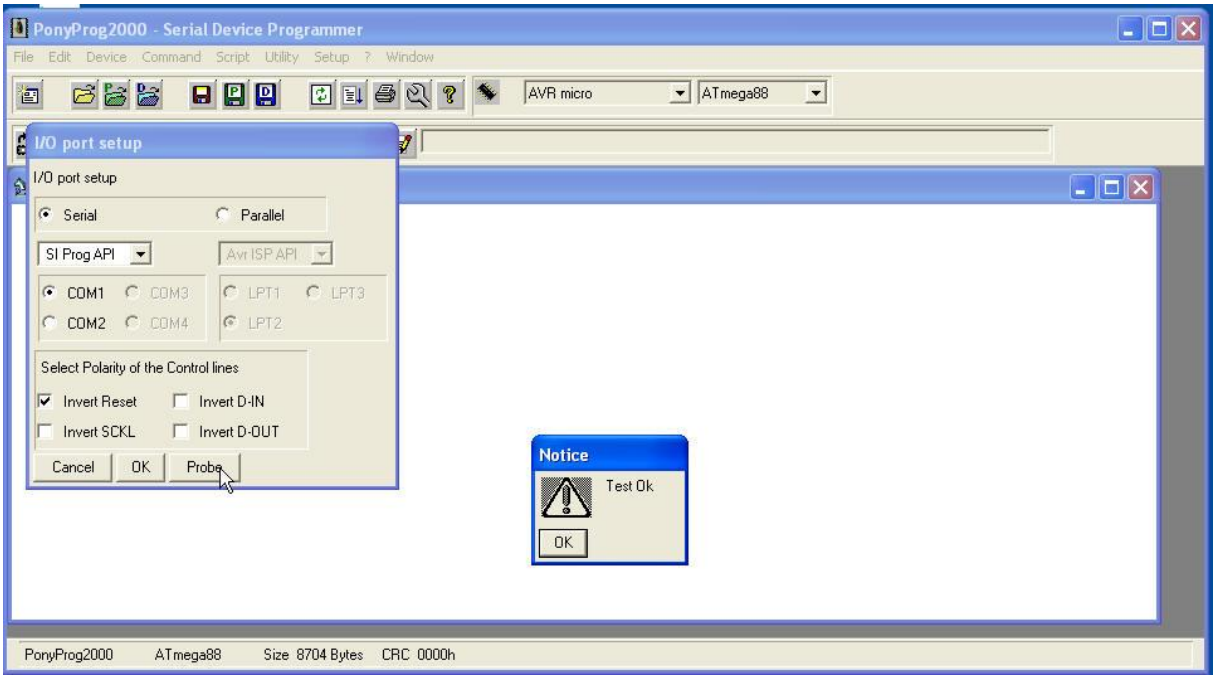
Dále vybereme programátor pomocí menu **Setup → Interface Setup …**



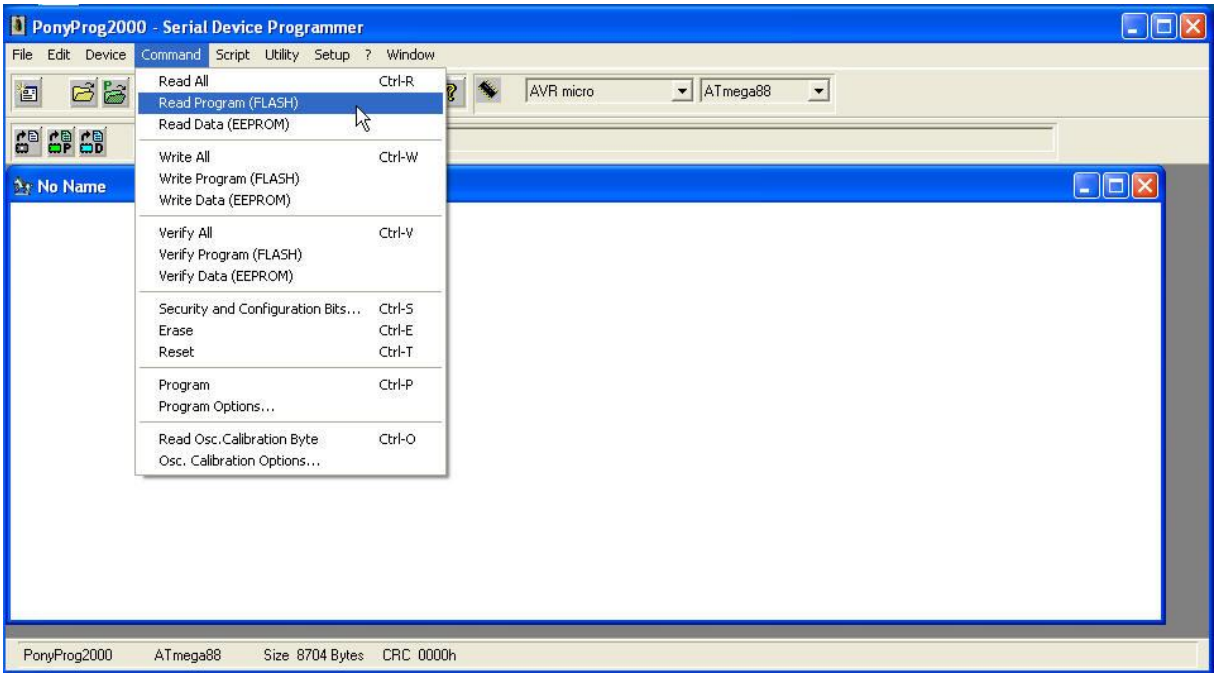
Dostaneme



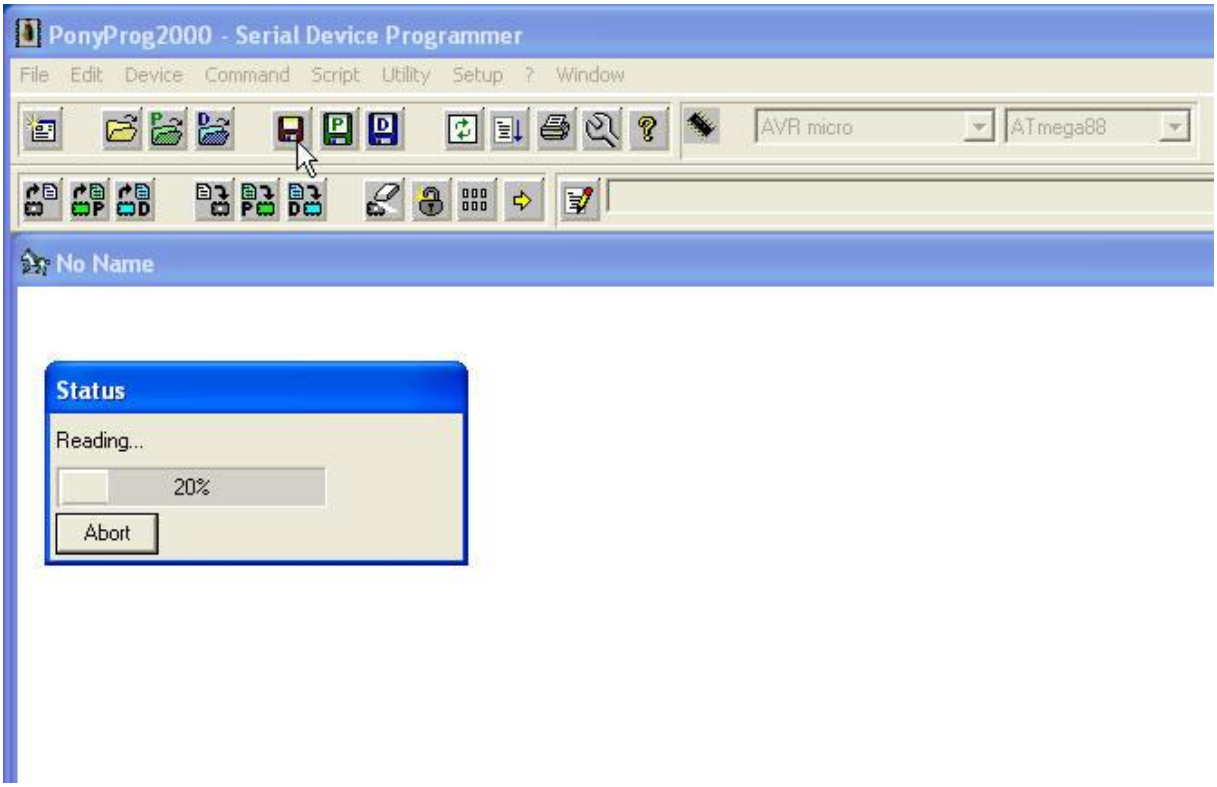
Vybral jsem **SI Prog API** a sériový port **COM1** a ještě jsem zaškrtnul **Invert Reset**. Kliknutím na tlačítko **Probe** zkontrolujeme, že přítomnost programátoru



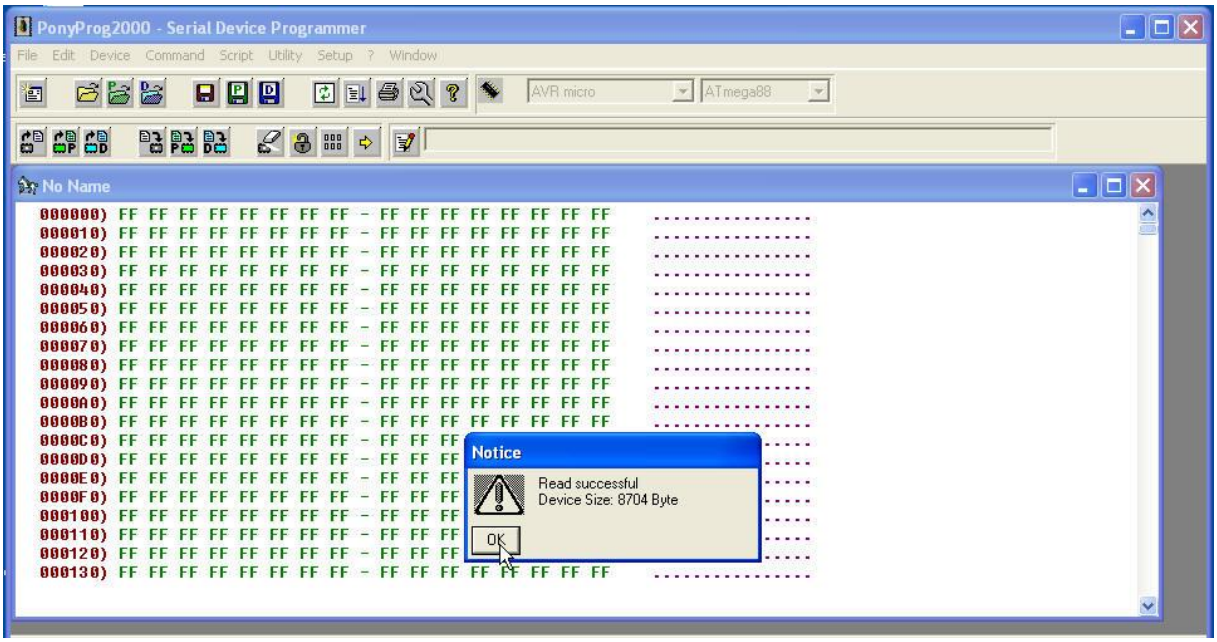
Ověříme si, že *PonyProg2000* komunikuje s připojeným *ATMega88*. Vybereme menu **Command → Read Program (FLASH)**



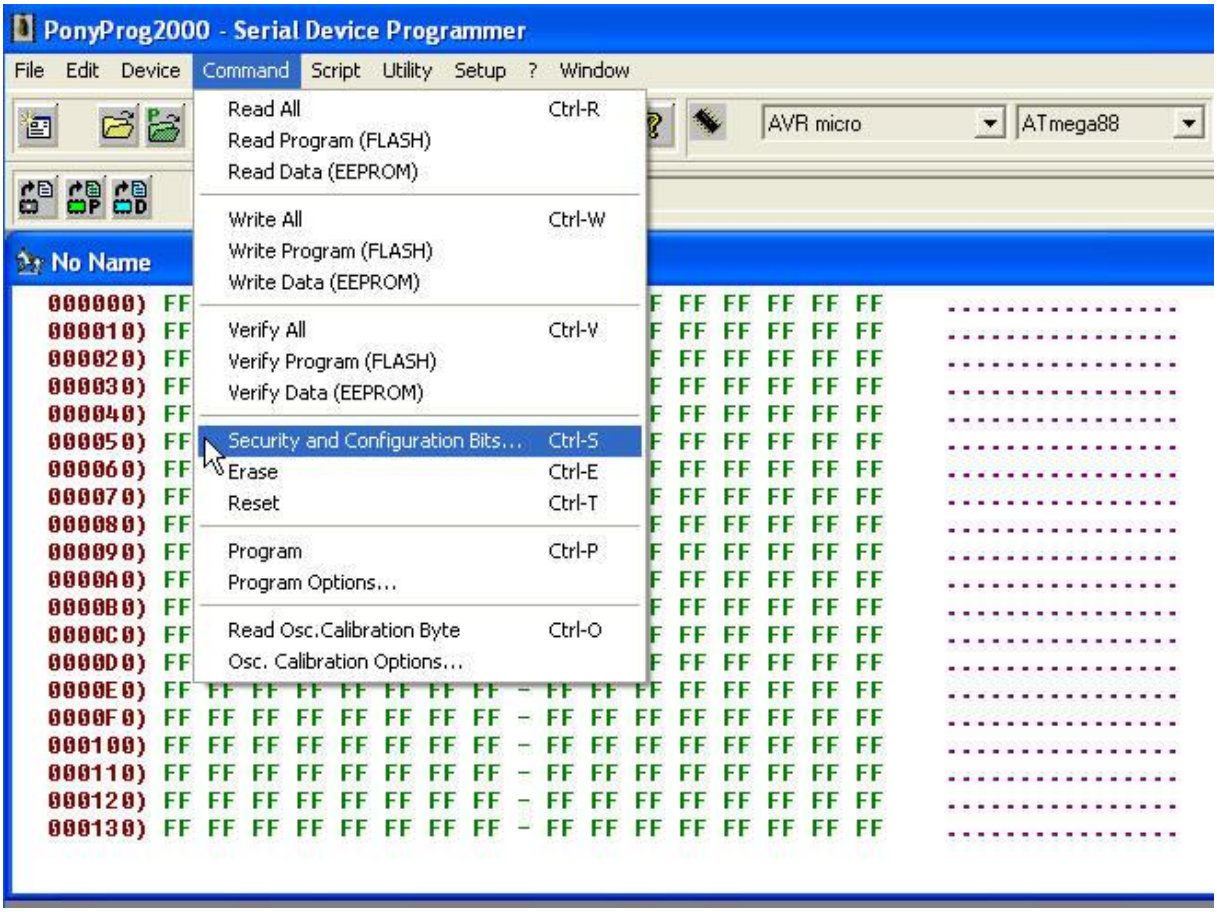
Obsah programové paměti Flash se začne načítat



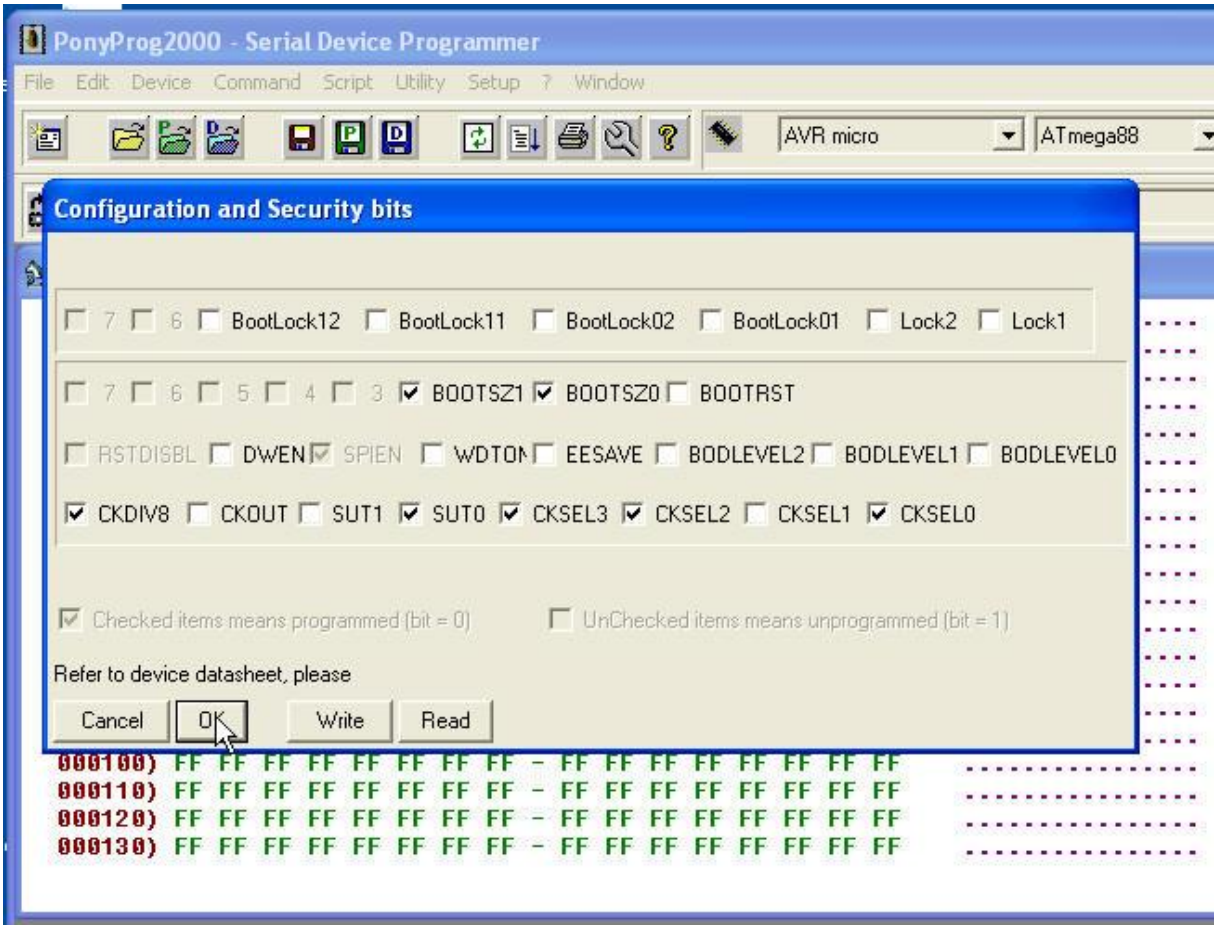
Takže dostaneme



Klikneme na tlačítko **OK** informačního okna Notice potvrzujícího úspěšnost čtení. Vidíme, že nový obvod ATMega88 má v programové paměti samé jedničky, tj v každém byte této paměti je **11111111** což je **FF** v hexadecimálním kódu. Ještě se podíváme na nastavení pojistek **(Fuses)** nového ARMega88. V menu vybereme **Command →Security and Configuretions Bits…**

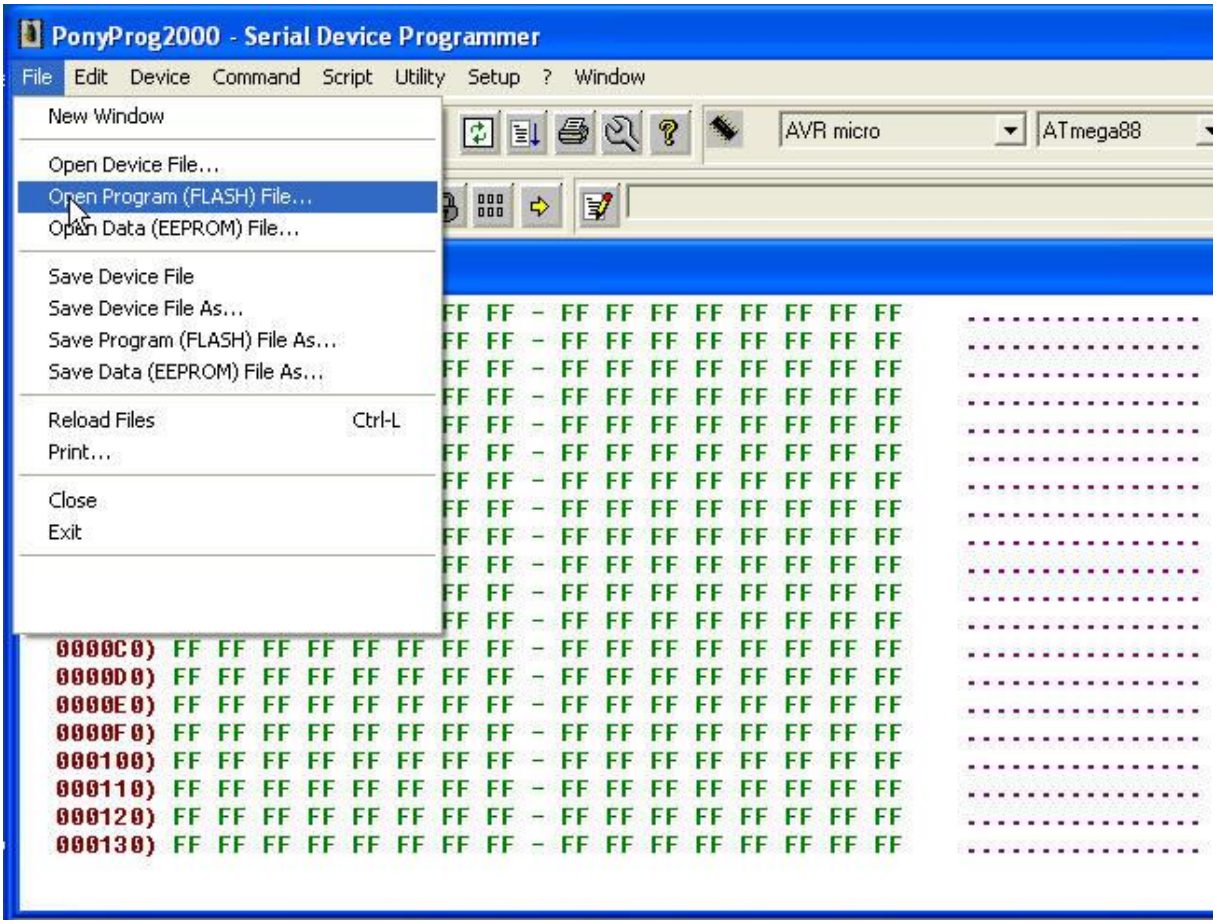

PonyProg2000 - Serial Device Programmer
File Edit Device Command Script Utility Setup ? Window
Read All Ctrl-R
Read Program (FLASH)
Read Data (EEPROM)
Write All Ctrl-W
Write Program (FLASH)
Write Data (EEPROM)
Verify All Ctrl-V
Verify Program (FLASH)
Verify Data (EEPROM)
Security and Configuration Bits... Ctrl-S
Erase Ctrl-E
Reset Ctrl-T
Program Ctrl-P
Program Options...
Read Osc.Calibration Byte Ctrl-O
Osc. Calibration Options...
AVR micro
ATmega88
No Name

Dostaneme

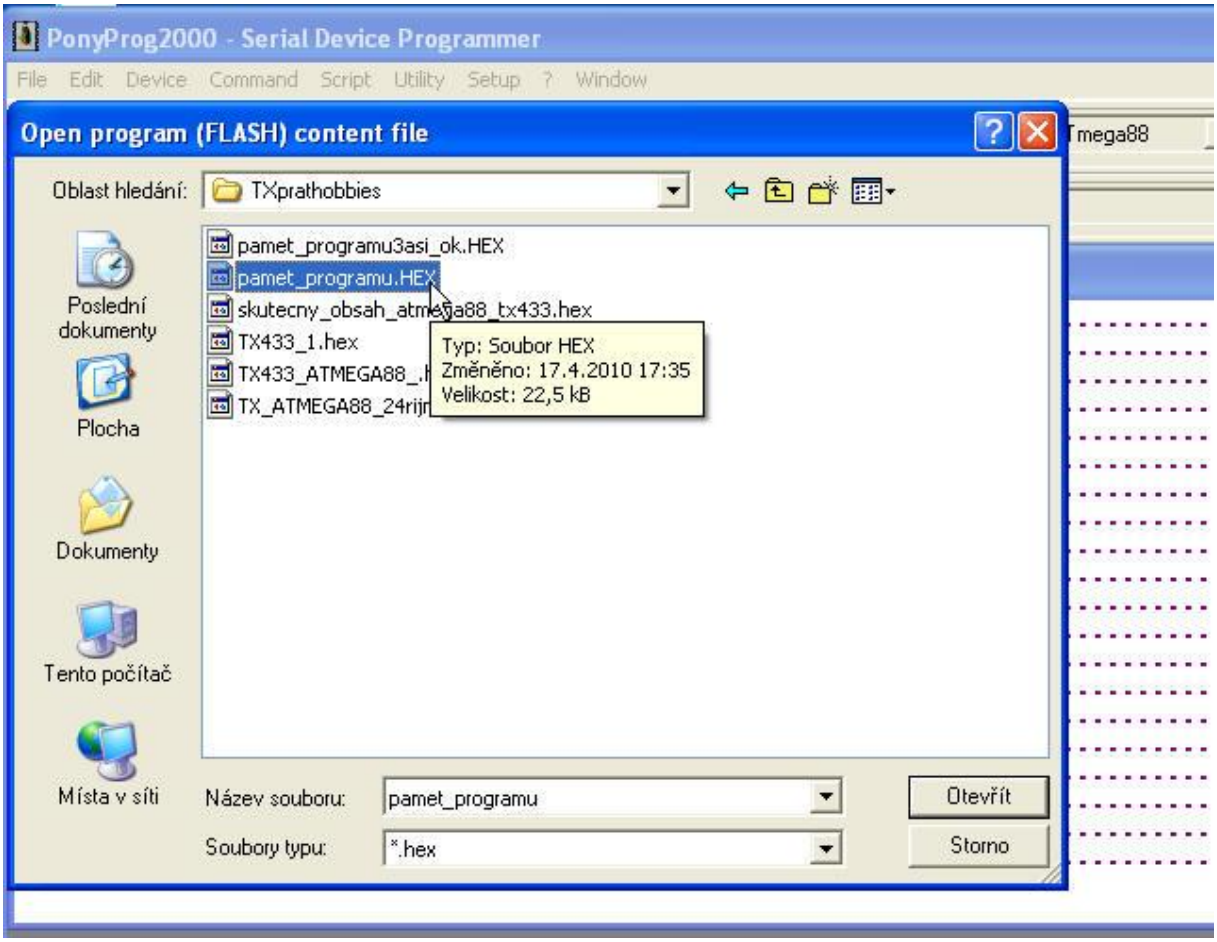


Výše popsaným způsobem lze získat i obsah programové paměti Flash naprogramovaného obvodu i nastavení pojistek tohoto obvodu. Tímto způsobem jsem získal i binární kód firmware vysílače *PrattHobbies.*

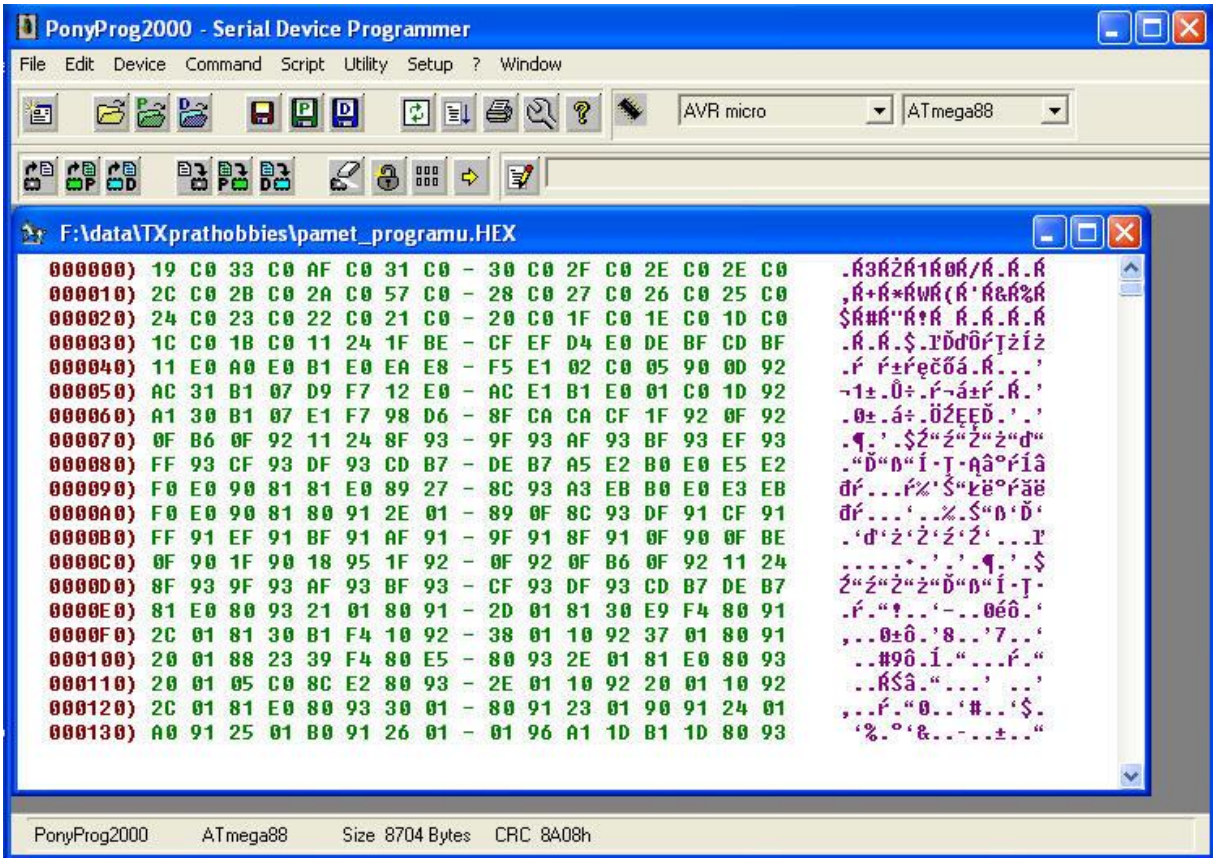
Nyní zapíšeme tento firmware do nové *ATMega88*. V menu vybereme **File →Open Program (FLASH) File**



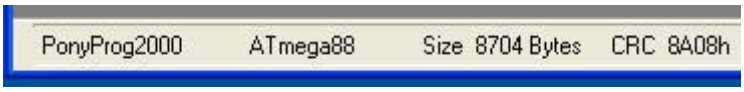
Vybereme příslušný soubor. Může to být jak binární soubor (extenze .bin) tak hexadecimální (.hex)



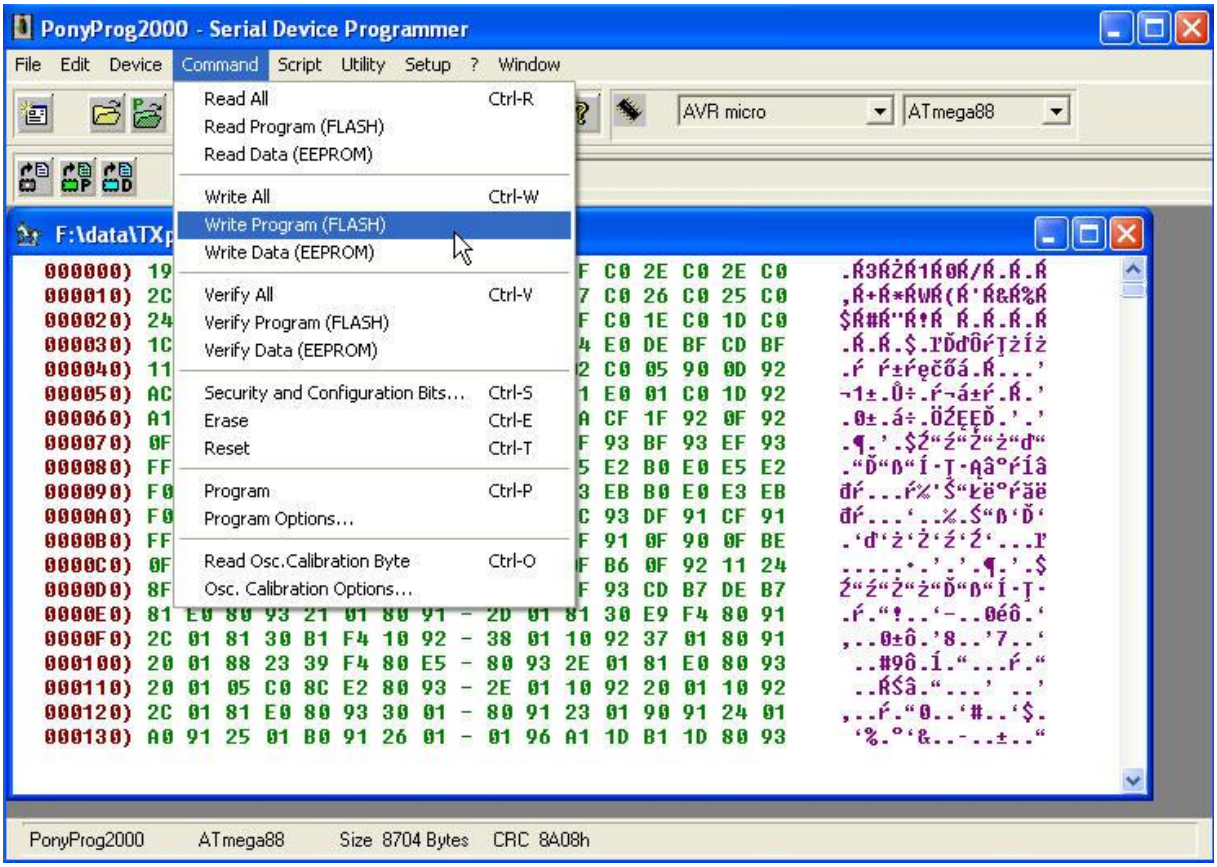
Jeho obsahem se naplní paměť programu *PonyProg2000* a zobrazí se v jeho okně:



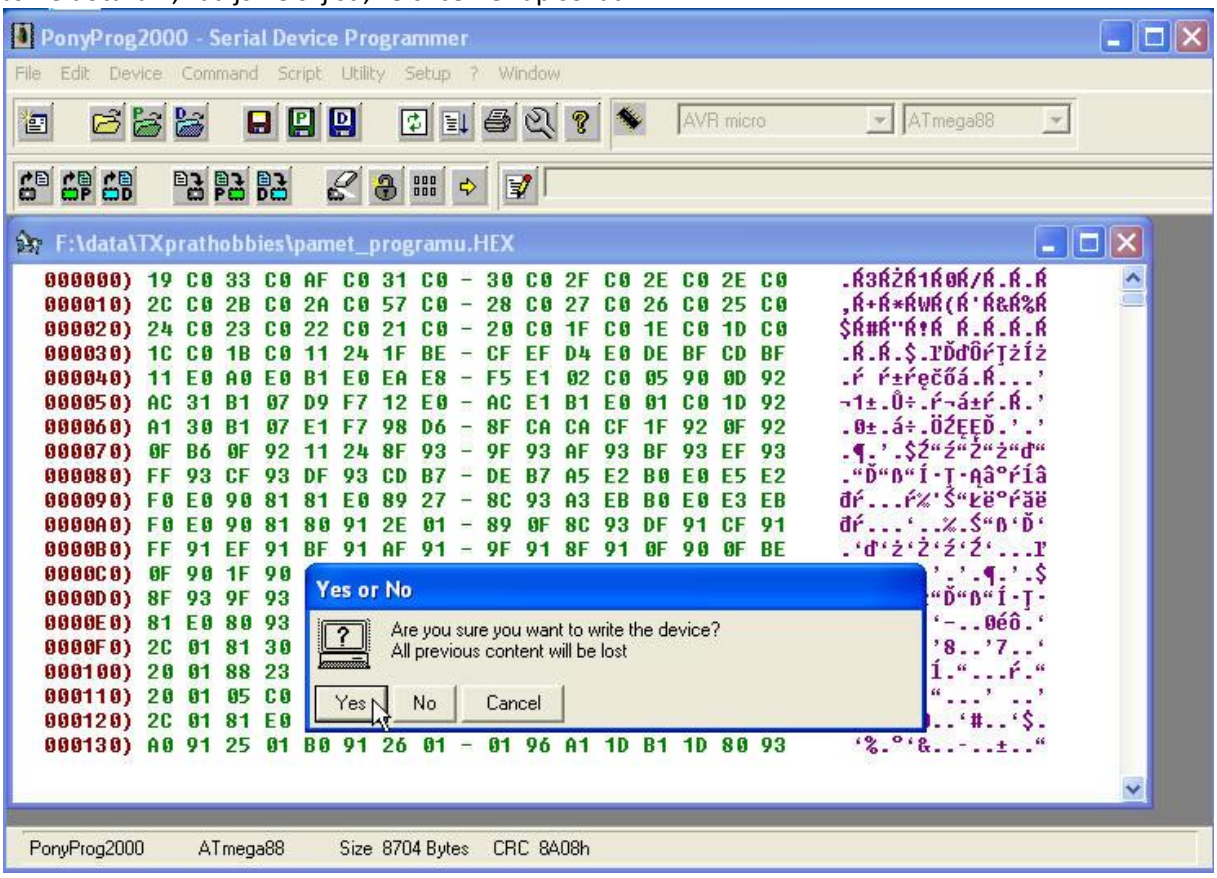
Všimněme si jeho délky a zejména kontrolního CRC:



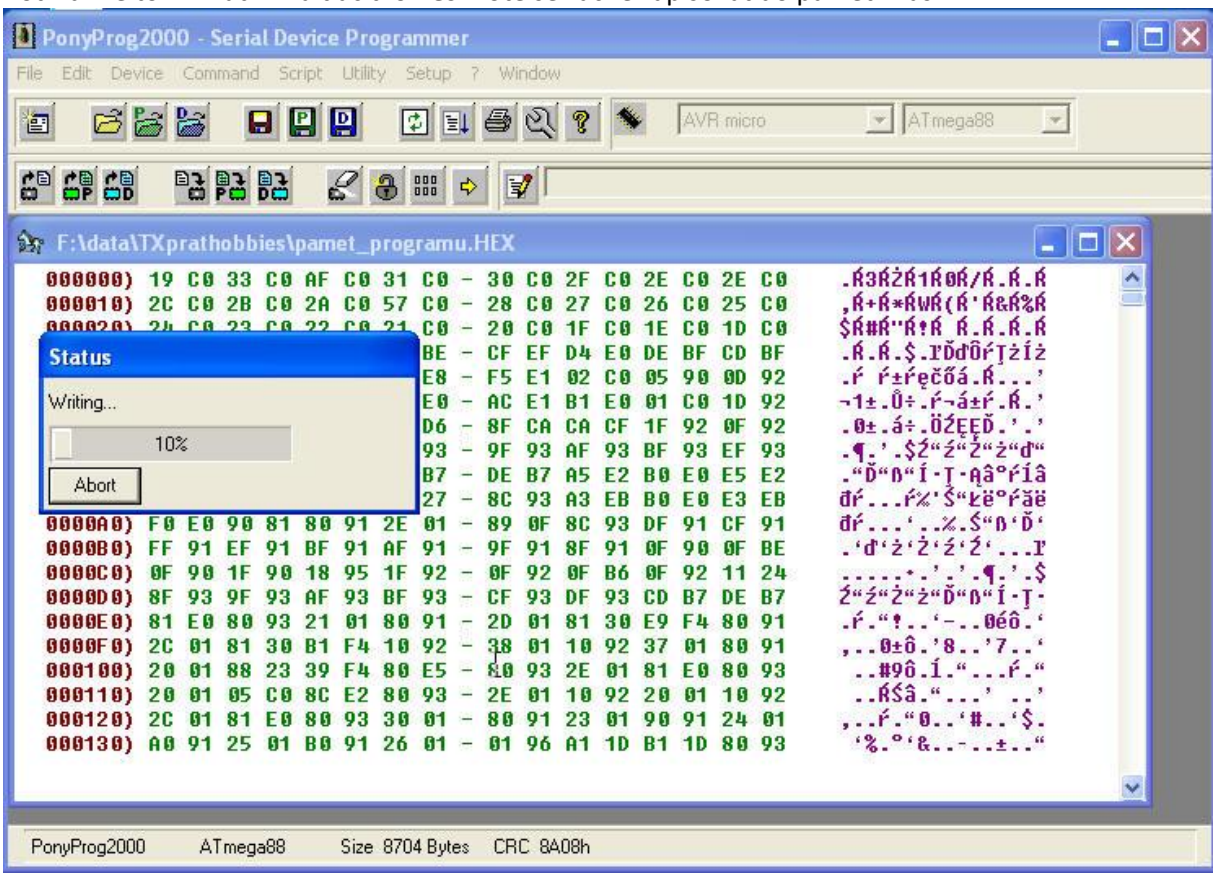
Tímto obsahem naplníme Flash programovaného *ATMega88*. V menu vybereme **Command →Write Program (FLASH)**



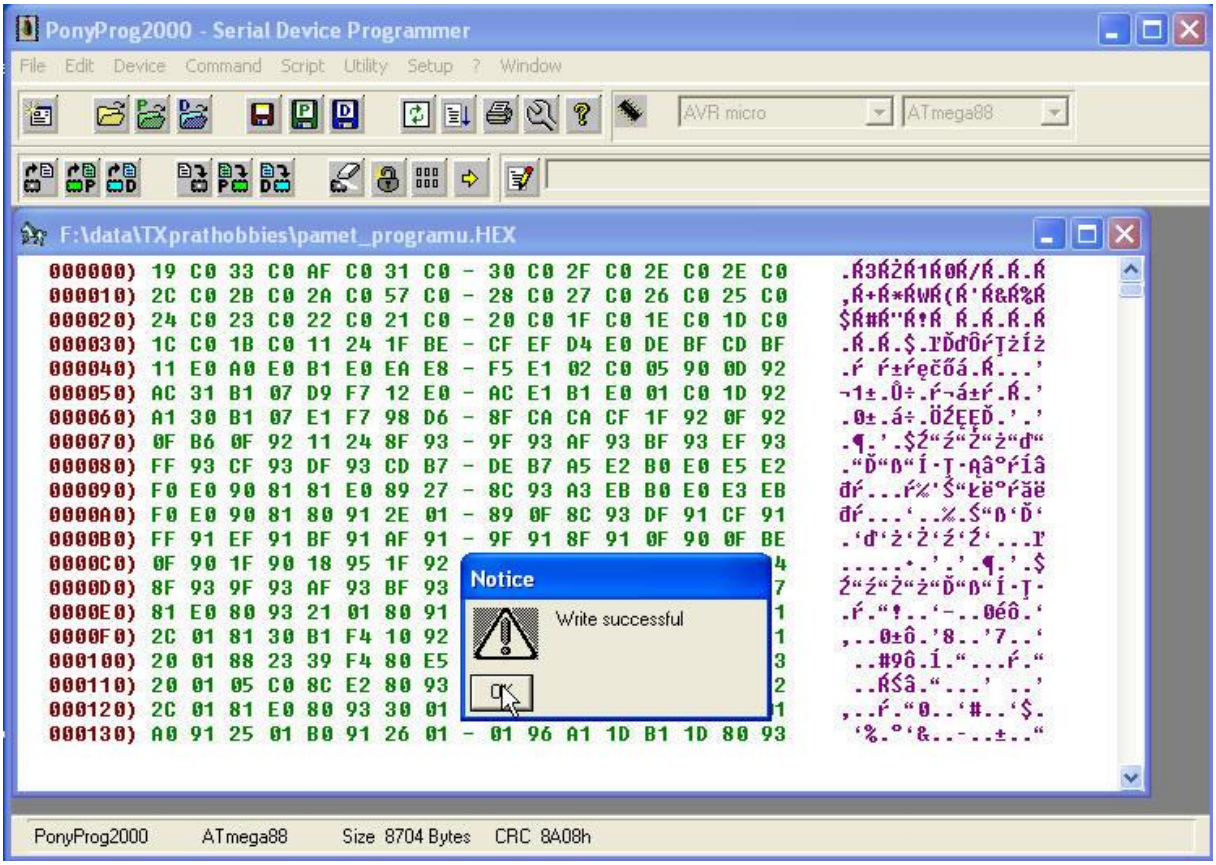
Jsme dotázáni, zda jsme si jisti, že chceme zapisovat



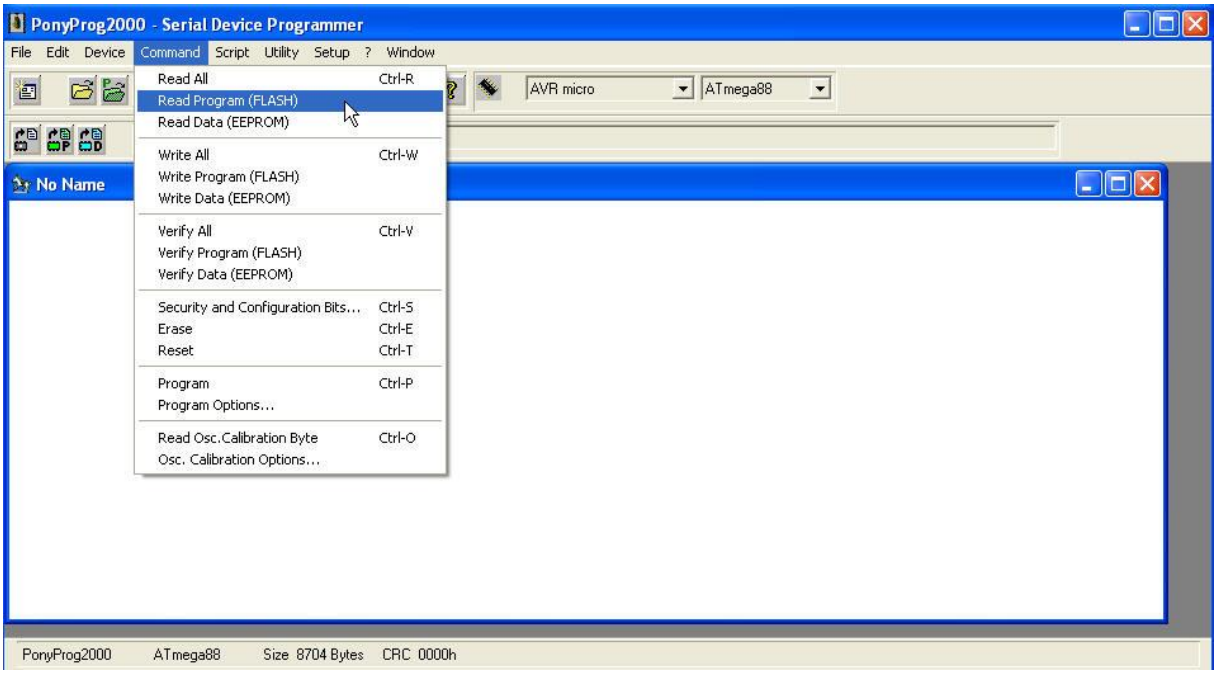
Potvrdíme to kliknutím na tlačítko **Yes.** Poté se začne zapisovat do paměti Flash



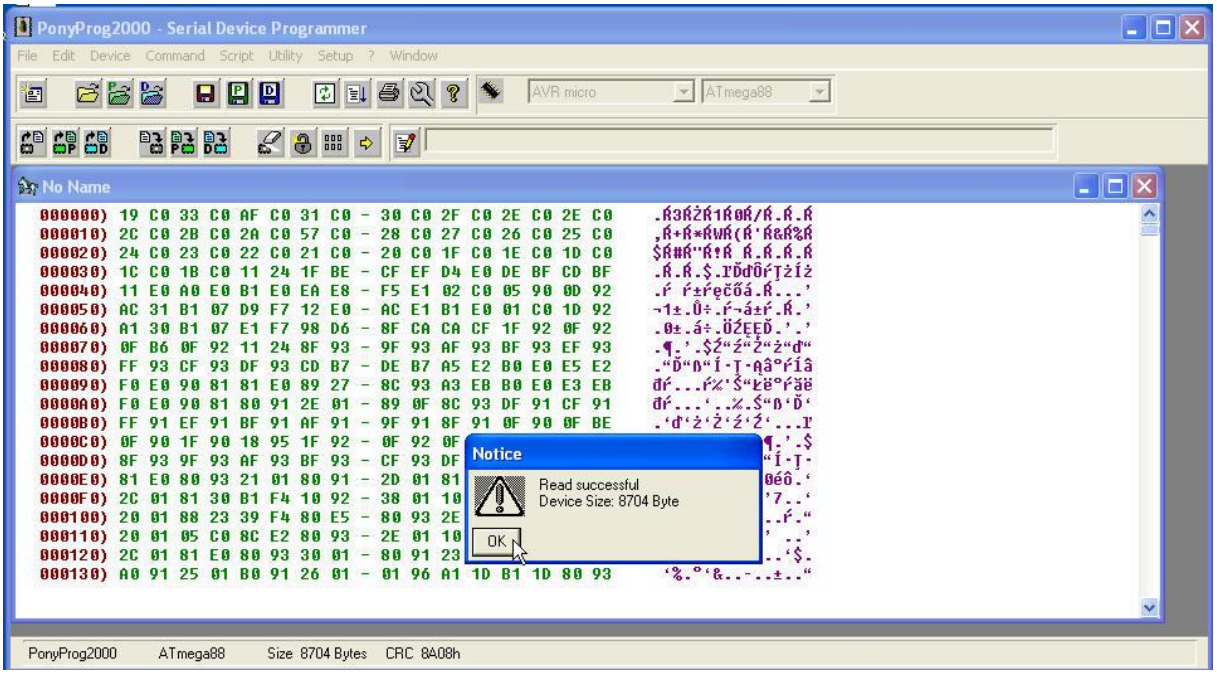
O úspěchu zápisu nás informuje okno **Notice**



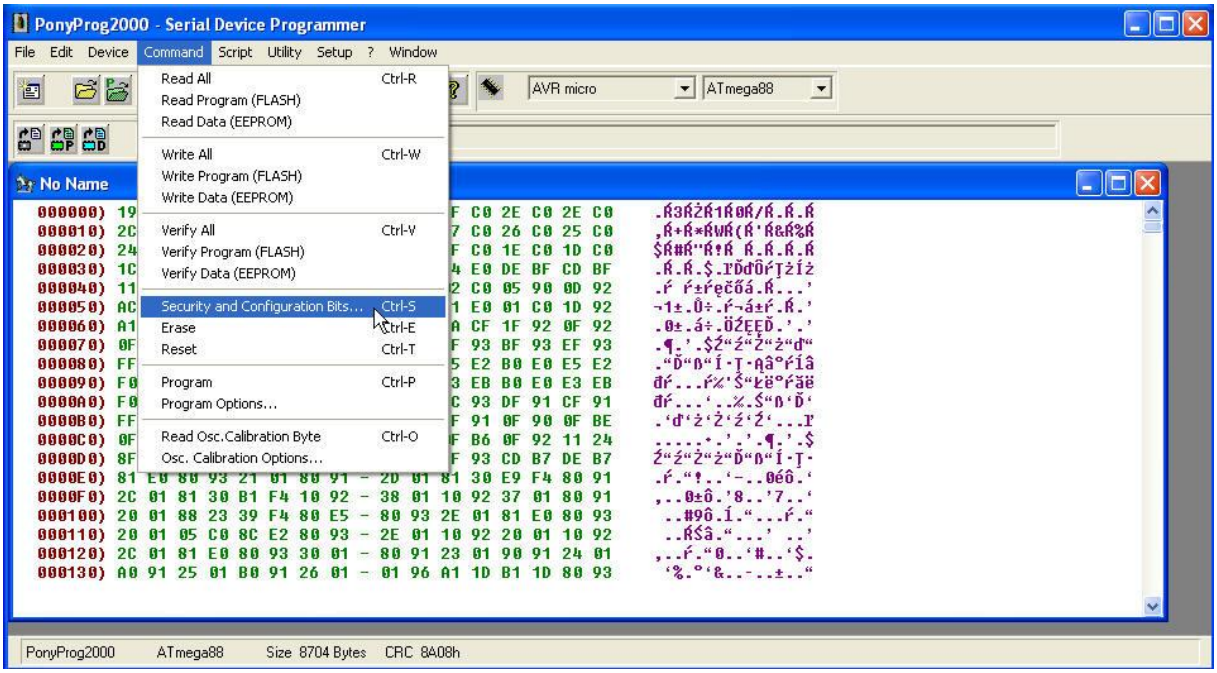
Potvrdíme **OK**. Nyní zkontrolujeme, že obsah flash paměti opravdu obsahuje požadovaný obsah. Proto přečteme obsah této paměti. V menu vybereme **Command → Read Program (FLASH)**



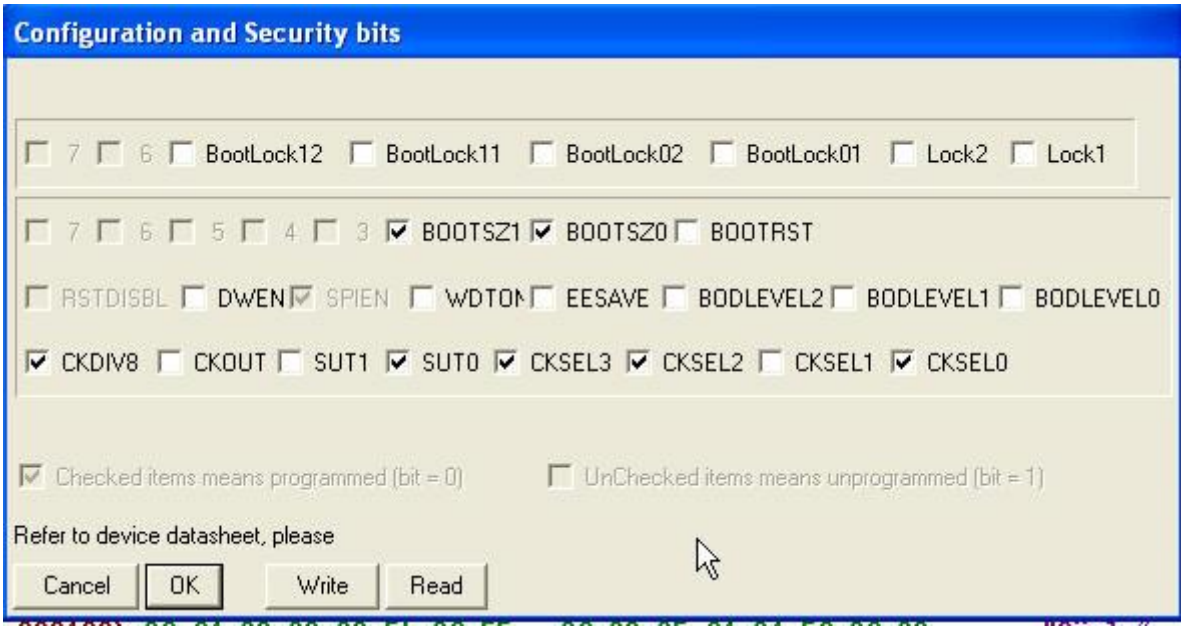
Dostaneme



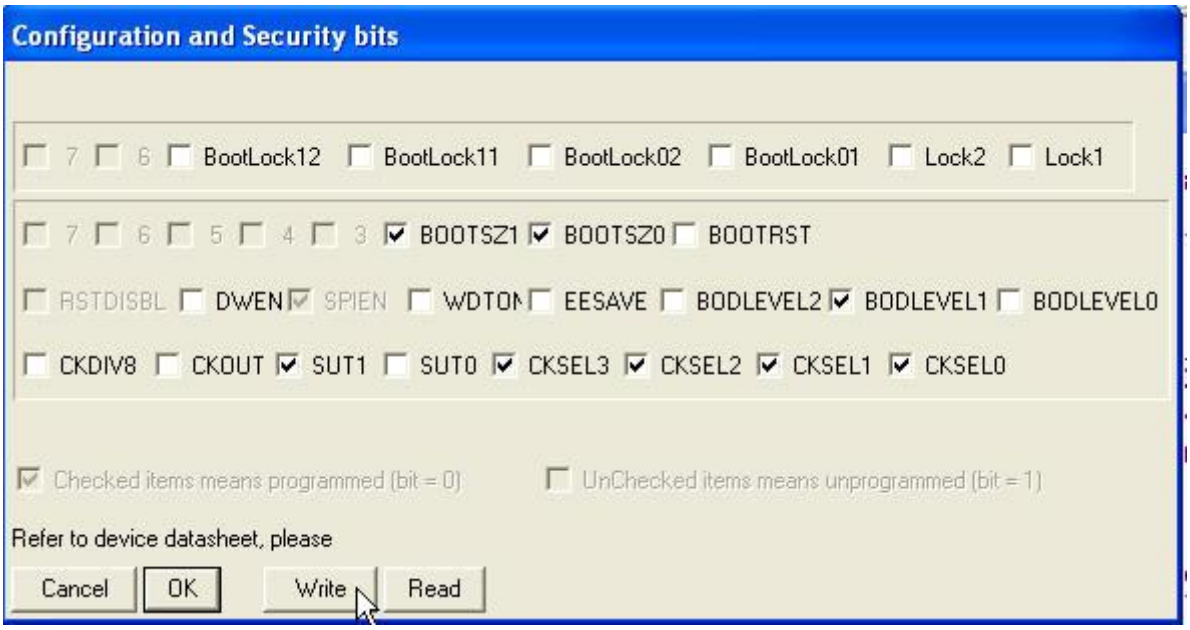
Čtení bylo úspěšné. Potvrdíme **OK**. Vidíme, že máme stejný CRC, jaký měl načtený soubor. Dále v menu vybereme **Command →Security and Configuretions Bits…**



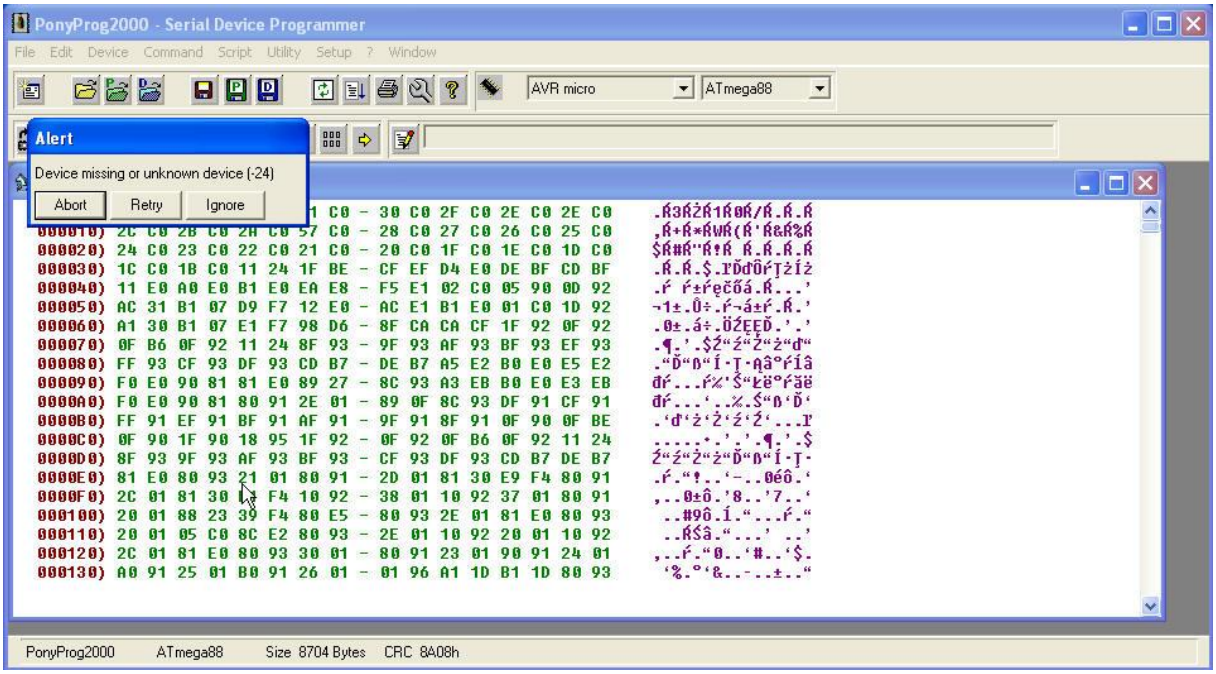
Uvidíme původní nastavení pojistek nového obvodu *ATMega88*



Přenastavíme je tak, aby byly stejné jako u vysílače *PrattHobbies*

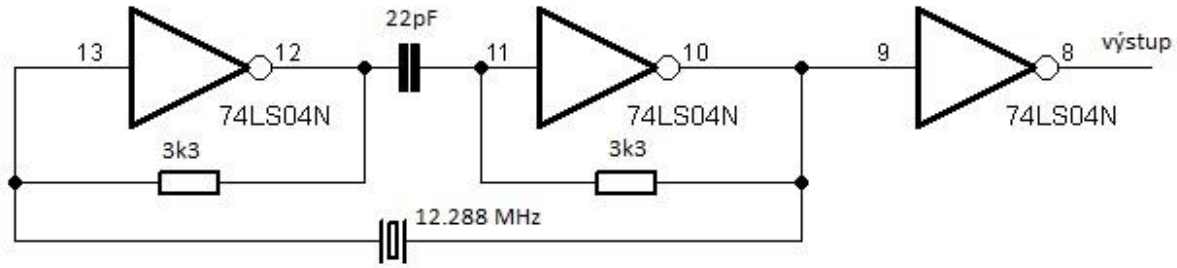


Klikneme na tlačítko **Write**. Objeví se



Jde o chybovou hlášku. Je způsobená tím, že program *PonyProg2000* vždy kontroluje úspěšnost své činnosti, tj pokud něco zapisujeme, snaží se poté přečíst zapsaná data a ty kontroluje s tím, co jsme zapsali. V našem případě se snaží přečíst stav pojistek (fuses). Ty byly u nové *ATMega88* nastaveny tak, že jako zdroj hodinových pulzů používala vnitřní RC generátor. Tím, že jsme nastavili zaškrtnutí všech bitů **CKSEL0** až **CKSEL3** jsme zvolili, že *ATmega88* bude jako zdroj hodin používat vnější generátor. Takže po zapsání tohoto nastavení *fuses* tlačítkem **write** přestala *ATMega88* pracovat, neboť k ní žádný vnější generátor nemáme připojený. Protože nyní nepracuje, objeví se chybová hláška při jakémkoli pokusu o práci s ní. Např. když chceme přečíst obsah programové paměti flash:

Připojíme k *ATMega88* vnější zdroj hodinových pulzů a pomocí programu *PonyProg2000* se přesvědčíme, že nyní již obvod pracuje. Můžeme si přečíst obsah jeho flash paměti i nastavení fuses. Zapojení vnějšího generátoru hodinových pulzů může být např.:

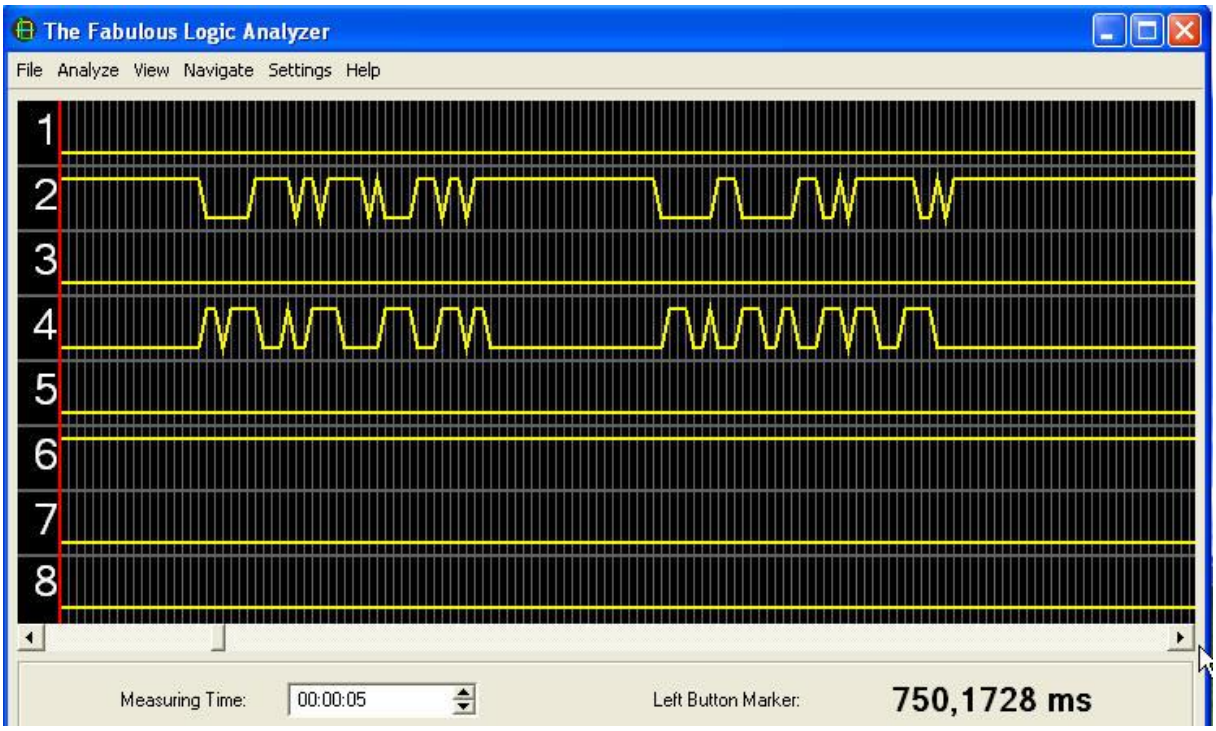


Chybovou hlášku PonyProg2000 o tom, že nelze komunikovat s ATMega88 tak již nedostaneme a nyní budeme provádět analýzu činnosti firmware PrattHobies. Nicméně musím nyní uvést jednu velice důležitou poznámku:

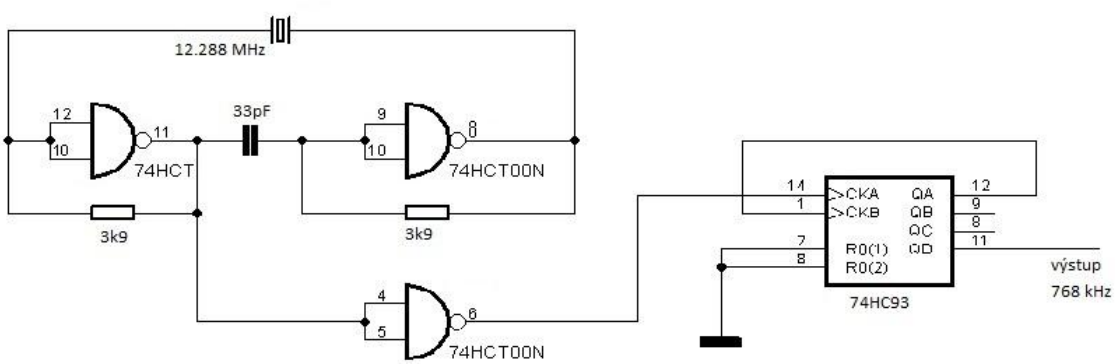
Při jakékoli změně fuses (pojistek) musíme být velice opatrní, být si jisti, že jsme pochopili jejich nastavení dle firemní dokumentace a že víme, co opravdu chceme. Špatné nastavení fuses může způsobit i to, že s obvodem nebudeme již moci komunikovat, opravit svou chybu atd. a že zdánlivě jediná možná akce s tímto obvodem bude jeho ekologická likvidace. Naštěstí je obvykle možná náprava návratem fuses do původního stavu pomocí paralelního programátoru. Principu paralelního programátoru využívá i projekt **FuseDoctor** http://luta.7u.cz/index.php?str=5 a http://diy.elektroda.eu/atmega-fusebit-doctor-hvpp/#update7en

Obrázek fusedoctor:

Obvod *ATMega88* má pak stejný hodinový kmitočet i firmware jako vysílač *PrattHobbies*. Jeho piny PB2, PB3, PB5 a PC6 připojíme ke vstupům logického analyzátoru s 74HC245, a provedeme záznam signálů z těchto výstupů. Dostaneme mj. např.



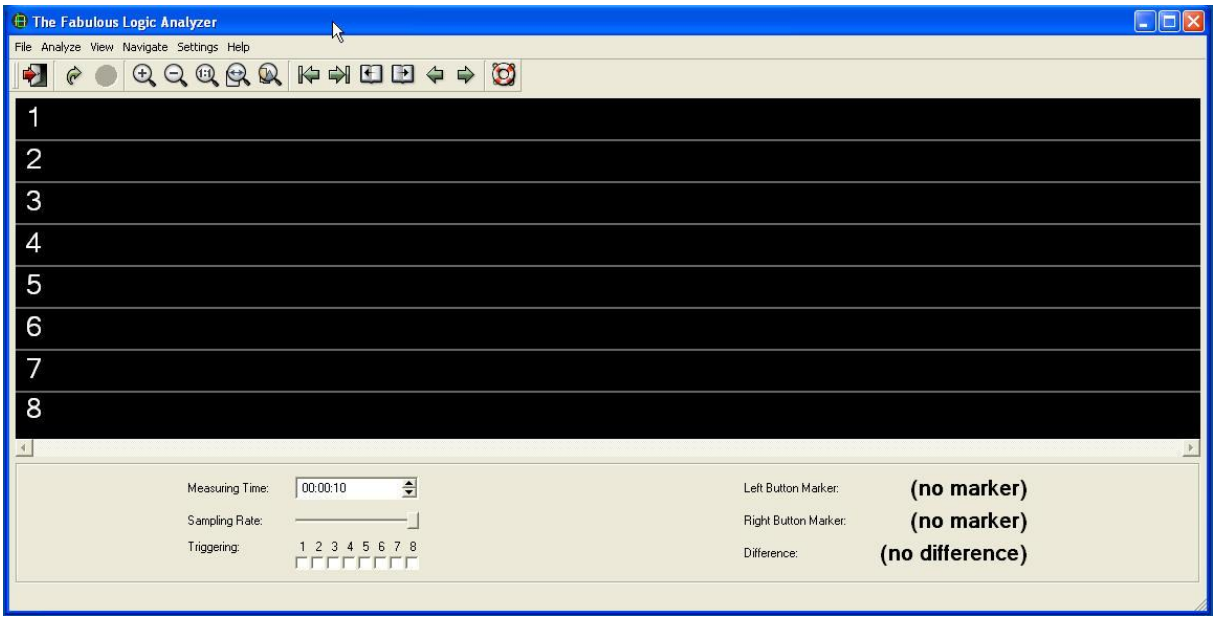
Signál v kanálu 2 jsou **DATA**, v kanálu 4 hodiny **CLK**. To, zda data obsahují **0** či **1** se vyhodnocuje se vzestupnou hranou hodinových impulzů. Ve výše uvedeném obrázku vidíme, že schází několik hodinových pulzů a že hrany mají pozvolný náběh i doběh apod., Je zřejmé, zobrazení těchto signálů je již *za hranicí schopností našeho logického analyzátoru*.
Proto pro účely analýzy použijeme jako zdroj hodinových impulzů obvod



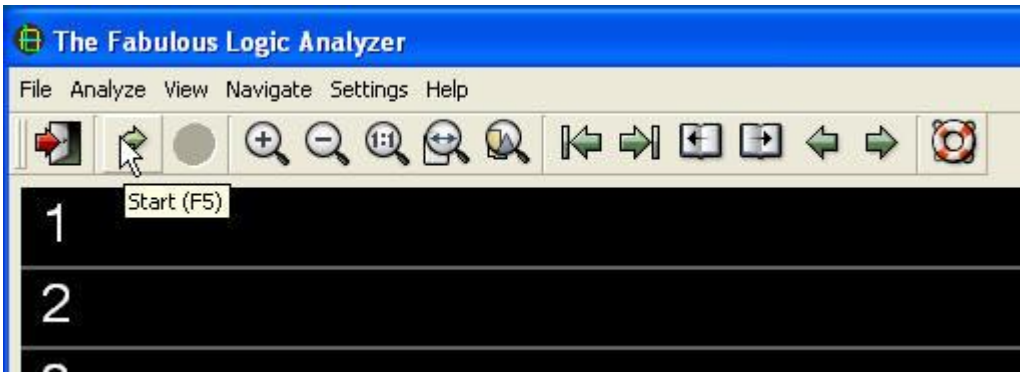
Tj. hodinový kmitočet je 16 x nižší, než 12.288 MHz u vysílače *PrattHobbies* a tedy všechny průběhy budou 16 x pomalejší. Tedy 16 x pomalejší musí být i rychlost sériového dálnopisného signálu předávající do *ATMega88* řídící příkazy (M, F, S …) tj místo 38 400 Bd to bude 2400 Bd.

Následujících několik obrázků bylo získáno při analýze inicializace a hodinovém kmitočtu 16 x menším, než u vysílače *Pratt Hobbies*. Kanál 4 bude zobrazovat **DATA**, kanál 6 hodinové pulzy **CLK** a

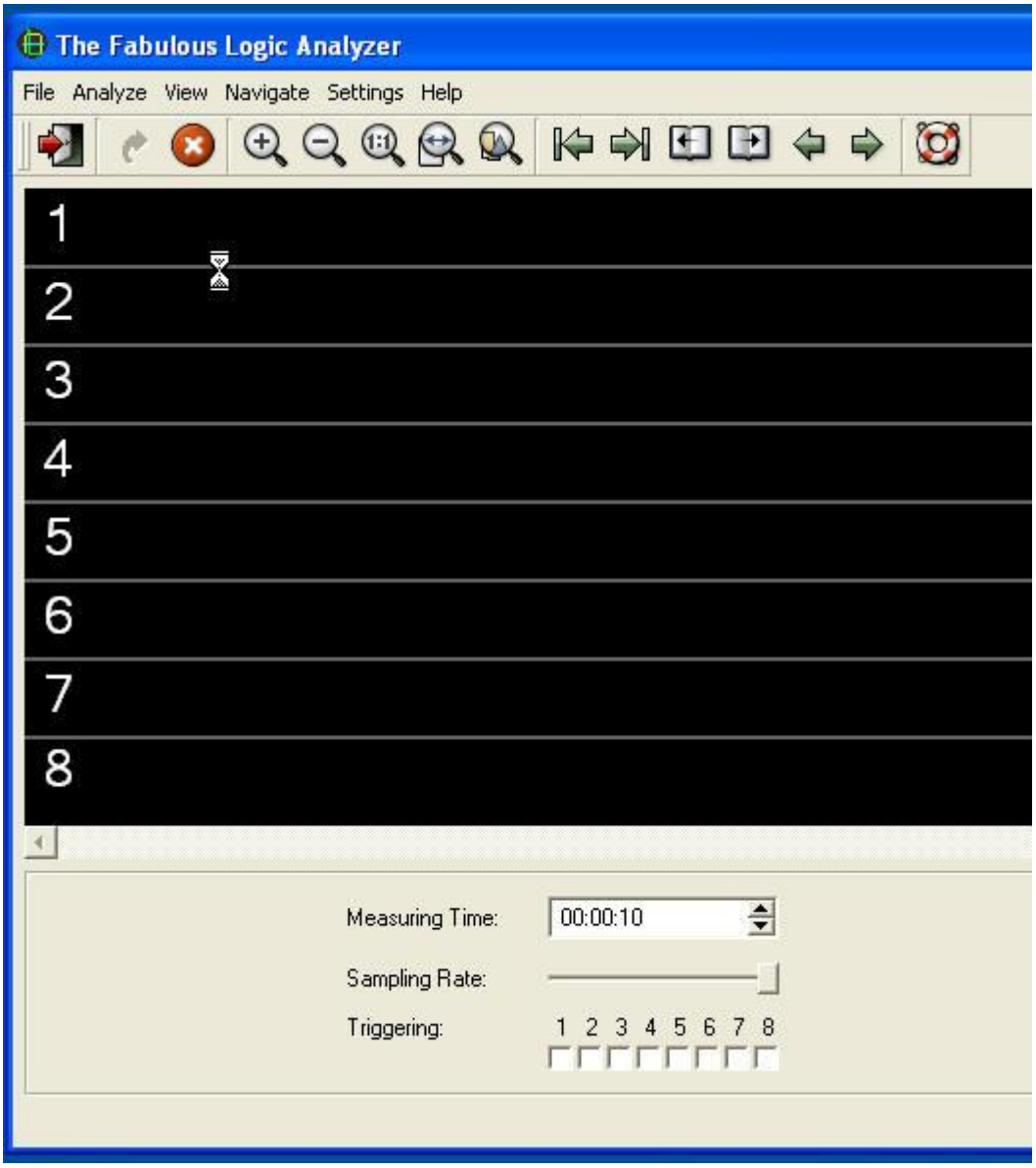
kanál 8 úrovně na vstupu **RESET/** obvodu *ATMega88.* Spustíme program **The Fabulous Logic Analyzer 0.1.2.** Zobrazí se okno tohoto programu:



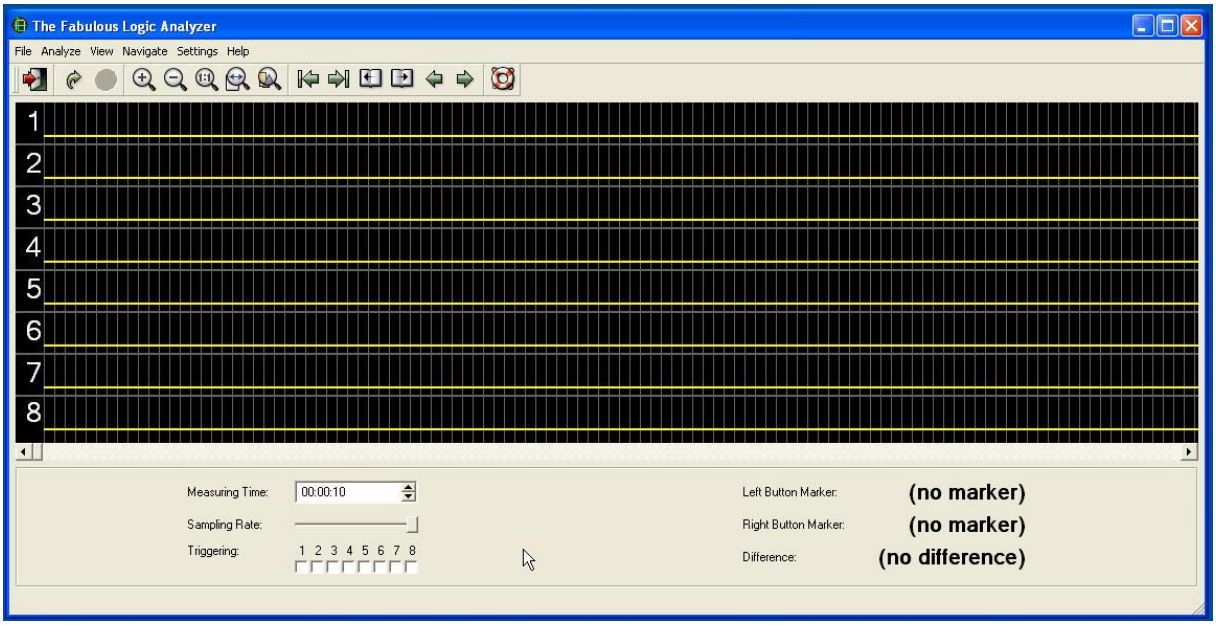
Stiskneme resetovací tlačítko u *ATMega88*, tj. nastavíme úroveň **0** na jeho resetovacím vstupu. Toto tlačítko zatím budeme držet stisknuté a klikneme na ikonku **Start** logického analyzátoru:



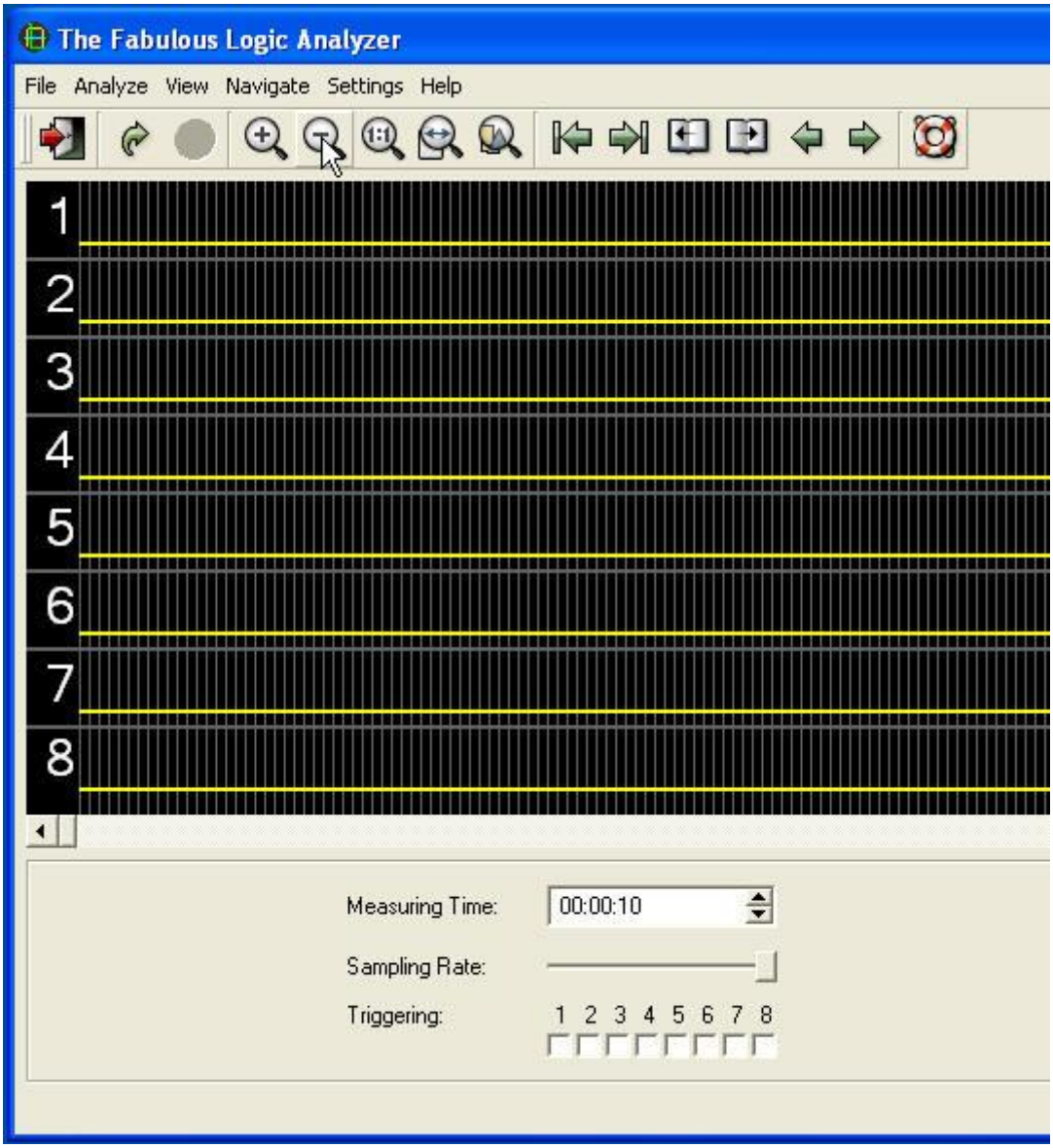
Nyní logický analyzátor snímá úrovně na všech 8 kanálech a zapisuje je do paměti. Přitom jsou jako kurzor zobrazovány přesýpací hodiny.

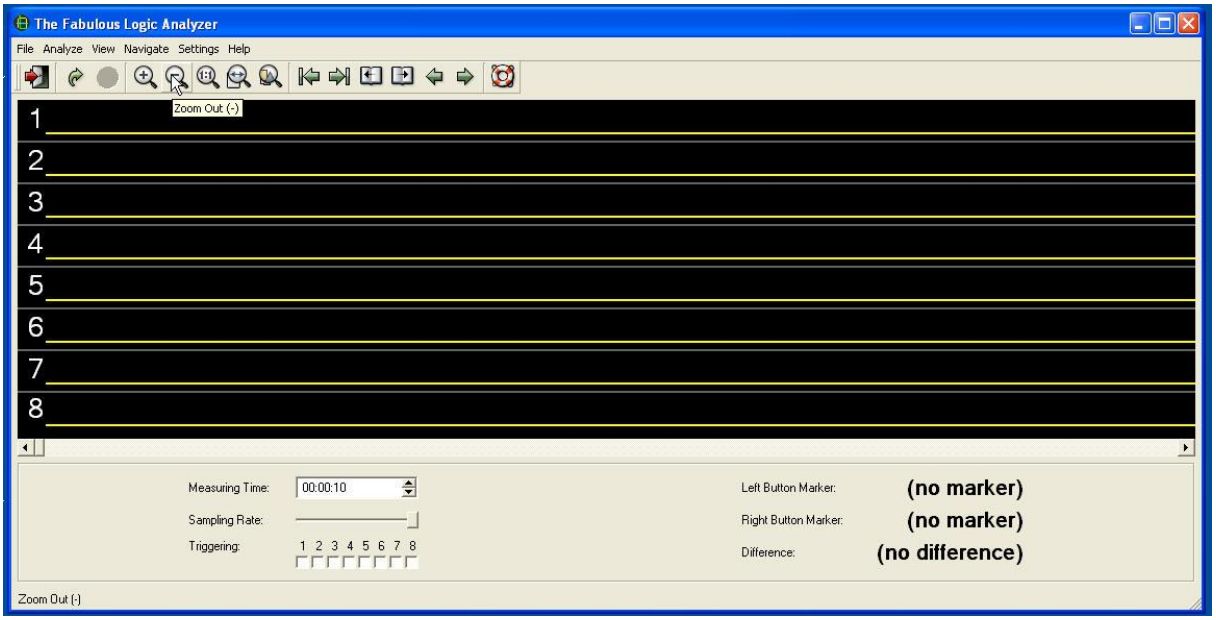


Po skončení záznamu se zobrazí zaznamenané průběhy:

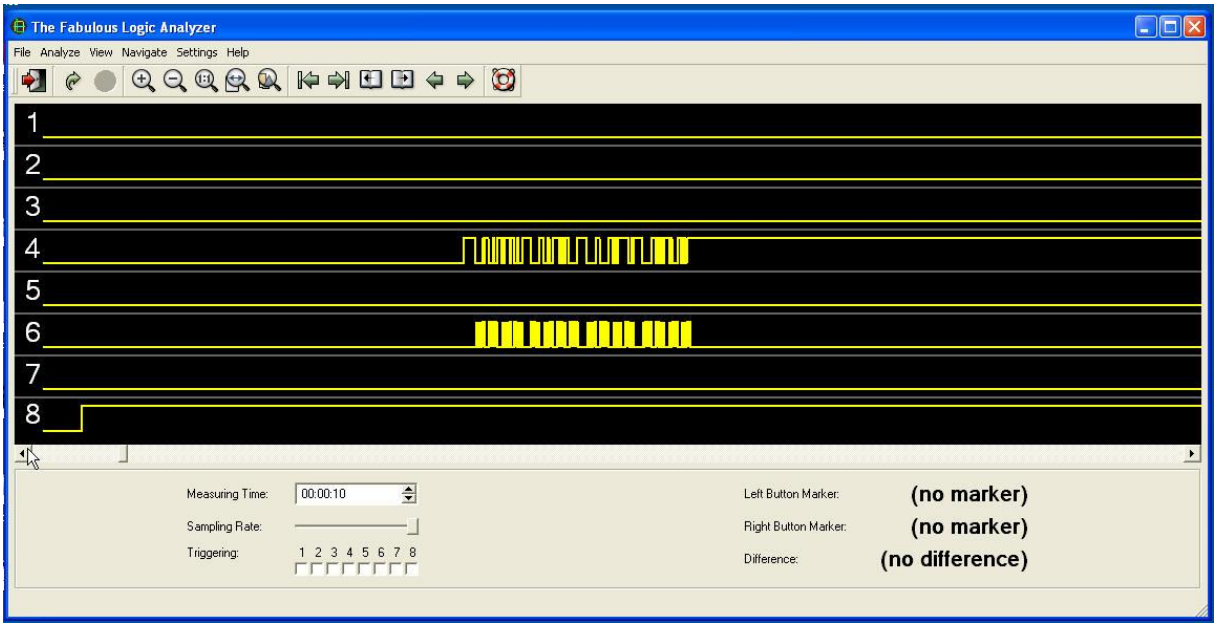


Několika kliknutími na ikonku – tj **Zoom Out** postupně zmenšíme měřítko zobrazovaných průběhů:

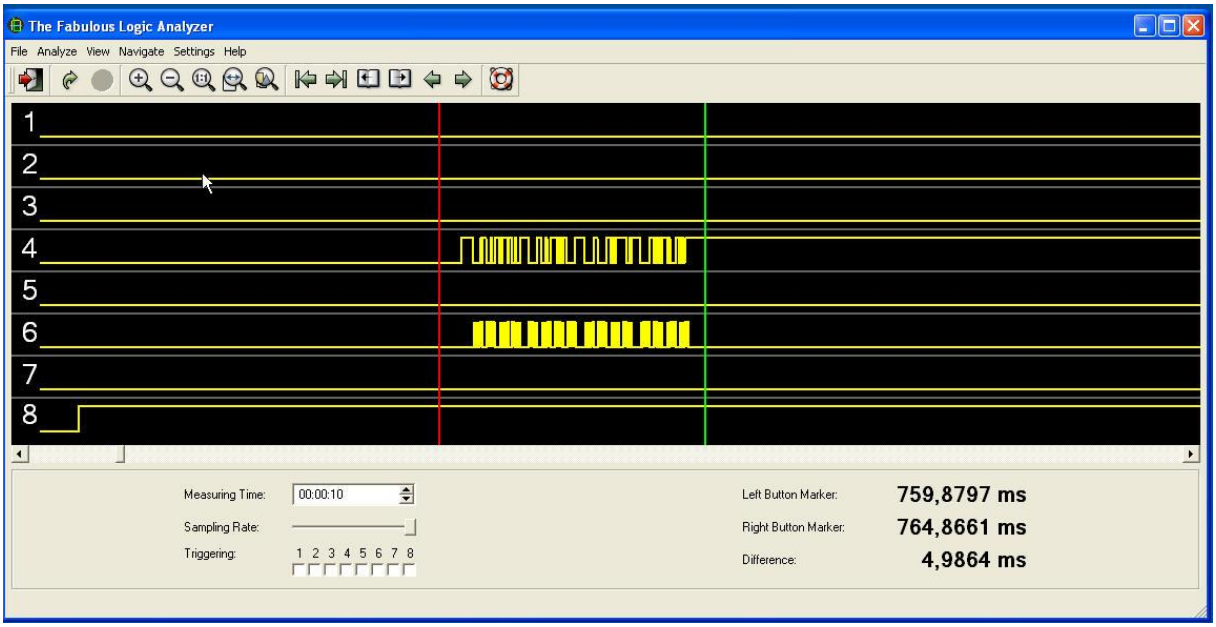

The Fabulous Logic Analyzer
File Analyze View Navigate Settings Help
1
2
3
4
5
6
7
8
Measuring Time:
00:00:10
Sampling Rate:
Triggering:
1 2 3 4 5 6 7 8



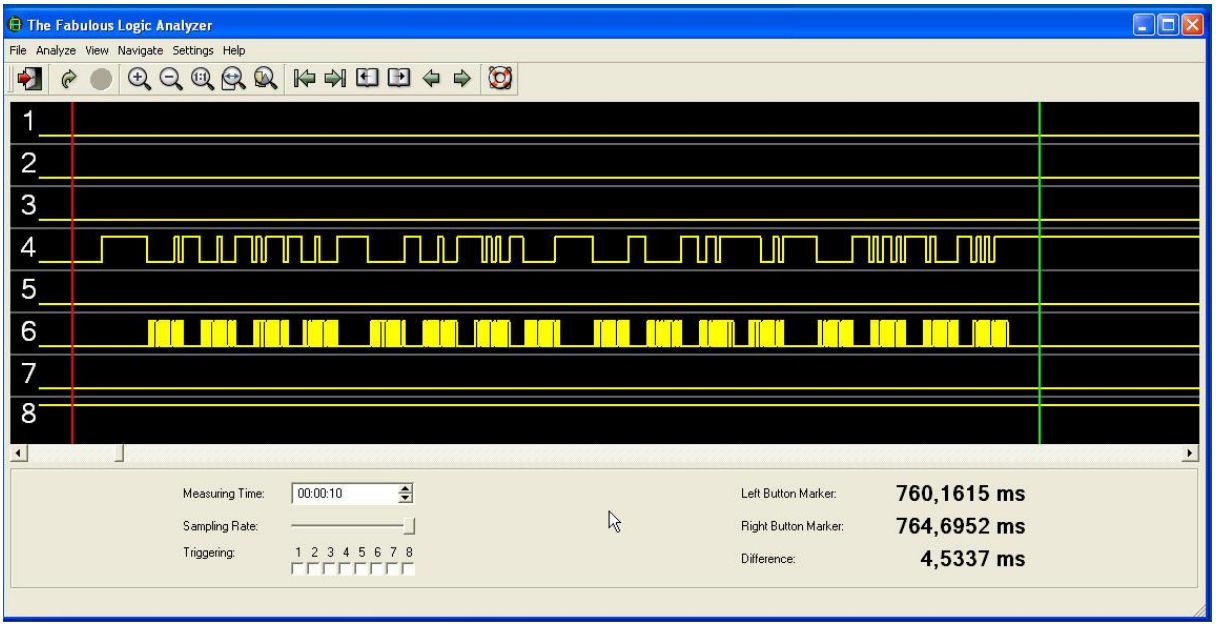
Pomocí posuvníku pod zobrazovanými průběhy v kanálech 1 až 8 najdeme zobrazené signály inicializace obvodu *ADF7012*. Při jejich hledání využijeme toho, že kanál 8 zobrazuje logický signál na pinu **RESET/** obvodu *ATMega88*. Na začátku má úroveň 0, neboť jsem držel stisknuté resetovací tlačítko. Poté, co jsem tlačítko uvolnil je úroveň zobrazovaná v kanálu 8 na úrovni 1. Po uvolnění resetovacího tlačítka se spustí program (firmware) v *ATMeg88a,* tj. nejprve se provede inicializace:



Před inicializační signály umístíme kurzor a klikneme levým tlačítkem, obdobně klikneme za průběhy pravým. V místech kliknutí se zobrazí červená a zelená čára:



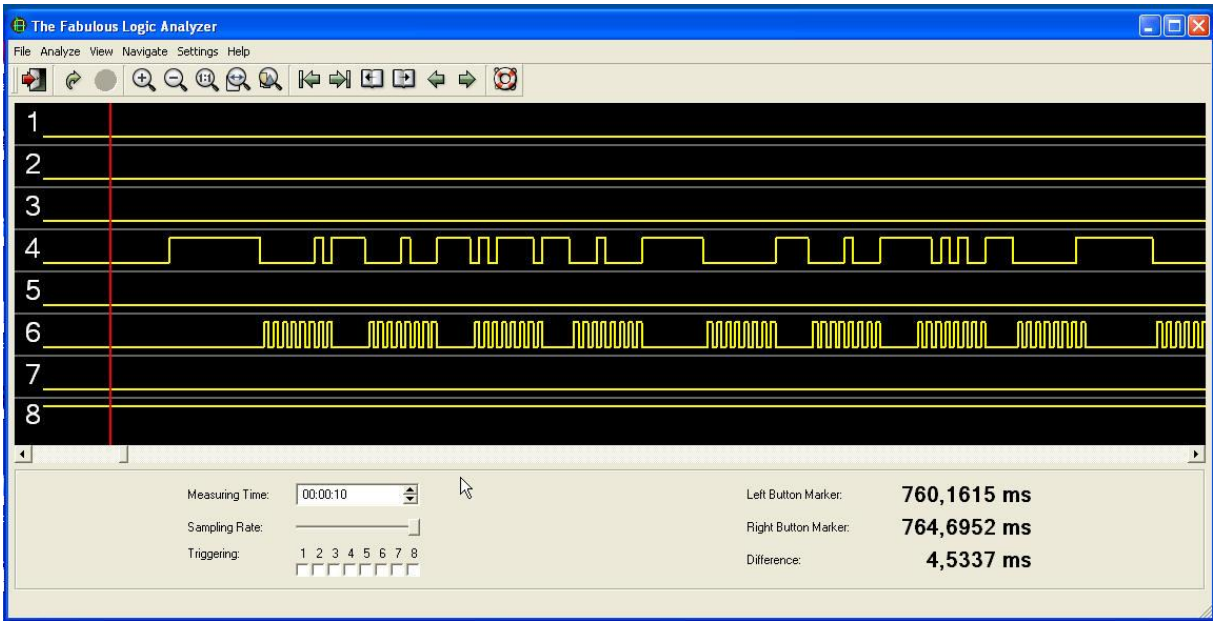
Pomocí **Zoom**ovacího tlačítka **+** postupně upravujeme velikost zobrazovaných průběhů:



Nyní je nejvhodnější čas, vysvětlit si, jak provádí firmware *PrattHobbies* vysílače inicializaci *ADF7012*. Současně připomínám, že jsem použil zdroj hodin pro *ATMega88* o kmitočtu 12.288MHz / 16 takže zobrazený časový interval mezi červenou a zelenou čárou bude u tohoto vysílače 4,5337 ms / 16.
Pro další studium výše uvedených průběhů si následně zvětšíme měřítko, budeme však vidět vždy jen část inicializační sekvence a proto si stručně popíšeme již nyní, kdy ji vidíme celou:
Kanál 6 zobrazuje hodinové pulzy generované **8bitovým** počítačem *ATMega88* pro vstup **Clk** (pin11) obvodu *ADF7012*. Kanál 4 pak zobrazuje inicializační data generované *ATMega88* pro vstup **DATA** (pin 12) obvodu *ADF7012*. Tato řídící **DATA** se předávají sériově coby 32bitová slova. Hodnota každého z těchto 32 bitů je obvodem *ADF7012* vyhodnocována vždy při náběžné hraně hodinových pulzů zobrazovaných kanálem 6. Firmware je napsán tak, že v inicializační části se do *ADF7012* přenesou celkem čtyři 32bitová slova a to nejprve 32bitů, které se v *ADF7012* zapíší do **Registru 0**, poté data pro **Registr 1** následovaná daty pro **Registr 2** a nakonec pro **Registr 3**. Na výše uvedeném

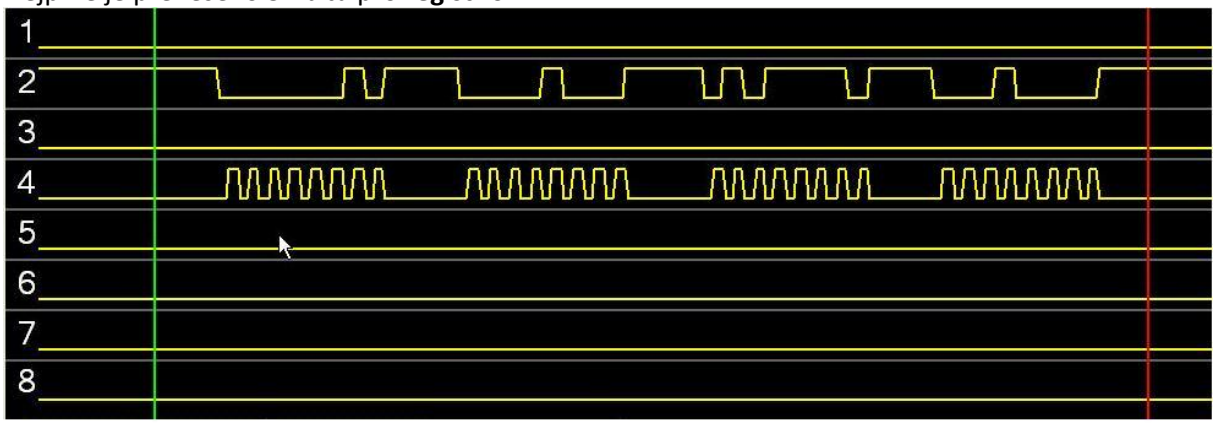
obrázku vidíme, v **kanálu 6** čtyři jakési posloupnosti čtveřic hodinových pulzů. Pokud zvětšíme měřítko zobrazení, vidíme, že tyto shluky jsou složeny vždy z 8 hodinových pulzů. Neboli každá čtveřice shluků pulzů obsahuje těchto pulzů 32 a slouží jako doprovod přenosu 32 bitů dat do některého z registrů. Nyní již můžeme rozumět výše uvedenému obrázku.

Poznamenávám, že z dokumentace *ADF7012* lze určit význam všech datových bitů pro data všech čtyř registrů. Z dokumentace však **nevyplývá**, že 32 hodinových pulzů by mělo mít stejnou délku a mít mezi sebou stejné vzdálenosti. Je jen definováno, že hodnota přenášených datových bitů je definována v okamžicích vzestupných hran hodinových pulzů. To že v našem případě je za každou osmicí hodinových pulzů prodleva je způsobeno tím, že *ATMega88* je osmibitový procesor a tedy zpracovává data po 8 bitech a mezi zpracováním dalších 8 bitů zpracovává procesor ještě několik dalších instrukcí, což spotřebovává čas a projevuje se těmi časovými prodlevami po každých 8 hodinových pulzech generovaných pro *ADF7012*.



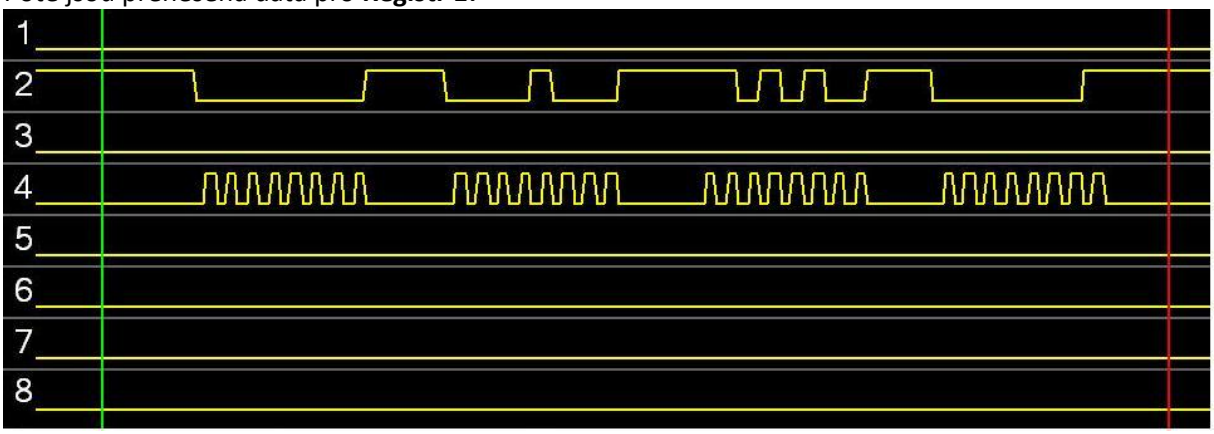
Vidíme, že zobrazené průběhy jsou vhodné k další analýze. Znovu zdůrazňuji, že přitom musíme vzít v úvahu, že hodnoty časů uváděné logickým analyzátorem jsou při použití hodinového kmitočtu 12.288 MHz 16 x menší, tj. např. diference 4,5337 ms by byla 4,5337/16 ms. Konečně jsme se tedy dostali ke studiu inicializačních dat.

Nejprve je přeneseno 32 bitů pro **registr 0**:



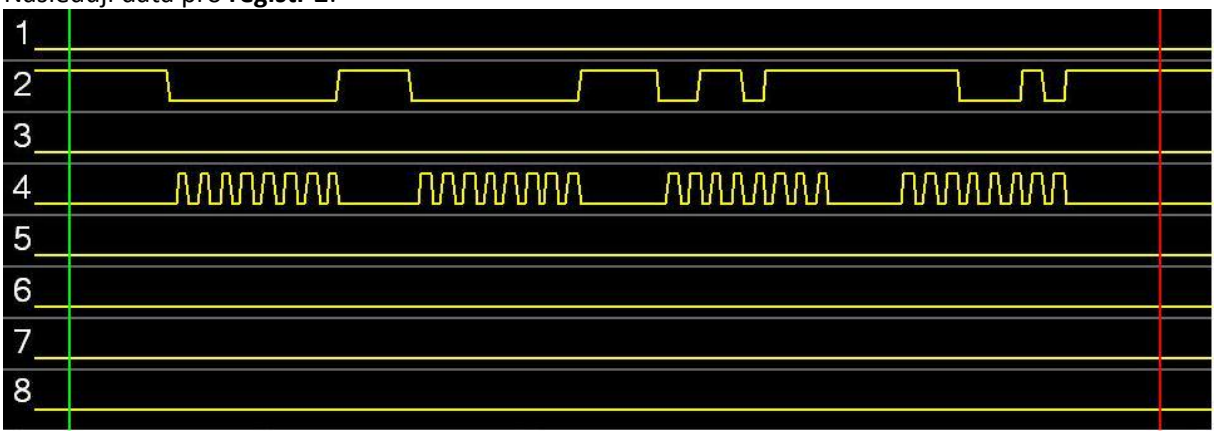
Z tohoto obrázku, popř. z obrázku s ještě větším měřítkem již můžeme zjistit data přenášená do Registru 0. Jsou **00000010 00001000 01011110 00010000** tj. 0x02085E10h Poslední dva bity jsou 00 tj data se skutečně týkají registru 0.

Poté jsou přenesena data pro **Registr 1:**



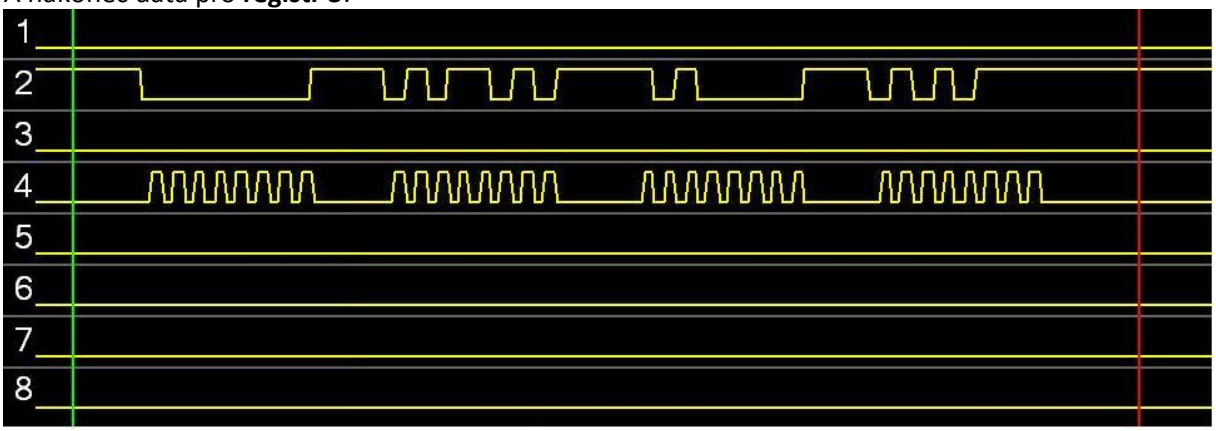
Nyní jsou přenášená data **00000000 00001000 11010100 00000001** tj. 0x0008D401 Poslední dvojice bitů je 01 tj. skutečně se data posílají pro registr 1.

Následují data pro **registr 2**:



Přenášená data jsou **00000000 00000000 00110111 11100010** tj. 0x000037E2h Poslední dva bity jsou 10 tedy dekadicky 2 (Registr 2)

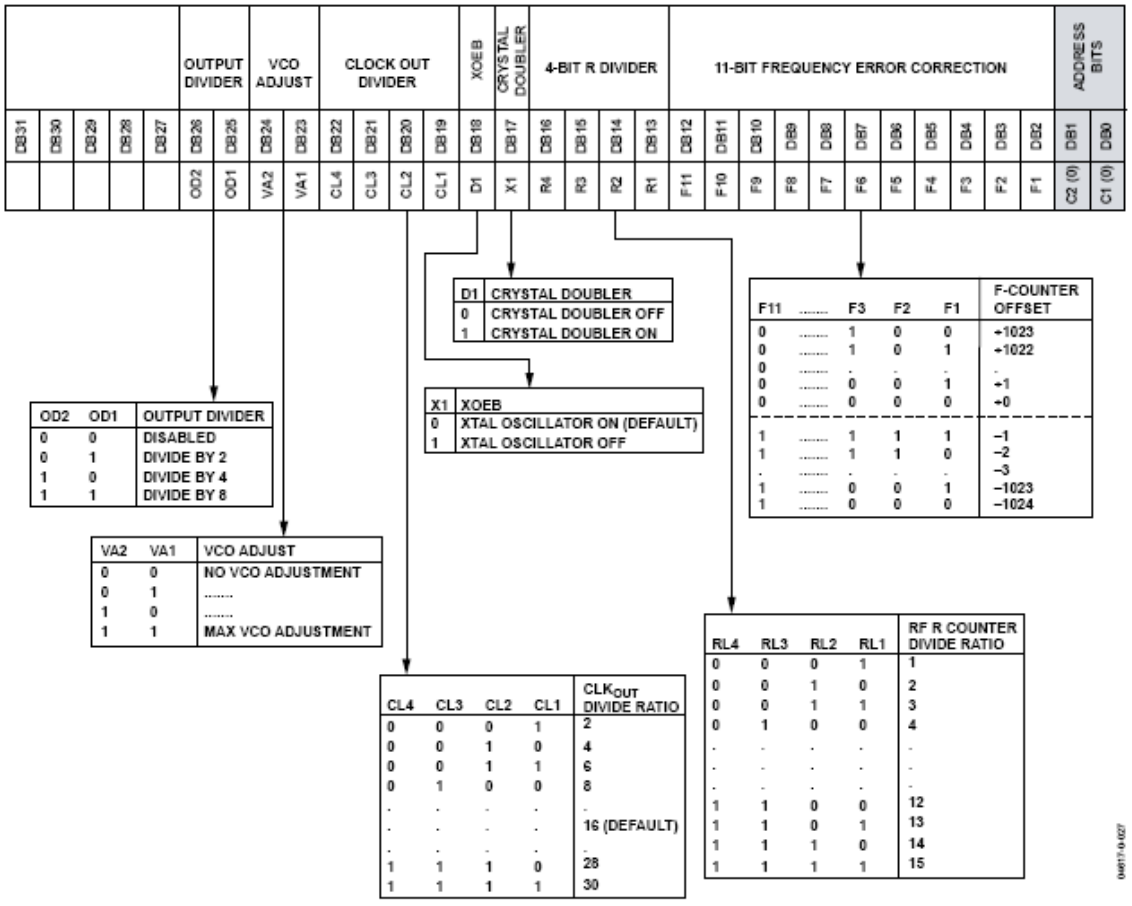
A nakonec data pro **registr 3**:



Přenášený obsah je **00000000 01011010 10100000 01010111** tj. 0x005AA057 poslední dva bity jsou 11 tj. dekadicky 3 (Registr 3)

Význam přenášených dat pro **registr 0** popisuje firemní dokumentace takto:

REGISTER 0: R REGISTER



Naše data pro **R registr** jsou **00000010 00001000 01011110 00010000** tj. 0x02085E10h První jednička počínajíc od leva tj od DB31 je v DB25. Tím je nastavena výstupní dělička na :2 (Tj. protože kmitočet VCO je cca 900MHz, bude výstupní kmitočet cca 450MHz - neboli námi požadovaných 433 až 434MHz). Další jednička je na bitu DB19. Protože je předcházena třemi nulami je CL4 CL3 CL2 CL1 rovné 0001 tj CLK_{out} DIVIDE RATIO je 2 neboli CLK_{out} má kmitočet krystalu 24.576 : 2 tj. 12.288 MHz. Dalším porovnáním zjistíme, že RL4 RL3 RL2 RL1 je 0010 neboli R Counter Divide Ratio je 2. Dále frequency error correction F11 F10 F9 … F1 je 11110000100 a adresové bity DB1 DB0 jsou 00.

Význam dat v **registru 1**:

REGISTER 1: N-COUNTER LATCH

	PRESCALER	8-BIT INTEGER-N								12-BIT FRACTIONAL-N										ADDRESS BITS	
DB23	DB22	DB21	DB20	DB19	DB18	DB17	DB16	DB15	DB14	DB13	DB12	DB11	DB10	DB9	DB8	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	P1	N8	N7	N6	N5	N4	N3	N2	N1	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	C2 (0)	C1 (1)

M12	M11	M10	.....	M3	M2	M1	MODULUS DIVIDE RATIO
0	0	0	.....	0	0	0	0
0	0	0	.....	0	0	1	1
0	0	0	.....	0	1	0	2
.	.	.	.....	.	.	.	.
.	.	.	.....	.	.	.	.
.	.	.	.....	.	.	.	.
1	1	1	.....	1	0	0	4092
1	1	1	.....	1	0	1	4093
1	1	1	.....	1	1	0	4094
1	1	1	.....	1	1	1	4095

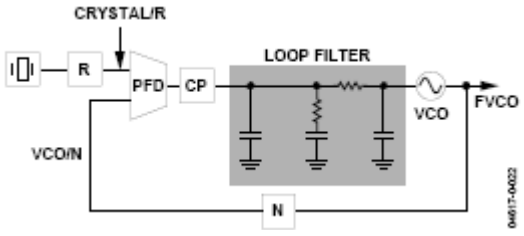
N8	N7	N6	N5	N4	N3	N2	N1	N-COUNTER DIVIDE RATIO
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	1	0	2
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
1	1	1	1	1	1	1	0	254
1	1	1	1	1	1	1	1	255

P1	PRESCALER
0	4/5
1	8/9

THE N-VALUE CHOSEN IS A MINIMUM OF $P^2 + 3P + 3$. FOR PRESCALER 8/9 THIS MEANS A MINIMUM N-DIVIDE OF 91. FOR PRESCALER 4/5 THIS MEANS A MINIMUM N-DIVIDE OF 31.

04617-0-023

Naše data pro **N-Counter Latch** jsou přenášená data 00000000 00001000 11010100 00000001. Porovnáme-li je s výše uvedeným popisem z firemní dokumentace vidíme, že znamenají že N_{int} = 0010 0011 = 23hex = 35 (dekadicky) N_{fract} = 0101 0000 0000 = 1280 (dekadicky) prescaler bit = 0.
Jaký je význam N_{int} a N_{fract} si nyní vysvětlíme. Obsah registru 1 tj. N-Counter Latch slouží k určení kmitočtu výstupního signálu vysílače *ADF7012*. Ten je generován obvodem fázové smyčky PLL:

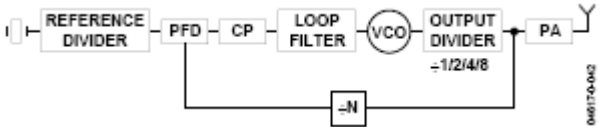


Princip PLL je následující: v fázově kmitočtovém komparátoru PFD je porovnáván kmitočet referenčního signálu s kmitočtem F_{VCO} napěťově řízeného oscilátoru VCO vyděleným N, tj F_{VCO}/N.
Jsou-li tyto kmitočty stejné, je chybové napětí na výstupu PFD nulová. Liší-li se tyto kmitočty, je na výstupu PFD chybové napětí úměrné chybě, tj. rozdílu kmitočtů. Z principu PFD však plyne, že chybové napětí obsahuje řadu složek, přičemž k doladění VCO na správný kmitočet (ustálená smyčka PLL) potřebujeme jen jeho stejnosměrnou složku. Proto jsou ostatní složky odfiltrovány **Loop Filtrem**. Na vlastnostech tohoto filtru závisí i doba ustálení regulační smyčky PLL.
my remarks: *CanSat Book for Students* – part.1 2011

Předpokládejme např. že referenční kmitočet je např. 1 MHz a dělící poměr **N** je 500. Pak v ustáleném stavu je výstupní kmitočet 500 MHz. Dělící poměr **N** však můžeme měnit a tím měnit (ladit) i výstupní kmitočet. Změníme-li N na 499 bude výstupní kmitočet 499 MHz atd. Tj. bude-li dělící poměr N celočíselný, bude (po ustálení PLL) výstupní kmitočet **N** násobkem referenčního kmitočtu. Obvod *ADF7012* je však navržen tak, že **N** nemusí být jen celé číslo. Skládá se z celočíselné části $\mathbf{N_{INT}}$ a zlomkové části $\mathbf{N_{FRAC}}$. Ta je 12 bitová, celočíselná část je 8 bitová. Pro výsledný výstupní kmitočet pak platí

$$F_{OUT} = F_{PFD} \times \left(N_{INT} + \frac{N_{FRAC}}{2^{12}} \right)$$

A podrobnější blokové schema vysílače je



Protože kmitočet krystalového oscilátoru je 24.576 MHz a reference divider R je roven 2, je F_{PFD}=12,228MHz. Dosadíme-li do výše uvedeného vzorce N_{INT} = 35 a N_{FRAC} = 1280, dostaneme

$f_{OUT} = f_{FPD} (N_{int} + N_{FRACT} / 2^{12}) =$ 12,288 . (35 + 1280/4096) = 12,228 . 35,3125 = 433,92

Neboli při inicializaci se nastavuje výstupní (defaultní) kmitočet na 433.92 MHz. Tento kmitočet byl zřejmě zvolen, protože jde o kmitočet ISM.

Pomocí tohoto vzorce snadno dopočítáme tabulku kmitočtů pro námi předpokládané pásmo:

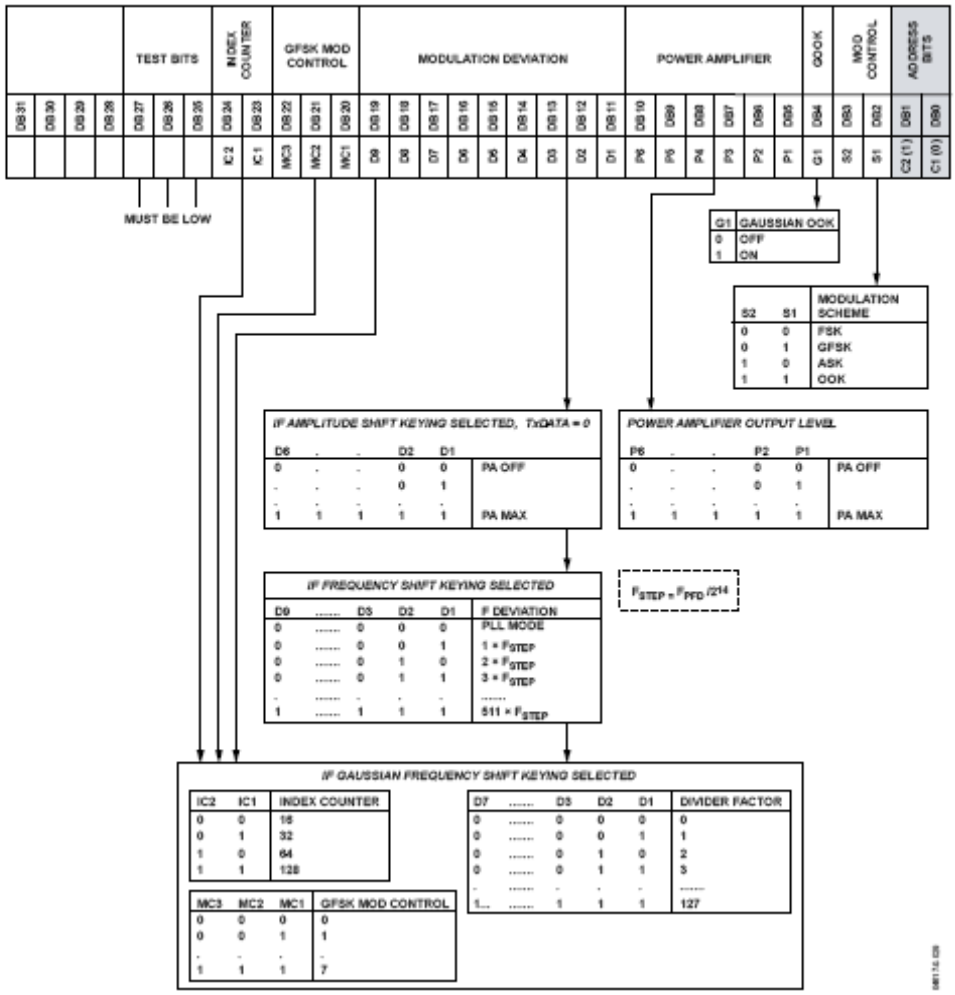
Frekvence v MHz	Hex – OBSAH N-REGISTRU	bin	Nfract	Nfrac	Nint	Nint
433.050	8CF79	10001100111101111001	001111011110	990	100011	35
433.100	8CFBD	10001100111110111101	001111101111	1007	100011	35
433.150	8CFFD	10001100111111111101	001111111111	1023	100011	35
433.200	8D03D	10001101000000111101	010000001111	1039	100011	35
433.250	8D07D	10001101000001111101	010000011111	1055	100011	35
433.300	8D0CD	10001101000011001101	010000110011	1075	100011	35
433.350	8D10D	10001101000100001101	010001000011	1091	100011	35
433.400	8D14D	10001101000101001101	010001010011	1107	100011	35
433.450	8D18D	10001101000110001101	010001100011	1123	100011	35
433.500	8D1D1	10001101000111010001	010001110100	1140	100011	35
433.550	8D215	10001101001000010101	010010000101	1157	100011	35
433.600	8D255	10001101001001010101	010010010101	1173	100011	35
433.650	8D299	10001101001010011001	010010100110	1190	100011	35
433.700	8D2DD	10001101001011011101	010010110111	1207	100011	35
433.750	8D31D	10001101001100011101	010011000111	1223	100011	35
433.800	8D361	10001101001101100001	010011011000	1240	100011	35
433.850	8D3A5	10001101001110100101	010011101001	1257	100011	35
433.900	8D3E5	10001101001111100101	010011111001	1273	100011	35
433.920	8D401	10001101010000000001	010100000000	1280	100011	35
433.950	8D429	10001101010000101001	010100001010	1290	100011	35
434.000	8D46D	10001101010001101101	010100011011	1307	100011	35
434.050	8D4AD	10001101010010101101	010100101011	1323	100011	35
434.100	8D4F1	10001101010011110001	010100111100	1340	100011	35
434.150	8D535	10001101010100110101	010101001101	1357	100011	35
434.200	8D575	10001101010101110101	010101011101	1373	100011	35
434.250	8D5B9	10001101010110111001	010101101110	1390	100011	35
434.300	8D5FD	10001101010111111101	010101111111	1407	100011	35

my remarks: *CanSat Book for Students* – part.1 2011

434.350	8D63D	10001101011000111101	010110001111	1423	100011	35
434.400	8D681	10001101011010000001	010110100000	1440	100011	35
434.450	8D6C5	10001101011011000101	010110110001	1457	100011	35
434.500	8D705	10001101011100000101	010111000001	1473	100011	35
434.550	8D749	10001101011101001001	010111010010	1490	100011	35
434.600	8D78D	10001101011110001101	010111100011	1507	100011	35
434.650	8D7CD	10001101011111001101	010111110011	1523	100011	35
434.700	8D811	10001101100000010001	011000000100	1540	100011	35
434.750	8D855	10001101100001010101	011000010101	1557	100011	35
434.800	8D895	10001101100010010101	011000100101	1573	100011	35

Pozn. Tato tabulka je shodná s tabulkou v **ESA Cansat Book** až na kód pro kmitočet 433.700 MHz. Je zřejmé, že v tabulce v *ESA Cansat Booku* je chyba. Současně je vidět, že v tabulce kódů pro volbu kmitočtů v *ESA Cansat Booku* je tento kód tvořen písmenem **F**, tj symbolem pro příkaz nastavení frekvence (další příkazy jsou rovněž označeny jednopísmennými symboly **M**, **S** …). Za písmenem **F** následuje v hexa kódu celý obsah registru 1, tj. včetně adresových bitů DB1 DB0 rovných 01. Nevýznamné nuly nalevo jsou v kódu vynechány a tak obsah všech bitů pro registr 1 dostaneme tak, že hexa kód převedeme na binární a poté zleva doplníme nulami na celkovou délku 32 bitů.

REGISTER 2: MODULATION REGISTER

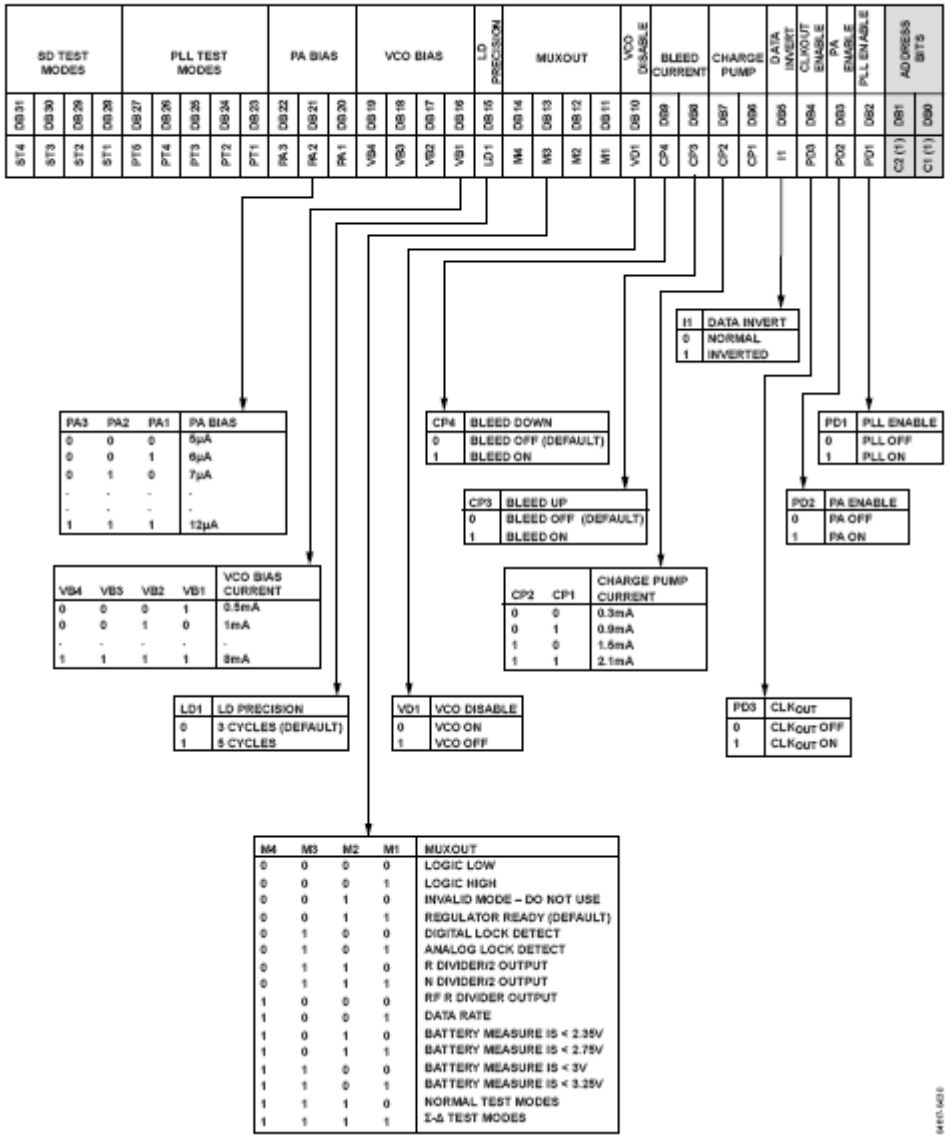


Přenášená data jsou **00000000 00000000 00110111 11100010**

Registr 2 je pojmenován **Modulation Registr**. Poslední (adresové)bity DB1 DB0 jsou 10 tj. dekadicky 2. Před nimi jsou dva nulé bity S1 a S2, což znamená modulace **FSK**. Ty předchází rovněž nulový bit DB4, znamenající vypnutí Gaussian OOK. Před ním je v DB5 až DB10 tj. P1 až P6 obsah 111111 neboli nastavení na maximální výkon koncového stupně vysílače. Většina zbývajících bitů je užita při jiných modulacích, než FSK, zejména při GFSK.

Registr 3:

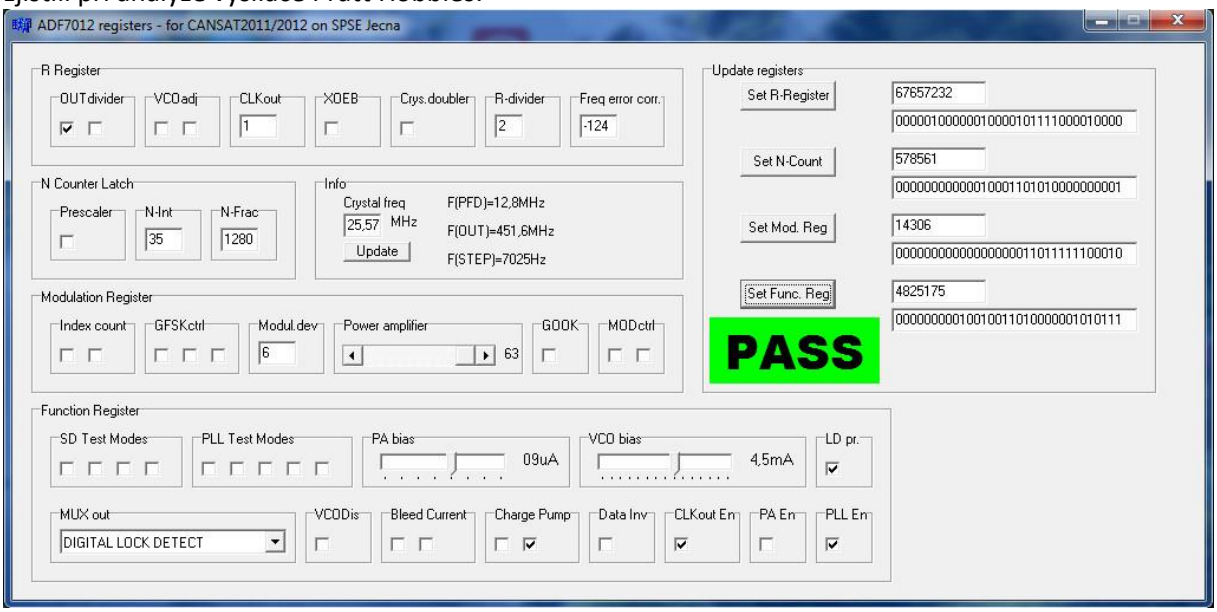
REGISTER 3: FUNCTION REGISTER



Přenášený obsah je **00000000 01011010 10100000 01010111**. Poslední dva bity jsou adresové a obsahují 11 tj. dekadicky 3 (registr 3). Jim předchází DB2 obsahující 1, což znamená zapnutý PLL. Jemu předchází nulový bit DB3 . Je velice důležitý, neboť určuje zapnutí (1) či vypnutí (0) koncového stupně PA. Po inicializaci se tedy žádný signál nevysílá. Nastaví se na 1 před tím, než se budou vysílat vlastní data (pakety) a po jejich odvysílání se zase nastavuje na nulu. Předchozí bit DB4 je nastaven na 1, neboli je zapnuto generování CLK_{OUT} . Nula v DB5 ukazuje že data nebudou invertována. Obdobně další předchozí bity určují nastavení vlastností *ADF7012* jako proud nábojové pumpy atd.

Vysílačem pro APRS v pásmu 144MHz s ATMega88 a ADF7012 se zabývala i diplomová práce [26] Martina Sabola přičemž autor diplomové práce na její CD příloze publikoval i program pro výpočet a

nastavení obvodu ADF7012. Protože M. Sabol poskytl i zdrojové kódy tohoto programu, mohl jsem ho upravit k využití pro návrh vysílačů CanSATu. Následující obrázek ukazuje tento program s nastavenými hodnotami pro inicializaci ADF7012, které vytvářejí stejné 4 řídící slova, které jsme zjistili při analýze vysílače Pratt Hobbies:

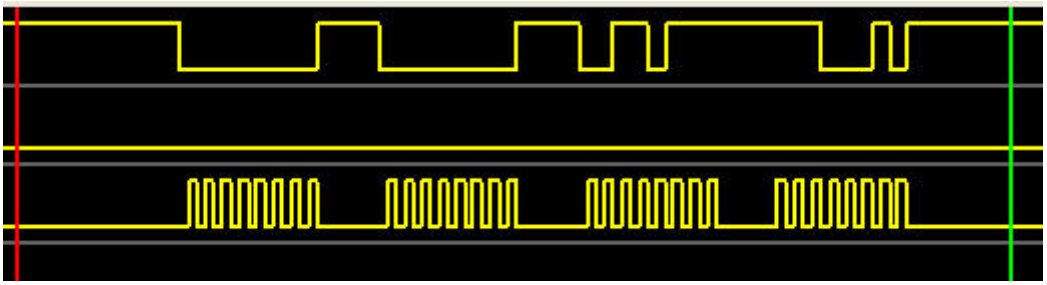


Při nastavování vysílače a kontroly jeho funkce se nám bude hodit pin MUXOUT. Je to výstupní pin jehož funkce je nastavena čtveřicí bitů DB14 DB13 DB12 DB11 obsahující bity M4 M3 M2 M1, jejichž význam je dán tabulkou uvedenou u popisu registru 3. V našem případě obsahuje 0100 mající význam DIGITAL LOCK DETECT. Slouží k indikaci toho, že regulační smyčka PLL je v ustáleném stavu (PLL je „zachycen") tj na svém výstupu poskytuje signal o nastaveném kmitočtu. Pokud by M4 M3 M2 M1 obsahovaly 0101, MUXOUT by poskytoval Analog Lock Detect. V těchto režimech budeme MUXOUT používat při nastavování vysílače.

Nyní již víme, jaké řídící 32 bitová slova posílá firmware *ATMega88* při inicializaci. Otestujeme ještě, jaké řídící slova posílá při provádění příkazů M, F a S zasílaných sériově do vstupu RxD procesoru *ATMega88*.

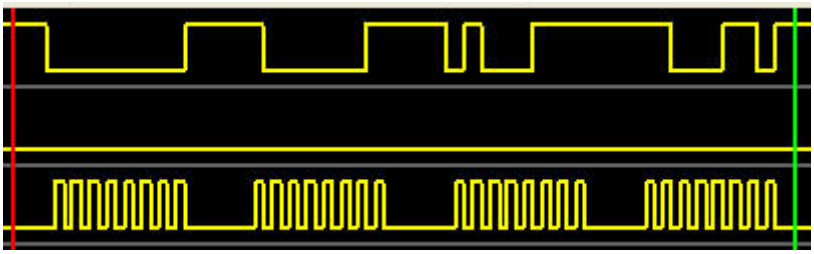
Nejprve nastavení modulace a rychlosti přenosu. K tomu slouží příkaz **M** ve tvaru **M**1200 nebo **M**9600 :

Nejprve pošleme do *ATMega88* příkaz **M1200**, logický analyzátor zobrazí:



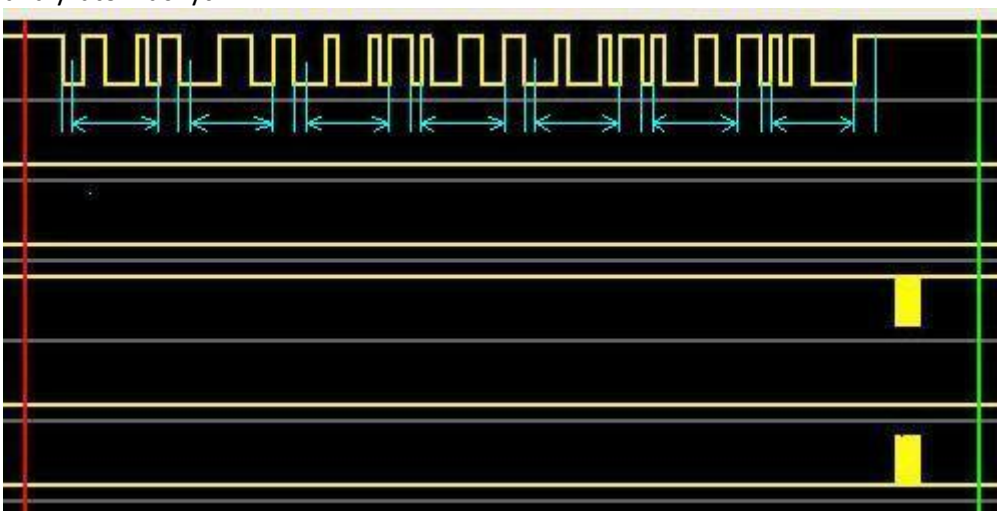
tj. **00000000 00000000 00110111 11100010** , což se zapisuje do registru 2. Je to naprosto totéž, co se do tohoto registru zapisuje při inicializaci.

Do *ATMega88* nyní pošleme příkaz **M9600**, logický analyzátor zachytí



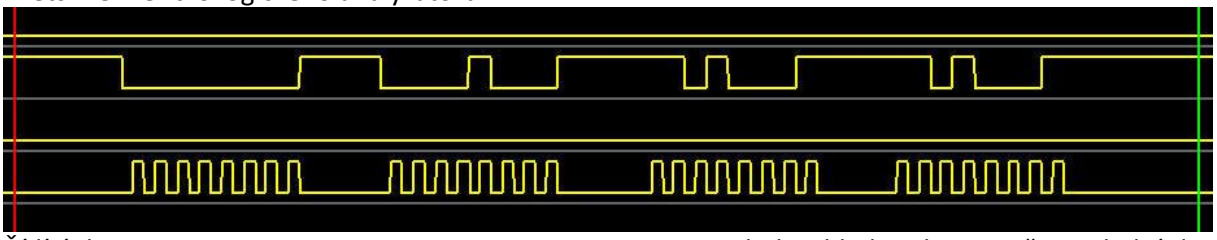
Což je **00000000 10000001 01000111 11000110**

Nyní otestujeme příkaz F, např F8D1D1 tj příkaz k nastavení kmitočtu na 433,500 MHz. Logický analyzátor zachytí



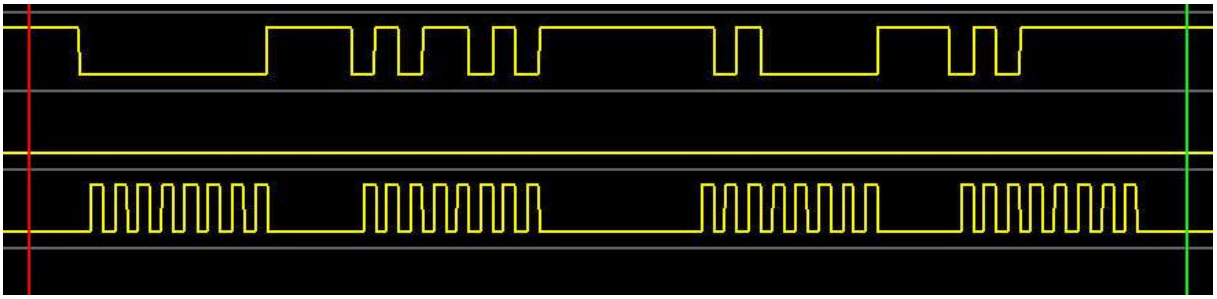
V horním kanálu je zachycen signál do RxD obvodu *ADF7012*. Do obrazku jsem zakreslil svislé čáry a dále vodorovné s šipečkami. Vidíme, že každý znak je předcházen jedním nulovým START pulzem a následován dvěma jedničkovými STOP pulzy. Vidíme, že je odesláno 7 znaků ASCII : F8D1D1 a CR (t.j OD). Binární reprezentace těchto ASCII jsou pro F 01000110 , pro 8 je 00111000, pro D 01000100, pro jedničku 00110001 a pro CR 00001101. Jednotlivé bity těchto znaků jsou ovšem vysílány v obráceném pořadí, tj. od nejméně významného d_0 , d_1 … d_7 .

Zvětšíme měřítko logického analyzátoru



Řídící data jsou **00000000 00001000 11010001 11010001** neboli vzhledem k tomu, že poslední dva bity jsou 01, bude jimi naplněn registr 1 (N-counter latch). Tato data také přesně odpovídají vypočítaným číslům N v tabulce kmitočtů.

Nakonec otestujeme příkaz **S**. Odešleme např Sabc následovaný CR. Logický analyzátor zobrazí řídící data



Což je **00000000 01011010 10100000 01011111** neboli hexadecimálně 0x5AA05Fh. Poslední dvě 11 potvrzují, že půjde o data do registru 3. Tento obsah se lisí od obsahu vytvářeného při inicializaci je v jediném bitu – zapnutí koncového stupně vysílače. Tj **PA on** se provádí pomocí **0x5AA05F**, a **PA off** pomocí **0x5AA057**. Řetězec znaků uvedený jako parametr se doplní o služební údaje tak, aby vytvořil paket dle AX25 a odešle se prostřednictvím PB0 počítače ATMega88 do pinu 4 tj. TxData obvodu ADF7012.

6.1.3.2 Naprogramování inicializační části firmware (Initial Firmware Coding and Programming)

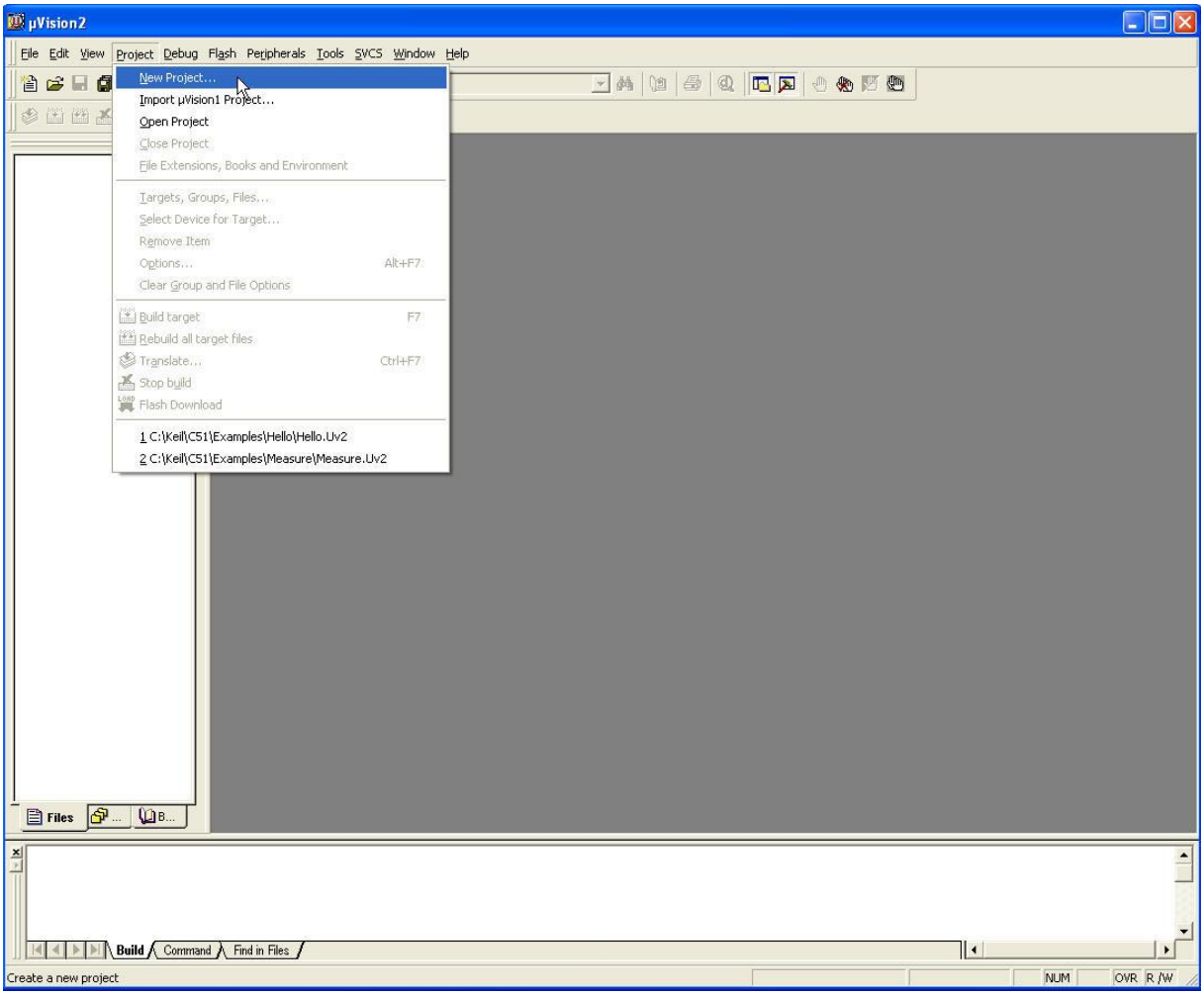
V inicializační části potřebujeme sériove poslat 4 řídící 32bitová slova 0x02085E10, 0x0008D401, 0x000037E2a 0x005AA057.

6.1.3.2.1 Jednočipy x51 (x51 Single Chip Computer)

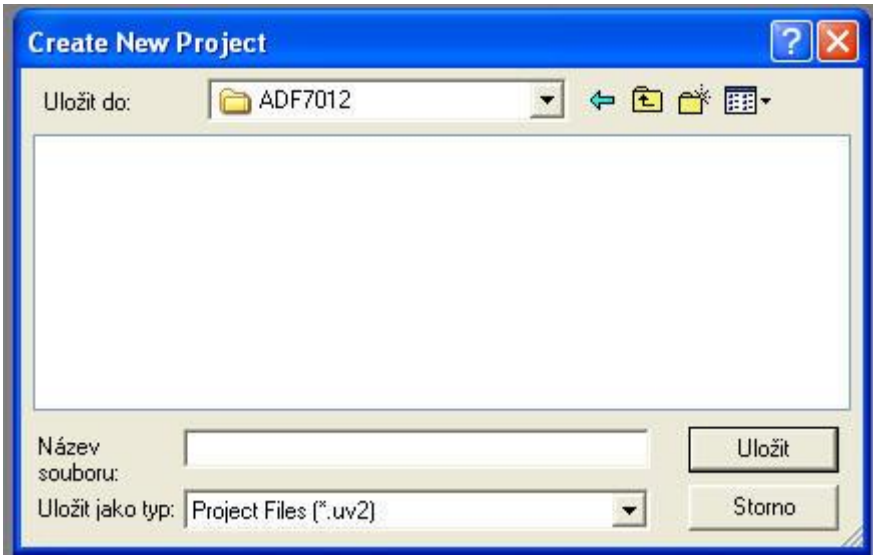
Uvedeme si kódy pro několi různých tříd jednočipových počítačů. Snad nejstarším dosud nejpoužívanějším 8bitovým MCU jsou MCU řady x51, tj kompatibilním s 8051. Příkladem může být AT89x51.

Příklad vytvoření kódu inicializace x51:

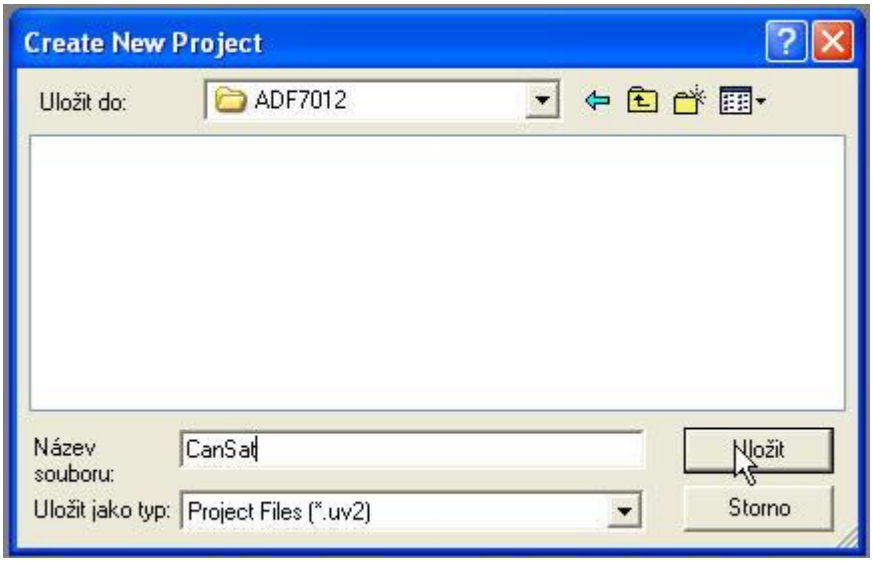
Jako vývojový sw jsem použil demo verzi **uVision2** firmy Keil Software.



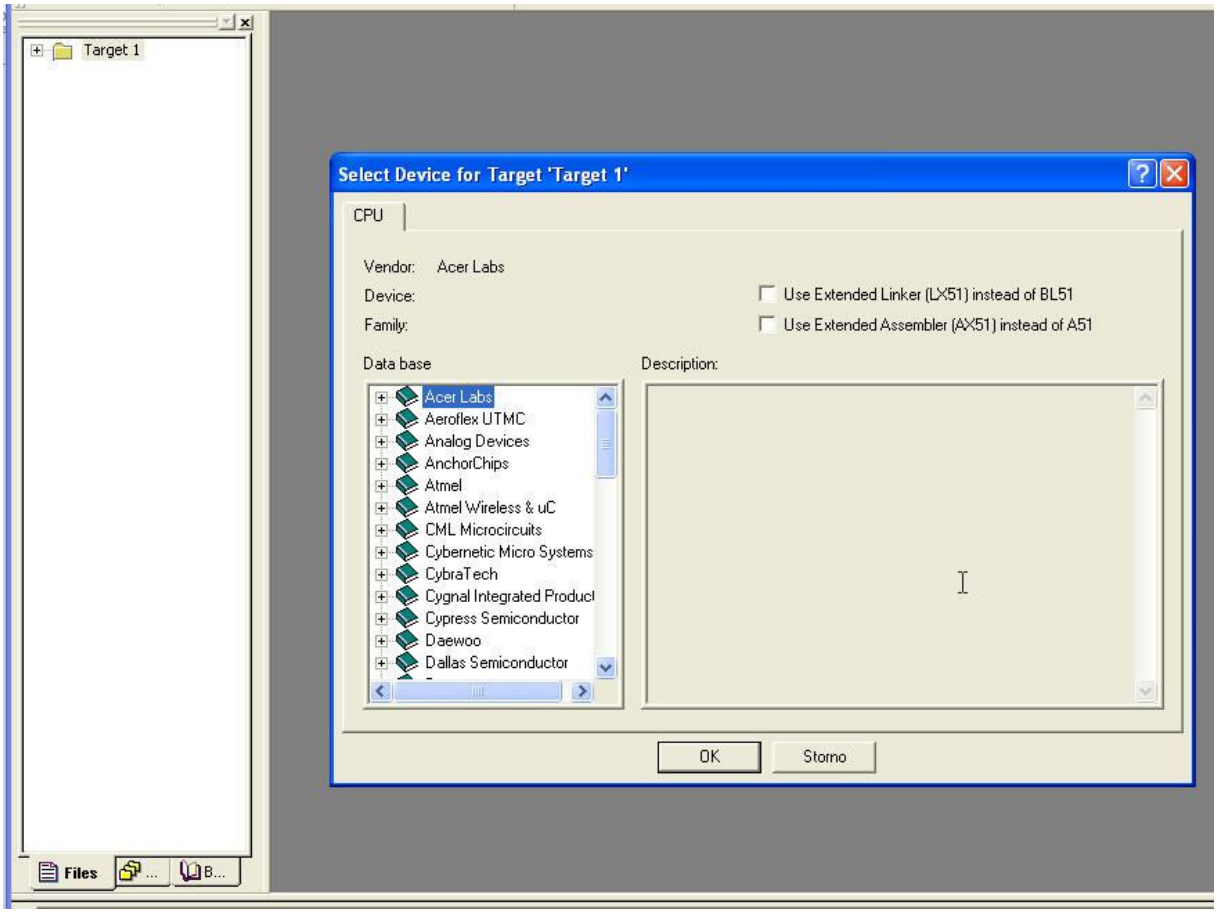
Po spuštění tohoto vývojového prostředí vybereme v menu **Project → New Project…** Objeví se okno



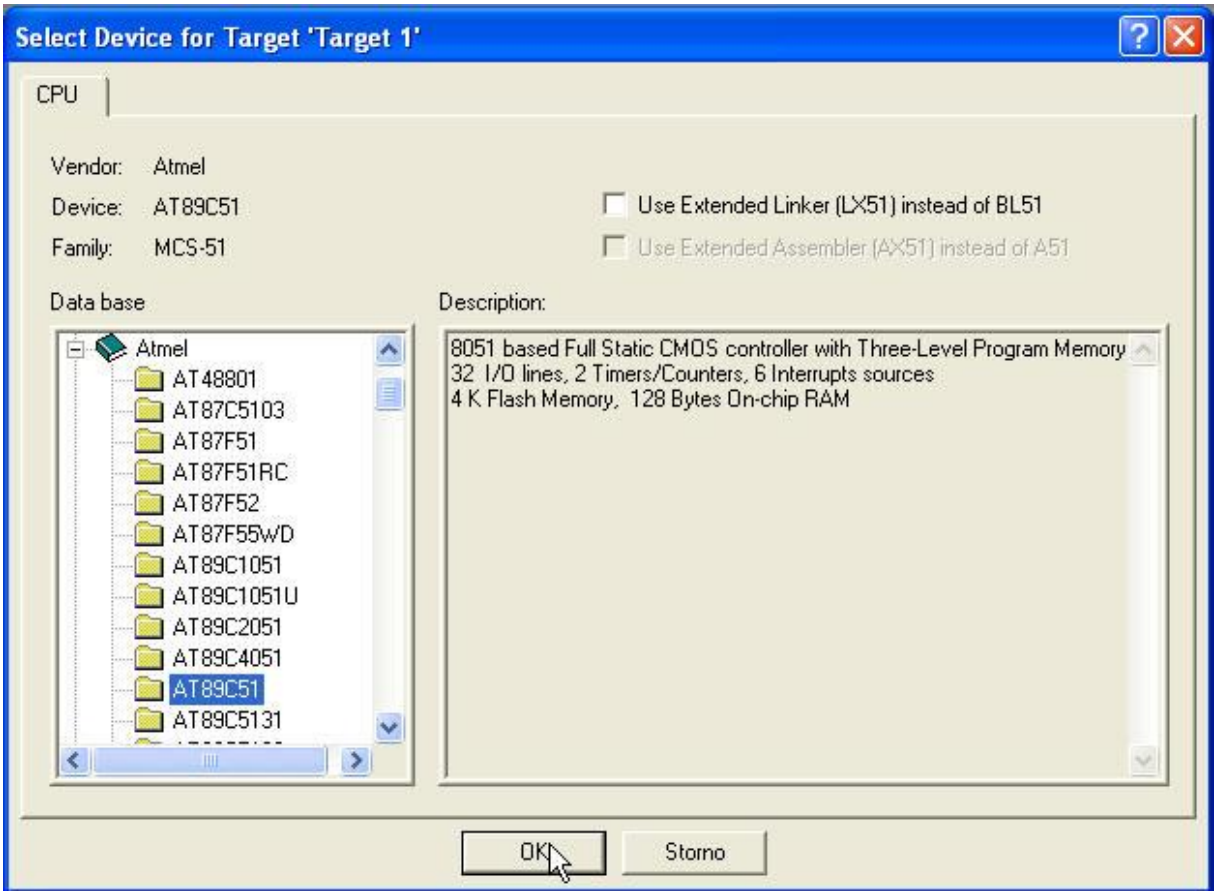
Zvolíme a vyplníme název souboru .uv2



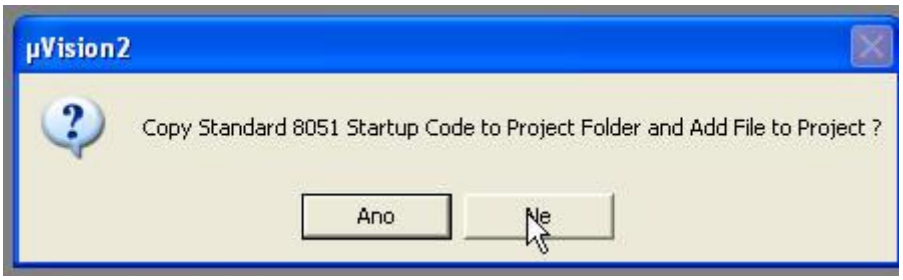
A klikneme na tlačítko **Uložit**. Dostaneme



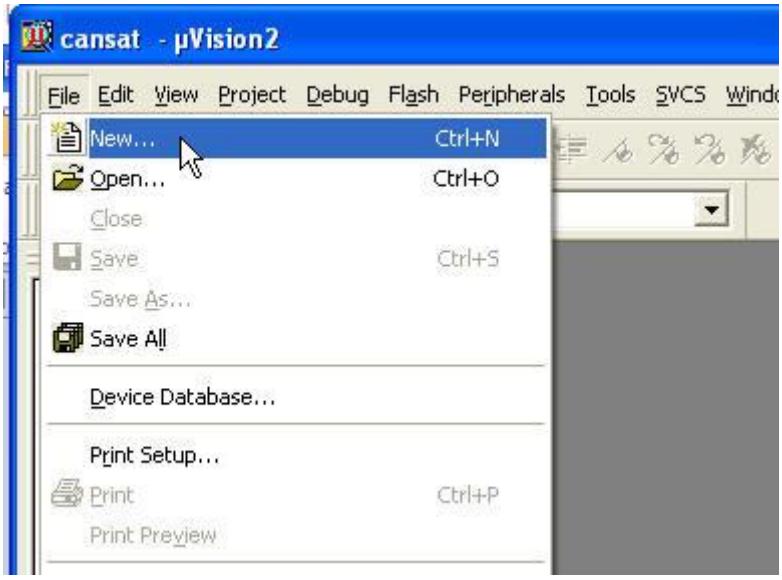
Spustil se automaticky wizard, nutící nás vybrat CPU. Zvolíme ATMEL



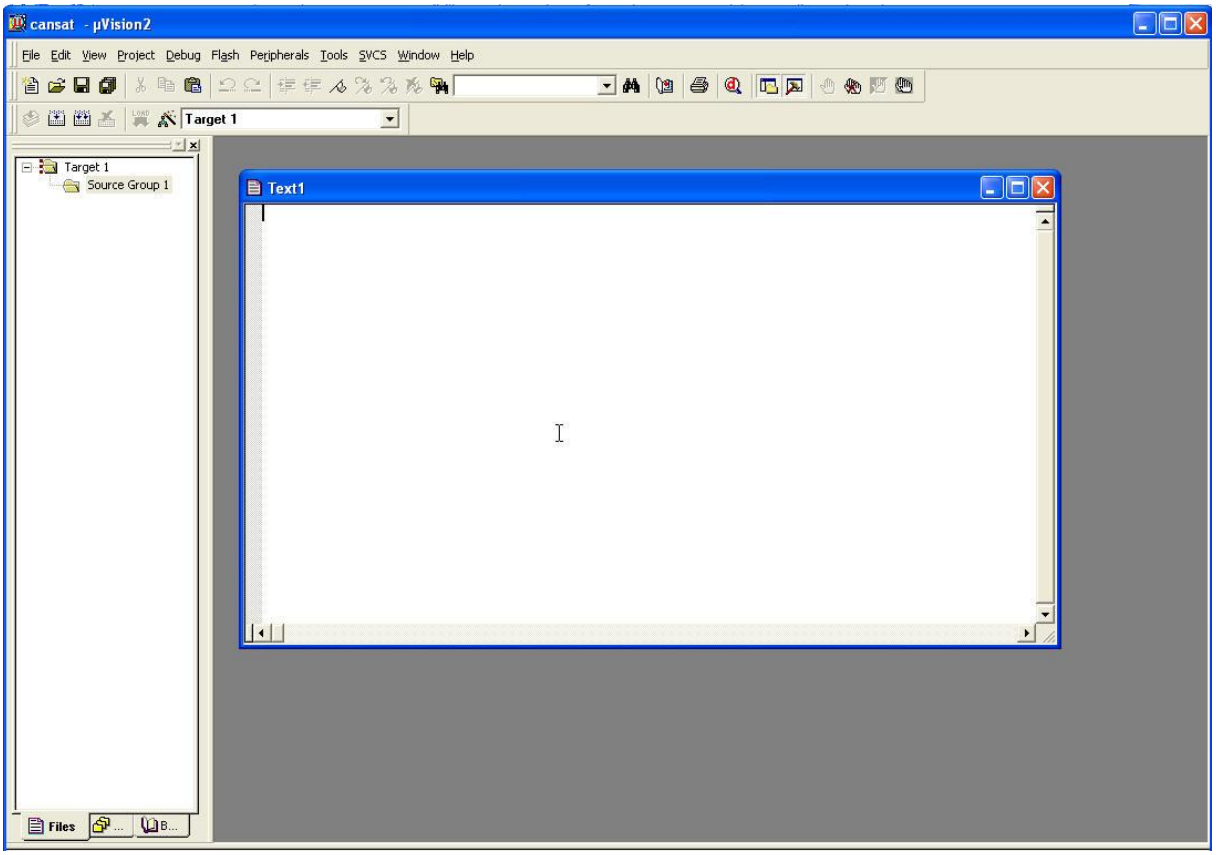
A následně vybereme **AT89C51**. Výběr potvdíme OK.



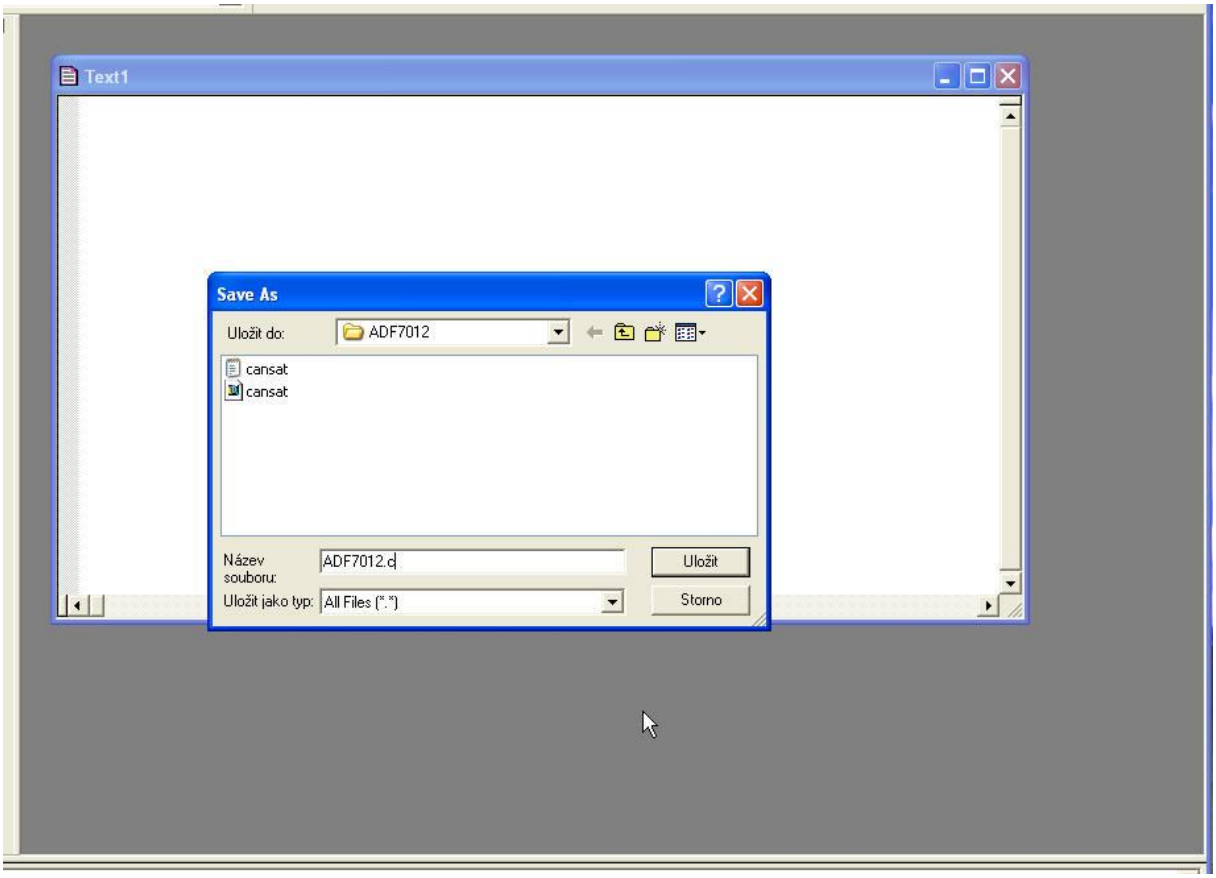
Na výše uvedenou nabídku odpovíme **Ne**. Poté vytvoříme nový prázdný soubor pro náš zdrojový kód:



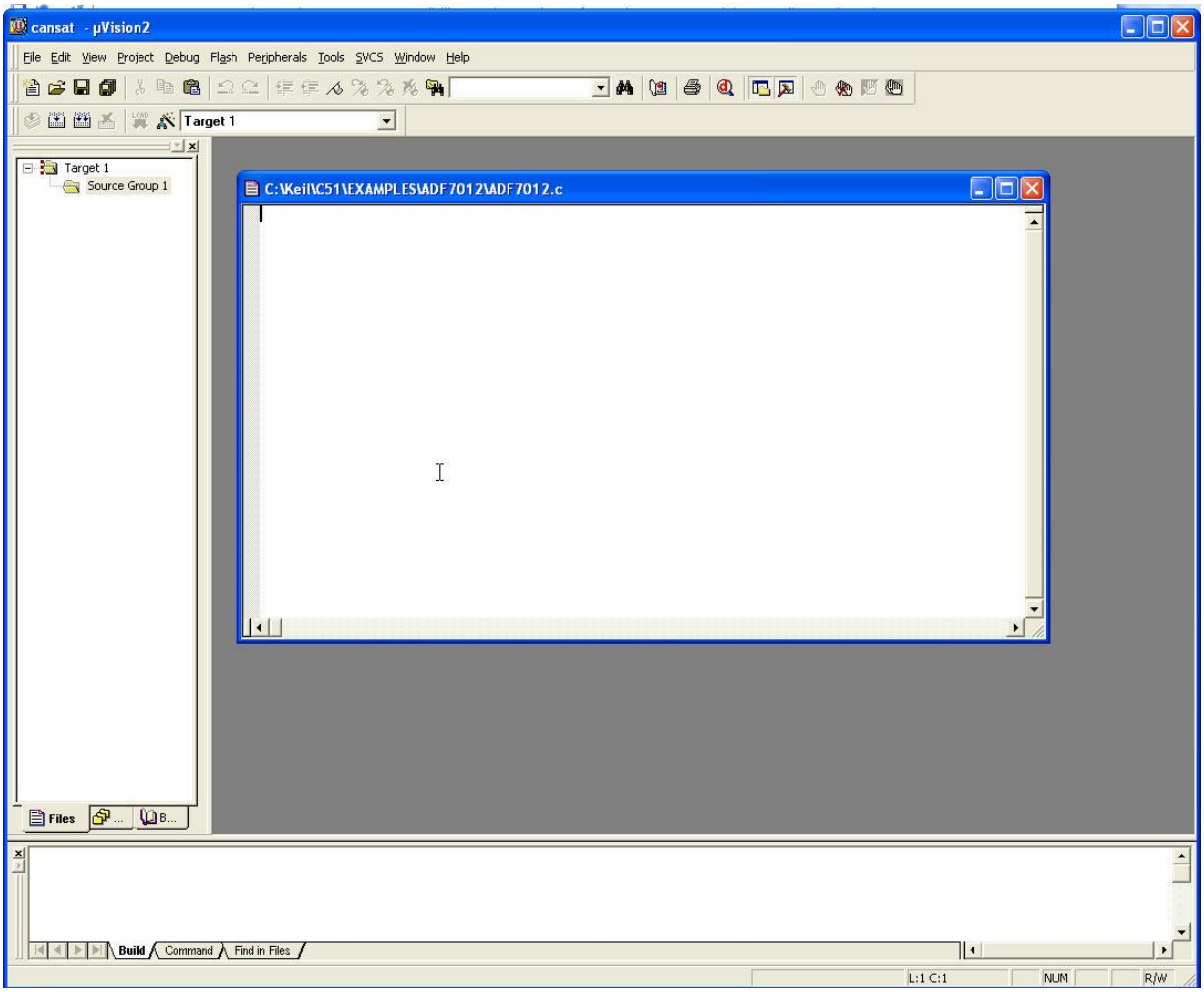
Uděláme to tak, že v menu vybereme **File →New…** Dostaneme



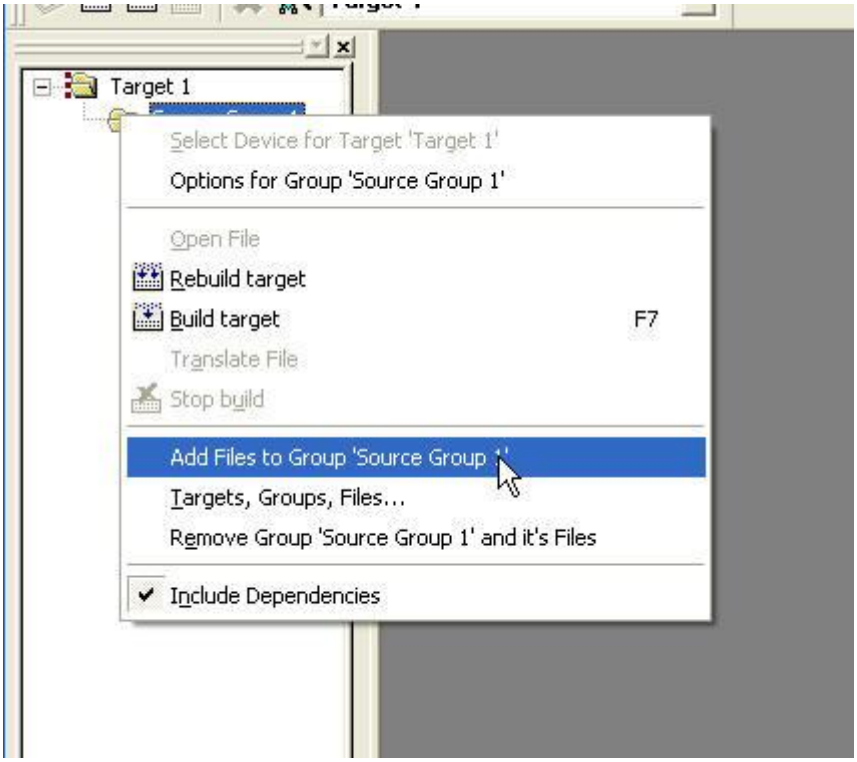
V menu dále vybereme menu vybereme **File →Save as…** Objeví se okno **Save as**, v němž vyplníme požadované jméno souboru



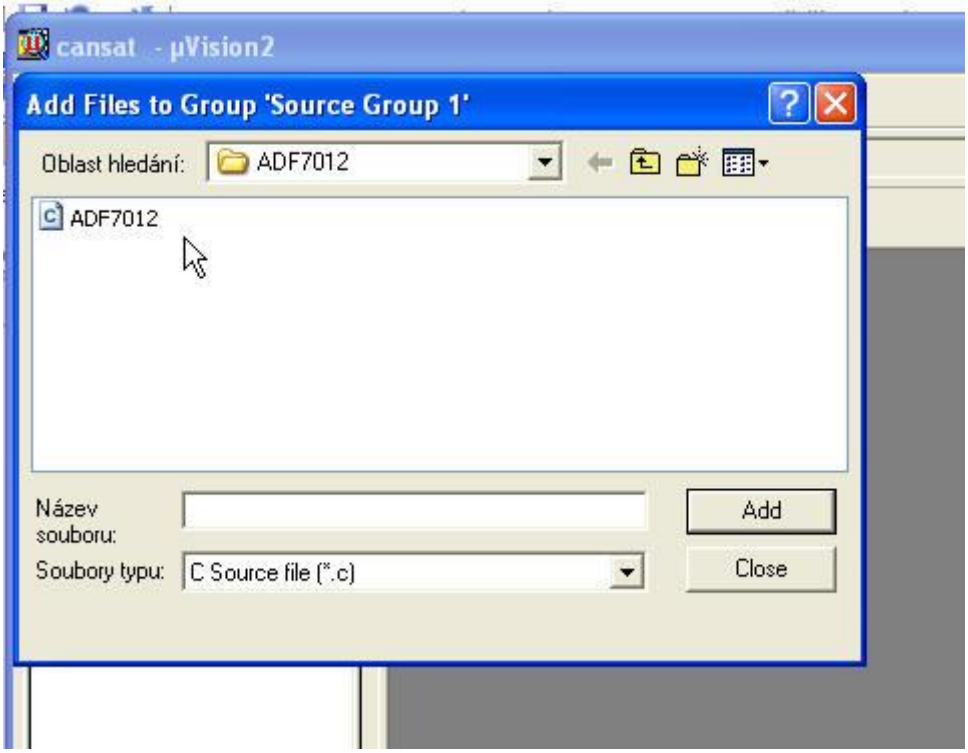
Zvolil jsem *ADF7012.c* a potvrdil tlačítkem **Uložit**



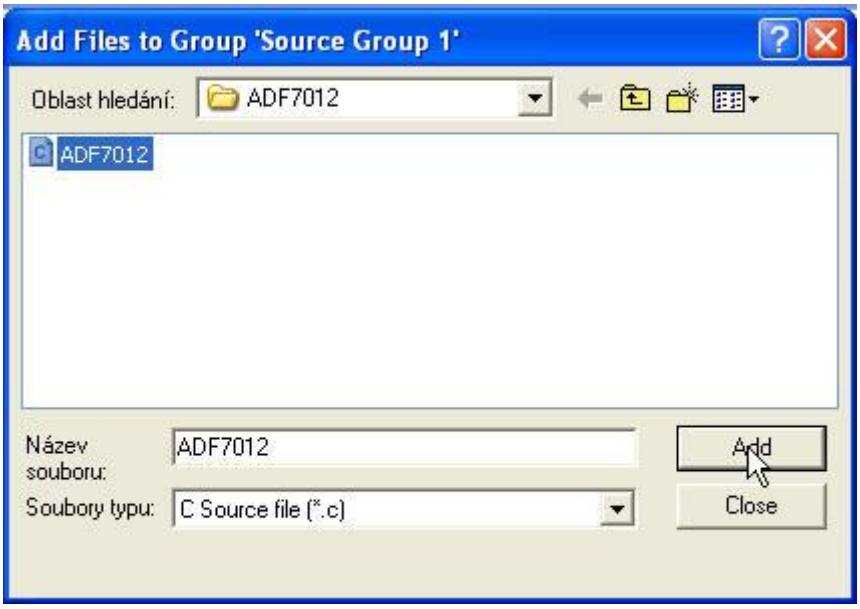
Vytvořený soubor *ADF7012.c* však dosud není součástí projektu. Proto



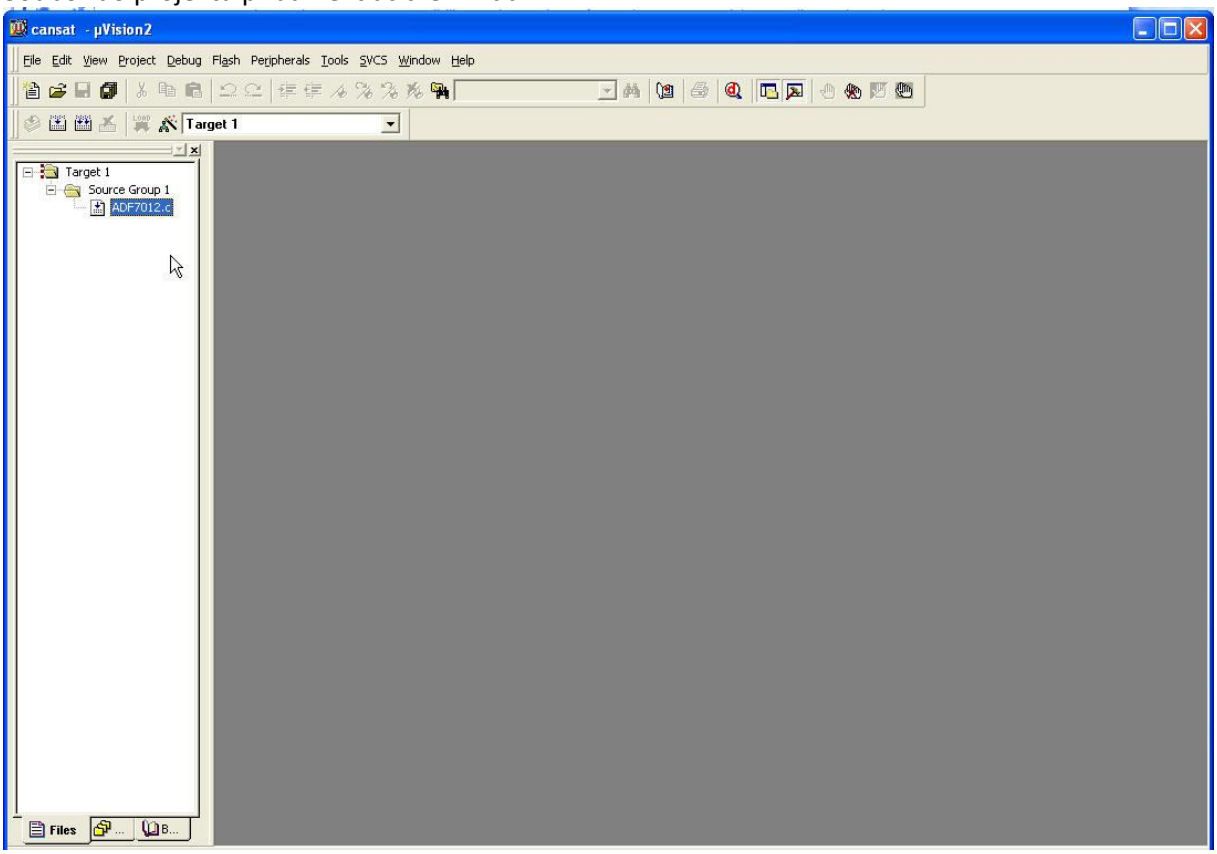
na ikonce **Source Group 1** pravým tlačítkem myši rozvineme místní menu a vybereme položku **Add Files to Group 'Source Group'** . Objeví se okno



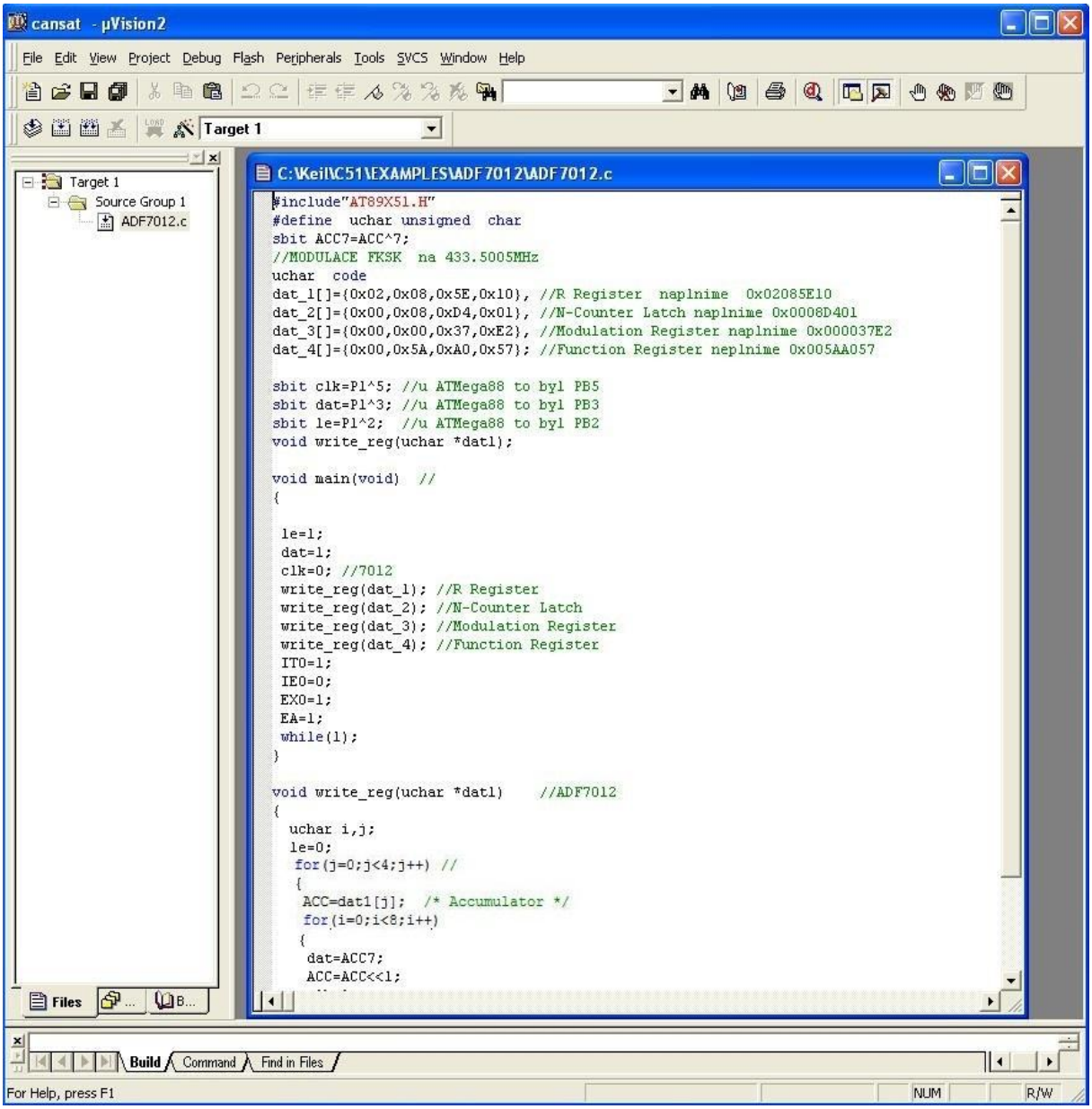
Vybereme soubor *ADF7012.c*



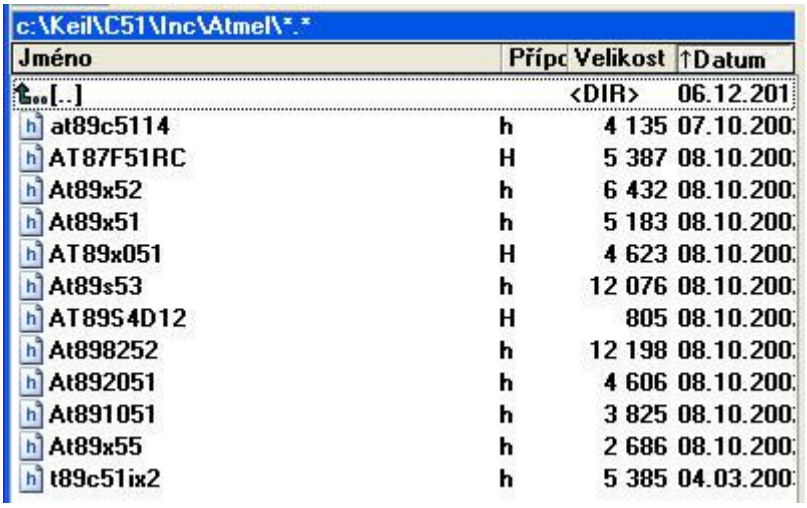
Soubor do projektu přidáme tlačítkem **Add**



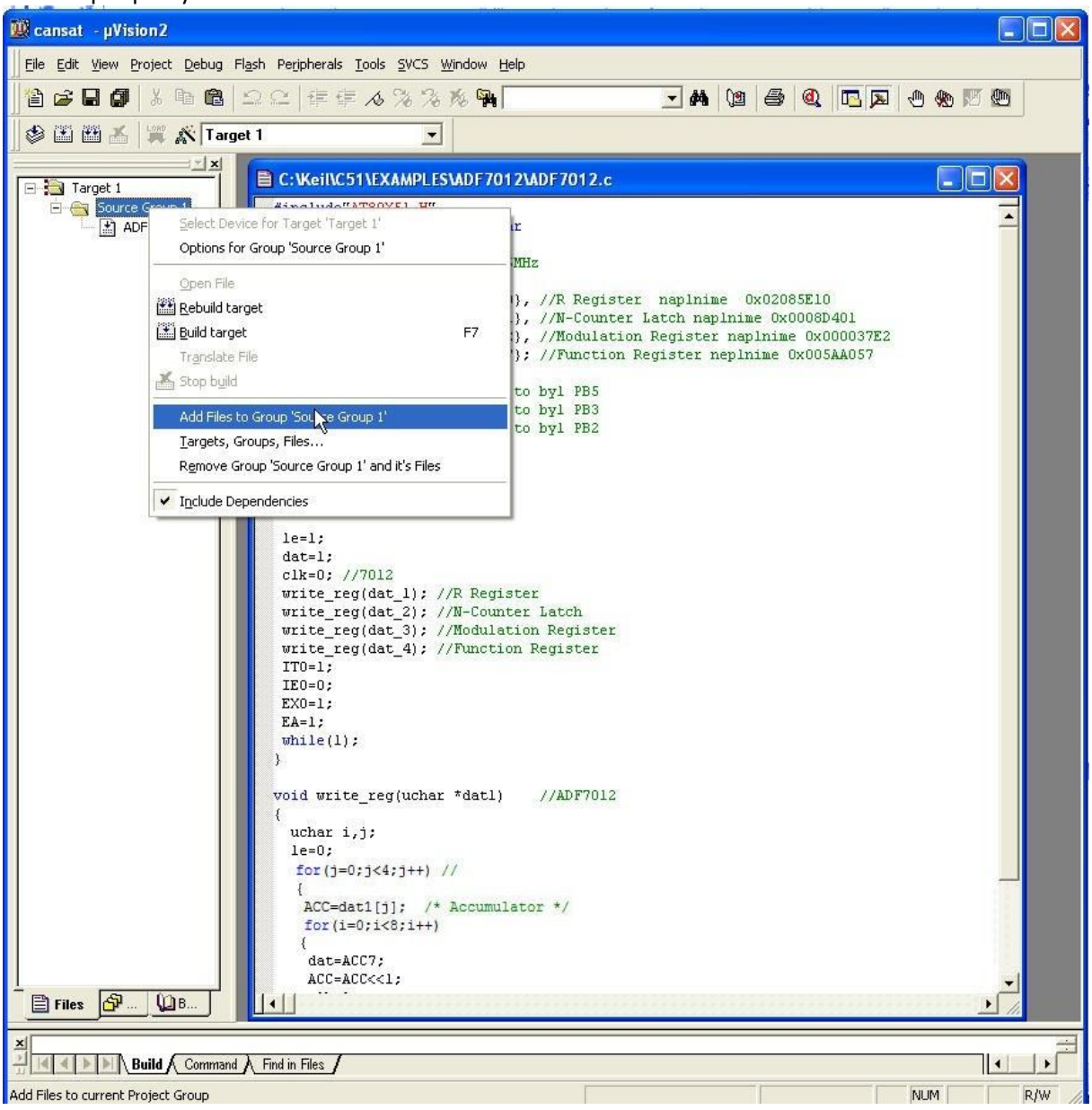
Klikneme na ikonky *ADF7012.c*. Tím se otevře dosud prázdný soubor, do kterého napíšeme kód:



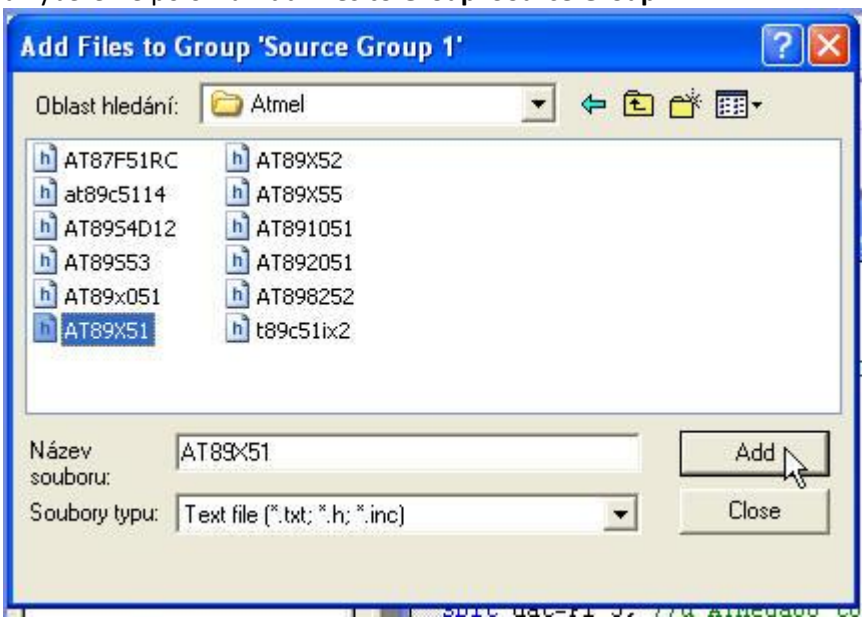
Funkci tohoto kódu se vysvětlíme na konci této podkapitily. Ještě musíme přidat hlavičkový soubor *AT89X51.H* . Tento soubor najdeme mezi hlavičkovými soubory v



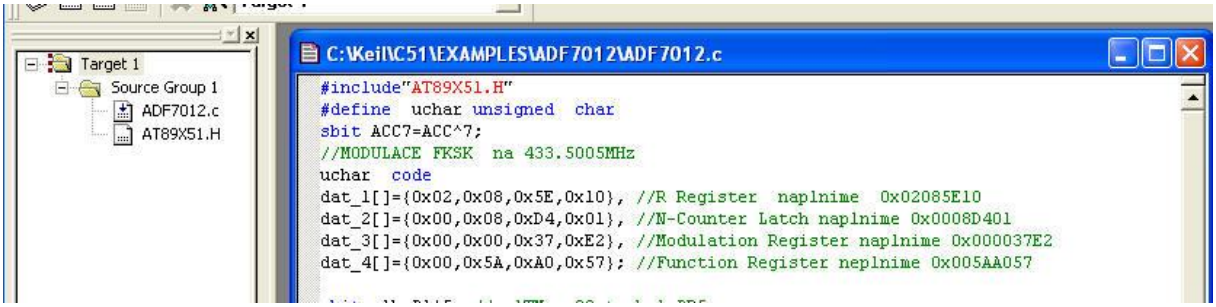
Takže opět pravým tlačítkem rozvineme místní menu



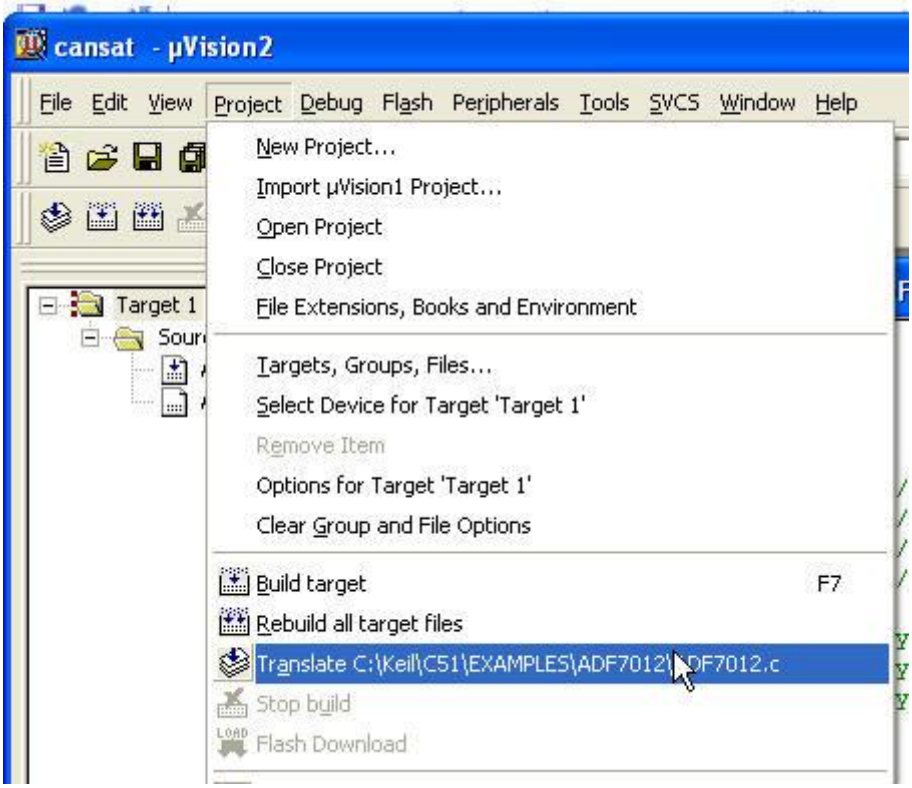
a vybereme položku **Add Files to Group 'Source Group'**



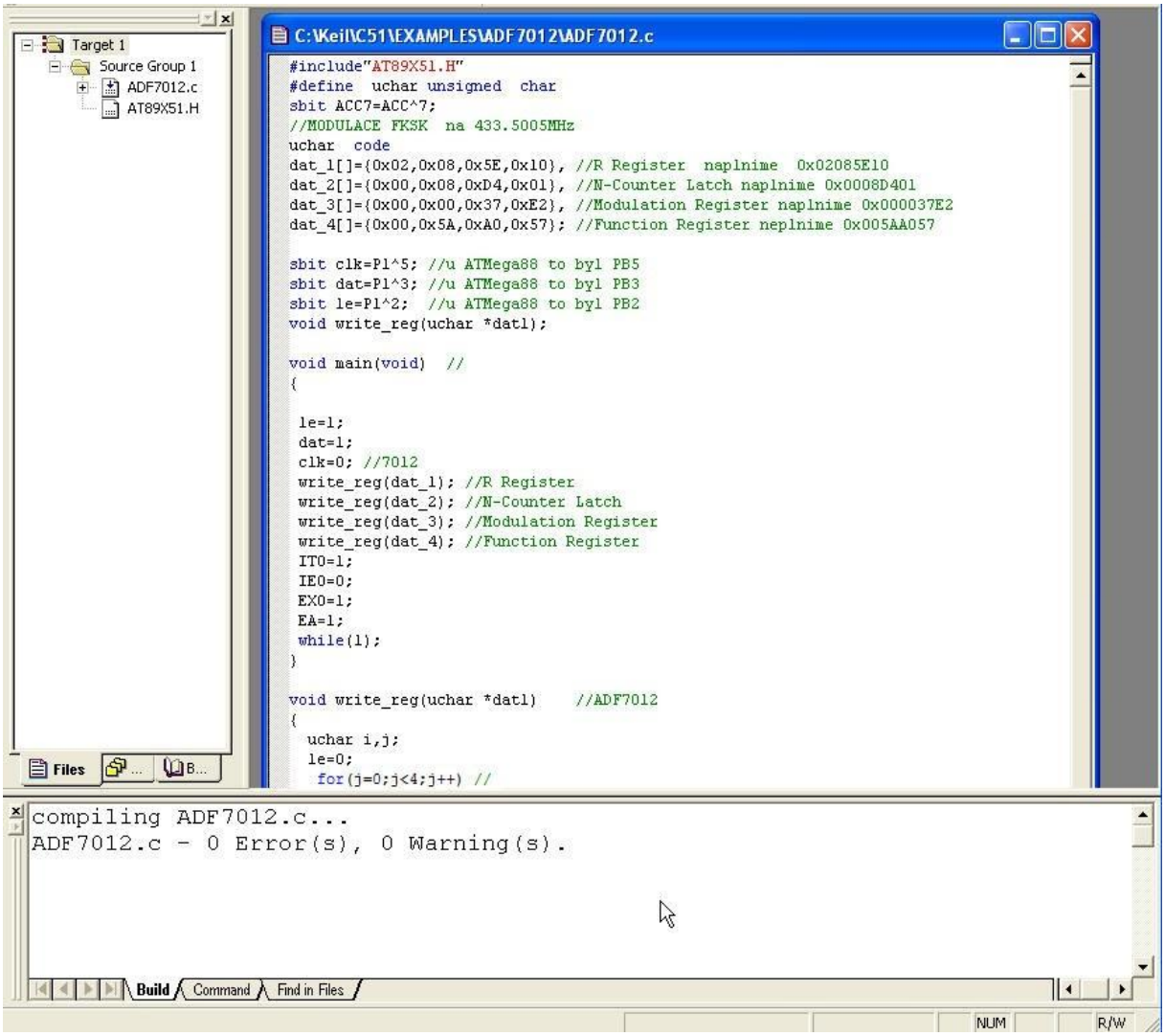
Najdeme soubor AT89X51.h a pomocí tlačítka **ADD** ho přidáme do projektu



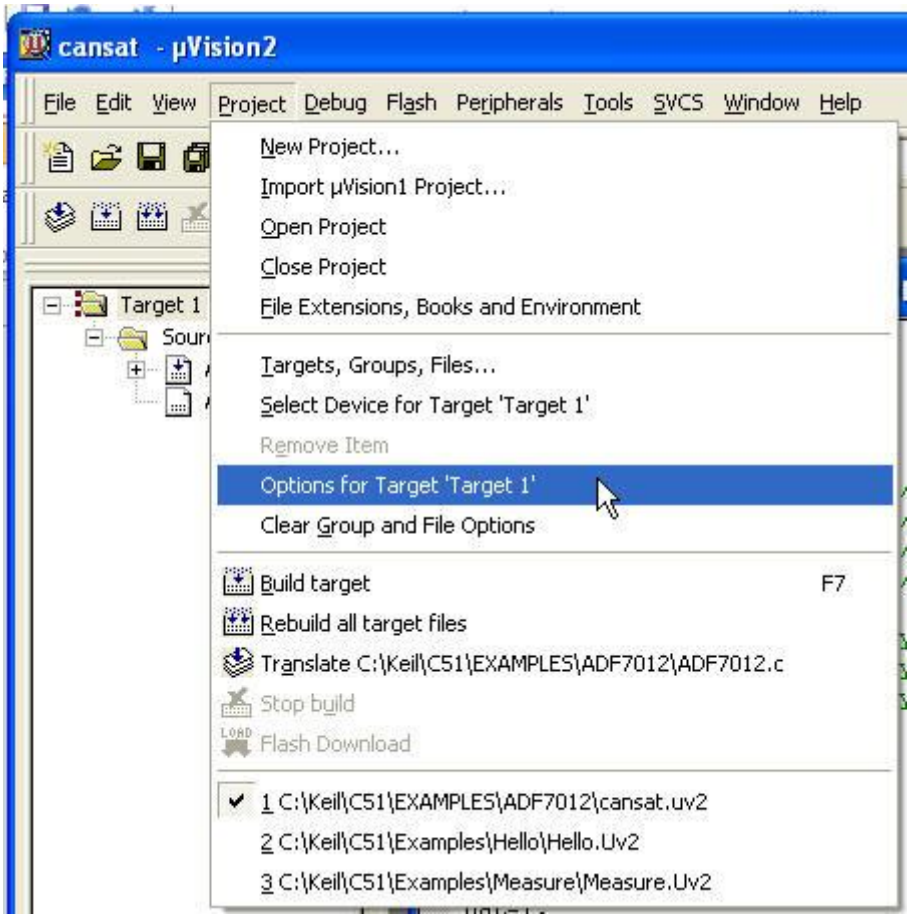
Zkusíme přeložit zdrojový kód.



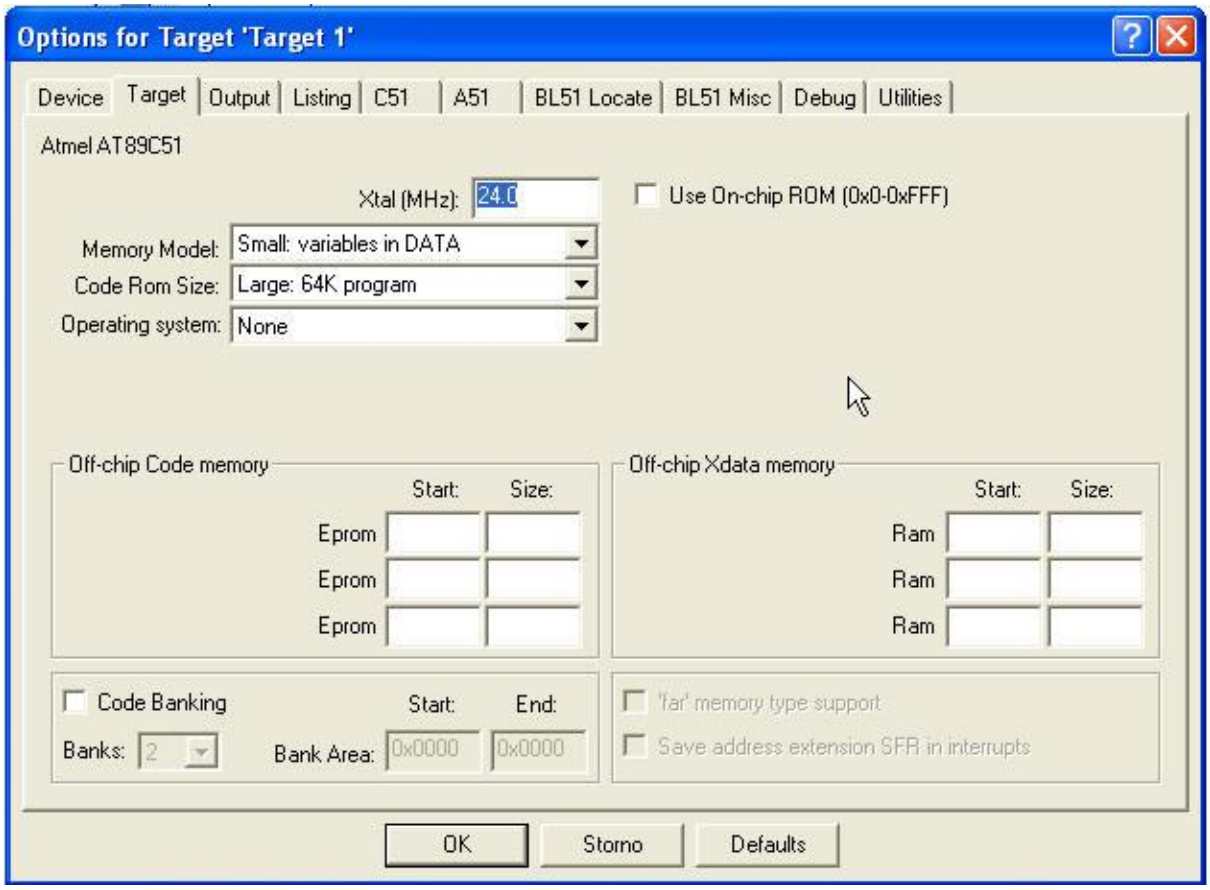
V menu vybereme **Project → Translate C…….**



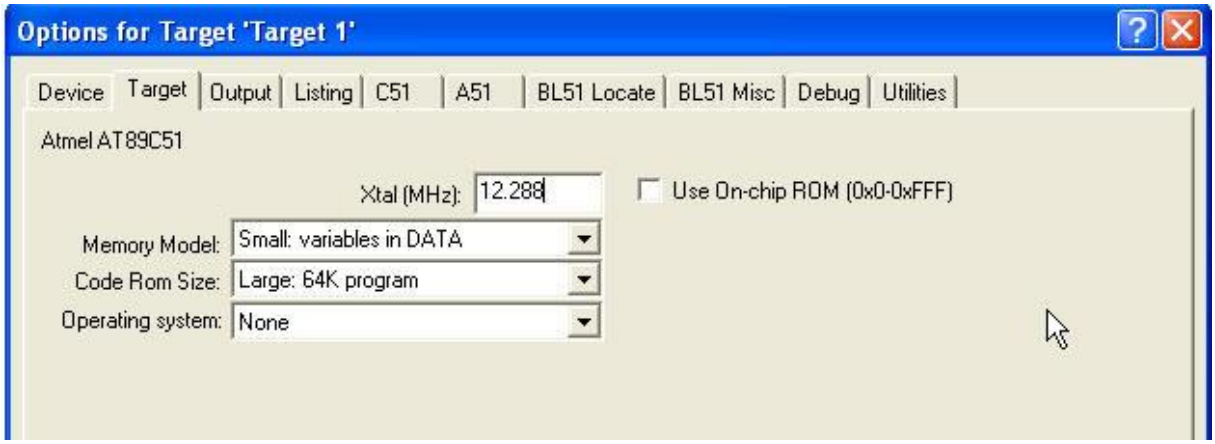
Vidíme, že překlad byl úspěšný, takže se pokusíme o vytvoření souboru HEX pro programátor (např. nám již známý *PonyProg2000)*. Musíme nastavit vlastnosti (**Options**) projektu:



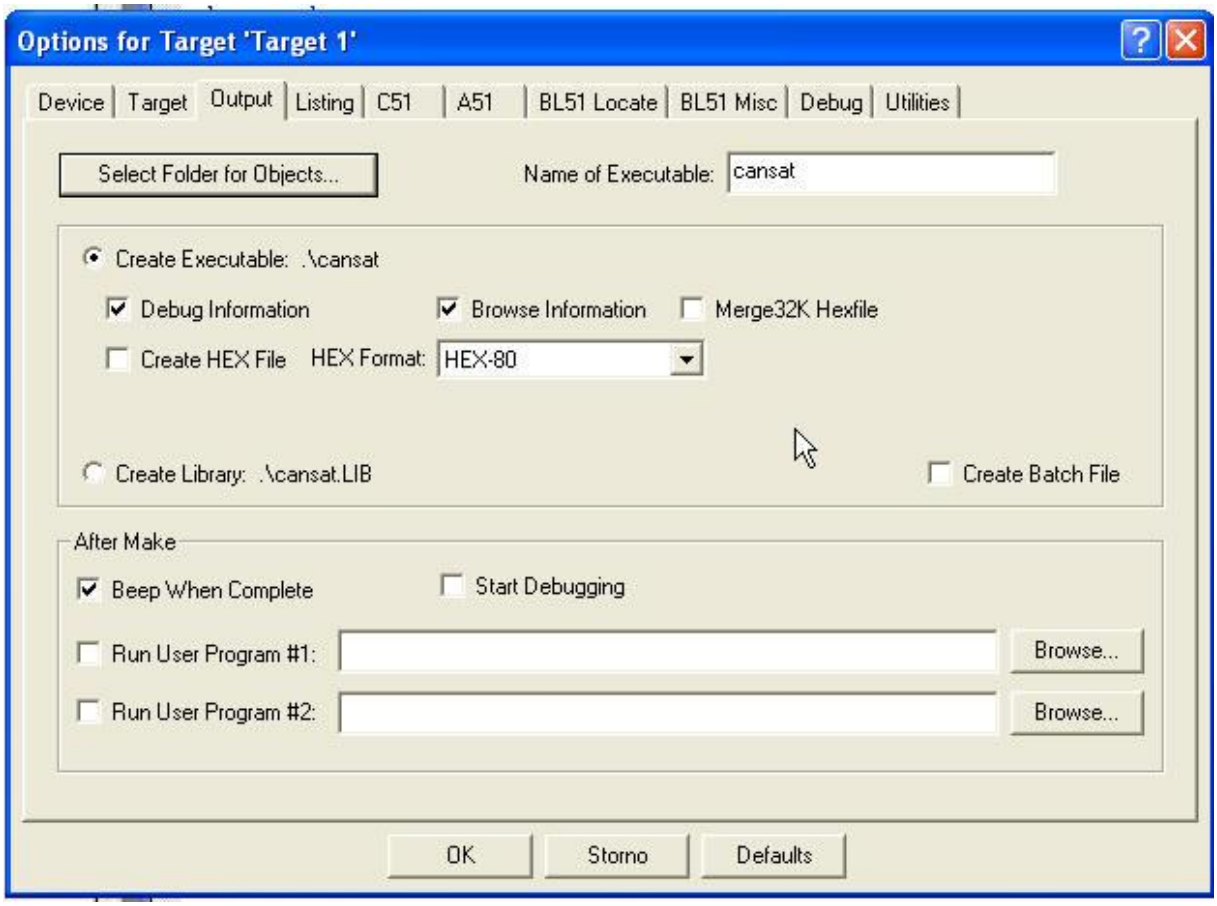
Proto vybereme v menu **Project -- >Options for …**



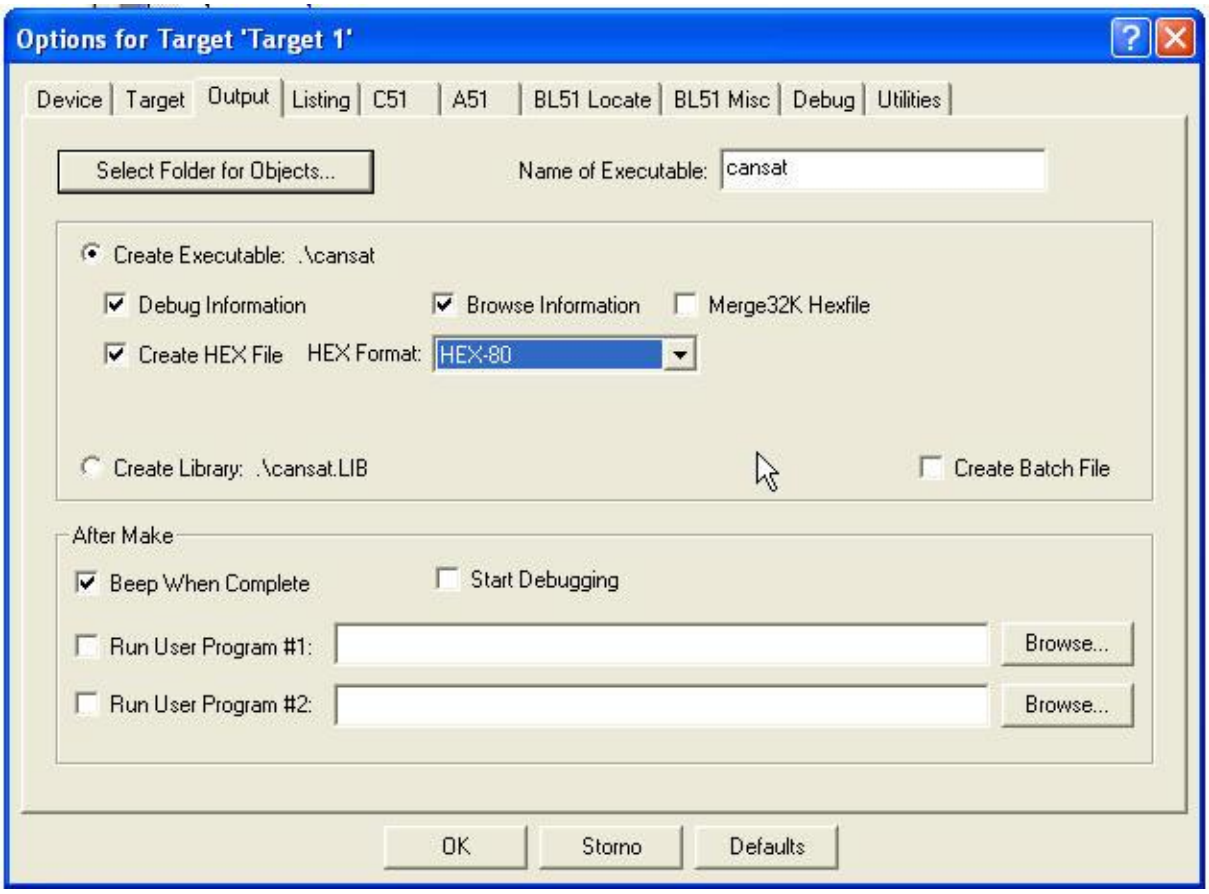
Vyplníme správný hodiný kmitočet



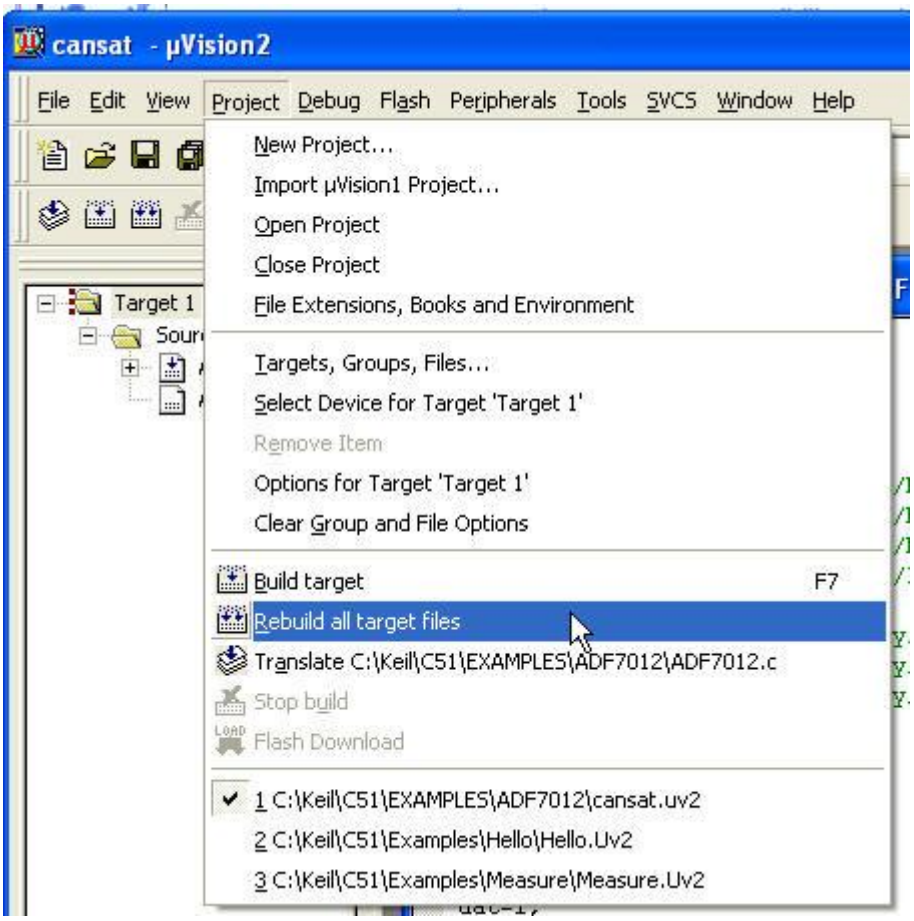
A ještě zvolíme, co má být výstupem (Output)



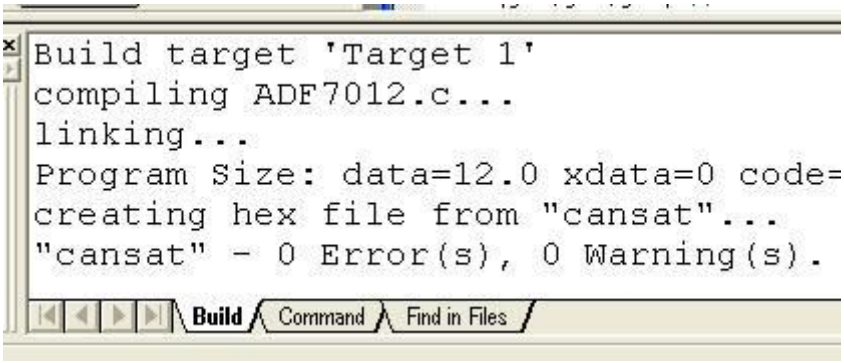
Zaškrtneme **Create HEX File**,



Potvrdíme **OK**.



Spustíme Rebuild all target files



Rebuild se podařil.

Máme i HEX soubor. Nyní si vysvětlíme, jak funguje inicializační kód :

```
#include"AT89X51.H"
#define  uchar unsigned  char
sbit ACC7=ACC^7;
//MODULACE FSK  na 433.500 MHz
uchar  code
dat_1[]={0x02,0x08,0x5E,0x10}, //R Register  naplnime  0x02085E10
dat_2[]={0x00,0x08,0xD4,0x01}, //N-Counter Latch naplnime 0x0008D401
dat_3[]={0x00,0x00,0x37,0xE2}, //Modulation Register naplnime 0x000037E2
dat_4[]={0x00,0x5A,0xA0,0x57}; //Function Register neplnime 0x005AA057

sbit clk=P1^5; //u ATMega88 to byl PB5, 89X51 má porty P0,P1 …
sbit dat=P1^3; //u ATMega88 to byl PB3
sbit le =P1^2; //u ATMega88 to byl PB2
void write_reg(uchar *dat1);

void main(void)  //
{
 le=1;
 dat=1;
 clk=0; //7012
 write_reg(dat_1); //R Register
 write_reg(dat_2); //N-Counter Latch
 write_reg(dat_3); //Modulation Register
 write_reg(dat_4); //Function Register
 IT0=1;
 IE0=0;
 EX0=1;
 EA=1;
 while(1);
}


void write_reg(uchar *dat1)     //ADF7012
{
  uchar i,j;
  le=0;
  for(j=0; j<4; j++) //
  {
   ACC=dat1[j];
   for(i=0; i<8; i++)
   {
```

```
    dat=ACC7;
    ACC=ACC<<1;
    clk=1;
       clk=0;
   }
  }
  le=1;
}
```

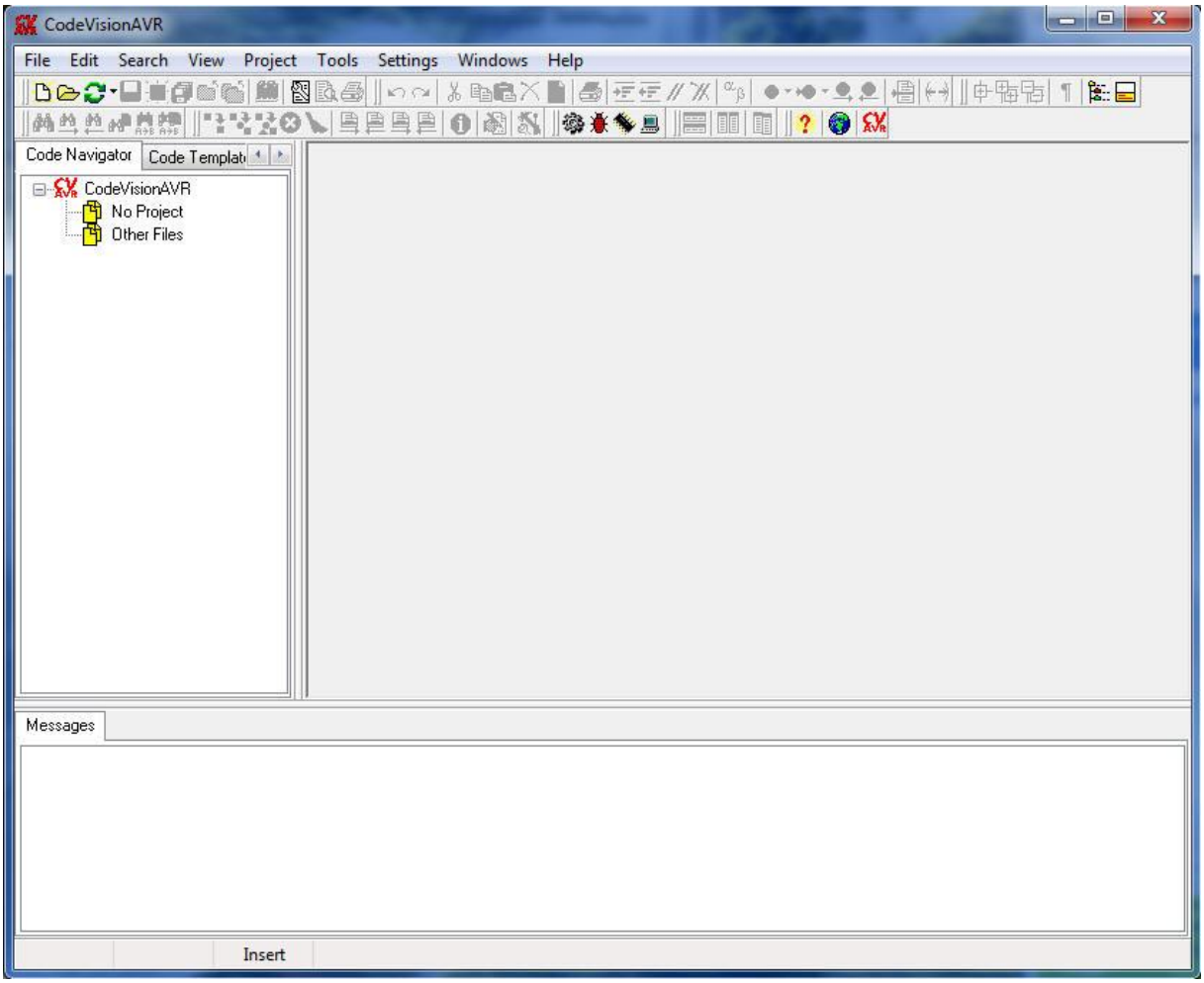
Proměnná ACC (akumulátor) je definována v hlavičkovém souboru v části registry takto
sfr ACC = 0xE0;
`dat_1, dat_2, dat_3 a dat_4` jsou byte pole obsahující vždy 4 prvky. Tyto prvky jsou tvořeny 8 bity. Každé z těchto polí tedy obsahuje celkem 32 bitů dat, kterými jsou řídící slova pro *ADF7012*. Protože 89x51 je osmibitový, rozdělili jsme tyto řídící slova na 4 x 8 = 32 bitů. K odvysílání těchto řídících slov jsme naprogramovali funkci `write_reg` , která jako parametr má ukazatele na čtyřprvkové pole bytů. V hlavní funkci `main` jsou nejprve nastaveny logické úrovně pro klidový stav signálů le, clk a dat (viz naše předchozí analýza či firemní dokumentace k *ADF7012*). Poté je 4 x volána funkce `write_reg` Nejprve s parametrem `dat_1`, poté s `dat_2`, dále s `dat_3` a nakonec s `dat_4`. Tím jsou odvysílány řídící slova do všech čtyř registrů *ADF7012*. Program pak běží v nekonečné smyčce `while(1);` Smyčka je zatím prázdná. Později by měla obsahovat kód pro příjem příkazů přes **RxD** a jeho interpretaci, jako např. tvorbu a vysílání AX25 paketů.
Ještě si vysvětlíme kód funkce `write_reg`, jehož pochopení je důležité a je využitelné i pro jiné 8bitové procesory:
Příkazem `le=0;` na začátku a `le=1;` na konci kódu této funkce vytváříme signál LE (Load Enable) pro *ADF7012*, který má byt na úrovni 0, po dobu vysílání dat **DATA** a hodin **CLK** do *ADF7012*. Dále obsahuje kód dvě vložené smyčky for. Vnější smyčka s řídící smyčkou j zabezpečuje , že se 4 x bude opakovat vnitřní kód . Neboli 4x se bude vysílat 8 hodinových pulzů pro hodiny **CLK** a 8 datových bitů signálu **DATA**. Vnitřní kód na svém začátku pošle příkazem `ACC=dat1[j];`do akumulátoru příslušných 8 bitů z 32 bitového řídícího slova. Pro j=1 je to prvních 8bitů, pro j=2 druhých 8 bitů atd.
Poté následuje příkaz for s řídící proměnnou i, který 8x opakuje svůj vnitřní kód. Ten na svém začátku pošle nejvíce významný bit (bit 7) do proměnné dat. Poté příkaz `ACC=ACC<<1;` provede posun obsahu akumulátoru ACC o jedno místo doleva, čímž se v bitu 7 akumulátoru objeví další bit řídícího slova. V proměnné dat je ovšem stále předchozí bit. Dále se příkazem `clk=1;` nastaví úroveň signálu CLK na 1, a protože vně smyčky for je clk=0, jde o vzestupnou hranu hodin CLK v době, kdy na výstupu DATA je již správná úroveň dat. Dalším příkazem `clk=0;` se ukončí hodinový pulz CLK. Protože vnitřní cyklus for proběhne 8x, vygeneruje se takto 8 datových bitů. Z tohoto popisu je také zřejmé, proč jsme při analýze řídících signálů logickým analyzátorem viděli „shluky“ osmic hodinových pulzů a mezi nimi časové prodlevy – ačkoli nemáme zdrojový kód *PrattHobbies* vysílače *ATMega88* je zřejmé, že řídící signály pro *ADF7012* se budou vytvářet obdobným způsobem.
Program jsem odzkoušel i s *AT89S8252* Atmel. Liší se pouze prvním řádkem, který je nyní

```
#include"AT898252.H"
```

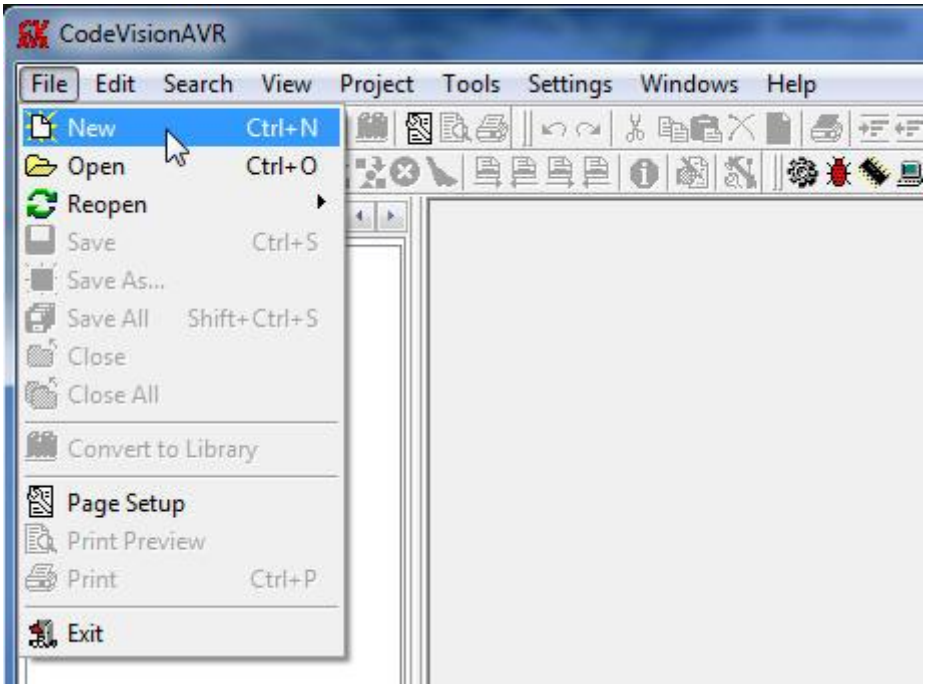
K naprogramování programové flash paměti můžeme použít *PonyProg2000* a stejné programátory, které jsme použili pro *ATMega88*.

6.1.3.2.2 Jednočipy ATMEL AVR (ATMEL AVR single chip computers)

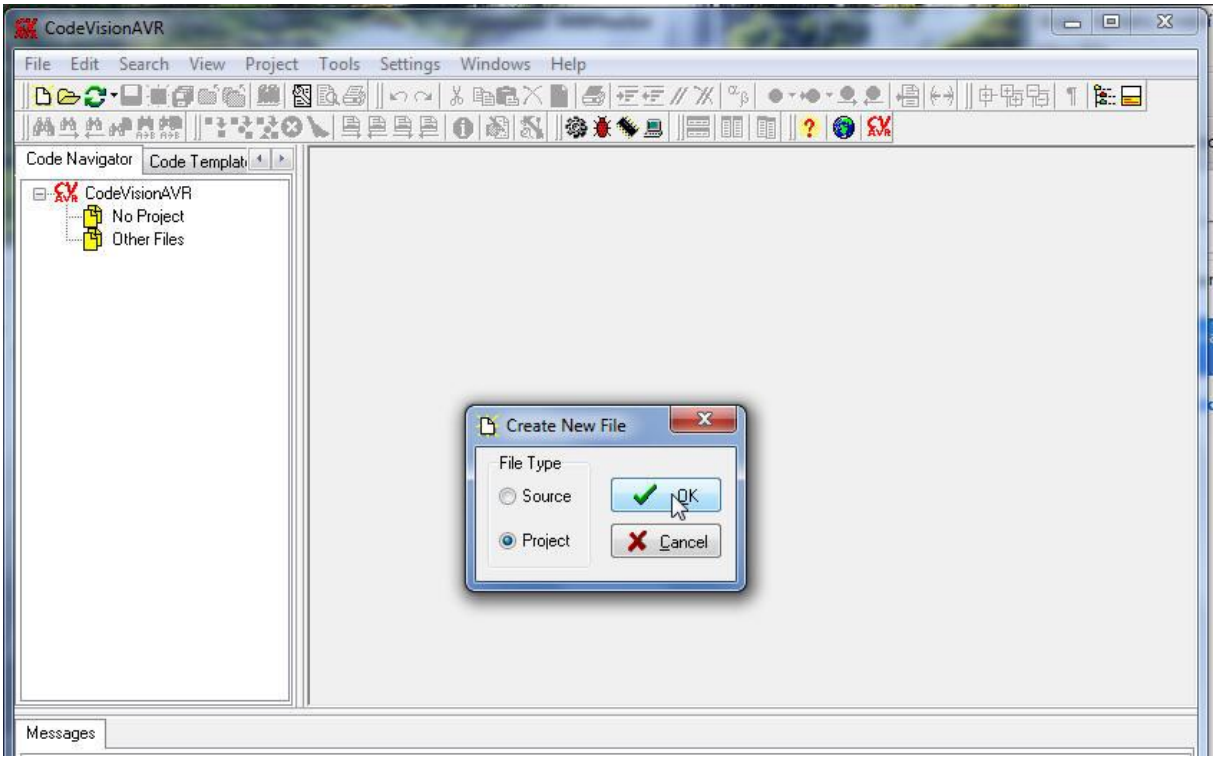
Mezi tyto jednočipy patří i ATMega88 použitá ve vysílači Pratt Hobies. Pokud s AVR začínáme, můžeme mít u tohoto jednočipu problémy s pojistkami (fuses) a v případě jejich nesprávného použití způsobit, že s tímto jednočipem již nebudeme moci pracovat.
Pro může být, alespoň při prvních pokusech s AVR, výhodné použít některý z členů rodiny AVR, který pojistky nemá. Základní instrukční soubor je stejný u všech členů rodiny AVR a program napsaný pro některý ze členů rodiny můžeme snadno upravit pro jiný AVR. Práci s ATMEL AVR si ukážeme s AT90S8515 na startkitu z [1]. Naprogramuje posílání řídících signálů CLK, DATA, LE do ADF7012 do 32bitových registrů tohoto obvodu. Program napíšeme v jazyce C v prostředí CodeVisionAVR [1], jehož free demoverze pro naše účely je plně postačující. Součástí CodeVisionAVR je wizard, který nám tvorbu programu usnadní a s jehož pomocí snadno upravíme program i pro jiné procesory AVR a jiné hodinové kmitočty. Po startu CodeVisioAVR se objeví úvodní okno programu:



V menu vyfereme **File - - New**

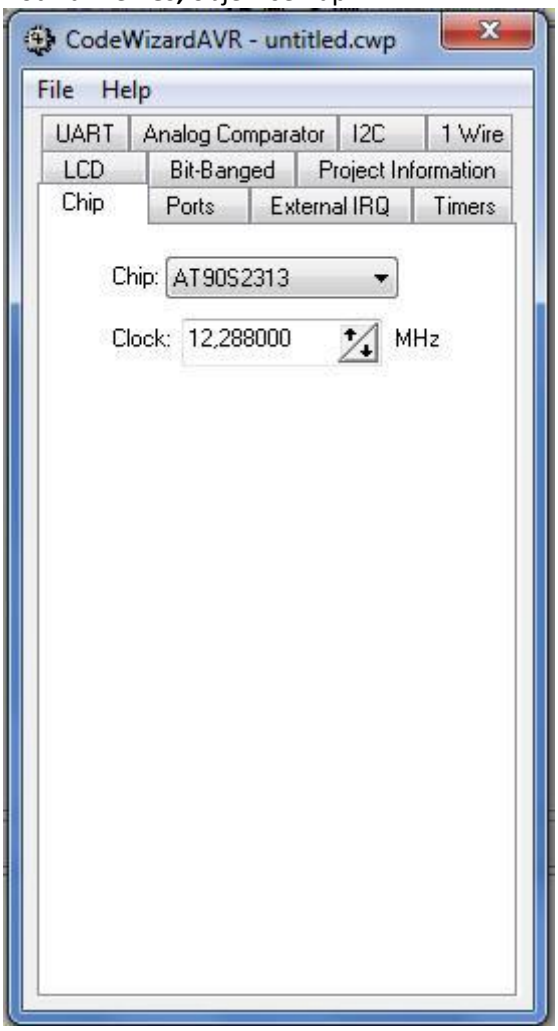


Objeví se

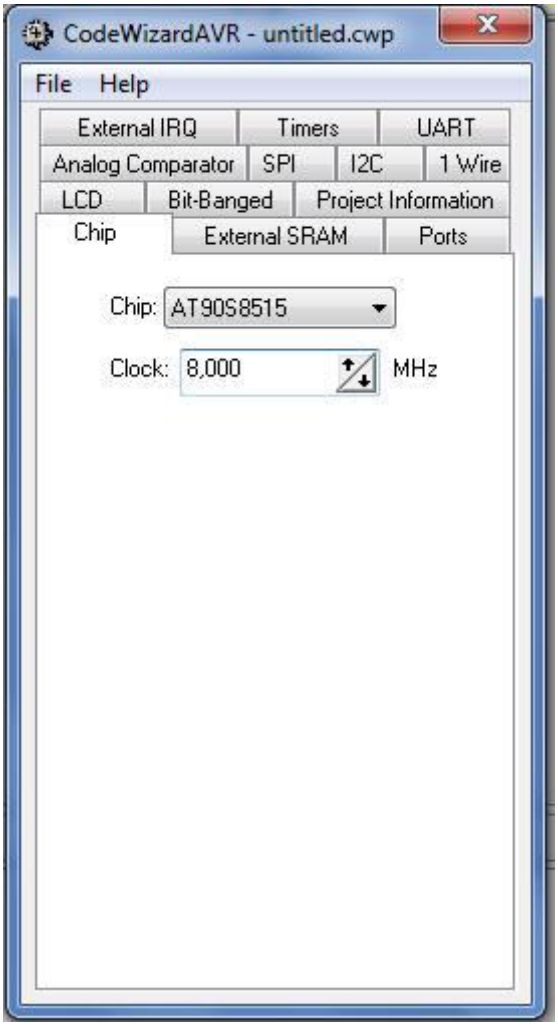


Vybereme Project a potvrdíme **OK**. Dostanem

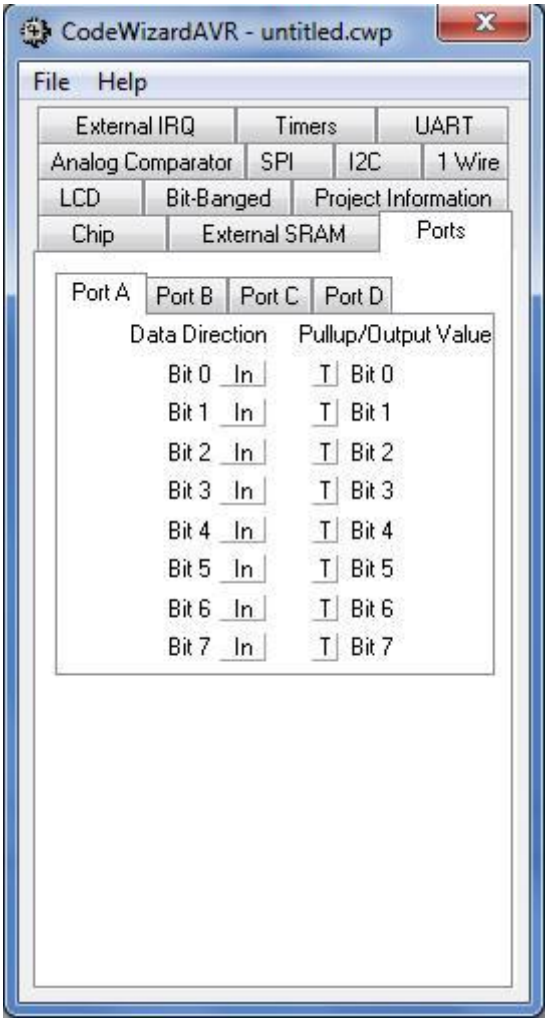
Potvrdíme **Yes,** objeví se např.



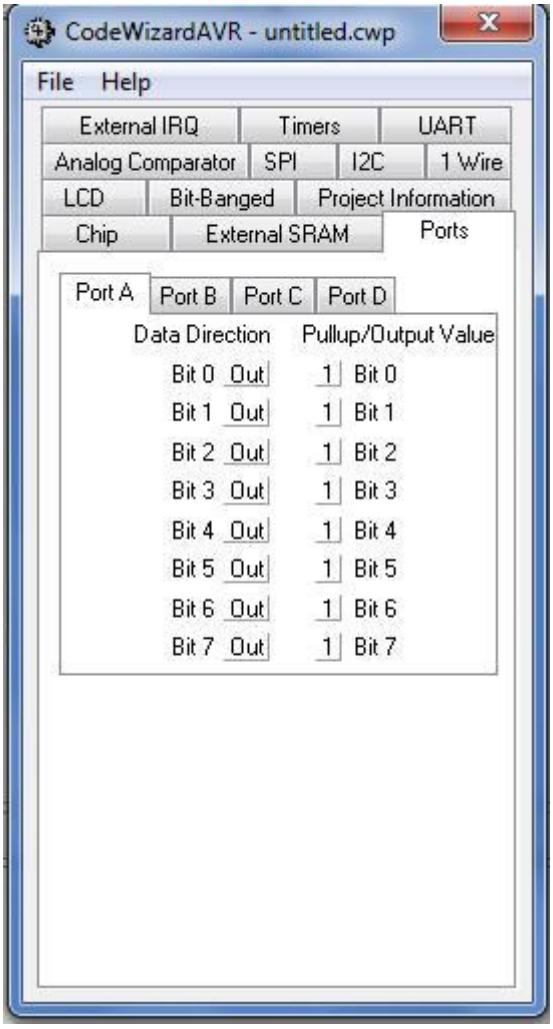
Nastavíe typ procesoru a jeho hodinový kmitočet (kmitočet krystalu)



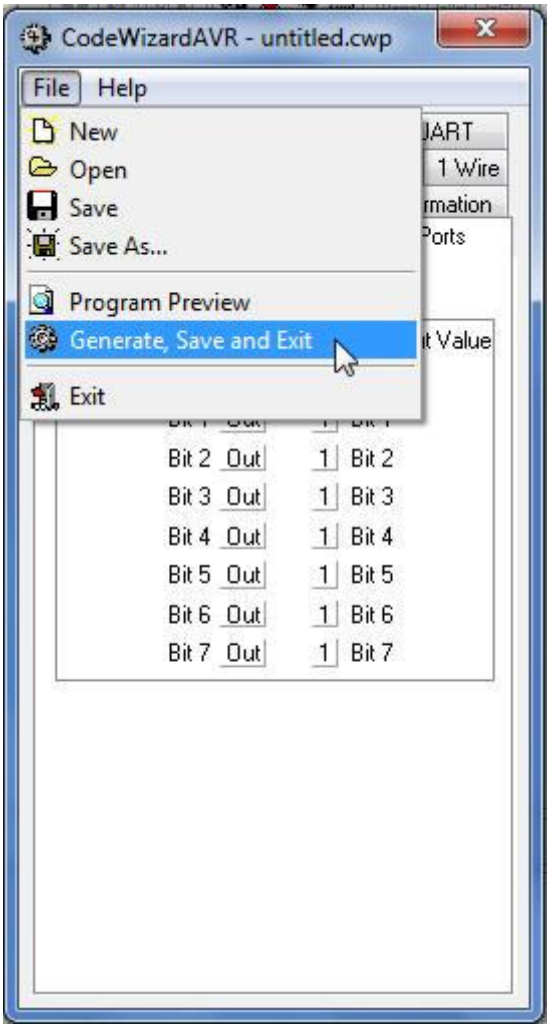
Přepneme na záložku **Ports**



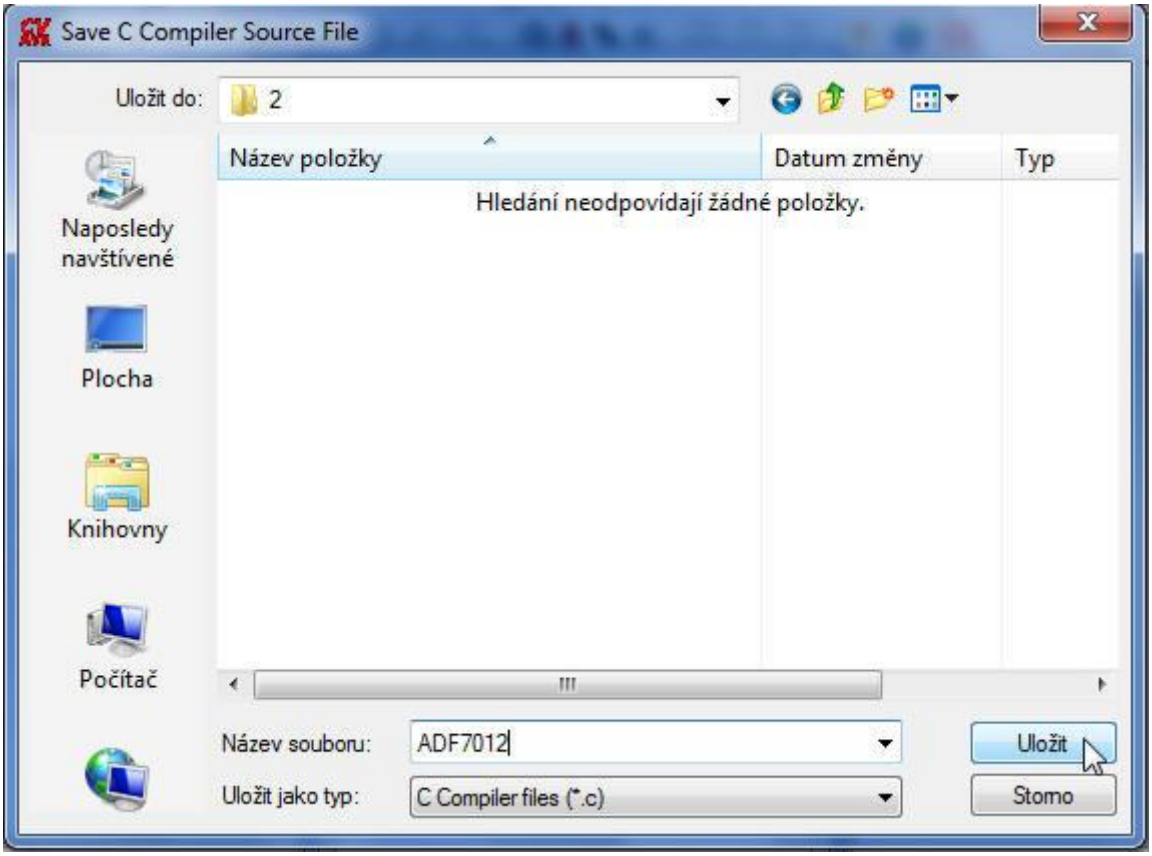
Piny Portu A nastavíme jako výstupní.



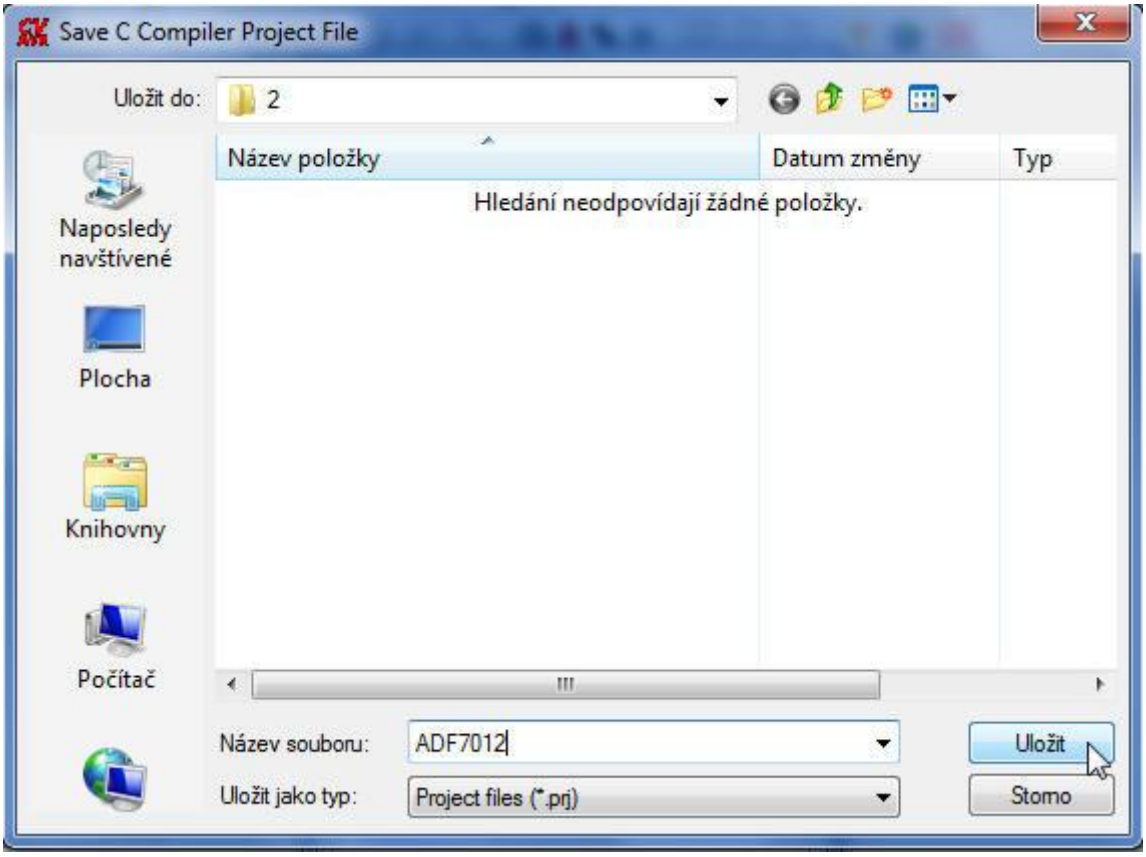
V menu výše uvedeného okna vybereme **File --- Generate, Save end Exit**



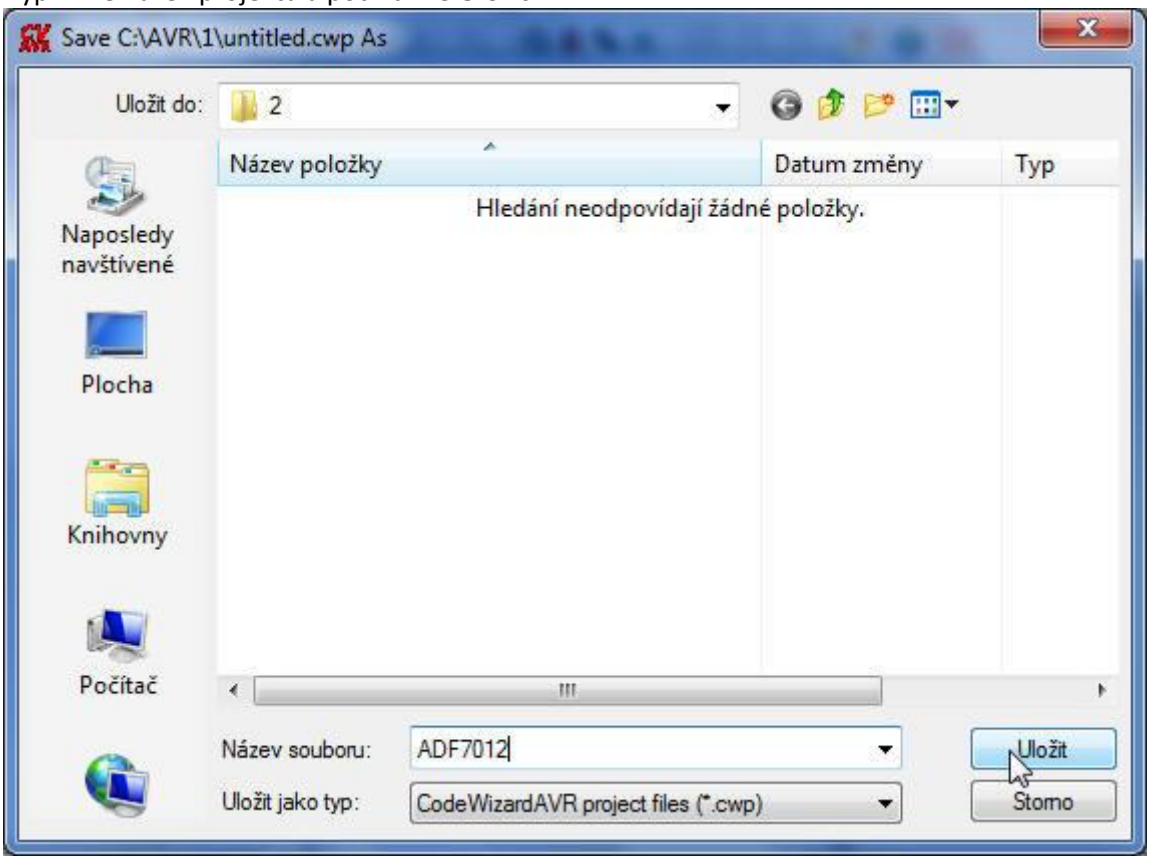
Vyplníme název souboru se zdrojovým kódem



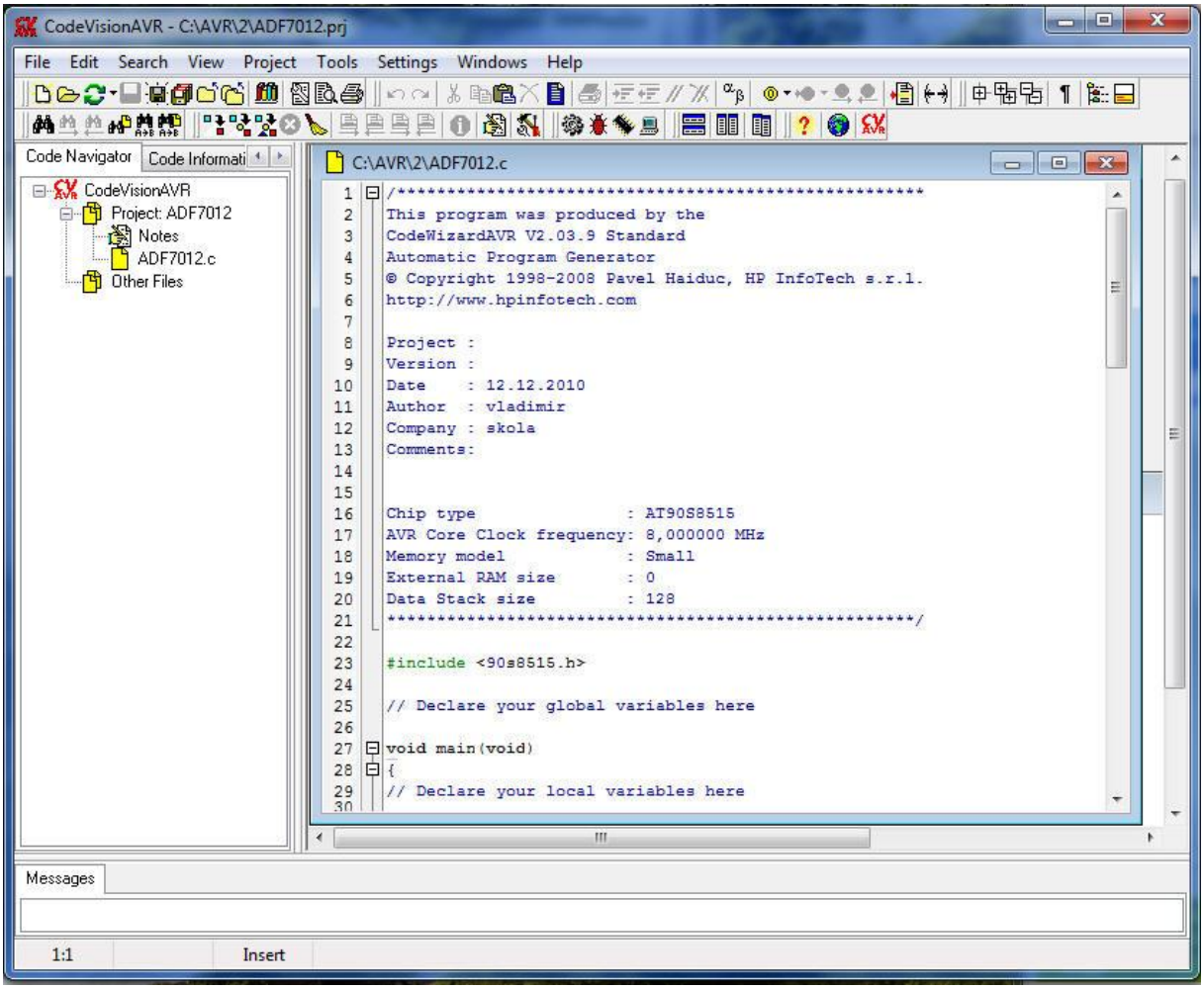
Klikneme na **Uložit**



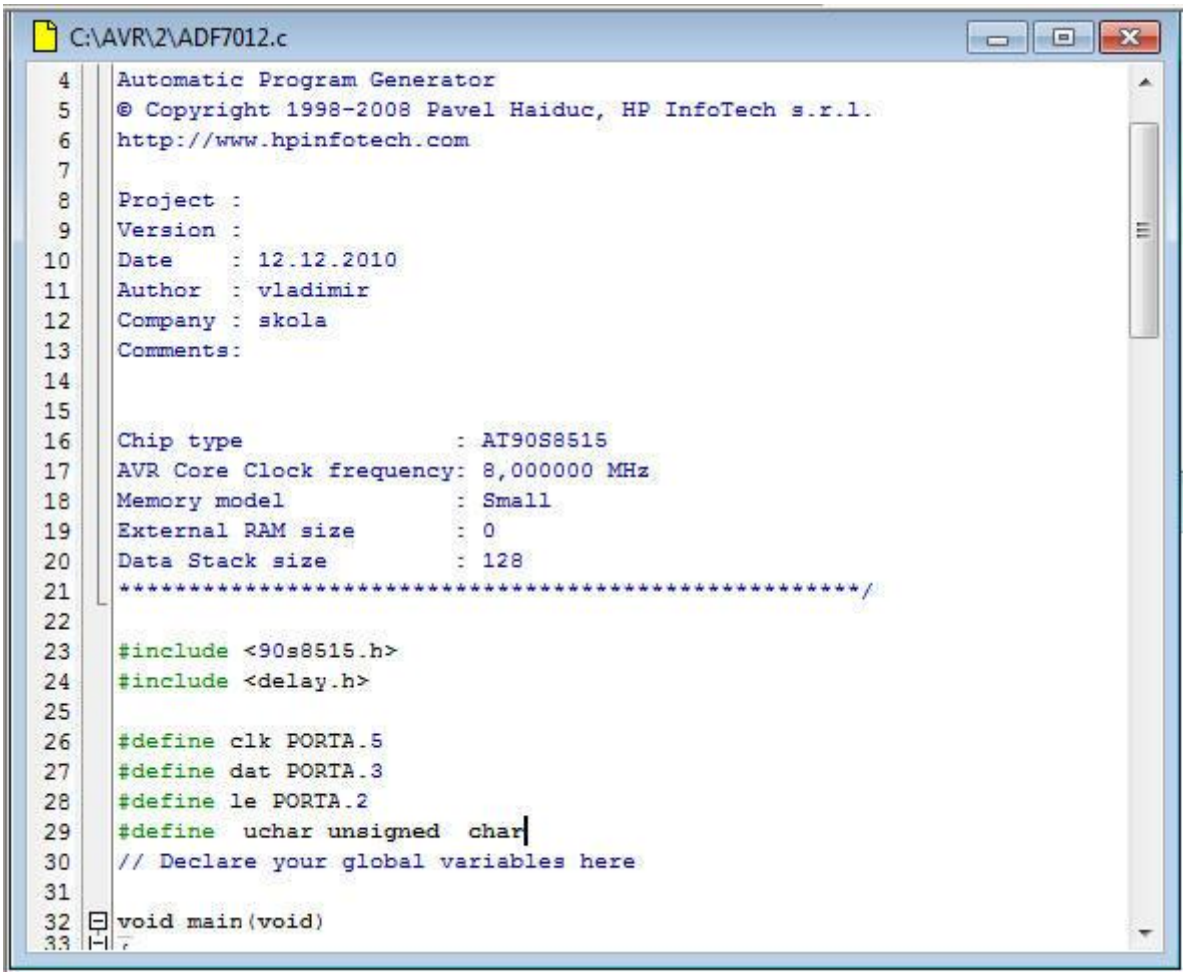
Vyplníme název projektu a potvrdíme **Uložit**.



Opět vyplníme název souboru klikneme na **Uložit**. Objeví se kostra kódu



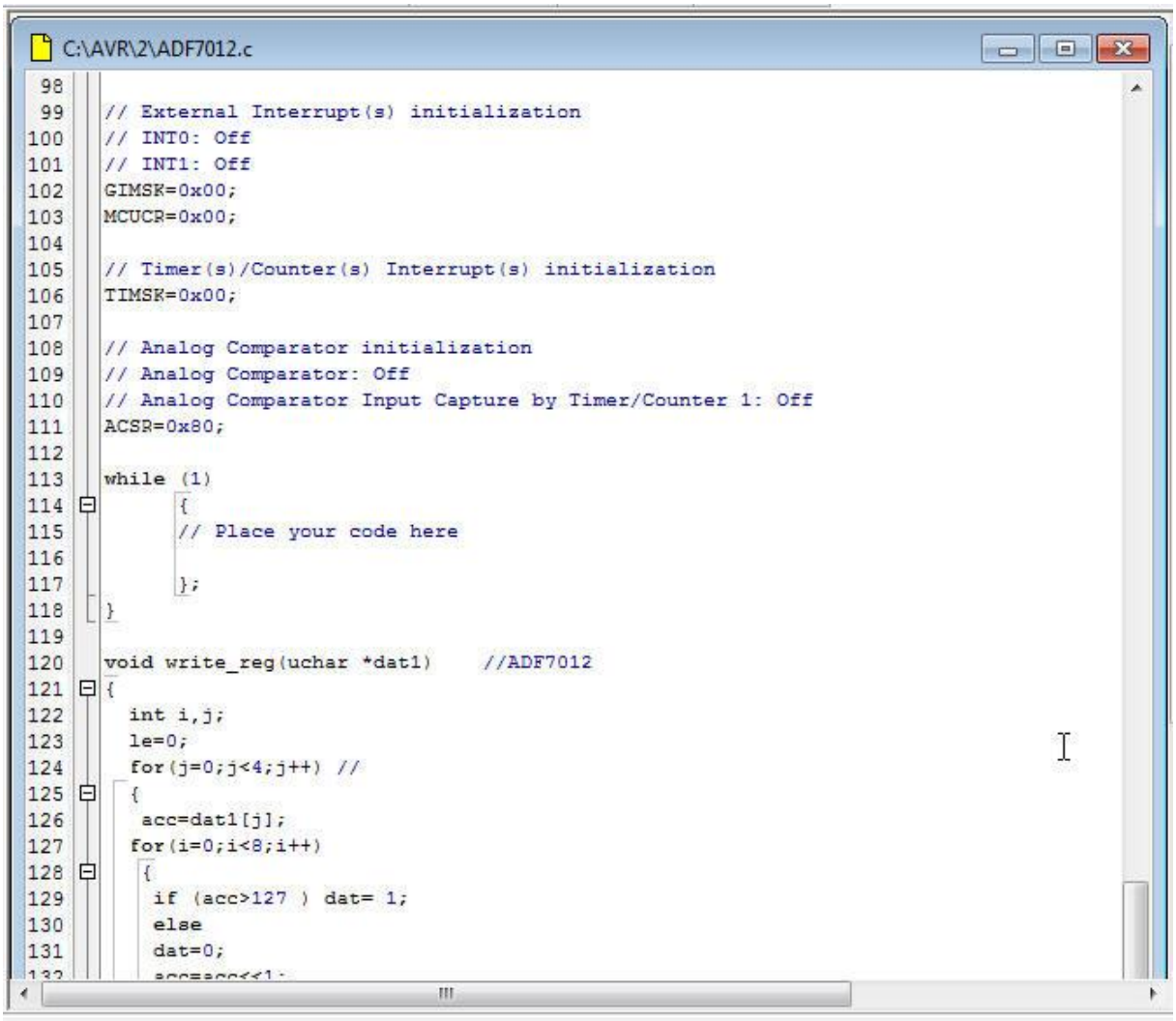
Přidáme hlavičkový soubor a definice


C:\AVR\2\ADF7012.c
Automatic Program Generator
© Copyright 1998-2008 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com
Project :
Version :
Date : 12.12.2010
Author : vladimir
Company : skola
Comments:
Chip type : AT90S8515
AVR Core Clock frequency: 8,000000 MHz
Memory model : Small
External RAM size : 0
Data Stack size : 128
*****/
#include <90s8515.h>
#include <delay.h>
#define clk PORTA.5
#define dat PORTA.3
#define le PORTA.2
#define uchar unsigned char
// Declare your global variables here
void main(void)

C:\AVR\2\ADF7012.c

```
19  External RAM size       : 0
20  Data Stack size         : 128
21  *****************************************************/
22
23  #include <90s8515.h>
24  #include <delay.h>
25
26  #define clk PORTA.5
27  #define dat PORTA.3
28  #define le PORTA.2
29  #define  uchar unsigned  char
30  // Declare your global variables here
31
32  #include <stdio.h>
33
34  // Declare your global variables here
35  uchar acc;
36  uchar dat_1[]={0x02,0x08,0x5E,0x10}; //R Register  naplnime  0x02085E10
37  uchar dat_2[]={0x00,0x08,0xD4,0x01}; //N-Counter Latch naplnime 0x0008D401
38  uchar dat_3[]={0x00,0x00,0x37,0xE2}; //Modulation Register naplnime 0x000037E2
39  uchar dat_4[]={0x00,0x5A,0xA0,0x57}; //Function Register naplnime 0x005AA057
40
41  void write_reg(uchar *dat1);
42
43  void main(void)
44  {
45  // Declare your local variables here
46
47  // Input/Output Ports initialization
```

Postupně doplňujeme zdrojový kód

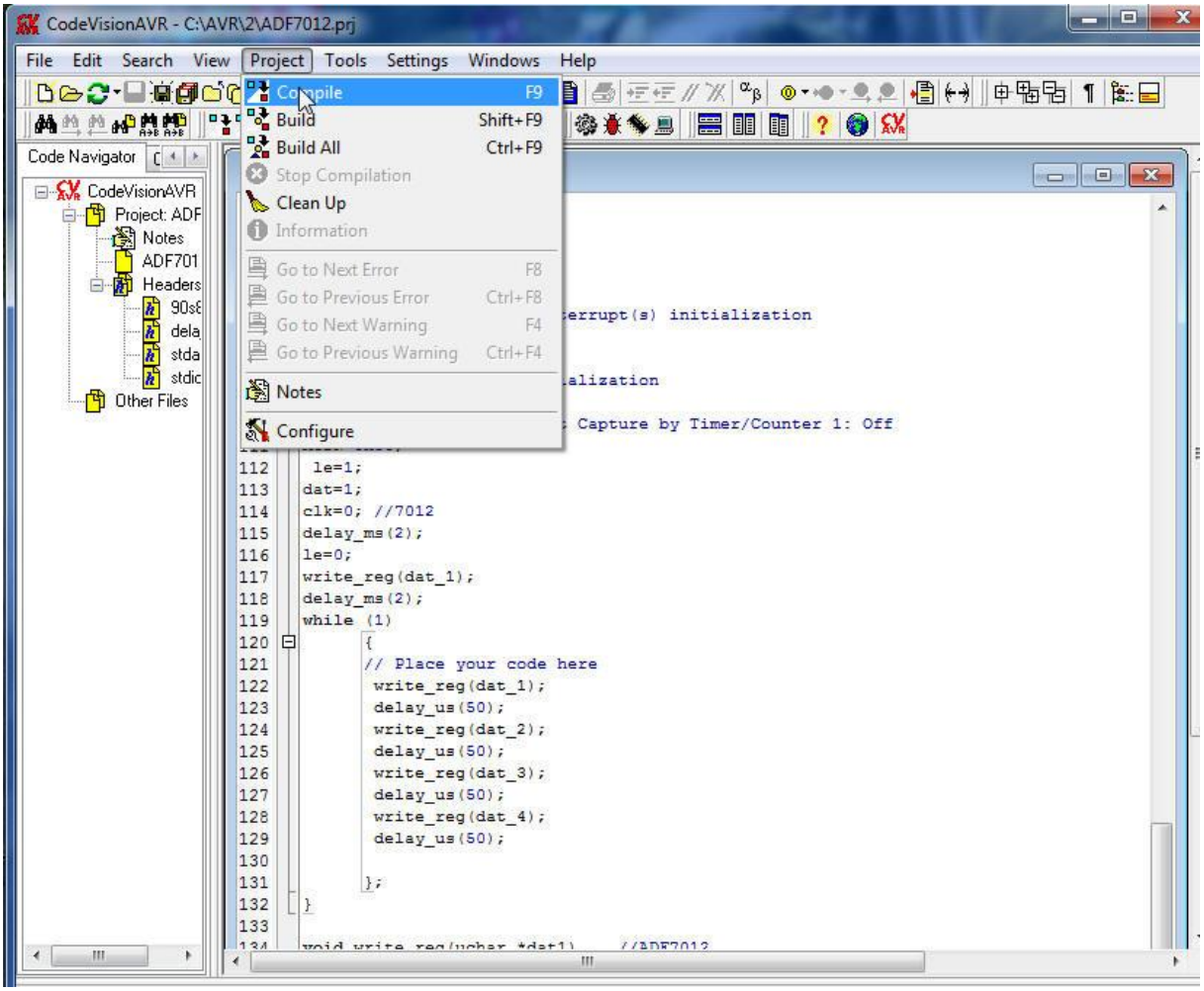

C:\AVR\2\ADF7012.c

```
 98
 99  // External Interrupt(s) initialization
100  // INT0: Off
101  // INT1: Off
102  GIMSK=0x00;
103  MCUCR=0x00;
104
105  // Timer(s)/Counter(s) Interrupt(s) initialization
106  TIMSK=0x00;
107
108  // Analog Comparator initialization
109  // Analog Comparator: Off
110  // Analog Comparator Input Capture by Timer/Counter 1: Off
111  ACSR=0x80;
112
113  while (1)
114        {
115        // Place your code here
116
117        };
118  }
119
120  void write_reg(uchar *dat1)     //ADF7012
121  {
122    int i,j;
123    le=0;
124    for(j=0;j<4;j++) //
125    {
126     acc=dat1[j];
127    for(i=0;i<8;i++)
128     {
129      if (acc>127 ) dat= 1;
130      else
131      dat=0;
132      acc=acc<<1;
```

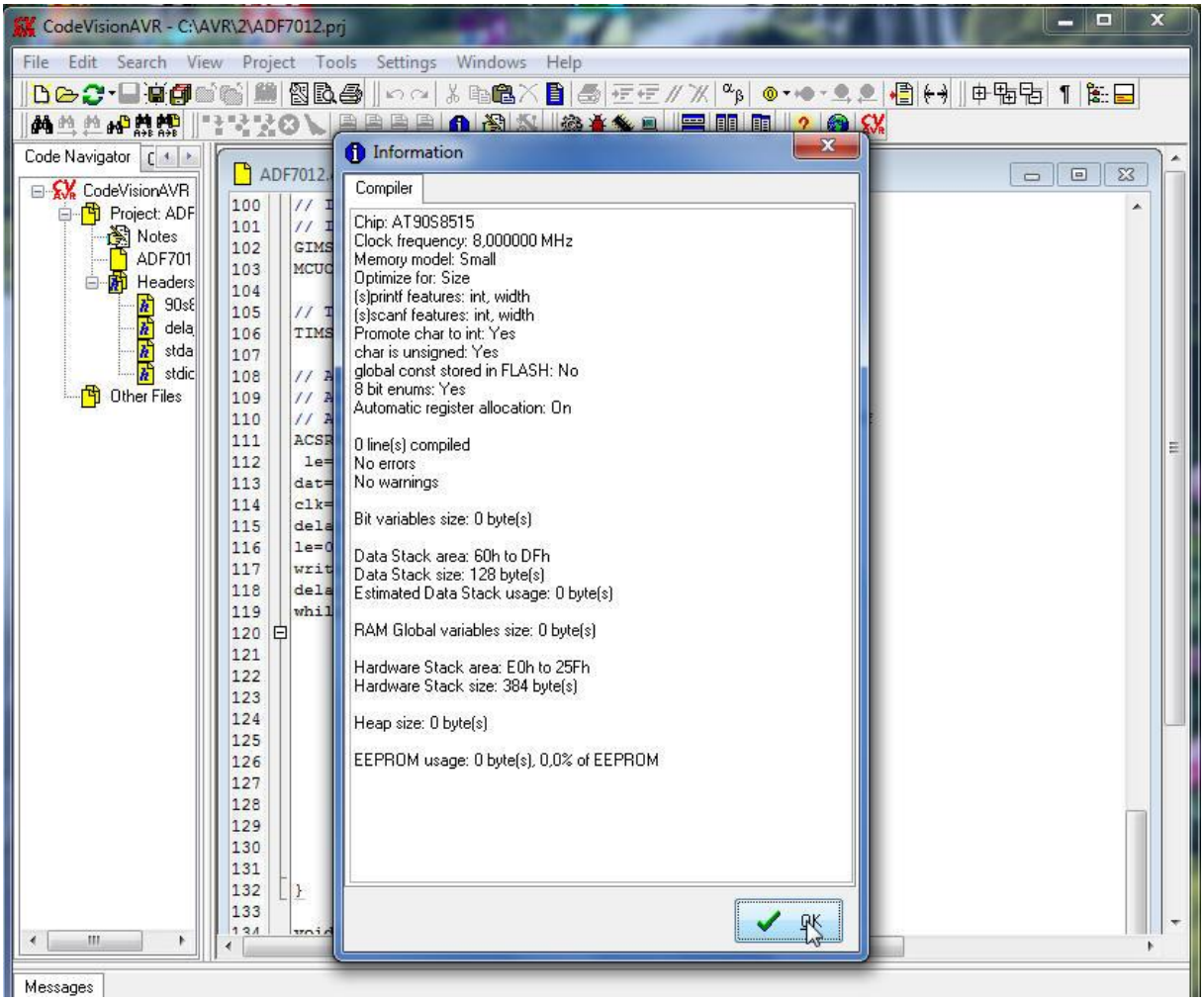
C:\AVR\2\ADF7012.c

```
 98
 99  // External Interrupt(s) initialization
100  // INT0: Off
101  // INT1: Off
102  GIMSK=0x00;
103  MCUCR=0x00;
104
105  // Timer(s)/Counter(s) Interrupt(s) initialization
106  TIMSK=0x00;
107
108  // Analog Comparator initialization
109  // Analog Comparator: Off
110  // Analog Comparator Input Capture by Timer/Counter 1: Off
111  ACSR=0x80;
112   le=1;
113  dat=1;
114  clk=0; //7012
115  delay_ms(2);
116  le=0;
117  write_reg(dat_1);
118  delay_ms(2);
119  while (1)
120        {
121        // Place your code here
122         write_reg(dat_1);
123         delay_us(50);
124         write_reg(dat_2);
125         delay_us(50);
126         write_reg(dat_3);
127         delay_us(50);
128         write_reg(dat_4);
129         delay_us(50);
130
131        };
```

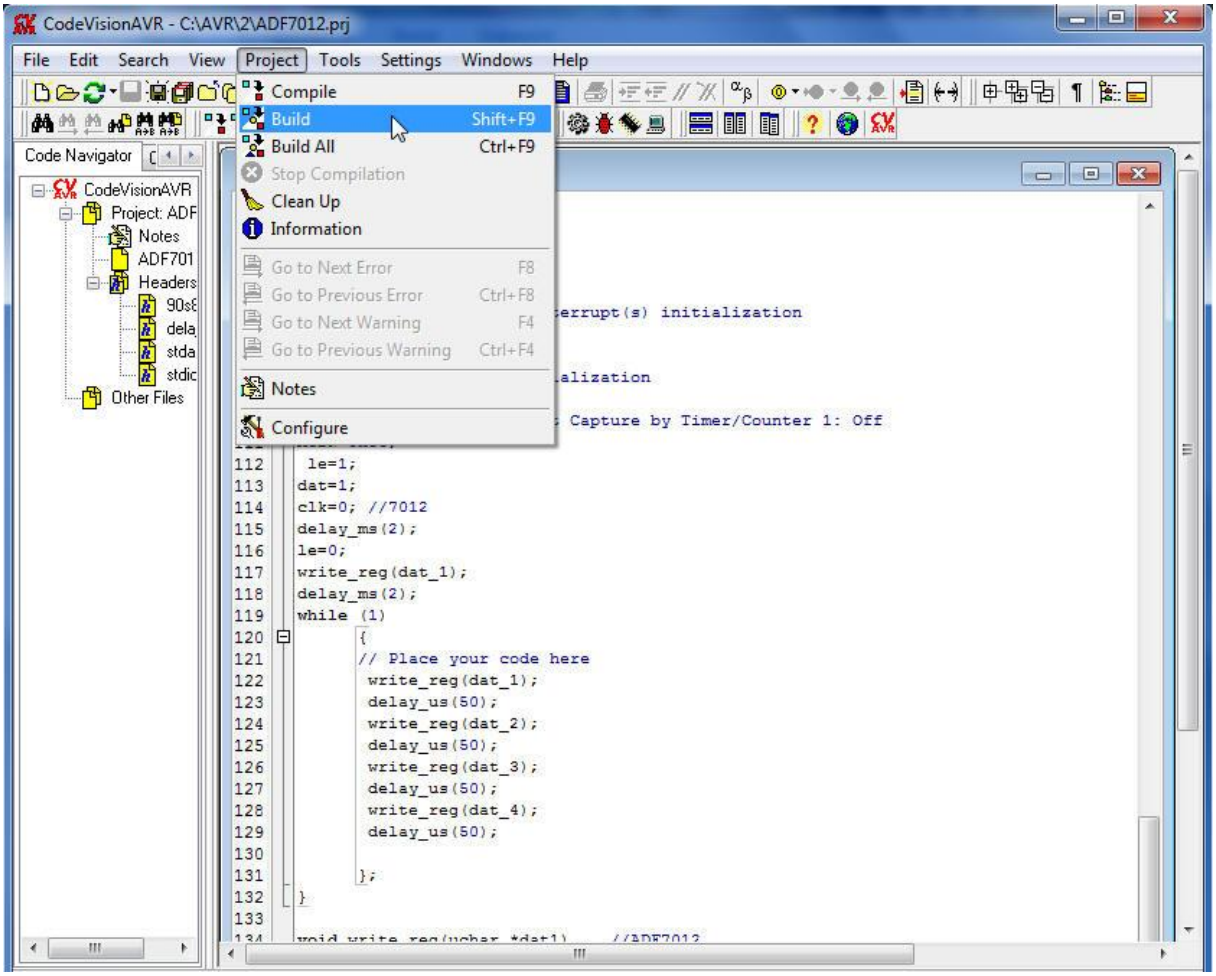
Zdrojový kód v jazyce C přeložíme výběrem v menu **Project – Compile**, dostaneme

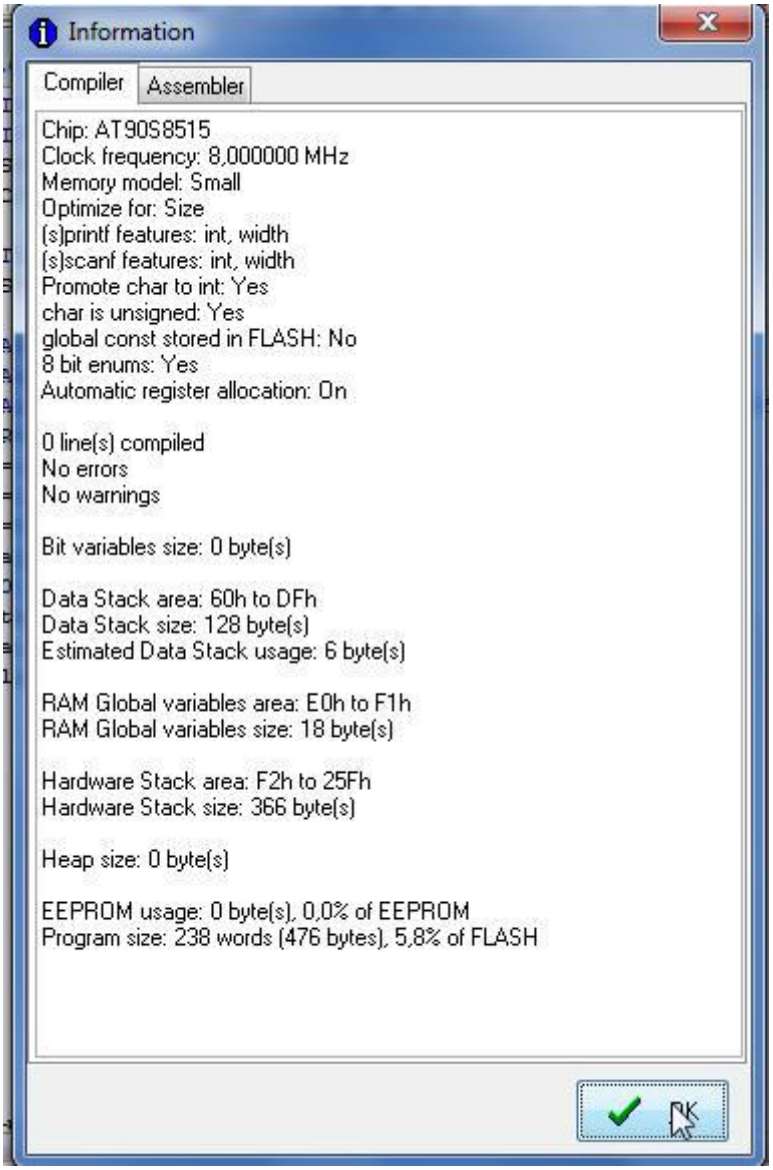


Provedeme překlad



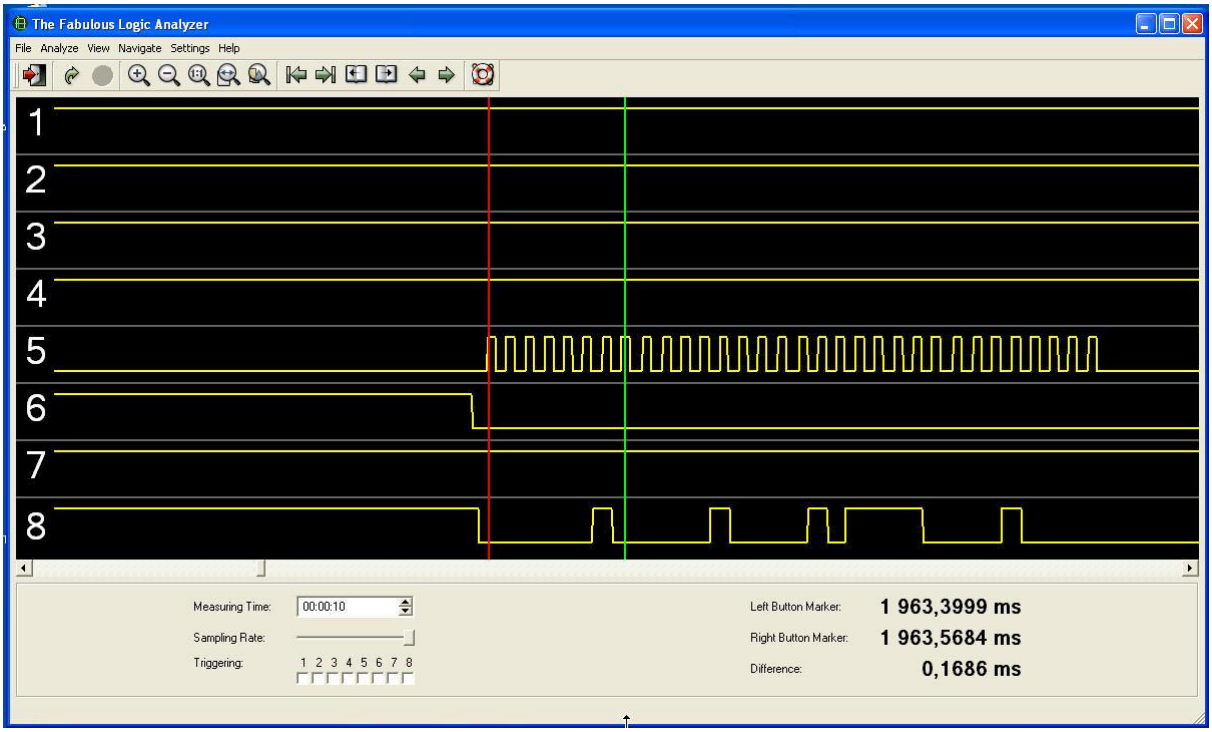
Překlad se podařil. Klikneme na tlačítko **OK**.
Zbývá provést linkování s knihovnami, spuštění make atd., To vše se provede výběrem v menu **Project – Build**


CodeVisionAVR - C:\AVR\2\ADF7012.prj
File Edit Search View Project Tools Settings Windows Help
Compile F9
Build Shift+F9
Build All Ctrl+F9
Stop Compilation
Clean Up
Information
Go to Next Error F8
Go to Previous Error Ctrl+F8
Go to Next Warning F4
Go to Previous Warning Ctrl+F4
Notes
Configure
Code Navigator
CodeVisionAVR
Project: ADF
Notes
ADF701
Headers
Other Files
112 le=1;
113 dat=1;
114 clk=0; //7012
115 delay_ms(2);
116 le=0;
117 write_reg(dat_1);
118 delay_ms(2);
119 while (1)
120 {
121 // Place your code here
122 write_reg(dat_1);
123 delay_us(50);
124 write_reg(dat_2);
125 delay_us(50);
126 write_reg(dat_3);
127 delay_us(50);
128 write_reg(dat_4);
129 delay_us(50);
130
131 };
132 }
133
Dostaneme

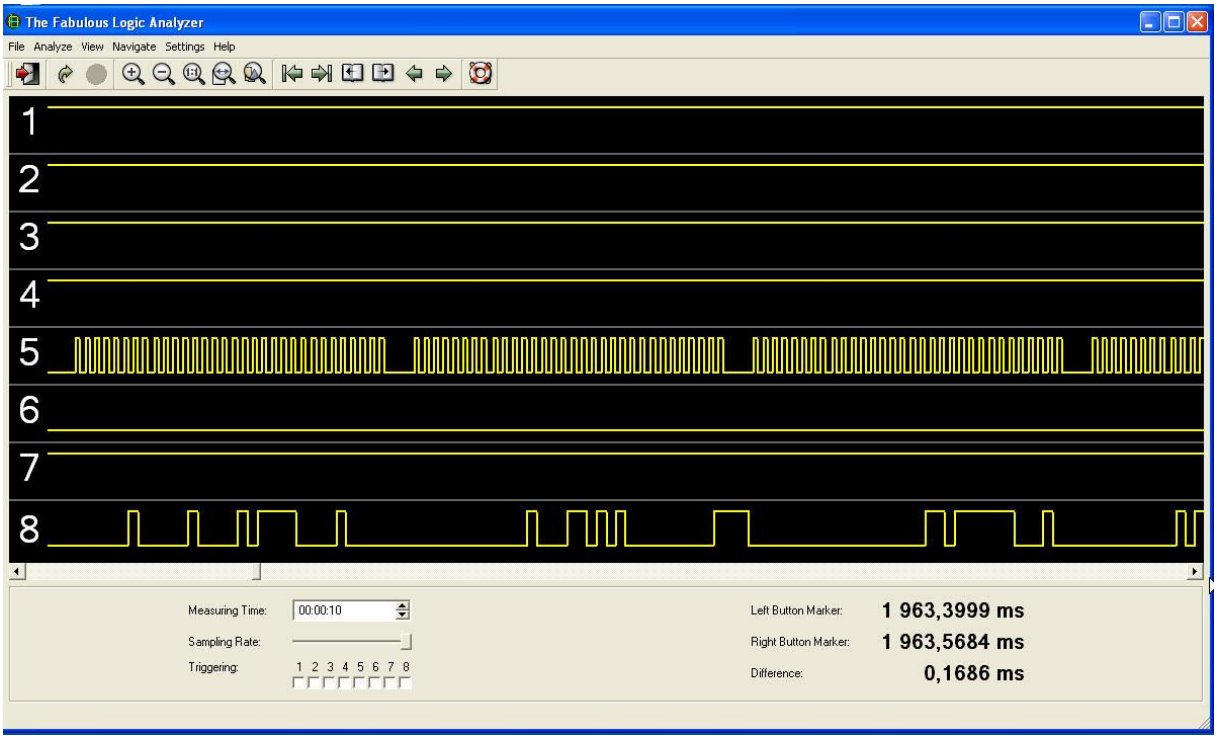


Potvrdíme **OK**. V adresáři projektu máme i HEX soubor, který použeje programátor (např. PonyProg2000) k naprogramování pameti flash počítače AT90S8515

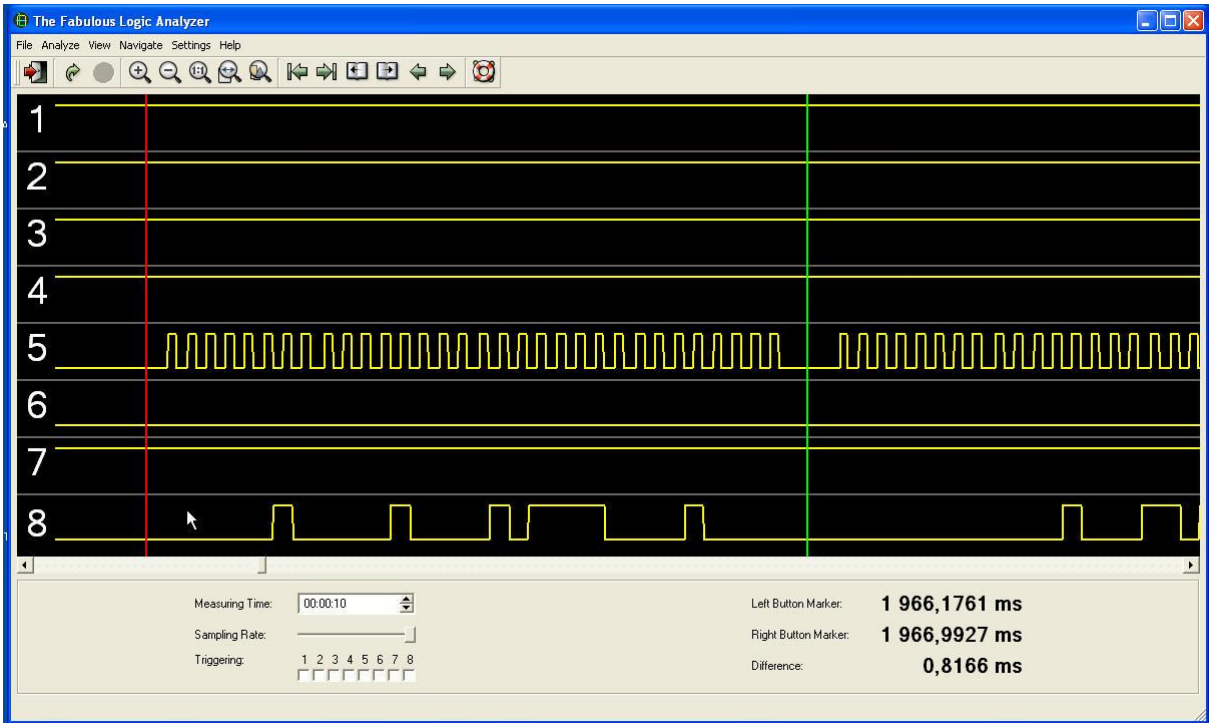
Zbývá pomocí logického analyzátoru zkontrolovat signály generované AT90S8515. Nejprve je odvysíláno 32 bitů do registru 0.

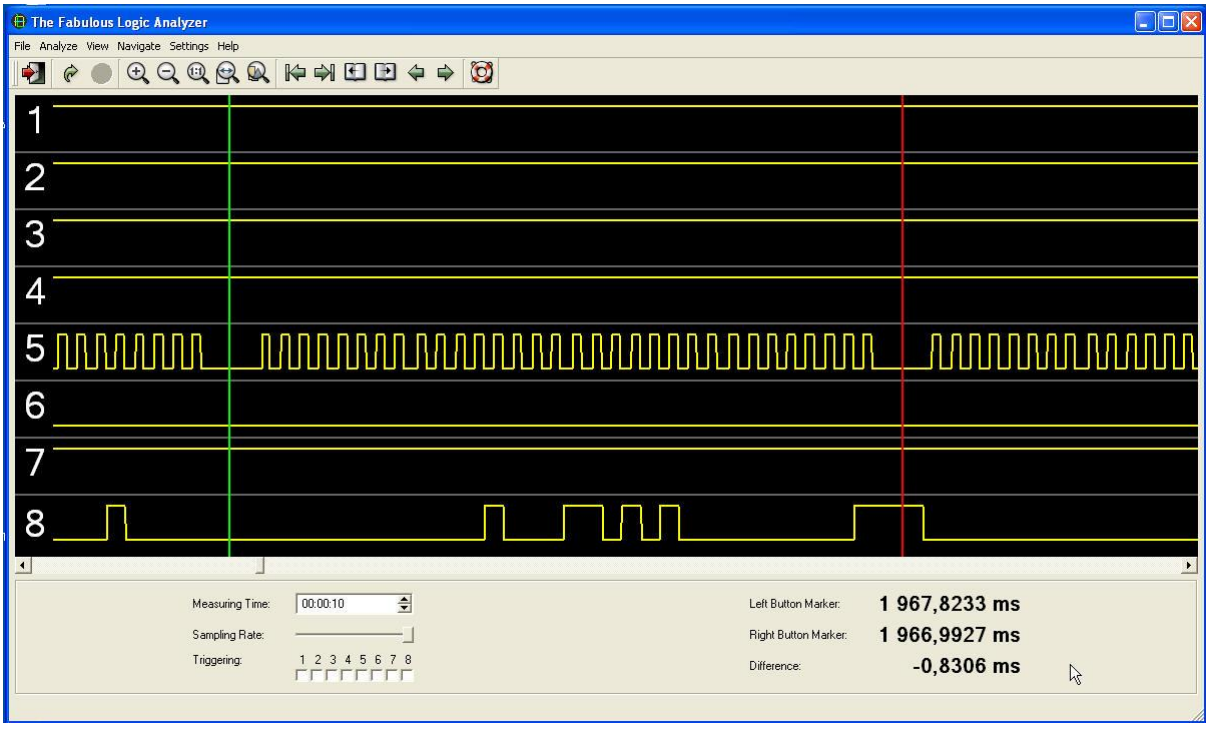


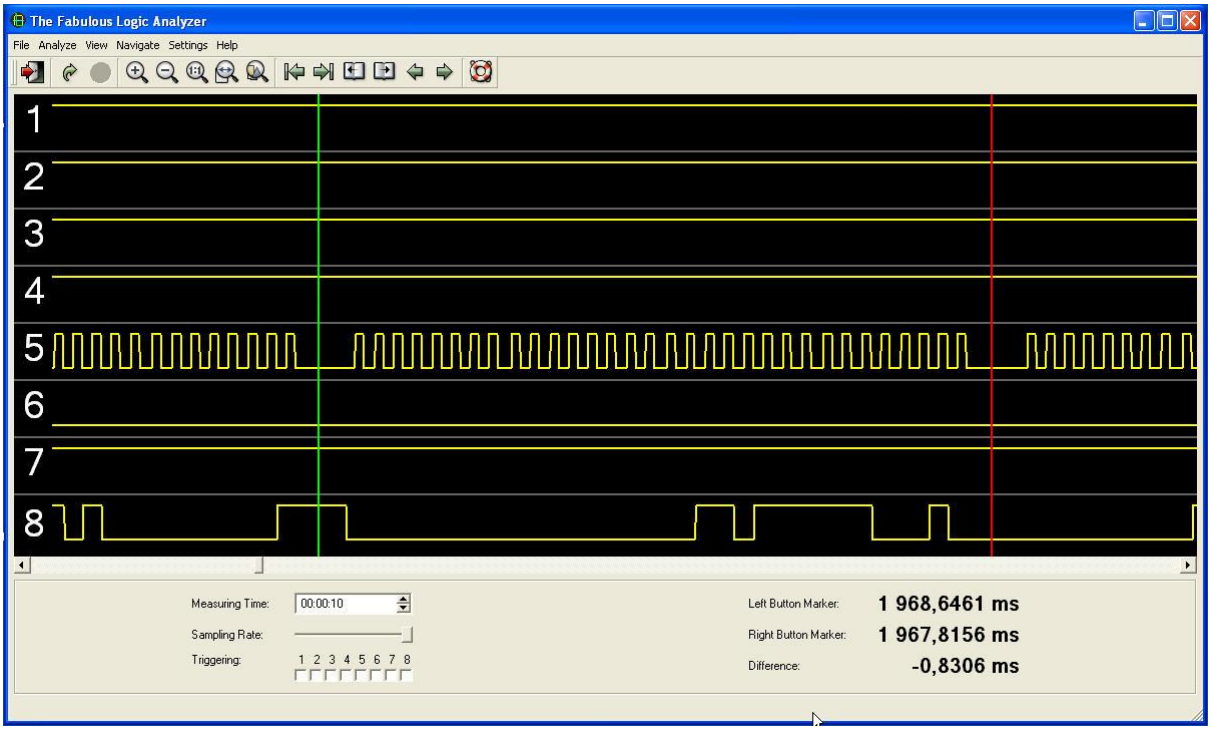
Poté jsem naprogramoval prodlevu a dále v smyčce while (1) čtveřici 32bitových řídících slov – stejná, jako v inicializační části firmware ATMega88 vysílače Pratthobies

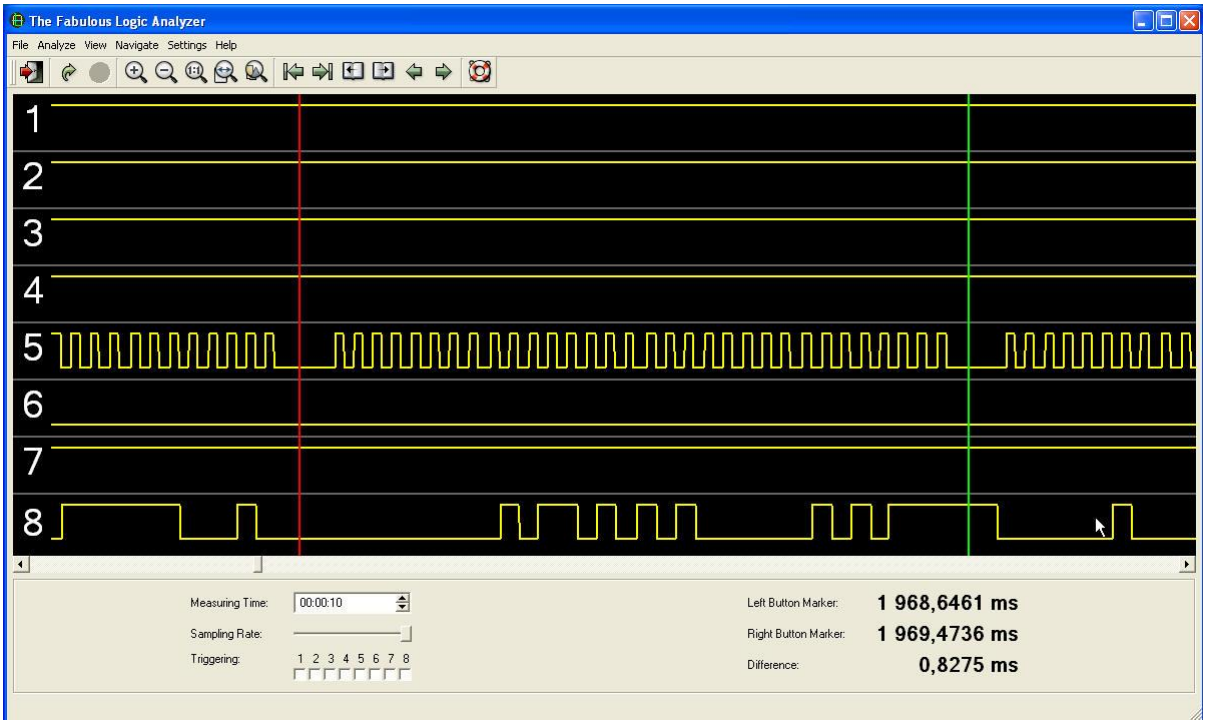


Zvětšíme zobrazení logického analyzátoru


The Fabulous Logic Analyzer
File Analyze View Navigate Settings Help
1
2
3
4
5
6
7
8
Measuring Time: 00:00:10
Sampling Rate:
Triggering: 1 2 3 4 5 6 7 8
Left Button Marker: 1 966,1761 ms
Right Button Marker: 1 966,9927 ms
Difference: 0,8166 ms


The Fabulous Logic Analyzer
File Analyze View Navigate Settings Help
1
2
3
4
5
6
7
8
Measuring Time: 00:00:10
Sampling Rate:
Triggering: 1 2 3 4 5 6 7 8
Left Button Marker: 1 967,8233 ms
Right Button Marker: 1 966,9927 ms
Difference: -0,8306 ms





Takže jsme si ověřili, že náš program pracuje tak, jak má. Zbývá popsat si náš zdrojový kód v jazyce C pro procesory ATMEL AVR. S drobnými úpravami ho použijeme i pro 32bitové procesory s jádrem ARM7.

```
/*****************************************************
This program was produced by the
```

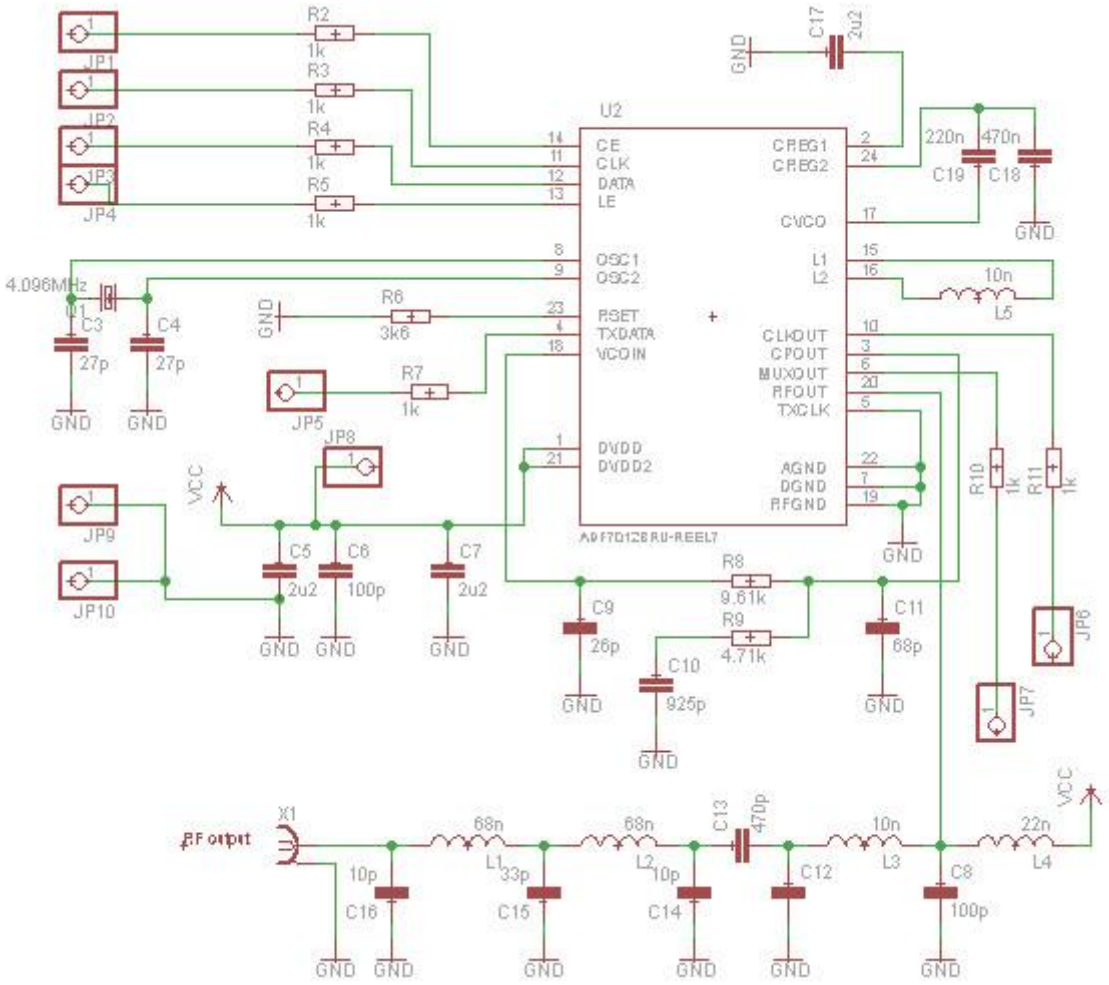
```
CodeWizardAVR V2.03.9 Demo
Automatic Program Generator
© Copyright 1998-2008 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 12.12.2010
Author  : vladimir
Company : skola
Comments:


Chip type               : AT90S8515
AVR Core Clock frequency: 8,000000 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 128
*****************************************************/

#include <90s8515.h>
#include <delay.h>

#define clk PORTA.5
#define dat PORTA.3
#define le PORTA.2
#define  uchar unsigned  char
// Declare your global variables here

#include <stdio.h>

// Declare your global variables here
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 naplnime
0x000037E2
uchar dat_4[]={0x00,0x5A,0xA0,0x57}; //Function Register naplnime
0x005AA057

void write_reg(uchar *dat1);

void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out
Func0=Out
// State7=1 State6=1 State5=1 State4=1 State3=1 State2=1 State1=1 State0=1
PORTA=0xFF;
DDRA=0xFF;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTB=0x00;
DDRB=0x00;
```

```
// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
TCCR0=0x00;
TCNT0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
GIMSK=0x00;
MCUCR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
 le=1;
dat=1;
clk=0; //7012
delay_ms(2);
le=0;
write_reg(dat_1);
delay_ms(2);
while (1)
```

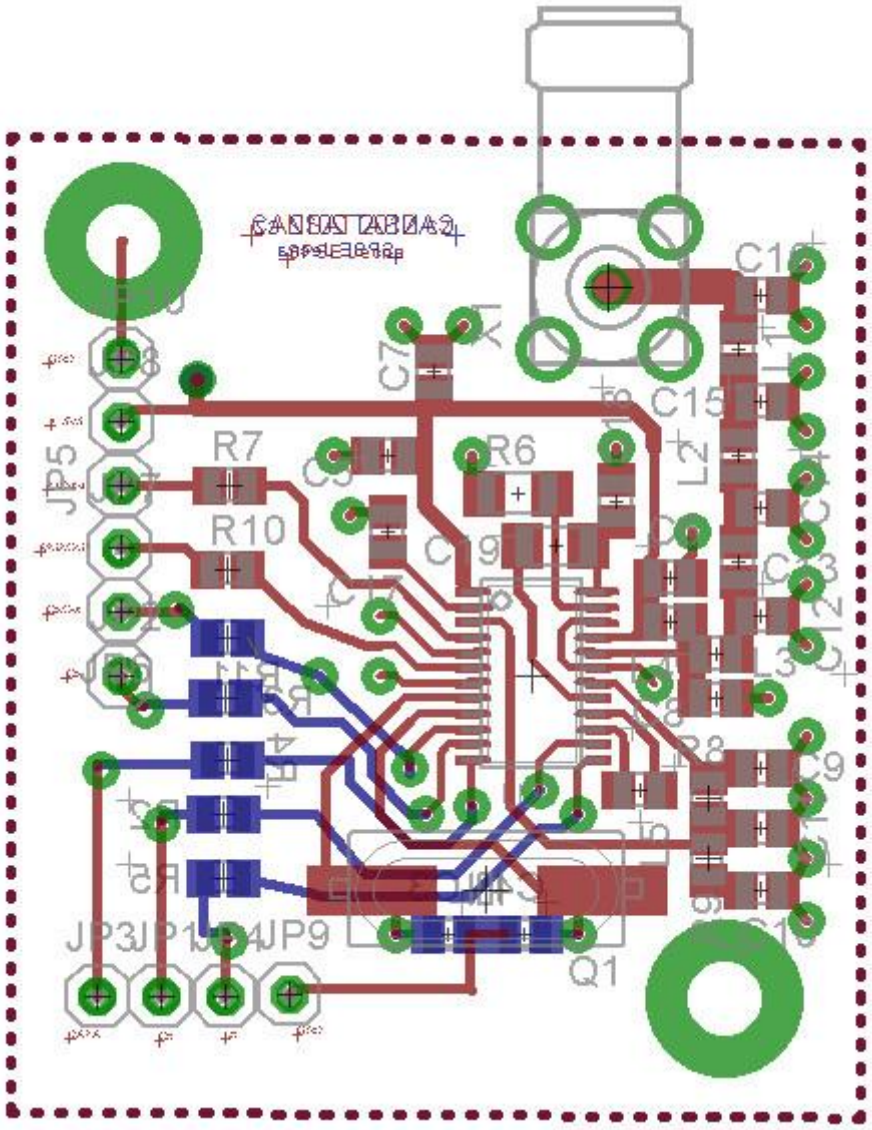
```
      {
      // Place your code here
       write_reg(dat_1);
       delay us(50);
       write_reg(dat_2);
       delay_us(50);
       write_reg(dat_3);
       delay_us(50);
       write_reg(dat_4);
       delay_us(50);

      };
}

void write_reg(uchar *dat1)     //ADF7012
{
  int i,j;
  le=0;
  for(j=0;j<4;j++) //
  {
   acc=dat1[j];
  for(i=0;i<8;i++)
   {
    if (acc>127 ) dat= 1;
    else
    dat=0;
    acc=acc<<1;
    delay_us(10);
    clk=1;
    delay_us(10);
    clk=0;



   }

 }
}
```

6.1.4 Konstrukce vysílače s obvodem ADF7012 a jeho testování (Transmitter design with ADF7012 chip and testing)

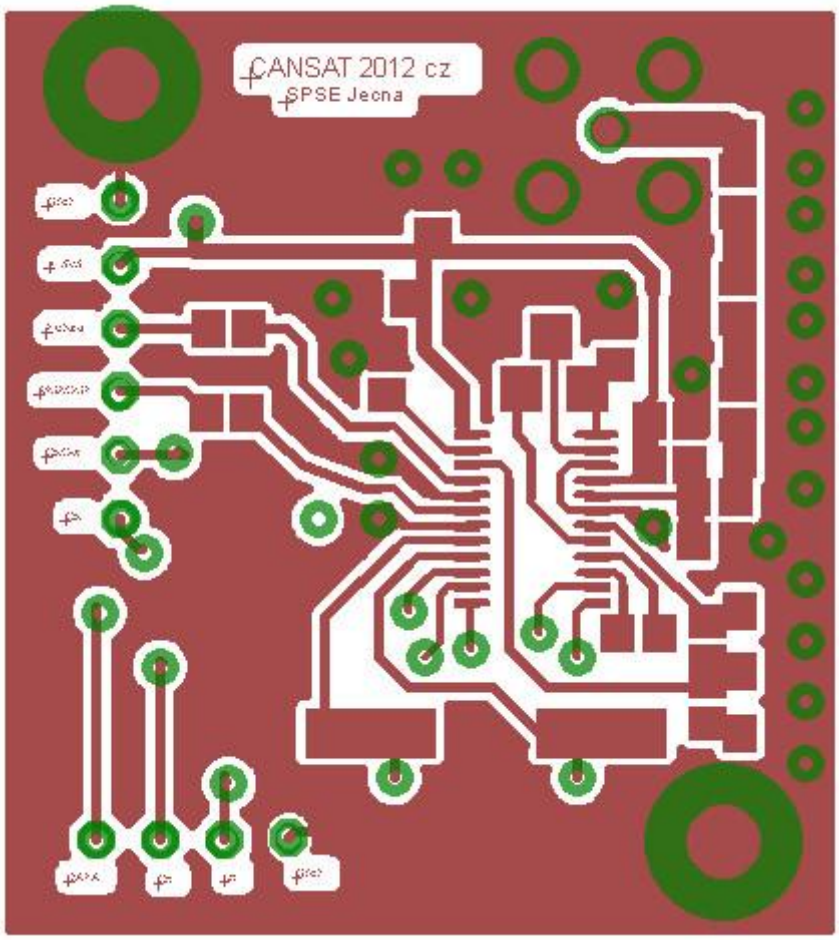
Zapojení vysílače je katalogové, ničím se neliší od zapojení a hodnot součástí v aplikačních listech Analog Devices.

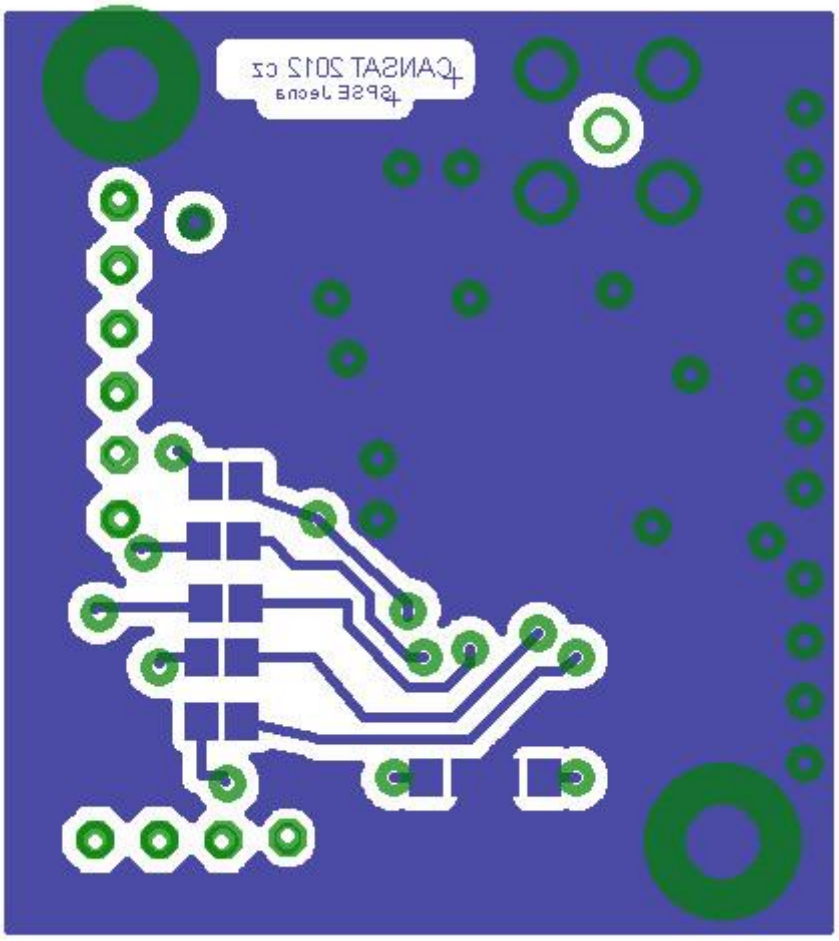


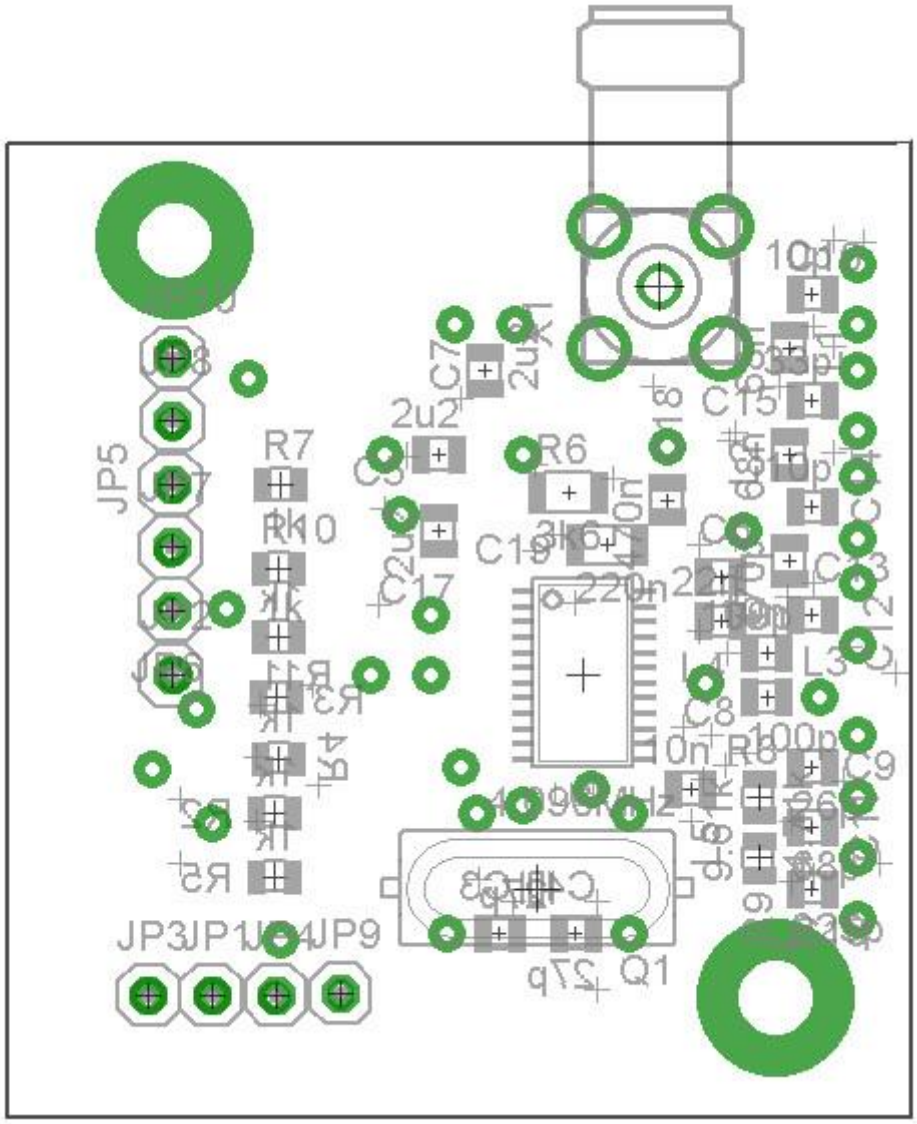
Pozn. Hodnoty L a C výstupního filtru i L3 VCO odpovídají kmitočtu 144,8MHz (je tam „čilý“ provoz APRS) – pro 433,9 MHz nutno upravit.
Schema bylo nakresleno v free verzi Eagle, sloužícího k návrhu PCB (plošných, tištěných spojů).

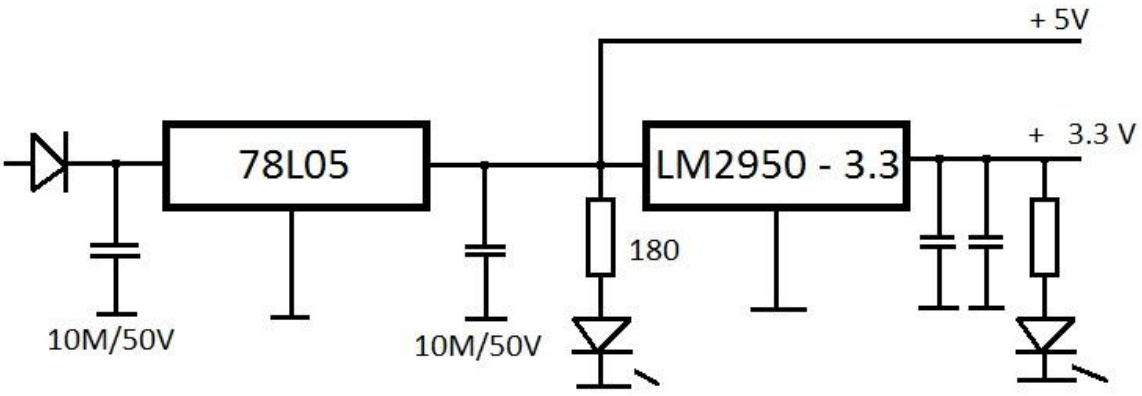


Ještě si zobrazíme obě strany spoje (nesmíme zapomenout v Eagle ještě aplikovat Tools – Ratsnet):


CANSAT 2012 cz
SPSE Jecna


CANSAT 2012 cz
SPSE Jecna



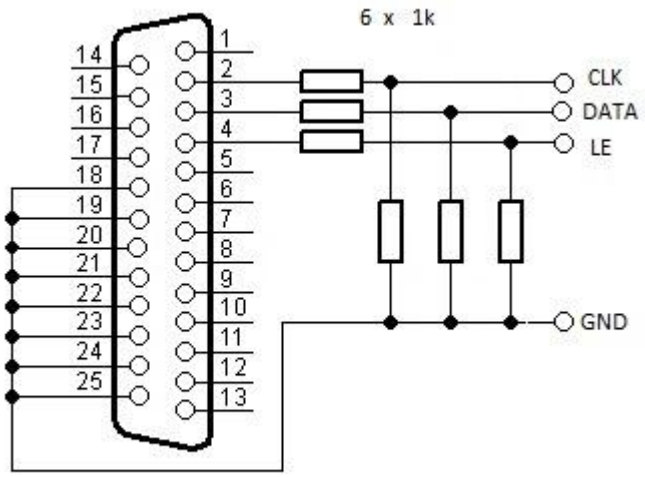


Zdroj 3.3V pro ADF7012 a 5V pro ATMega88

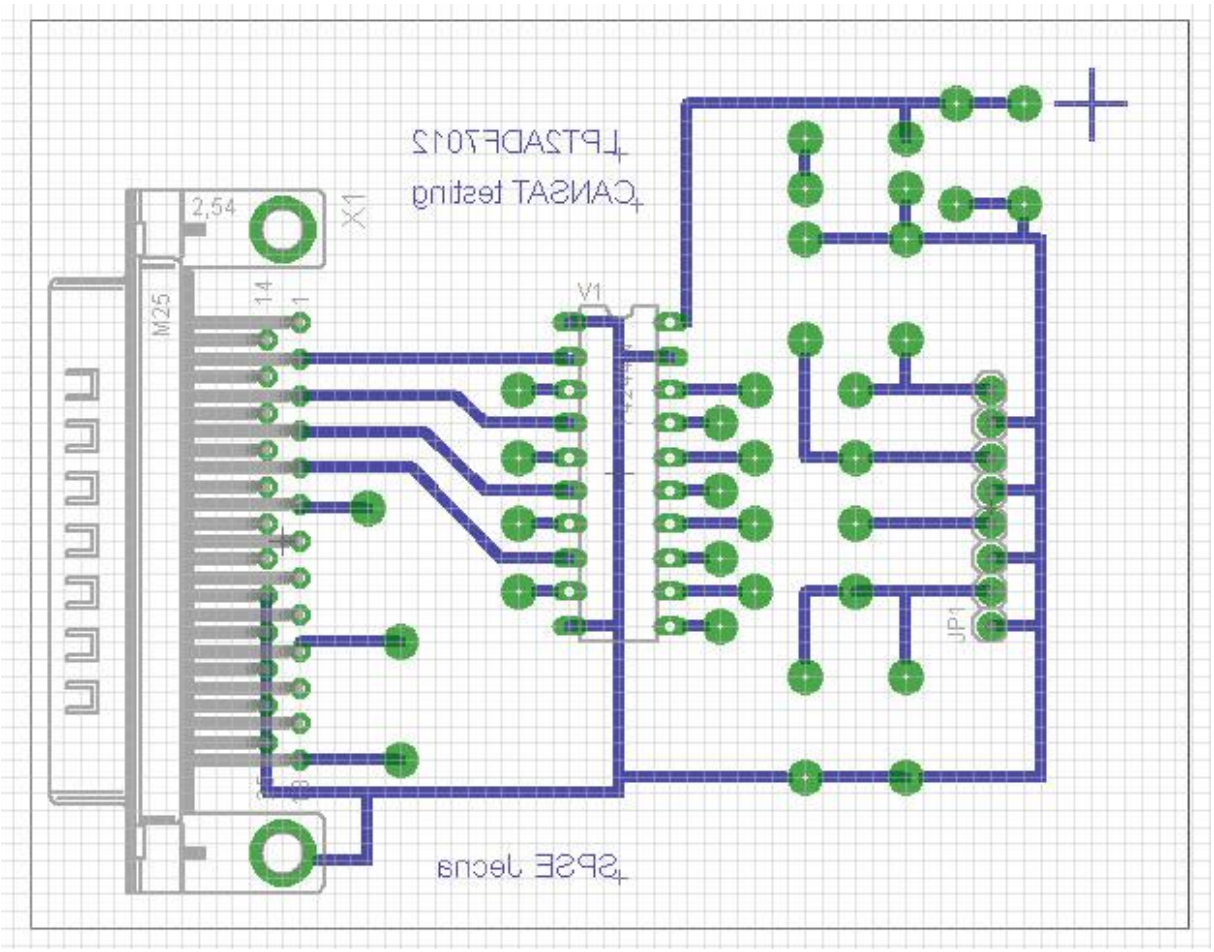
my remarks: *CanSat Book for Students* – part.1 2011

6.1.4.1 Testování pomocí sw firmy Analog Devices
(Testing with help of Analog Devices software)

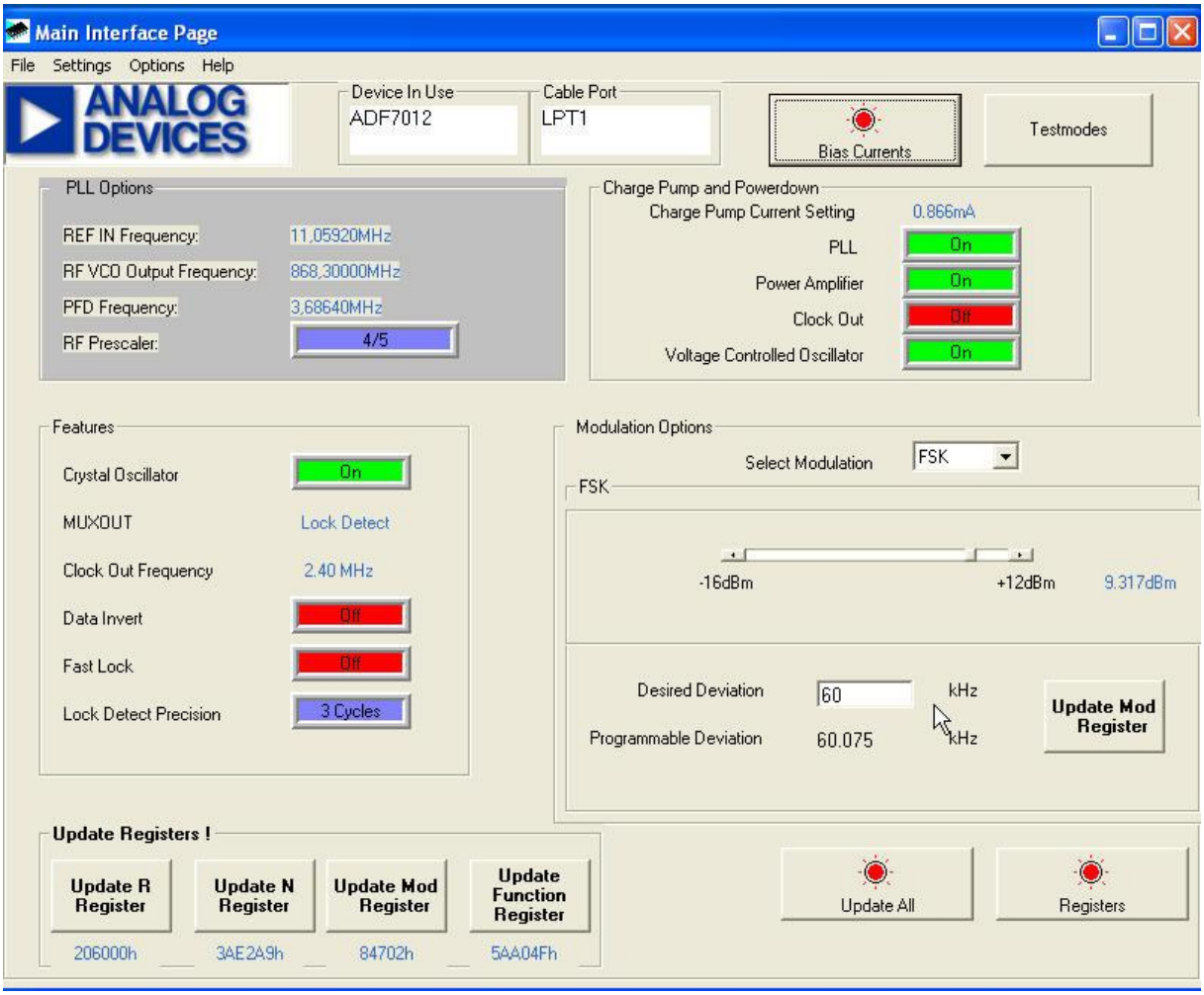
Pro nastavení či oživení destičky vysílače s *ADF7012* potřebujeme tento obvod inicializovat posláním čtveřice řídících 32bitových slov a popř. do něj dále posílat nějaká řídící slova, např. pro volbu kmitočtu. Dále potřebujeme nějaký zdroj „dat“. Jde o podobnou situaci, jako když se vývojáři seznamují s funkcí *ADF7012* pomocí startkitu firmy *ADF7012*. Jeho zapojení se velice podobá našemu zapojení a tak můžeme použít free software takového startkitu. Získáme ho za stránek AD spolu s dokumentací k startkitu. AD jich pro *ADF7012* má několik, liší se však pouze v maličkostech. Jsou označeny např. *EVAL-ADF7012EBx*. Z jeho schematu vyčteme, že s PC je propojen prostřednictvím paralelního portu LPT počítače. Obslužný sw přes tento paralelní port posílá tři řídící signály pro obvod *ADF7012*. Konkrétně do jeho pinů 11 (**CLK**), 12 (**DATA**) a 13 (**LE**). Signály z LPT však neposílá do pinů obvodu *ADF7012* přímo, ale přes odporové děliče složené z dvojic odporů 1k. Tím se úroveň signálů z LPT (cca 5V) sníží na polovinu. To je nutné vzhledem k tomu, že obvod *ADF7012* je napájen 3,3 V.



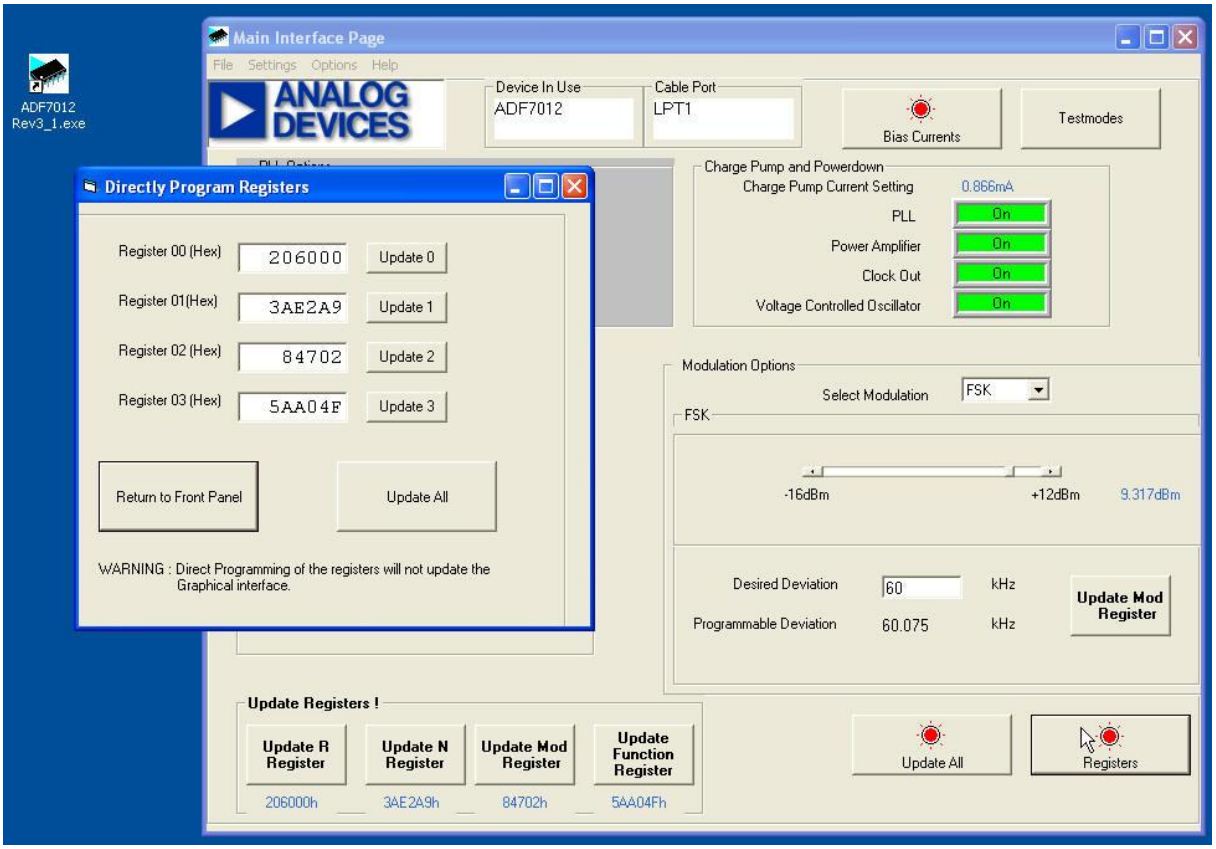
Lepší je ale použít zapojení obsahující ještě obvod 74244



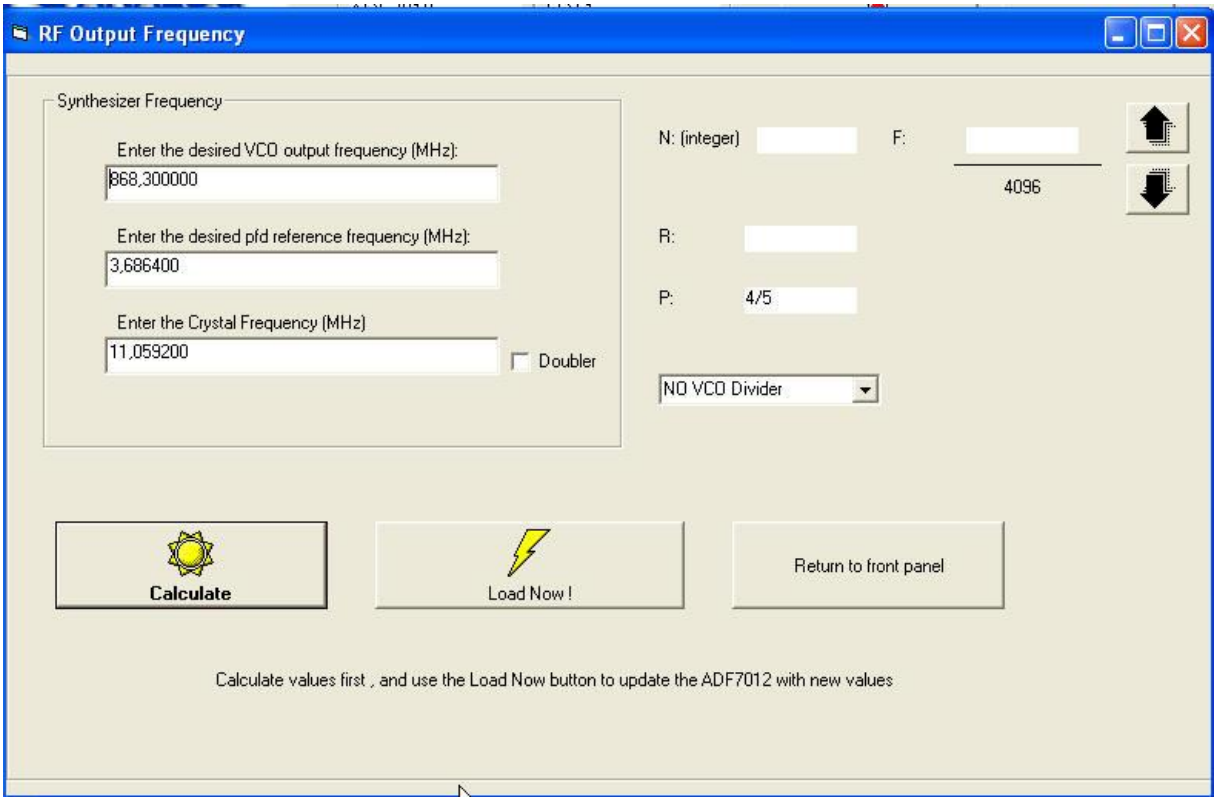
Ze stránek Analog Devices stáhneme a poté nainstalujeme **ADF70xx Evaluation Software**. Odzkoušel jsem verzi 3.1, která ještě předpokládá ovládání startkitu přes paralelní port PC. Některé vyšší verze již očekávají jen USB. Úvodní okno tohoto sw vypadá takto:



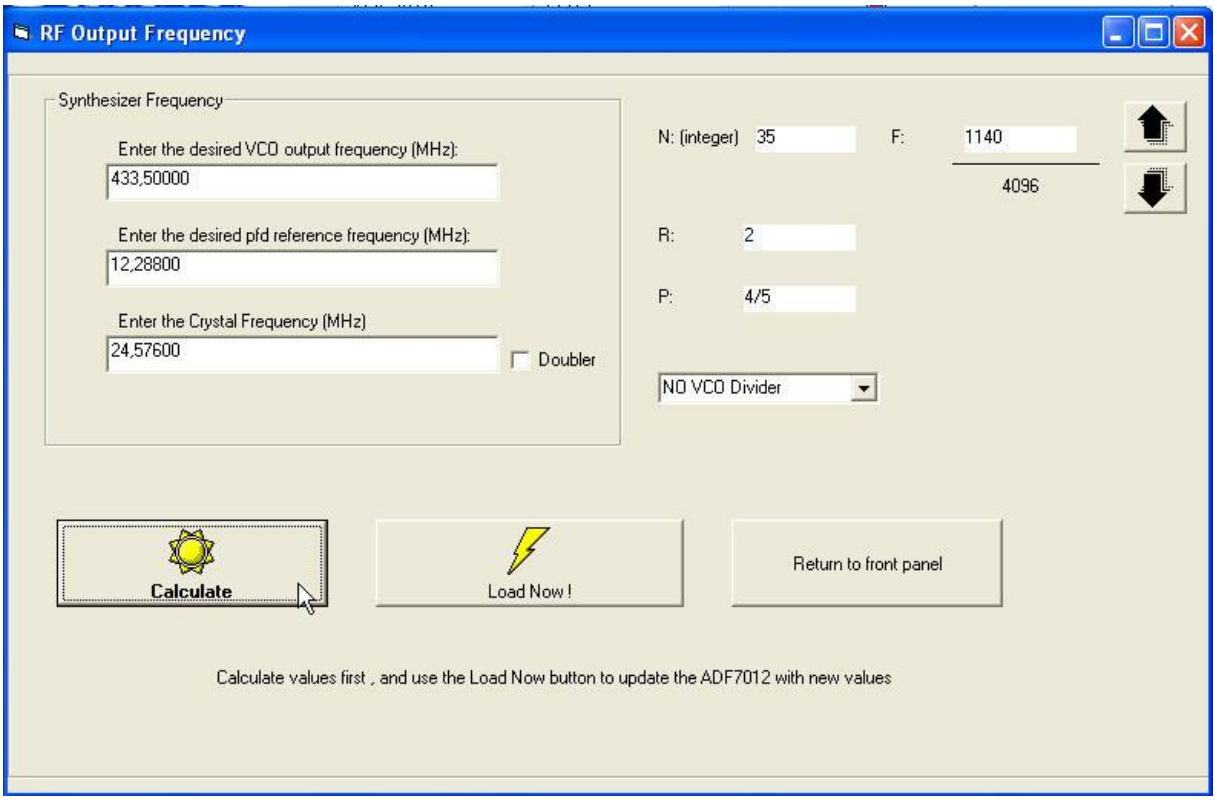
Tento program slouží jednak jako jakási „kalkulačka“ pro výpočet hodnot dat, vysílaných v 32bitových slovech do registrů *ADF7012* a dále k jejich vyslání přes lpt do **CLK**, **DATA** a **LE** vstupů obvodu *ADF7012*. Nemusíme ovšem hodnoty obsahu řídících registrů před jejich odesláním do **ADF7012** počítat. Můžeme je nastavit ručně. Stačí kliknout na tlačítko **Registers**. Poté se objeví okno



Vyplníme námi požadovaný obsah registrů a klikneme na tlačítko **Update All**. Můžeme ovšem obsah registrů spočítat pomocí tohoto sw. Ukážeme si to na našem příkladě. V úvodnímokně v části PLL options vidíme defaultní kmitočet 868.300 MHz, Ref kmitočet atd. Klikneme na hodnotu výstupního kmitočtu. Objeví se.

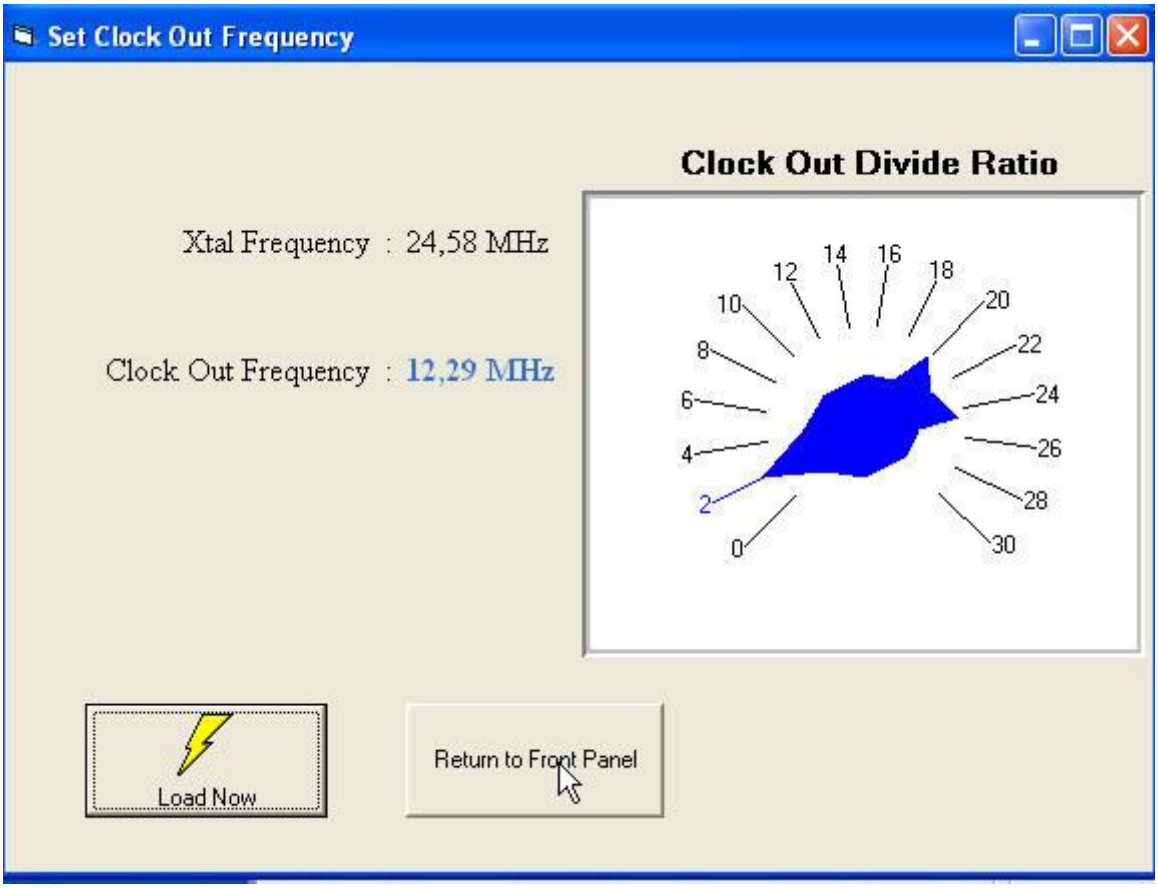


Dosadíme námi požadované hodnoty kmitočtů VCO, reference frekvence a kmitočtu krystalu. Poté klikneme na tlačítko **Calculate**. Dostaneme

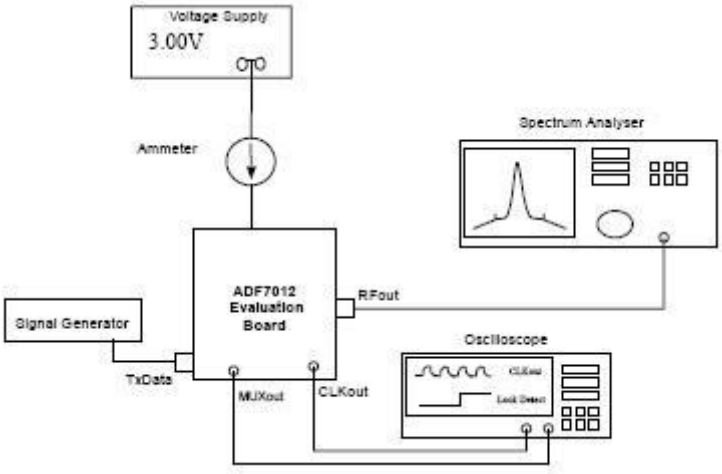


Vidíme, že se spočítalo N_{int} i N_{frac}, R, P . Spočítané hodnoty jsou stejné, jaké jsme uvedli v předchozích podkapitolách. Klikneme ještě na tlačítko **Load Now**. Nově spočítaná hodnota N registru se objeví pod příslušným tlačítkem. Je stejná, jakou jsme již spočítali.

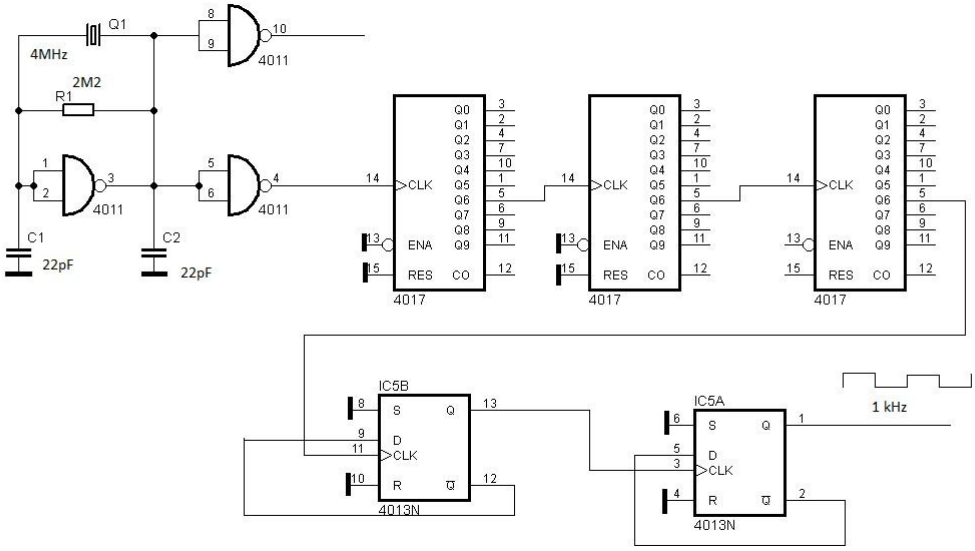
Obdobně můžeme určit pomocí této „kalkulačky“ i další požadované hodnoty registrů *ADF7012*. V některých případech se nepodaří získat stejné hodnoty, které jsme zjistili u vysílače PrattHobies. Např. poloviční kmitočet krystalu má být 12.288 MHz. Sw však zobrazí zaokrouhlenou hodnotu.



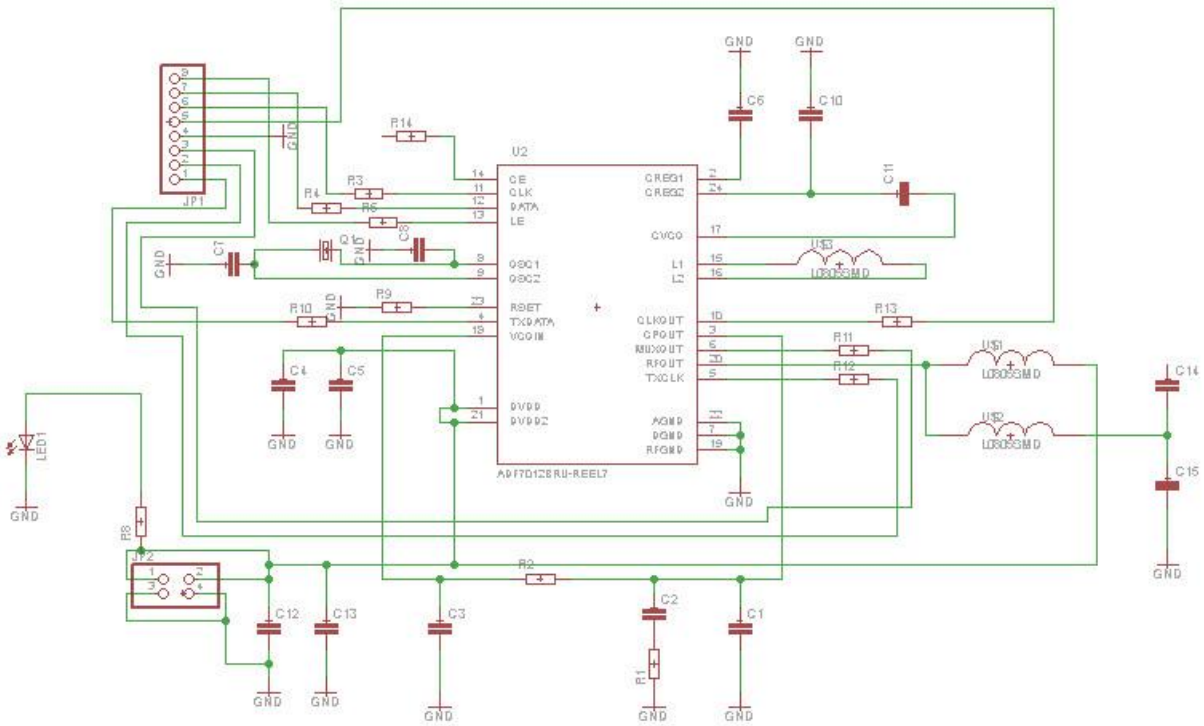
V manuálu k Evaluation Boardu *EVAL-ADF7012Ebx* je uvedeno i zapojení měřícího pracoviště:

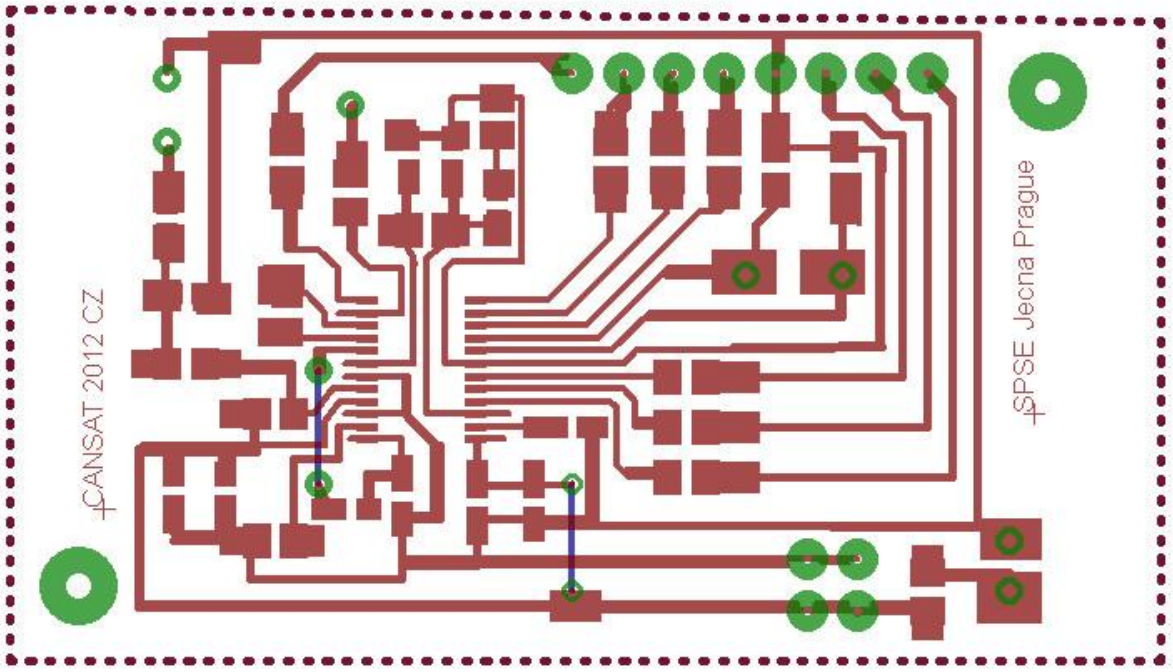


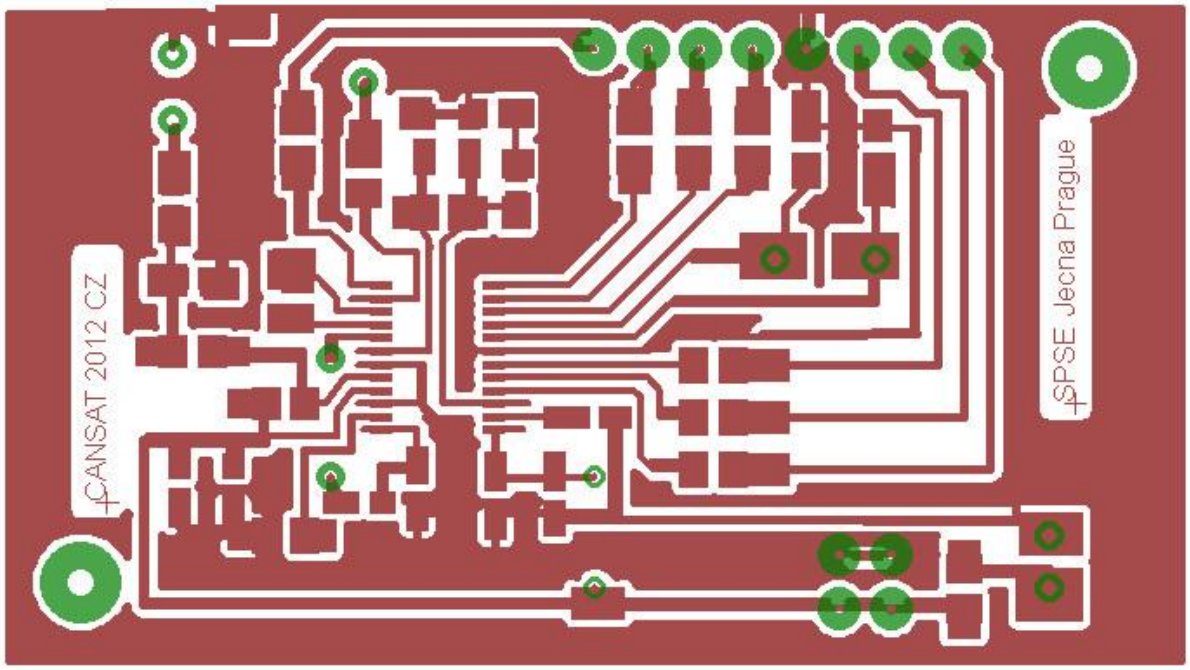
Skutečnost, že jako zdroj dat pro vstup TxData používají signální generátor mě inspirovala k použití téhož. K ověření funkce vysílače totiž ještě nepotřebuje AX25 pakety či jiný zdroj dat nesoucích informací. Stačí zdroj napětí obdélníkového průběhu o vhodném kmitočtu a napěťové úrovni. Obdélníkový průběh vpodstatě znamená, že do TxData posíláme signál 0101010101….. .
Pro účely testování jsem realizoval zdroj obdélníkového průběhu s opakovací frekvencí 1kHz realizovaný pomocí obvodů CMOS napájených 3,3 V. Výstup tohoto generátoru můžeme proto přímo propojit s **TxData** obvodu *ADF7012*.

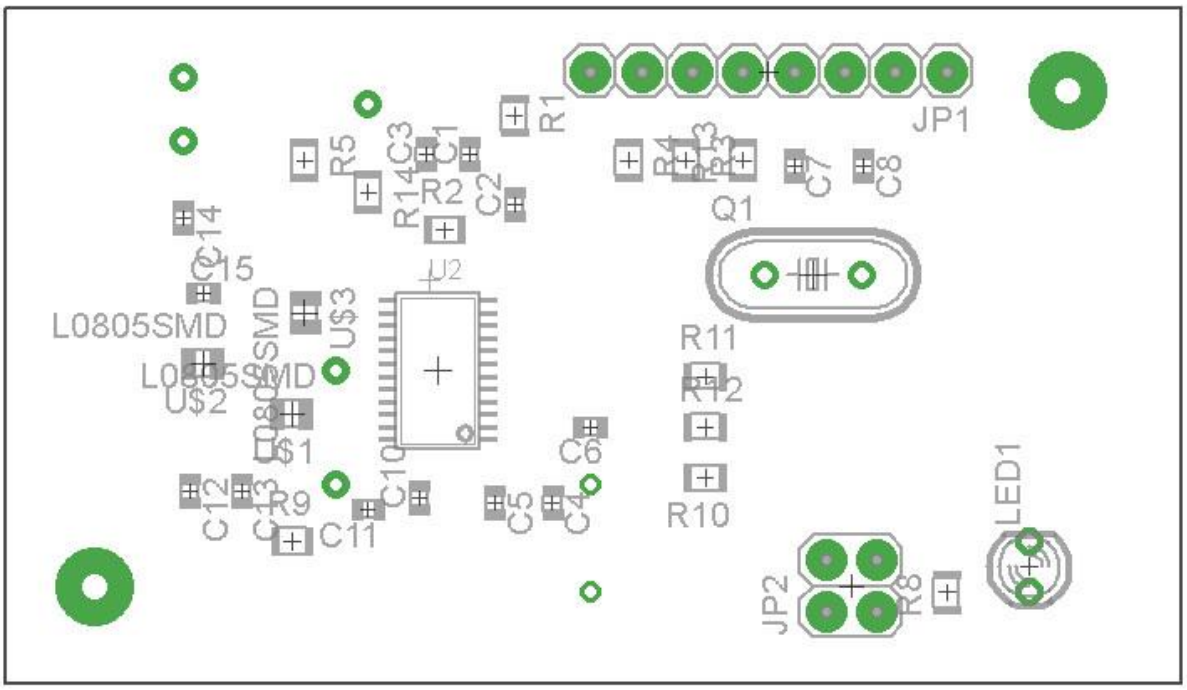


Pro první seznámení s obvodem ADF7012 vyrábí firma Analog Devices tzv. Evaluation Boards for ADF7012 pro které ja také výše uvedený sw. Doporučuji se vyrobit obdobný startkit:


GND
C6
C10
C11
U2
R14
R3
R4
R5
C8
Q1
C7
R9
R10
C4
C5
R13
R11
R12
U$1
U$2
C14
C15
LED1
R8
JP2
R2
C12
C13
C3
C2
C1
R1
ADF7012BRU-REEL7


CANSAT 2012 CZ
SPSE Jecna Prague





6.1.4.2 Testování pomocí příkazů F (Testing with help F comands)

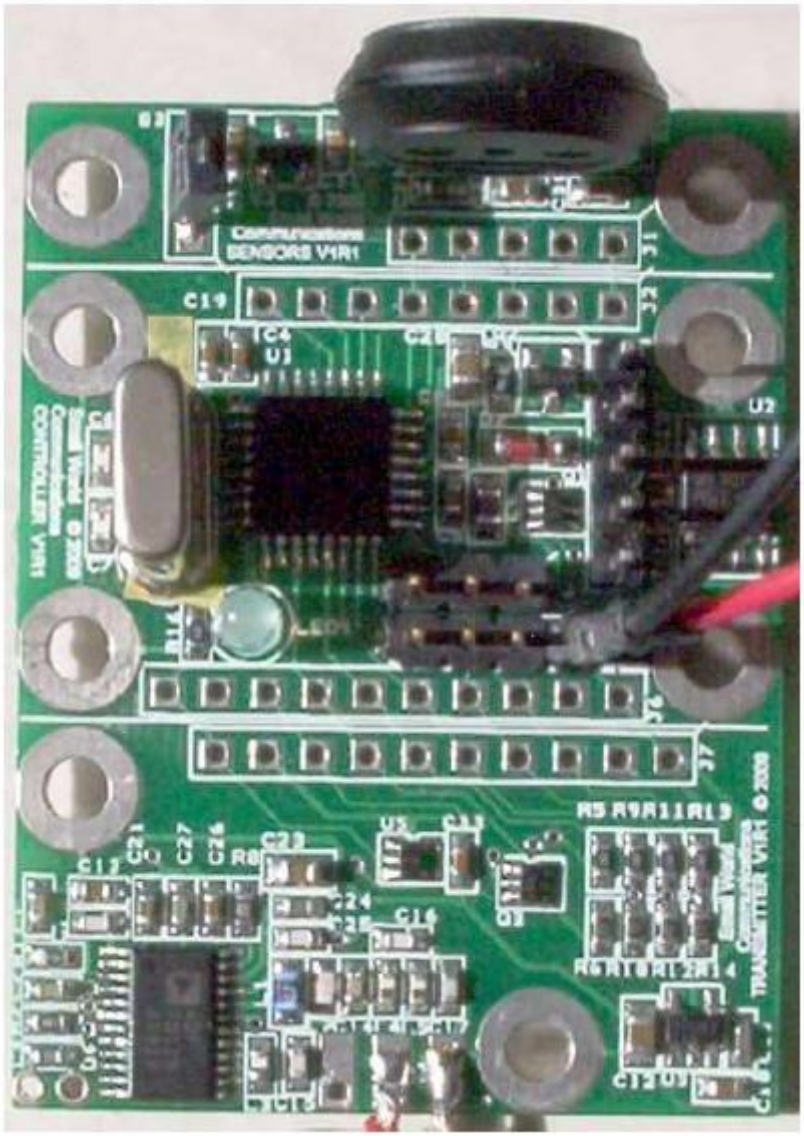
Firmware *ATMega88* je naprogramováno tak, že pro nastavení kmitočtu vysílače je „dálnopisným“ signálem do *ATMega88* odeslán příkaz **F** složený z písmena **F** následovaného haxeznaky uvedené v tabulce. Snadno zjistíme, že příkaz **F** je implementován tak, že za ním uvedené hexaznaky převede do binárního kódu a ten se doplní zleva nulami na celkových 32 bitů a ty se odešlou do *ADF7012*. Pokud použijeme znaky uvedené v tabulce, bude výsledný 32 bitový kód zakončen dvojicí 01. Půjde tedy skutečně o obsah N registru. Můžeme však použít za písmenem **F** hexakód libovolného

32bitového binárního čísla, a to bude odesláno do *ADF7012*. Tj můžeme takto do *ADF7012* odeslat libovolný obsah do kteréhokoli ze čtyř registrů *ADF7012*.

6.1.4.3 Konstrukce vysílače s ADF7012 a jeho nastavování

Stavebnice CanSatu Pratthobies obsahuje tři desky PCB – destičku čidel, palubní počítač a vysílač obsahující ATMega88 a *ADF7012*. Konstrukci celého zařízení můžeme zjednodušit tak, že použijeme jediný jednočipový počítač, který bude zastávat funkci palubního počítače zpracovávajícího signály z čidel a zároveň bude zajišťovat komunikaci, tj řídit obvod *ADF7012* a dodávat mu data zabalená do paketů AX25.

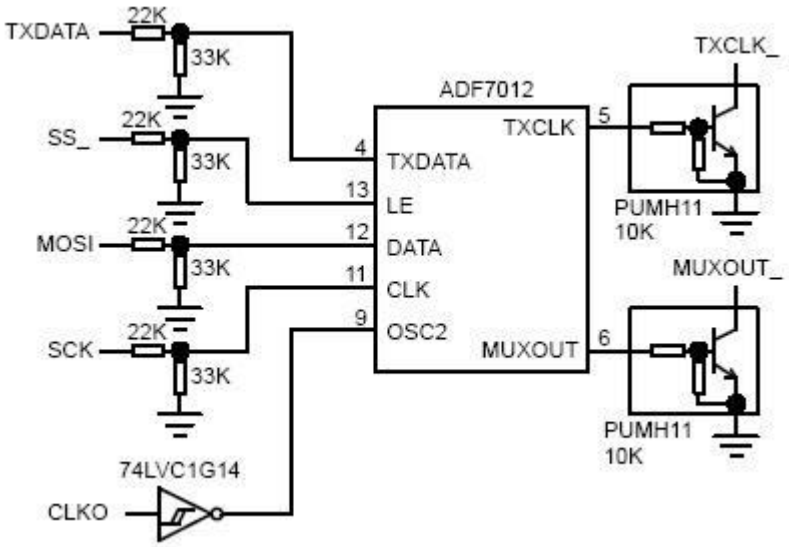
Takovou konstrukci používá např. firma Small World Communications v projektu Tinsat http://www.sworld.com.au/products/tinsat.html . Konstrukční provedení je zřejmé z obrázku:



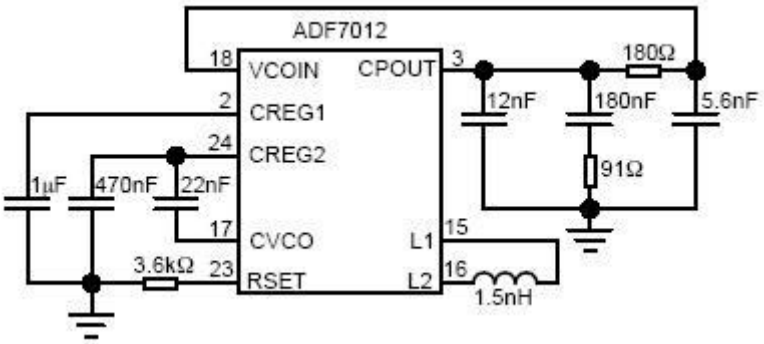
Horní destička obsahuje čidla, včetně nám známého čidla tlaku MPX4115, Prostřední destička pak jednočipový počítač ATMEL ATMega328P a krystal 12.288 MHz. Hodinový signál 12.288 MHz je také přiveden na dolní destičku obsahující vysílač *ADF7012*.

Ten vysílá na ISM kmitočtu 916.36 MHz AX25 pakety modulací AFSK rychlostí 1200bit/s či FSK resp. GFSK rychlostí 9600 bit/s. Protože k napájení vysílače *ADF7012* je potřeba 3.3V, použili nízkodriftový

stabilizátor firmy Texas Instruments TPS76433, který 5V použité pro napájení ATMega328 snižije na 3.3V pro vysílač *ADF7012*. Protože *ATMega328* má signály vyšších úrovní, než dovolují vstupy *ADF7012*, používají přizpůsobovací obvod:

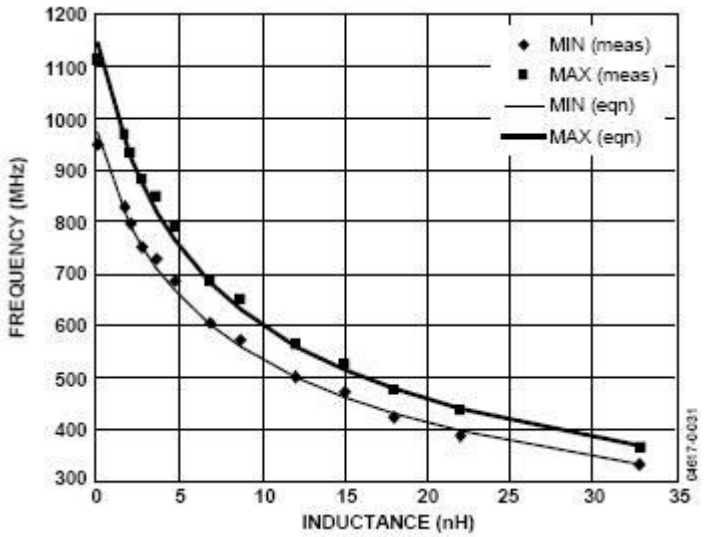


Inspirací pro nás může být i filtr v zpětnovazebné smyčce PLL pro řízení VC0.



Poznámka
Cívka 1,5nH mezi piny L1 a L2 je součástí VCO pro kmitočet 916.36MHz. Pro naši aplikaci v pásmu 433MHz bude hodnota této cívky větší.

Potřebnou hodnotu indukčnosti cívky mezi vývody L1 a L2 určující střední kmitočet VCO lze vyčíst z grafu z dokumentace Analog Devices pro vysílač *ADF7012*:



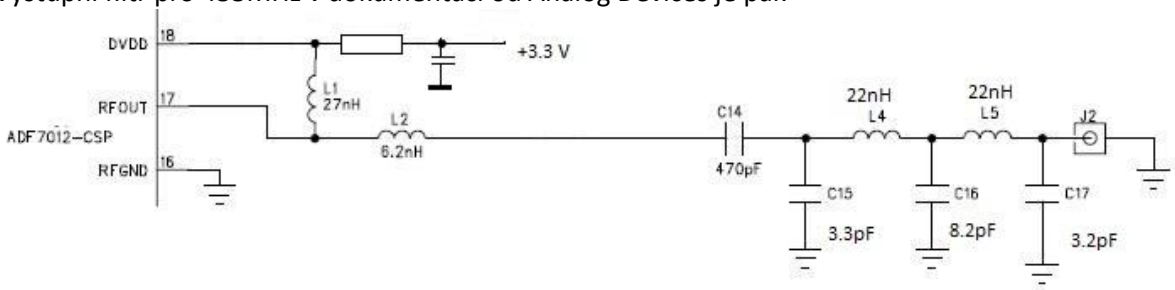
Typické hodnoty cívky VCO lze vyčíst také z dokumentace Analog Devices pro startkity *ADF7012 Eval EB5:*

315MHz 36nH
433MHz 19nH
868MHz 1.9nH
915MHz 1.6nH

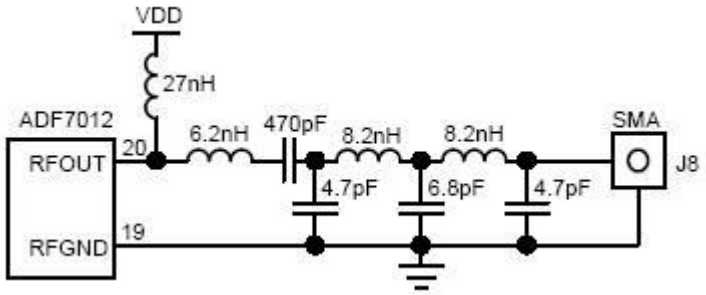
V dokumentaci pro samotný *ADF7012* uvádí výrobce čtyři aplikační zapojení včetně hodnot součástí pro kmitočty 315 MHz, **433,92MHz**, 868 MHz a 915MHz. Zde lze nalést jak hodnoty cívky VCO, tak součástí výstupního filtru.
Pro 433.92 MHz doporučují indukčnost **22nH** v provedení výrobce Coilcraft, typ 0603-CS-22NXJBU.

Pro naše účely se zdá proto vhodný typ TL.18nH SMD 0805 2% prodávaný po 2Kč v prodejně GM Electronics http://www.gme.cz/cz/elektronicke-soucastky/indukcnosti-tlumivky/03012.html?p=7 pod Skl.č: 965-112

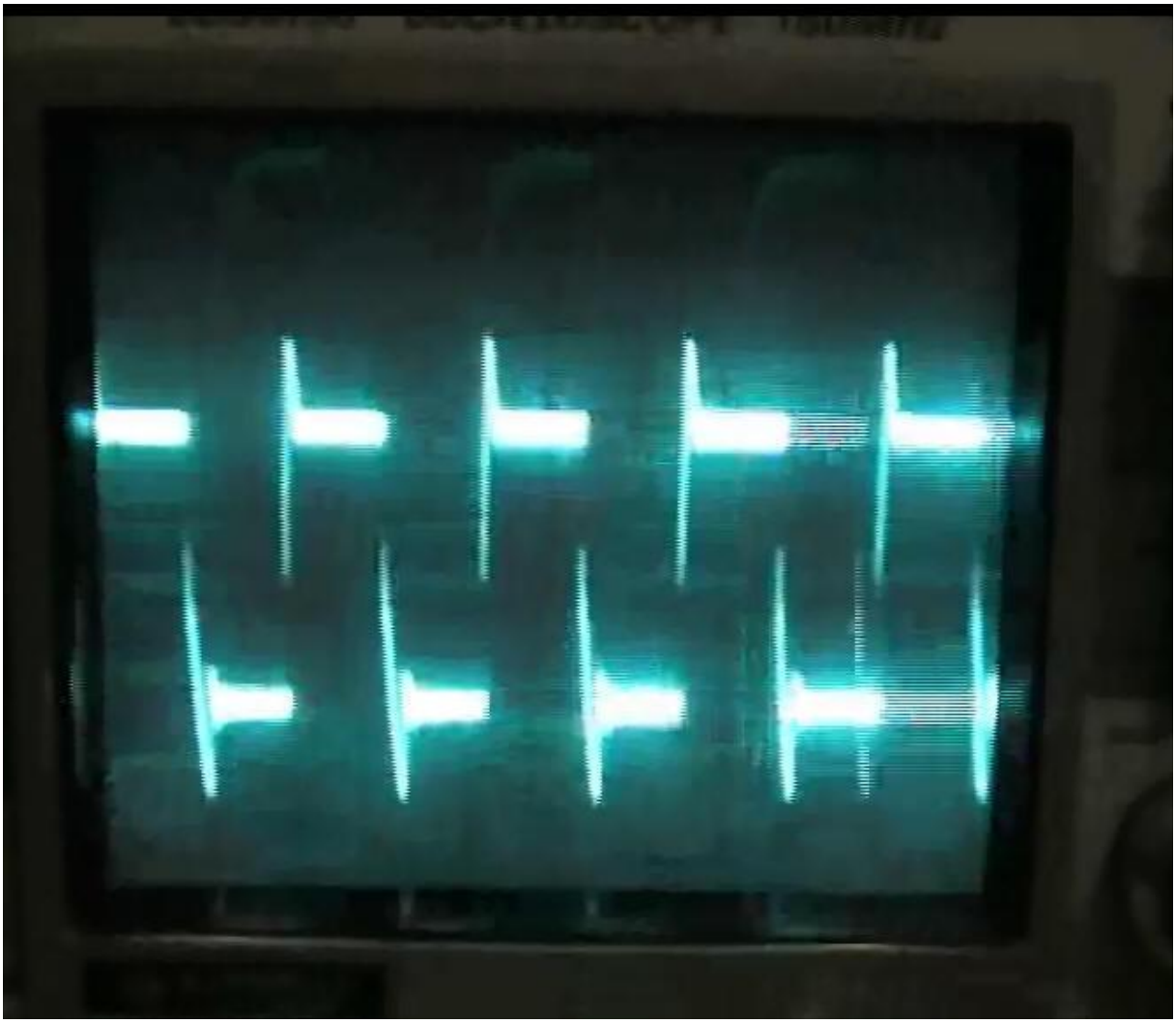
Výstupní filtr pro 433MHz v dokumentaci od Analog Devices je pak



Výstupní filtr pro 916 MHz je zapojen podle obr.:



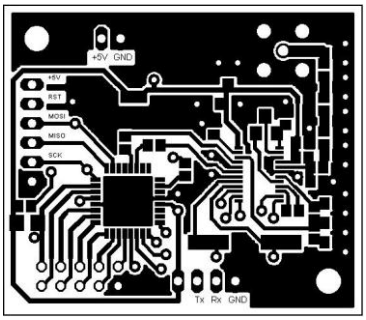
Základní ověření funkce vysílače s *ADF7012* provedeme v několika krocích:
Nejprve se pomocí logického analyzátoru zkontrolujeme čtveřicí řídících 32 bitových slov vyráběných např. nějakým procesorem ATMEL AVR (kap. **6.1.3.2.2**). Pokud tato řídící slova pošleme do *ADF7012*, můžeme se pomocí přijímače přesvědčit o silném signálu cca 24,576MHz, což je kmitočet krystalu, který slouží k řízení hodin obvodu *ADF7012*. Dále bychom měli slyšet silný signál i na polovičním kmitočtu, tj 12,288MHz. Nastavili jsme totiž dělící poměr děličky zapojené v obvodu *ADF7012* na :2. Tím jsme si ověřili, že funguje komunikace mezi jednočipovým počítačem a obvodem *ADF7012* a že tento obvod je řídícími signály ovládán.
Pokud by tomu tak nebylo, byl by defaultní dělící poměr :16 a tak bychom na přijímači kromě signálu na 24.576 MHz zachytili také signál na 1.536 MHz.
Dále můžeme zkontrolovat výstupní signál vysílače např. přijímačem naladěným na kmitočet, který jsme nastavili pomocí řídícího slova. Pokud tento signál nezachytíme a nemáme např. špatnou hodnotu cívky VCO, je problém obvykle v tom, že smyčka PLL není v ustáleném stavu, resp. Že PLL se nezachytil. Obvykle bude problém v RC filtru. Máme-li k dispozici osciloskop, můžeme s jeho pomocí kontrolovat, zda PLL je zasynchronizován. K tomu slouží vývod č.6 obvodu *ADF7012* označený MUXOUT. Protože tento vývod slouží i k jiným kontrolním účelům, musí být obvod *ADF7012* naprogramován čtveřicí bitů M1 až M4 (DB11 až DB14) řídícího slova pro poslední, funkční (R3) registr obvodu *ADF7012* na funkci vývodu MUXOUT označenou jako **DIGITAL LOCK DETECT** (popř. na ANALOG LOCK DETECT). Na funkci **DIGITAL LOCK DETECT** je nastaven pomocí posledního řídícího slova v init části a zůstává tak nastavován i když toto řídící slovo posíláme pozměněné za účelem zapnutí koncového stupně PA. Pokud je PLL „zavěšen“ je na MUXOUT signál o logické úrovni cca 3V. Ve skutečnosti tento signál ještě obsahuje střídavou složku cca 40mV viz obr.:



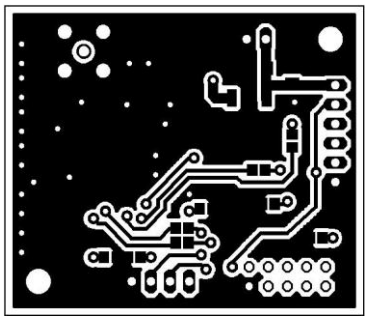
(poz. Osciloskop nastaven na střídavý zesilovač, 20mV/dílek)

6.1.5 ARPS vysílač z diplomové práce VUT Brno

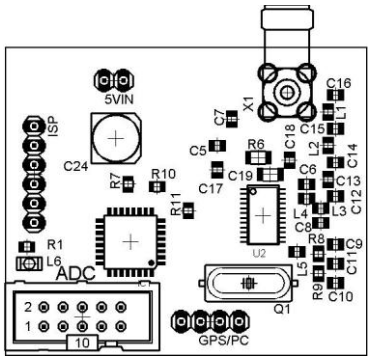
Sabol Martin ve své diplomové práci: „FM VYSÍLAČ APRS TELEMETRICKÝCH DAT V PÁSMU 144MHZ“ [26] velice dobře popsal systém APRS, funkci obvodu ADF7012 a vytvořil vysílač telemetrických dat v radioamatérském pásmu 2m, které pro APRS využívá kmitočet 144,8 MHz. Kromě vlastní diplomovové práce jsou k dispozici i **veškeré zdrojové kódy** jak pro ATMega88, tak pro PC sloužící k nastavování vysílače a dále soubory PCB návrhového sw Eagle. Můžeme si tedy jeho vysílač vyrobit a rovnou používat do Cansatu, pokud nám nebude vadit, že používá pásmo 2m. Ostatně jedna ze tří variant vysílače Pratt Hobbies je rovněž určena pro toto pásmo. Vysílač ing.Sabola lze ovšem vyrobit i pro pásmo 70cm (433 MHz), což lze jen doporučit. Pak bude moci zcela nahradit vysílač Pratt Hobbies v soutěži ESA pro středoškoláky. S vysílačem PrattHobbies má společné to, že obsahuje ATMega88 a ADF7012. Jeho sw vybavení jak ATMega88, tak PC lze považovat za zdařilejší a navím máme k dispozici zdrojové kódy. Protože navíc umožňuje připojení GPS modulu a řady čidel, může v řadě případů sloužit i jako palubní počítač Cansatu. Tím ušetříme váhu, spotřebu i rozměry části elektroniky. Pro představu uvádíme několik obrázků vysílače z diplomové práce:



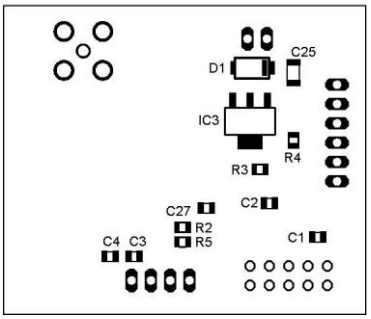
Pohled ze strany „TOP“



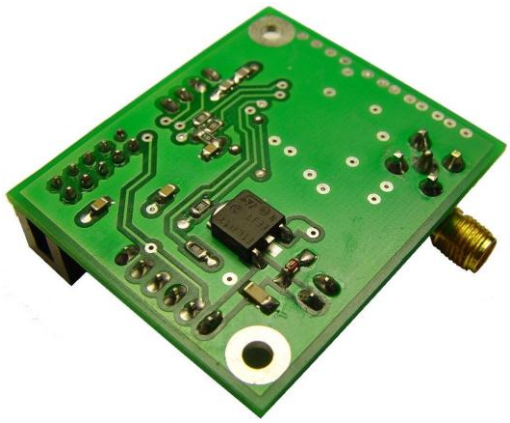
Pohled ze strany „Bottom“



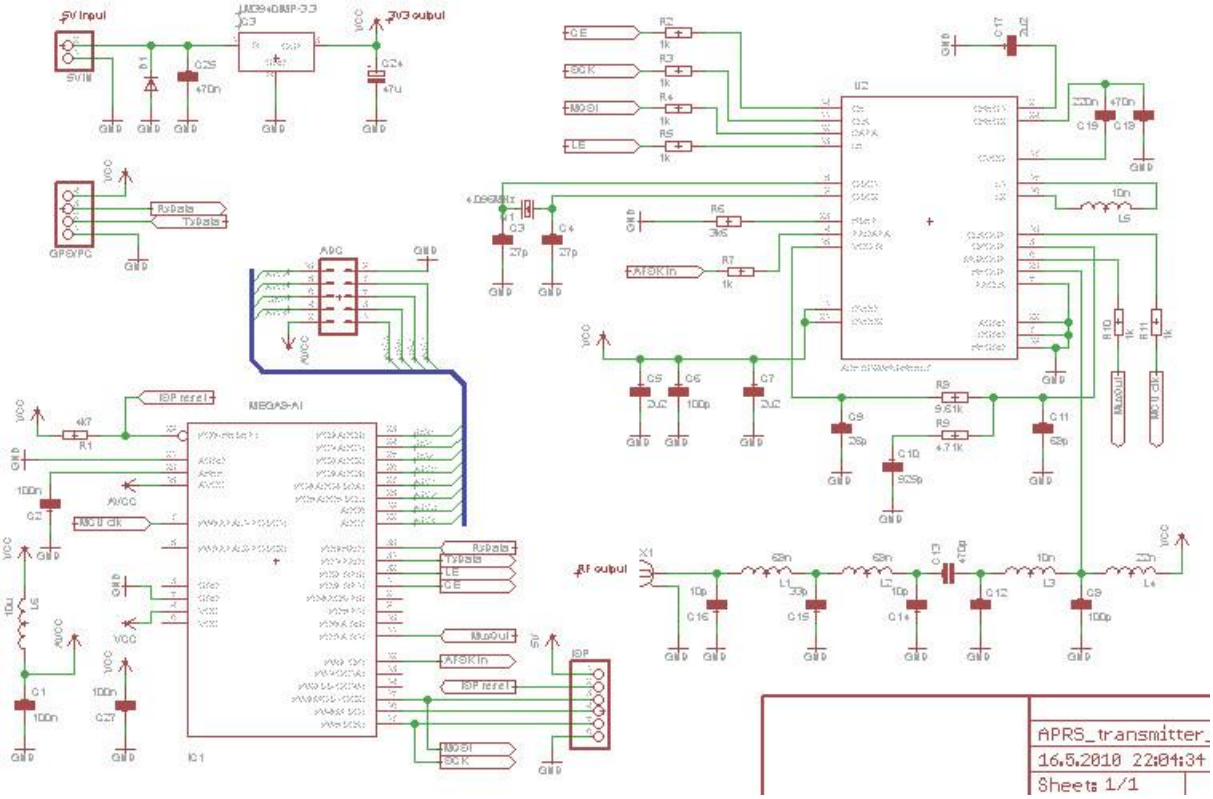
Osazení ze strany „TOP“



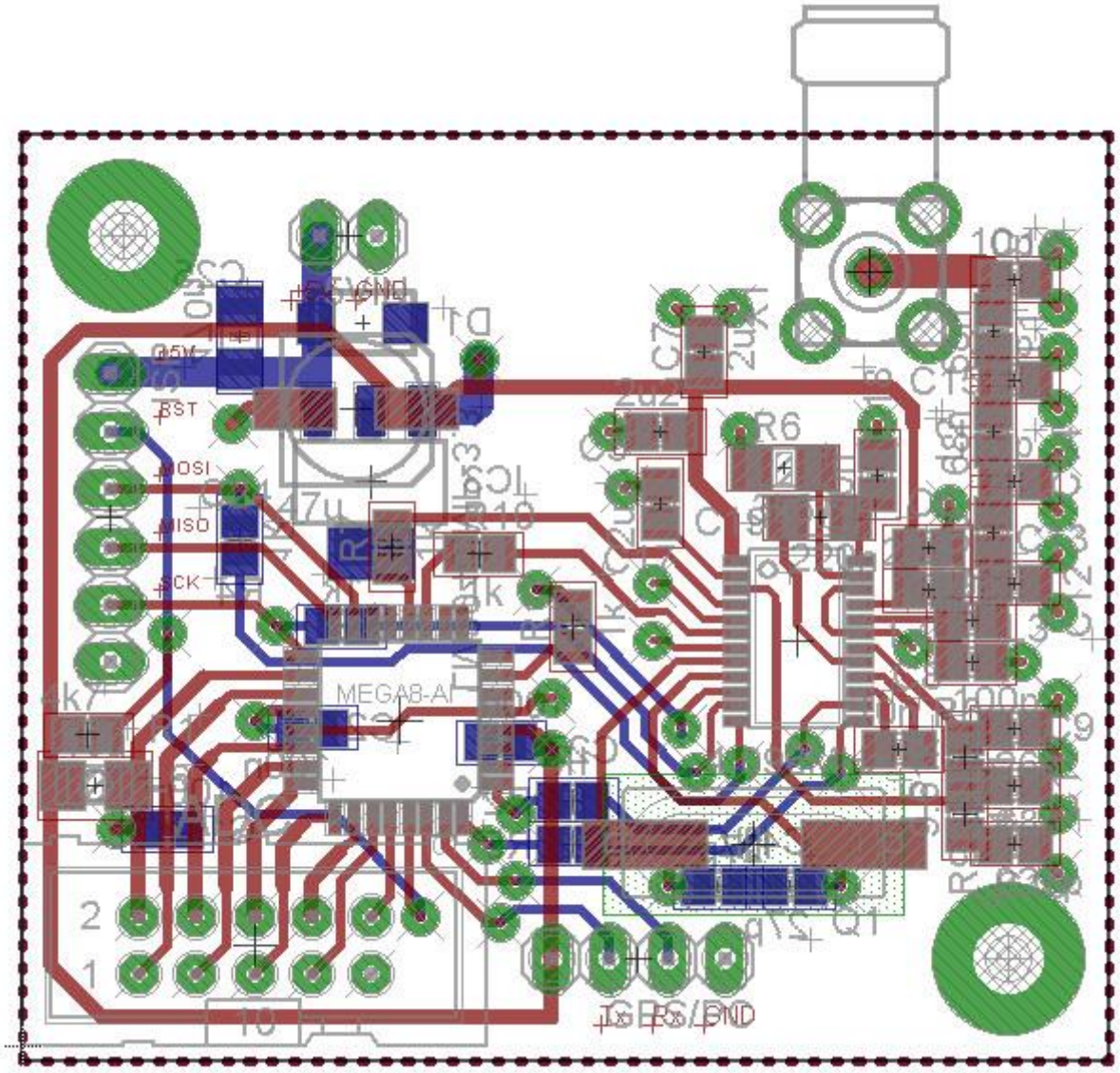
Osazení ze strany „Bottom“



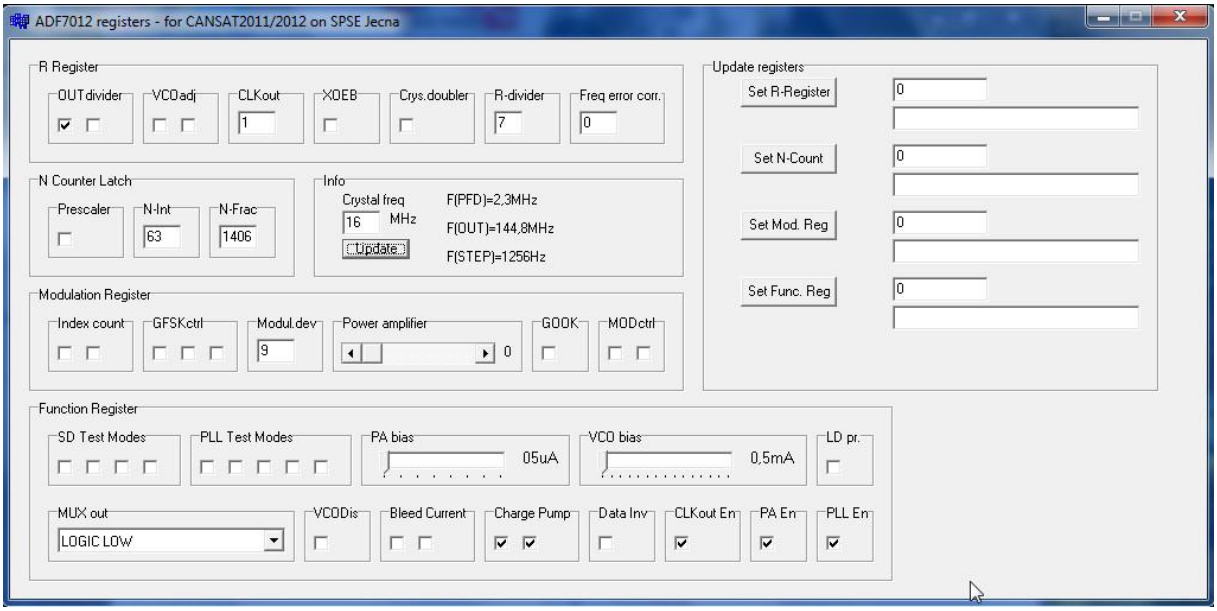
Uvedeme též schema


5V input
3V3 output
GPS/PC
ADC
MEGA8-AI
ISP reset
AFSK in
Af output
APRS_transmitter_
16.5.2010 22:04:34
Sheets 1/1

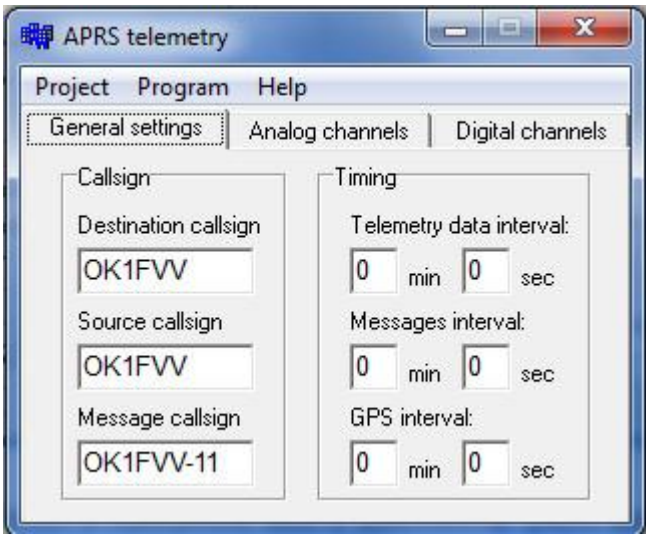
A PCB v Eagle

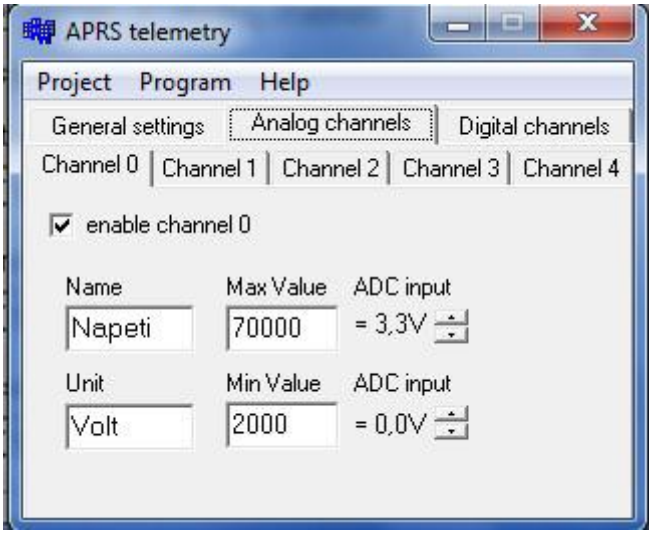


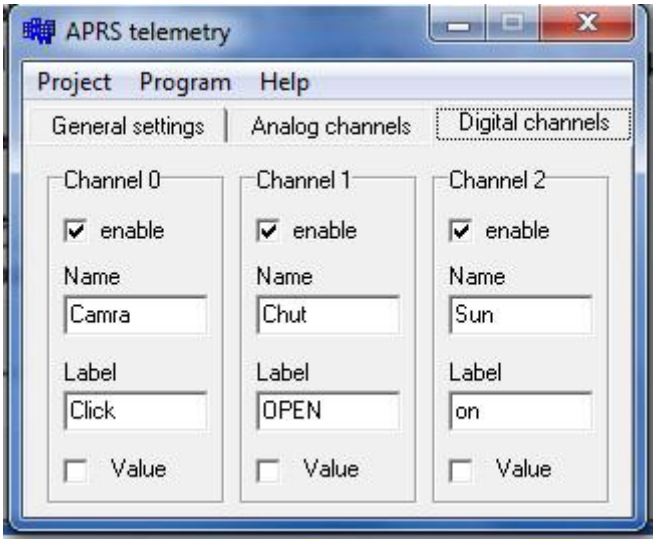
Tištěný spoj použijeme beze změny i pro pásmo 70cm. Změníme však kmitočet krystalu. Vhodný je např. 24,576 MHz, stejně jako v TXu Pratt Hobbies. Dále provedeme drobnou změnu v firmware ATMega 88 spočívající ve změně nastavení kmitočtu ADF7012. Můžeme též upravit sw pro PC tak, aby místo defaultních hodnot pro 144,8 MHz měl hodnoty pro 433,92 MHz .
Pokud jde o sw na PC (napsaný v prostředí Borland C++ Builder, zdrojáky na DVD) máme dva programy. Jeden (již zmiňovaný) slouží pro inicializaci ADF7012



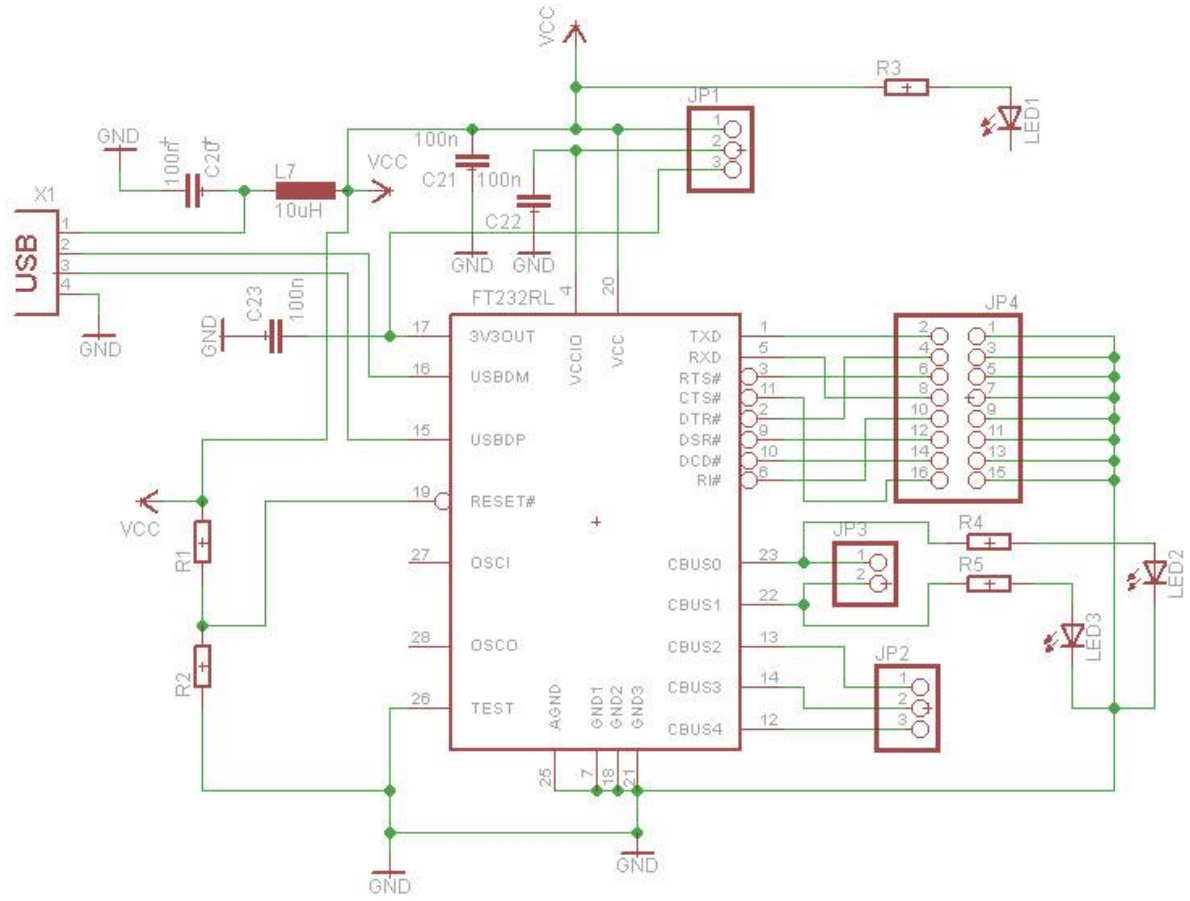
Další sw slouží k nastavení ARPS telemetrie (ADC kanálů apod.)

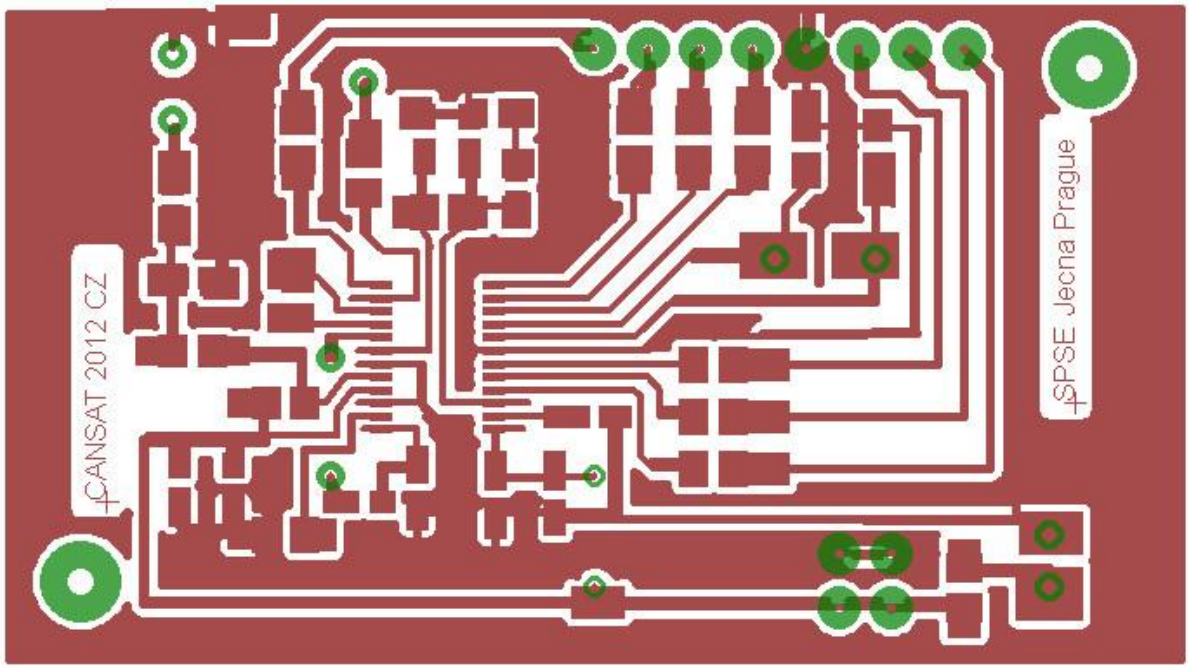






Oba tyto programy komunikují s ATMega88 vysílače prostřednictvím sériového kanálu rychlostí 9600baud, 8 datových bitu, jeden stop bit a bez paritního bitu. Přitom předpokládá ze strany PC použití USB kanálu a dále převodníku USB na sériový kanál. Tento převodník je založen na obvodu FT232RL. Oba programy kontrolují i přítomnost převodníku z USB na UART s FT232RL. Jeho zapojení je uvedeno v diplomové práci, PCB však již nikoli. Proto jsme použili vlastní s PCB opět v Eagle.
Obdobný převodník najdeme i na http://www.mlab.cz


VCC
R3
JP1
LED1
GND
100n
C20
L7
10uH
VCC
100n
C21
C22
X1
USB
GND
GND
GND
C23
100n
FT232RL
3V3OUT
USBDM
USBDP
RESET#
OSCI
OSCO
TEST
VCCIO
VCC
AGND
GND1
GND2
GND3
TXD
RXD
RTS#
CTS#
DTR#
DSR#
DCD#
RI#
CBUS0
CBUS1
CBUS2
CBUS3
CBUS4
JP4
JP3
JP2
R4
R5
LED2
LED3
VCC
R1
R2
GND
GND


CANSAT 2012 CZ
SPSE Jecna Prague

6.1.6 Výkonový zesilovač PA s obvodem ADL5601ARKZ-R7
(Power amplifier PA with ADL5601ARKZ-R7 chip)

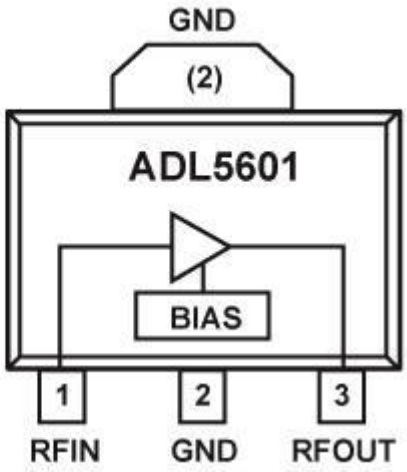
U startkitů CanSatu od PrattHobies s *ADF7012* při ESA 2010 soutěži byla předpokládaná vzdálenost mezi vysílačem CanSatu a přijímačem pozemní stanice cca 1 km. Při použití vysílače s *ADF7012* na kmitočtu cca 433,9 MHz a s λ/4 vysílací anténou tvořenou drátkem této délky a několikaprvkou Yagi anténou na přijímací straně toto zařízekí k radiovému spojení bylo postačující.

S pomocí sponzorů se nám ale může poštěstit mít možnost, např. pomocí stratosférického balonu, vypustit CanSat do větších výšek. Radiové spojení bude sice stále uskutečňováno za přímé viditelnosti vysílače a přijímače, nicméně signál z vysílače bude na straně příjmu slabší a při spojení s přistávajícím CanSatem při jeho nízké výšce dokonce může dojít k situaci, že mezi vysílačem a přijímačem již nebude přímá viditelnost a že úroveň přijímaného signálubude slabá. Může být proto vhodné zvýšit výkon vysílače.

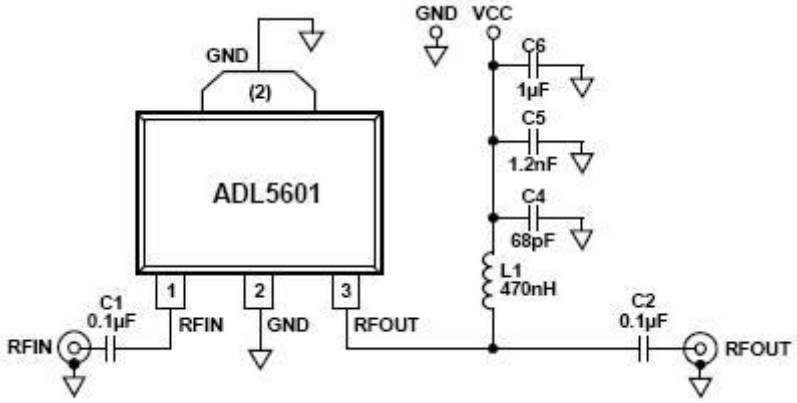
Toho lze docílit tím, že signál vysílaný vysílačem *ADF7012* zesílíme koncovým stupněm. Jako příklad, jak lze takový PA zhotovit si uvedeme možnost použití integrovaného PA zesilovače firmy Analog Devices *ADL5601*, který se mi od firmy AD podařilo získat zdarma jako tzv. **free samples**, obdobně, jako free samples obvodu *ADF7012*.

ADL5601 je 15dB lineární zesilovač pro kmitočty 50 MHz až 4GHz. Jeho vstupní i výstupní impedance je obvyklých 50 Ω. Jeho principielní zapojení ukazuje obr:

FUNCTIONAL BLOCK DIAGRAM FOR ADL5601

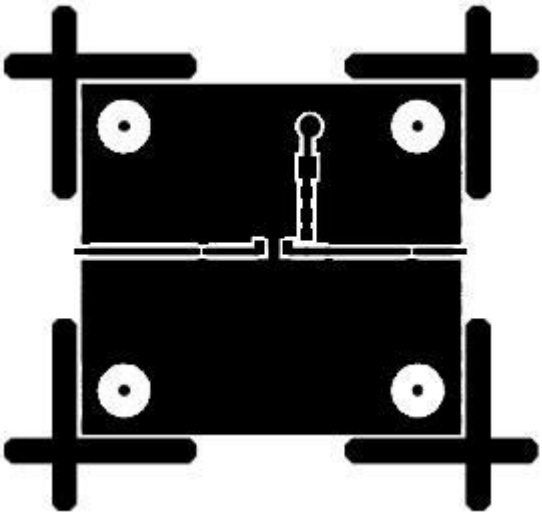


Typické aplikační schema tohoto obvodu doporučené výrobcem je:



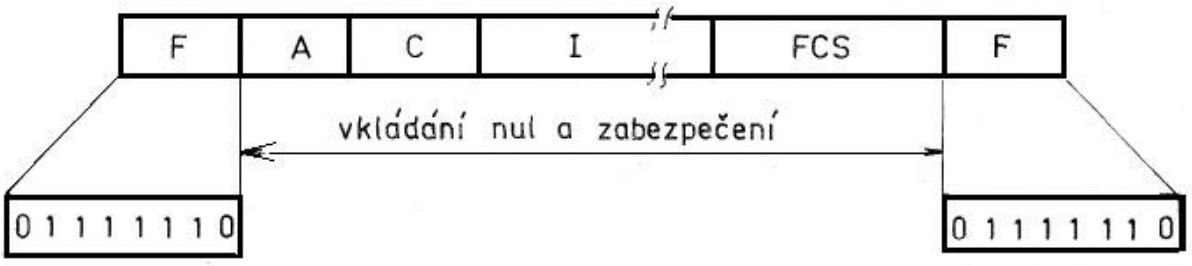
my remarks: *CanSat Book for Students* – part.1 2011

Napájecí napětí je 5V, obrazec PCB plošného spoje, použitý výrobcem v evaluation boardu ukazuje obr.:



6.1.7 Pakety AX25 (AX25 Packets)

Zopakujeme si, jak obevně vypadají pakety AX25, který vznikl aplikací profesionálního protokolu X.25. Ten pak vychází z HDLC. Základní jednotkou přenášených dat je rámec (Frame). V každém rámci se přenášená data samostatně zabezpečují. Struktura rámce je znázorněna na obrázku.



Struktura rámce řídicích postupů SDLC a HDLC

F - křídelní značka (8 bitů); A - adresové pole (8 bitů); C - řídicí pole (8 bitů); I - informační pole (libovolná délka); FCS - zabezpečovací pole (16 bitů)

Rámec se dělí na pole. Každý rámec začíná i končí křídelní značkou (Flag). Její tvar je rovněž na obrázku. Pak následuje adresové pole A. Toto pole vždy obsahuje adresu druhotné stanice. Další pole v pořadí je řídicí pole C, které obsahuje informace potřebné pro řízení činnosti. Za řídicím polem je v některých typech rámců informační pole I. Jeho délka může být libovolná (řídicí postupy SDLC však povolují délku rovnou jen celistvému násobku osmi bitů, HDLC nemá žádné omezení), tedy i nulová. Další pole FCS (Frame Check Sequence) přenáší 16 zabezpečovacích bitů rámce. Pak následuje koncová křídelní značka, která může být současně začáteční křídelní značkou dalšího rámce.

Aby nemohla být křídelní značka napodobena, vkládá se automaticky při vysílání do polí A, C, I a FCS vždy za 5 po sobě následujících bitů hodnoty 1 jeden bit hodnoty 0. Tyto vložené nuly mají další výhodu v tom, že zavádějí přechody charakteristických stavů do předávaných dat a tím slouží k udržování bitové synchronizace. Tyto vložené nuly se nepočítají do zabezpečení ani do významu polí. Na přijímací straně se opět před dalším zpracováním z přijímaného toku bitů odstraňují.

Takto je zaručeno, že se kromě křídelní značky nikde nevyskytne bezprostředně za sebou více než 5 bitů hodnoty 1. Při kontrole zabezpečení samozřejmě přijímací stanice nepozná, že přijímá pole FCS, dokud nevyhodnotí křídelní značku za polem FCS.

Rámce se pro přenos sdružují do posloupností rámců, které se mohou vysílat nepřetržitě za sebou. V posloupnosti rámců se jednotlivé rámce číslují a přidělená čísla rámců se přenášejí spolu s rámcem na protější stanici. K tomu slouží řídicí pole C. **Ns** označuje počet vyslaných rámců, **Nr** označuje počet správně přijatých rámců. Číslování se děje modulo 8. Na přijímací straně se zjišťuje, zda má přijatý rámec očekávané číslo v pořadí rámců. V případě neshody se nezapočítá do **Nr** a buď se vyřadí, nebo uschová do doby, než budou platně přijaty všechny předcházející rámce. Tak se zabrání přehození pořadí nebo ztrátě rámce. Na vysílací straně se rámce cyklicky opakují, přičemž se z tohoto opakovacího cyklu vždy potvrzený rámec vyřadí a nahradí v pořadí dalším rámcem.
Tímto způsobem je možné maximálně využít případné zpoždění potvrzování platně přijatých rámců způsobené konečnou dobou šíření informace na datovém okruhu.
K vyžádání odpovědi (potvrzení) slouží bit P/F (Poll/Final) v poli C. Prvotní stanice užívá tento bit ve významu P (Poli - výzva k odpovědi), druhotná stanice užívá bit P/F ve významu F (Finál - ukončení odpovědi).
Prvotní stanice nastaví bit P do stavu l, jestliže žádá od druhotné stanice odpověď. Naopak druhotná stanice nastaví bit F do stavu l v poli C posledního rámce své odpovědi. Tento způsob komunikace mezi stanicemi ovsem vyžaduje, aby prvotní stanice před vysláním rámce s bitem P ve stavu l dostala od druhotné stanice potvrzení (tj. rámec s bitem F ve stavu 1) na předcházející rámec s bitem P ve stavu 1. Jinými slovy, v každém okamžiku může na datovém spoji existovat pouze jeden nepotvrzený rámec s bitem P ve stavu 1.

bit	0	1	2	3	4	5	6	7
informační formát	Nr			P/F	Ns			0
dohlížecí formát	Nr			P/F	[1]		0	1
nečíslovaný formát	[2]			P/F	[2]		1	1

Formát řídicího pole rámce

1 Kód dohlížecího povelu (odpovědi)

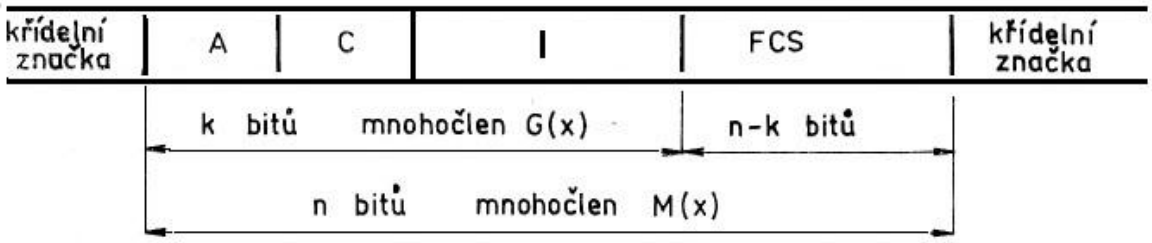
2 Kód nečíslovaného povelu (odpovědi)

Samozřejmě provoz může být plně duplexní, tj. i druhotná stanice může současně přenášet informaci do prvotní stanice. Tím se však nijak nemění úloha bitu P/F. K potvrzování rámců slouží stavy čítačů Nr a Ns, nikoli bit P/F.

Jak je patrno z formátu pole C (viz obrázek), existují tři typy rámců: informační, dohlížecí a nečíslovaný. Pro vlastní přenos informací slouží informační typ rámce, další dva typy rámců se užívají pro řízení činnosti na datovém okruhu.
Řídicí postupy HDLC umožňují v případě potřeby rozšířit pole A i pole C. Tím se jednak zvětší počet adresovatelných stanic, jednak počet řídicích a dohlížecích povelů i odpovědí.

Zabezpečení přenášené informace

Bezchybný příjem zprávy se ověřuje pomocí rámcové zabezpečovací posloupnosti přenášené v poli FCS. Posloupnost FCS je generována vysílačem, připojena k vysílané zprávě a prověřována přijímačem. Umístění FCS v rámci je patrno z obrázku:



Zabezpečení rámce

Zabezpečení přenášené zprávy se děje takto:

1. Bity zprávy, která má být zabezpečena (počet těchto bitů je *k),* se vyjádří ve tvaru mnohočlenu *G (x)* tak, že binární hodnoty jednotlivých bitů tvoří koeficienty mnohočlenu. Při tom se bitu následujícímu bezprostředně za počáteční křídelní značkou přiřadí nejvyšší řád bez ohledu na význam bitů ve vlastní zprávě.

2. Je definován tzv. generační mnohočlen *P (x)* stupně n — *k,* který má hodnotu $P(x) = x^{16} + x^{12} + x^5 + 1$. *(n* je počet bitů zabezpečené zprávy).

3. Posloupnost FCS je definována jako jedničkový doplněk zbytku po dělení *R (x),* který se získá dělením modulo 2 výrazu $x^{n-k} . G(x) + x^k(x^{15} + x^{14} + \ldots + x + 1)$ generačním mnohočlenem *P (x).* Platí tedy

$$\frac{x^{n-k} . G(x) + x^k (x^{15} + x^{14} \ldots + x + 1)}{P(x)} = Q(x) + \frac{R(x)}{P(x)},$$

kde *Q(x)* je podíl a *R(x)* je zbytek po dělení.

Tyto zdánlivě složité matematické operace je možno jednoduše realizovat pomocí posuvných registrů se zpětnými vazbami. Jak je z definice patrné, je zbytek po dělení *R(x)* nejvýše řádu 15, tedy FCS bude mít délku 16 bitů. Násobení x^{n-k} . *G(x)* znamená vlastně posun zprávy o *n-k* (tedy 16) bitů vlevo (směrem k vyšším řádům), což se uskuteční automaticky tím, že se za zprávu vyjádřenou mnohočlenem *G(x)* připojí FCS. Přičtení výrazu $x^k (x^{15} + x^{14} + \ldots + x + 1)$ odpovídá nastavení počáteční hodnoty zbytku po dělení (tj. obsahu posuvného registru realizujícího požadované výpočty) na 1111 ... 111 (»samé jednicky«). Kdyby se totiž vycházelo z počáteční hodnoty zbytku nulové, nebylo by možno detekovat ztrátu, nebo naopak přidání bitů s hodnotou 0 na začátku zprávy bezprostředně za počáteční křídelní značkou.

Obdobně skutečnost, že FCS je definována jako jedničkový doplněk zbytku po dělení *R(x),* zabraňuje nedetekované ztrátě, nebo přidání bitů s hodnotou 0 na konci zprávy před posloupností FCS.

Na přijímací straně probíhá obdobný proces, jako na straně vysílací. Jestliže byla zpráva přenesena bez chyby, bude mít zbytek po dělení vždy konstantní hodnotu 0001110100001111 (od řádu x^{15} po x°) v okamžiku ukončení příjmu pole FCS. Bude-li tedy zbytek po dělení mít jinou hodnotu, znamená to, že zpráva byla přenesena chybně. V ojedinělých případech může i při chybně přenesené zprávě vzniknout správný zbytek po dělení. Pak ovšem nebude chyba detekována. Tato pravděpodobnost je však velmi malá.

6.2 Komunikace na ISM pásmech (Communication on ISM bands) a předpisy pro vysílání vysilači malých výkonů (Low Power Transmitter Legislation)

Výstižně popisuje ISM http://cs.wikipedia.org/wiki/P%C3%A1smo_ISM **Pásma ISM** (industrial, scientific and medical) jsou pásma pro rádiové vysílání v oborech průmyslovém, vědeckém a zdravotnickém. Jsou to pásma volná, což znamená že je v nich při použití homologovaného (schváleného) zařízení dovolen provoz bez licenčních poplatků, avšak bez garance proti rušení. Podmínky pro provoz takovýchto zařízeních jsou stanoveny generálními licencemi, které vydal Český telekomunikační úřad (GL-12/R/2000, případně GL-30/R/2000).
Tato pásma používají např. radioamatéři nebo vysílačky RC modelů apod., vedle těchto využití se využívají také komerčně.
Použití vysílačů ISM obvykle předpokládá použití homologovaných vysílačů. Předpokládáme-li použití našeho vysílače jako zařízení pro pásmo ISM doporučuji prostudovat výše uvedené předpisy v jejich aktuální verzi. Při soutěžích mimo území ČR ovšem musíme dodržovat předpisy platná pro příslušná území.
Další možností, kterou vřele doporučuji, je provozovat Cansat vysílače pod radioamatérskou koncesí. I zde musí vysílač splňovat jisté minimální technické podmínky, není však požadována jejich homologizace. Účelem radioamatérských licencí je mj. i technické vzdělávání jejich držitelů a to včetně vývoje a provozování vysílacích zařízení. Této možnosti využívá i autor těchto skript a soutěžní tým SPŠE Ječná. Pro případné zájemce z jiných škol doporučuji navštívit stránky Českého radioklubu http://www.crk.cz/CZ a Českého telekomunikačního úřadu http://www.ctu.cz/ctu-informuje/jak-postupovat/radiove-kmitocty/informace-o-vyuzivani-radiovych-kmitoctu.html , popř. též kontaktovat některý z místních radioklubů. Radioamaterské vysílání musí splňovat podmíky vyhlášky č.156/2005 Sb. o technických a provozních podmínkách amatérské radiokomunikační služby. Ze stránek Českého telekomunikačního úřadu si ještě uvedeme:

Amatérská služba

Vysílací rádiové zařízení amatérské služby je určeno ke sportovní činnosti, technickému sebevzdělávání a studiu. Žadatel o vydání individuálního oprávnění k využívání kmitočtů amatérské služby musí být držitelem průkazu odborné způsobilosti pro obsluhu amatérských vysílacích rádiových stanic odpovídající třídy, jehož vydání je podmíněno úspěšným vykonáním zkoušky odborné způsobilosti k obsluze podle § 26 zákona. V souladu s přijetím Doporučení CEPT T/R 61-01 (The Conference of European Postal and Telecommunications Administrations) Českou republikou, mohou být amatérská rádiová vysílací zařízení na základě individuálního oprávnění k využívání rádiových kmitočtů vydaného Úřadem krátkodobě (maximálně tři měsíce) provozována v zemích, které na tuto úmluvu přistoupily, aniž by jejich držitelé museli zažádat o vydání jakéhokoliv dalšího oprávnění.
Cizí státní příslušníci mohou požádat o vydání individuálního oprávnění k využívání rádiových kmitočtů pro amatérskou radiokomunikační službu v České republice. Žádost o udělení individuálního oprávnění doloží kopií stávajícího platného oprávnění vydaného v zemi trvalého bydliště žadatele k vysílání v příslušné zemi.
Amatérské vysílací rádiové zařízení se nesmí používat k podnikatelské činnosti.

7.Obsah DVD přílohy

Datasheet vysílače Pratt Hobbies
Datashett Analog Devices
Datasheet Atmel ATMEGA
Terminálový program
Firmware
AGW engine
AGW uživatelský software
Zdrojové kódy pro .NET nad AGW engine
Datasheet Freescale Semiconductors čidla tlaku
Webcam
Diplomové práce z VUT Brno o APRS

Literatura

[1] Váňa V.: ATMEL AVR. Programování v jazyce C, BEN Praha 2003, ISBN 80-7300-102-0
[2] Váňa V.: ATMEL AVR. Popis procesoru a instrukční soubor. BEN Praha 2003, ISBN 80-7300-083-0
[3] Váňa V.: ATMEL AVR.Assembler. BEN Praha 2003, ISBN 80-7300-093-8
[4] Váňa V.: ATMEL AVR. Pascal, BEN Praha 2004, ISBN 80-7300-113-6
[5] Váňa V.: ATMEL AVR. Programování v jazyce Bascom. BEN Praha 2004. ISBN 80-7300-115-2
[6] Váňa V.: Začínáme s HC08, BEN Prana
[7] Váňa V.: ARM pro začátečníky. BEN Praha 2009, ISBN 978-80-7300-246-6
[8] Václavík R., Lajšner P.: Packet Radio od A skoro až do Z. BEN Praha 1996,ISBN 80-901984-8-1
[9] Frejlach K.: Internet, Amprnet a paket-radio. Praha 2000. ISBN 80-238-52-X
[10] Frejlach K.: Nové režimy radioamatérského provozu. Praha 2001. ISBN 80-238-6814-4
[11] Frejlach K.: Praktická příručka pro paket-radio. Praha 1997. ISBN 80-238-0689-0
[12] Frejlach K.: Digitální radioamatérský provoz. Praha 1998. ISBN 80-238-2039-7
[13] http://www.pratthobbies.com
[14] Torstein Wang, Dag Martin Nilsen: The CANSAT Book.NAROM, Jan 29, 2010
[15] Dag-Martin Nilsen: Can Sat Data Handling and transmission. ESA ESTAC Presentation. Febr. 2010
[16] http://agwpeport.codeplex.com
[17] http://eludium.stensat.org/mcguire/projects/FX-25/FX-25_performance.htm
[18] http://www.aprs.org/doc/APRS101.PDF
[19] http://www.sv2agw.com/ham/agwpe.htm
[20] http://www.spsejecna.net/CANSAT
[21] http://www.spsejecna.net/CANSAT/STRETECH2010/index.htm
[22] http://arctanb.wordpress.com/2010/08/21/esa-cansat-competition-2010
[23] Talanda O.:"GPS Sledovací systém pro automobil", bakalářská práce FE VUT Brno 2008. http:// www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=7612
[24] Pokorný V.: Vývojový modul s 32bitovým procesorem typu ARM. Bakalářská práce VUT Brno 2009 http://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=18710
[25] Jůn L.: Vývojový modul s 32bitovým procesorem typu ARM. Diplomová práce VUT Brno 2009 www.urel.feec.vutbr.cz/~fryza/downloads/dp_jun_09.pdf
[26] Sabol Martin: „FM VYSÍLAČ APRS TELEMETRICKÝCH DAT V PÁSMU 144MHZ", Diplomová práce VUT Brno 2010 www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=26883
[27] Bohátka Jan: „FM vysílac APRS telemetrických dat v pásmu 144MHz", Diplomová práce VUT Brno 2011 http://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=38165

[28] Klíma Martin: „FM VYSÍLAČ APRS TELEMETRICKÝCH DAT V PÁSMU 435 MHZ“, Diplomová práce VUT Brno 2011 viz http://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=38201
[29] http://www.garydion.com/projects/whereavr
[30] http://www.aprs.cz/
[31] http://wiki.argentdata.com/index.php/OpenTracker_USB
[32] http://www.ui-view.org/

Dodatek 1 - firmware

Soubor AX25.c

```
#define TxDxData    0b000111111

#define COUNTER_BIT           115   //=2^8-(827us/(TCCR0/11,0592Mhz))
#define COUNTER_1200    115         //=2^8-(833us/(TCCR2/11,0592Mhz))
#define COUNTER_2200    177         //=2^8-(454us/(TCCR2/11,0592Mhz))

int txDelay=50;               // Number of 6.7ms delay cycles (send flags)
//************************************************************************
// Variables
//************************************************************************
unsigned char loop_delay;
static char Phase,Clk;
unsigned int Counter2;
static unsigned int crc;
extern char message_gps;

void DelayBit(void);
void ChangePhase(void);
void ax25sendHeader(void);
void ax25sendByte(unsigned char txbyte);
void ChangeFreq(void);
void ax25crcBit(int lsb_int);
void ax25address(unsigned char txbyteA);

//************************************************************************
// Interrupt
//************************************************************************
ISR (TIMER2_OVF_vect){
      TCNT2 = Counter2;
      ChangePhase();
      }

//************************************************************************
// Functions
//************************************************************************
void ax25sendHeader(void){
      crc = 0xFFFF;
      //Initialize the crc register
      UCSR0B &= ~(1<<RXCIE0);
//RX0 Complete Interrupt Disable
//    UCSR1B &= ~(1<<RXCIE1);
//RX1 Complete Interrupt Disable

      DelayBit();
```

```
// Pause for the bit to be sent

      Phase=FALSE;
      Counter2=COUNTER 1200;

      TCNT2 = COUNTER_1200;
      TCCR2B |= ((0<<CS22) | (1<<CS21) | (1<<CS20));
      //ma byt TCCR0 = 0b00000011         CLK/64
      TIMSK2 |= (1<< TOIE2);
      //Timer/Counter2 Overflow Inerrupt Enable

      for (loop_delay = 0 ; loop_delay < txDelay ; loop_delay++)
            {
            (ax25sendByte(0x7E));
            }
}

//*************************************************************************/
void ax25sendByte(unsigned char txbyte){
      char loop;
      static char bitbyte;
      static char bit_zero;
      static unsigned char sequential_ones;

      bitbyte = txbyte;       // Bitbyte will be rotated through
      for (loop = 0 ; loop < 8 ; loop++)  // Loop for eight bits in the byte
            {
            bit_zero = bitbyte & 0x01;// Set aside the least significant
bit
            if (txbyte == 0x7E)     // Is the transmit character a flag?
                  {
                  sequential_ones = 0;// it is immune from sequential 1's
                  }
                  else  // The transmit character is not a flag
                        {
                        (ax25crcBit(bit_zero)); // So modify the checksum
                        }

            if (!(bit_zero))        // Is the least significant bit low?
                  {
                  sequential_ones = 0;// Clear the number of ones we have
sent
                  ChangeFreq();
                  }
                  else        // Else, least significant bit is high
                        {
                        if (++sequential_ones >= 5)         // Is this the
5th "1" in a row?
                              {
                              DelayBit(); // Go ahead and send it
                              ChangeFreq();
                              sequential_ones = 0;          // Clear the
number of ones we have sent
                              }
                        }
            bitbyte >>= 1;// Shift the reference byte one bit right
            DelayBit(); // Pause for the bit to be sent
      }
}
```

```
//*********************************************************************
void DelayBit(void){
      TCNT0 = COUNTER BIT;
      TCCR0B |= ((0<<CS02) | (1<<CS01) | (1<<CS00)); //ma byt
TCCR0=0b00000011  CLK/64

      loop_until_bit_is_set(TIFR0,0);
      TIFR0 = TIFR0 & 0b00000001;
      TCCR0B |= ((0<<CS02) | (0<<CS01) | (0<<CS00)); //off
      }

//*********************************************************************
void ax25sendBye(void){

      ax25sendByte(0x7E);                 // Send a flag to end the packet

      TCCR0B = 0x00;                      //Timer/Counter stopped
      TIMSK2 = 0x00;                      //Timer/Counter2 Overflow Inerrupt
disadle
      TCCR2B = 0x00;                //Timer/Counter stopped

      message_gps = 0;
      UCSR0B |= (1<<RXCIE0);        //RX0 Complete Interrupt Enable
//    UCSR1B |= (1<<RXCIE1);        //RX1 Complete Interrupt Enable
      }

//*********************************************************************
void ChangeFreq(void){
      if (Counter2 == COUNTER_1200)
            {
            Counter2 = TCNT2;
            TCNT2 =((((Counter2 - COUNTER_1200)>>1)) + COUNTER_2200);
            Counter2 = COUNTER_2200;                        // 2200 Hz
            }
            else
                  {
                  Counter2 = TCNT2;
                  if (Counter2 >= 247)          //(Counter2 >= 252)
                        {
                        TCNT2 = COUNTER_1200;
                        ChangePhase();
                  //    TCNT2 = COUNTER_1200+(((((Counter2 -
COUNTER_2200)<<1)) + COUNTER_1200)-256);
                        }
                        else
                              TCNT2 = ((((Counter2 - COUNTER_2200)<<1)) +
COUNTER_1200);
                  Counter2 = COUNTER_1200;                  // 1200 Hz

                  }
}

//*********************************************************************
void ax25crcBit(int lsb_int){
      static unsigned short   xor_int;

      xor_int = crc ^ lsb_int;      // XOR lsb of CRC with the latest bit
      crc >>= 1;                    // Shift 16-bit CRC one bit to the right
```

```
      if (xor_int & 0x0001)   // If XOR result from above has lsb set
            {
            crc ^= 0x8408;    // Shift 16-bit CRC one bit to the right
            }
}

//***************************************************************************
void ax25address(unsigned char txbyteA){
      ax25sendByte(txbyteA<<1);
      }

//***************************************************************************
void m_puts(unsigned char *text2 ){
      unsigned char i=0,temp2;
      do
            {
            temp2 = text2[i];
            if(temp2==0) break;
            ax25sendByte(temp2);
            i++;
            }
      while(temp2>0);
  }

//***************************************************************************
void ChangePhase(void){
      if (Phase == TRUE)
            {
            PORTB |= TxDxData;            //PC2 = 1
            Phase = FALSE;
            }
        else
            {
            PORTB &= ~TxDxData;       //PC2 = 0;
            Phase = TRUE;
            }
      }
//***************************************************************************
void ax25sendCrc(void){
      static unsigned char    crchi;

      crchi = (crc >> 8)^0xFF;
      ax25sendByte(crc^0xFF);             // Send the low byte of the crc
      ax25sendByte(crchi);                // Send the high byte of the crc
      ax25sendByte(0x7E);                 // Send a flag to end the packet

      crc = 0xFFFF;
      }
```

Soubor GPS.c

```
//***************************************************************************
//* Variables
//***************************************************************************
extern int Temp;
```

```
extern unsigned char mSecondas_clo, Secondas_clo, Minutes_clo, Hours_clo,
Days_clo, Months_clo,Years_clo;

char buffer[6];
char message_gps;
static unsigned char mSecondas_gps, Secondas_gps, Minutes_gps, Hours_gps,
Days_gps, Months_gps,Years_gps;
static unsigned char commacount;
static char bufferindex;
static char pointcount;
static char line;
static int Temp3;

char LatitudeP_gps=' ';
char LongitudeP_gps=' ';
int LatitudeGps=0;
int LatitudeM_gps=0;
int LongitudeGps=0;
int LongitudeM_gps =0;
int Course_gps=0;
int Speed_gps=0;
int Altitude_gps=0;

void NMEA(void);
void message_GPRMC(void);
void message_GPGGA(void);

//**************************************************************************
void NMEA(void){
switch(Temp)
  {
      case '$':
            commacount = 0;               //počítá znaky
            bufferindex = 0;        //číslo znaku v řetezci
            message_gps =0;               //typ věty
            pointcount = FALSE;     //jestli hodnota obsahuje desetinou část
            line = 0;               //která hodnota z věty je právě vysílana
            break;
      case ',':                     //we dont break as the code is shared
            commacount = 0;         //note this means packet header is 1
            bufferindex = 0;        //wipe all these so they can be reused
            memset(buffer,' ',6);
            pointcount = FALSE;
            line++;
            break;
      case '.':
            Temp3 = atoi(buffer);
            commacount++;                 //we need to be able to detect
number of points in the case of altitude
            bufferindex = 0;
            memset(buffer,' ',6);
            pointcount = TRUE;
            break;
      default:
            commacount++;
            bufferindex++;          //increase the position in the buffer
            if(bufferindex<6)
                  {
                  buffer[bufferindex] = Temp;
                  if (message_gps==1) message_GPRMC();
```

```
                  if (message_gps==2) message_GPGGA();
                  }

            if((buffer[3] == 'R') && (buffer[4] == 'M') && (buffer[5] ==
'C'))
              {
                  message_gps=1;
                  }
            if((buffer[3] == 'G') && (buffer[4] == 'G') && (buffer[5] ==
'A'))
              {
                  message_gps=2;
                  }
            break;
      }
}

//*********************************************************************
void message_GPRMC(){
switch(line){
      case 1:
            {
            switch(commacount)
                  {
                  if (commacount > 11) commacount =0;
                  case 2: Hours_gps = (atoi(buffer));      //UTC + 1hodina

                    bufferindex = 0;
                    break;
                  case 4: Minutes_gps = atoi(buffer);
                    bufferindex = 0;
                    break;
                  case 6: Secondas_gps = atoi(buffer);
                    bufferindex = 0;
                    break;
                  case 7 : bufferindex = 0;
                    break;
                  case 9:mSecondas_clo = atoi(buffer);
                    if ((Hours_gps < 24) && (Minutes_gps < 60) &&
(Secondas_gps < 60))
                      {
                      Hours_clo = Hours_gps;
                      Minutes_clo = Minutes_gps;
                      Secondas_clo = Secondas_gps;
                      mSecondas_clo = mSecondas_gps;
                        }
                    break;
                  }
            }
            break;
      case 3:
            {
            switch (commacount)
                  {
                  case 4 : LatitudeGps = atoi(buffer);
                        bufferindex = 0;
                        memset(buffer,' ',6);
                        break;
                  case 8 : LatitudeM_gps = atoi(buffer);
                        break;
```

```
                  default : break;
                  }
            }
            break;
      case 4:
            if (Temp =='N') LatitudeP_gps='N';
                  else LatitudeP_gps='S';
            break;
      case 5:
            {
            if (commacount == 5)
                  {
                  LongitudeGps =  atoi(buffer);
                  }
            if (commacount == 8)
                  {
                  LongitudeM_gps =  atoi(buffer);
                  }
            }
            break;
      case 6:
            {
            if (Temp =='E') LongitudeP_gps='E';
                  else LongitudeP_gps='W';
            }
            break;
      case 7:
            if (pointcount) Speed_gps = Temp3;
            break;
      case 8:
            if (pointcount) Course_gps = Temp3;
            break;
      case 9:
            {
            if (commacount > 8) commacount =0;
            switch(commacount)
                  {
                  case 2: Days_gps = atoi(buffer);
                    bufferindex = 0;
                    break;
            case 4: Months_gps = atoi(buffer);
                    bufferindex = 0;
                    break;
                  case 6: Years_gps = atoi(buffer);
                    bufferindex = 0;
                    if ((Months_gps <=12) && (Days_gps <=31) && (Years_gps
>=10))
                        {
                        Months_clo = Months_gps;
                        Days_clo = Days_gps;
                        Years_clo = Years_gps;
                        }
                    break;
                  default: break;
                  }
            }
            break;
      default:  break;
      }
}
```

```
//*************************************************************************

void message GPGGA(){
switch(line)
    {
      case 8:
            bufferindex = 0;
            break;
      case 9:
            if (pointcount) Altitude_gps = Temp3;
            break;
      default: break;
      }
}
```

Dodatek 2 – ATMega88 firmware Hex výpis programové paměti Flash TXu Stensat (PrattHobbies)

```
00000000 19 C0 33 C0 AF C0 31 C0 30 C0 2F C0 2E C0 2E C0
00000010 2C C0 2B C0 2A C0 57 C0 28 C0 27 C0 26 C0 25 C0
00000020 24 C0 23 C0 22 C0 21 C0 20 C0 1F C0 1E C0 1D C0
00000030 1C C0 1B C0 11 24 1F BE CF EF D4 E0 DE BF CD BF
00000040 11 E0 A0 E0 B1 E0 EA E8 F5 E1 02 C0 05 90 0D 92
00000050 AC 31 B1 07 D9 F7 12 E0 AC E1 B1 E0 01 C0 1D 92
00000060 A1 30 B1 07 E1 F7 98 D6 8F CA CA CF 1F 92 0F 92
00000070 0F B6 0F 92 11 24 8F 93 9F 93 AF 93 BF 93 EF 93
00000080 FF 93 CF 93 DF 93 CD B7 DE B7 A5 E2 B0 E0 E5 E2
00000090 F0 E0 90 81 81 E0 89 27 8C 93 A3 EB B0 E0 E3 EB
000000A0 F0 E0 90 81 80 91 2E 01 89 0F 8C 93 DF 91 CF 91
000000B0 FF 91 EF 91 BF 91 AF 91 9F 91 8F 91 0F 90 0F BE
000000C0 0F 90 1F 90 18 95 1F 92 0F 92 0F B6 0F 92 11 24
000000D0 8F 93 9F 93 AF 93 BF 93 CF 93 DF 93 CD B7 DE B7
000000E0 81 E0 80 93 21 01 80 91 2D 01 81 30 E9 F4 80 91
000000F0 2C 01 81 30 B1 F4 10 92 38 01 10 92 37 01 80 91
00000100 20 01 88 23 39 F4 80 E5 80 93 2E 01 81 E0 80 93
00000110 20 01 05 C0 8C E2 80 93 2E 01 10 92 20 01 10 92
00000120 2C 01 81 E0 80 93 30 01 80 91 23 01 90 91 24 01
00000130 A0 91 25 01 B0 91 26 01 01 96 A1 1D B1 1D 80 93
00000140 23 01 90 93 24 01 A0 93 25 01 B0 93 26 01 DF 91
00000150 CF 91 BF 91 AF 91 9F 91 8F 91 0F 90 0F BE 0F 90
00000160 1F 90 18 95 1F 92 0F 92 0F B6 0F 92 11 24 2F 93
00000170 3F 93 4F 93 5F 93 8F 93 9F 93 AF 93 BF 93 EF 93
00000180 FF 93 CF 93 DF 93 CD B7 DE B7 80 91 22 01 81 30
00000190 41 F4 A5 E2 B0 E0 E5 E2 F0 E0 80 81 81 60 8C 93
000001A0 08 C0 A5 E2 B0 E0 E5 E2 F0 E0 90 81 8E EF 89 23
000001B0 8C 93 80 91 1C 01 90 91 1D 01 A0 91 1E 01 B0 91
000001C0 1F 01 80 70 98 70 A0 70 B0 70 00 97 A1 05 B1 05
000001D0 21 F0 81 E0 80 93 2F 01 02 C0 10 92 2F 01 80 91
000001E0 1C 01 90 91 1D 01 A0 91 1E 01 B0 91 1F 01 80 70
```

```
000001F0 90 70 A1 70 B0 70 00 97 A1 05 B1 05 21 F0 81 E0
00000200 80 93 29 01 02 C0 10 92 29 01 80 91 2C 01 81 30
00000210 31 F4 80 91 20 01 91 E0 89 27 80 93 20 01 90 91
00000220 20 01 80 91 2F 01 98 27 80 91 29 01 89 27 80 93
00000230 22 01 80 91 1C 01 90 91 1D 01 A0 91 1E 01 B0 91
00000240 1F 01 9C 01 AD 01 22 0F 33 1F 44 1F 55 1F 80 91
00000250 22 01 99 27 87 FD 90 95 A9 2F B9 2F 82 2B 93 2B
00000260 A4 2B B5 2B 80 93 1C 01 90 93 1D 01 A0 93 1E 01
00000270 B0 93 1F 01 81 E0 80 93 30 01 DF 91 CF 91 FF 91
00000280 EF 91 BF 91 AF 91 9F 91 8F 91 5F 91 4F 91 3F 91
00000290 2F 91 0F 90 0F BE 0F 90 1F 90 18 95 CF 93 DF 93
000002A0 CD B7 DE B7 21 97 0F B6 F8 94 DE BF 0F BE CD BF
000002B0 89 83 E0 EC F0 E0 80 81 99 27 80 72 90 70 00 97
000002C0 C1 F3 E6 EC F0 E0 89 81 80 83 21 96 0F B6 F8 94
000002D0 DE BF 0F BE CD BF DF 91 CF 91 08 95 CF 93 DF 93
000002E0 CD B7 DE B7 24 97 0F B6 F8 94 DE BF 0F BE CD BF
000002F0 9B 83 8A 83 02 C0 89 81 D1 DF EA 81 FB 81 80 81
00000300 89 83 89 81 8C 83 8C 81 88 23 11 F0 81 E0 8C 83
00000310 8C 81 2A 81 3B 81 2F 5F 3F 4F 3B 83 2A 83 88 23
00000320 51 F7 24 96 0F B6 F8 94 DE BF 0F BE CD BF DF 91
00000330 CF 91 08 95 CF 93 DF 93 CD B7 DE B7 E0 EC F0 E0
00000340 80 81 88 23 DC F7 E6 EC F0 E0 80 81 99 27 87 FD
00000350 90 95 DF 91 CF 91 08 95 CF 93 DF 93 CD B7 DE B7
00000360 22 97 0F B6 F8 94 DE BF 0F BE CD BF E0 EC F0 E0
00000370 80 81 88 23 2C F0 8F EF 9F EF 9A 83 89 83 02 C0
00000380 1A 82 19 82 89 81 9A 81 22 96 0F B6 F8 94 DE BF
00000390 0F BE CD BF DF 91 CF 91 08 95 CF 93 DF 93 CD B7
000003A0 DE B7 26 97 0F B6 F8 94 DE BF 0F BE CD BF 9C 83
000003B0 8B 83 7E 83 6D 83 19 82 14 C0 89 81 28 2F 33 27
000003C0 8D 81 9E 81 28 17 39 07 64 F4 EB 81 FC 81 8A 81
000003D0 80 83 8B 81 9C 81 01 96 9C 83 8B 83 89 81 8F 5F
000003E0 89 83 A8 DF 8A 83 8A 81 8D 30 39 F7 89 81 99 27
000003F0 26 96 0F B6 F8 94 DE BF 0F BE CD BF DF 91 CF 91
00000400 08 95 CF 93 DF 93 CD B7 DE B7 24 97 0F B6 F8 94
00000410 DE BF 0F BE CD BF 9C 83 8B 83 8C DF 8A 83 8A 81
00000420 89 3C 10 F0 88 EC 8A 83 19 82 0C C0 83 DF EB 81
00000430 FC 81 80 83 8B 81 9C 81 01 96 9C 83 8B 83 89 81
00000440 8F 5F 89 83 99 81 8A 81 98 17 80 F3 24 96 0F B6
00000450 F8 94 DE BF 0F BE CD BF DF 91 CF 91 08 95 CF 93
00000460 DF 93 CD B7 DE B7 21 97 0F B6 F8 94 DE BF 0F BE
00000470 CD BF 89 83 EE E4 F0 E0 89 81 80 83 ED E4 F0 E0
00000480 80 81 88 23 DC F7 21 96 0F B6 F8 94 DE BF 0F BE
00000490 CD BF DF 91 CF 91 08 95 CF 93 DF 93 CD B7 DE B7
000004A0 25 97 0F B6 F8 94 DE BF 0F BE CD BF 6A 83 7B 83
000004B0 8C 83 9D 83 A5 E2 B0 E0 E5 E2 F0 E0 80 81 8B 7F
000004C0 8C 93 8A 81 9B 81 AC 81 BD 81 8B 2F 99 27 AA 27
000004D0 BB 27 89 83 89 81 C3 DF 8A 81 9B 81 AC 81 BD 81
000004E0 CD 01 AA 27 BB 27 89 83 89 81 B9 DF 8A 81 9B 81
000004F0 AC 81 BD 81 89 2F 9A 2F AB 2F BB 27 89 83 89 81
00000500 AE DF 8A 81 89 83 89 81 AA DF A5 E2 B0 E0 E5 E2
00000510 F0 E0 80 81 84 60 8C 93 25 96 0F B6 F8 94 DE BF
00000520 0F BE CD BF DF 91 CF 91 08 95 CF 93 DF 93 CD B7
00000530 DE B7 80 91 21 01 88 23 E1 F3 10 92 21 01 DF 91
00000540 CF 91 08 95 CF 93 DF 93 CD B7 DE B7 24 97 0F B6
```

```
00000550 F8 94 DE BF 0F BE CD BF 9C 83 8B 83 1A 82 19 82
00000560 06 C0 E3 DF 89 81 9A 81 01 96 9A 83 89 83 29 81
00000570 3A 81 8B 81 9C 81 28 17 39 07 9C F3 24 96 0F B6
00000580 F8 94 DE BF 0F BE CD BF DF 91 CF 91 08 95 CF 93
00000590 DF 93 CD B7 DE B7 22 97 0F B6 F8 94 DE BF 0F BE
000005A0 CD BF 8A 83 80 91 35 01 90 91 36 01 81 70 89 83
000005B0 80 91 35 01 90 91 36 01 95 95 87 95 90 93 36 01
000005C0 80 93 35 01 80 91 2A 01 90 91 2B 01 81 70 90 70
000005D0 88 23 51 F0 80 91 35 01 90 91 36 01 80 58 9F 4F
000005E0 90 93 36 01 80 93 35 01 80 91 2A 01 90 91 2B 01
000005F0 95 95 87 95 90 93 2B 01 80 93 2A 01 99 81 8A 81
00000600 89 27 81 30 C1 F4 20 91 35 01 30 91 36 01 88 E0
00000610 90 E0 82 27 93 27 90 93 36 01 80 93 35 01 20 91
00000620 2A 01 30 91 2B 01 84 E8 90 E0 82 27 93 27 90 93
00000630 2B 01 80 93 2A 01 22 96 0F B6 F8 94 DE BF 0F BE
00000640 CD BF DF 91 CF 91 08 95 CF 93 DF 93 CD B7 DE B7
00000650 25 97 0F B6 F8 94 DE BF 0F BE CD BF 8D 83 1C 82
00000660 1B 82 5B C0 8D 81 99 27 87 FD 90 95 81 70 90 70
00000670 9A 83 89 83 80 91 31 01 90 91 32 01 00 97 41 F4
00000680 80 91 27 01 90 91 28 01 00 97 11 F4 89 81 7F DF
00000690 89 81 9A 81 00 97 71 F4 81 E0 80 93 2C 01 80 91
000006A0 30 01 88 23 E1 F3 10 92 30 01 10 92 38 01 10 92
000006B0 37 01 2B C0 10 92 2C 01 80 91 30 01 88 23 E1 F3
000006C0 10 92 30 01 80 91 37 01 90 91 38 01 01 96 90 93
000006D0 38 01 80 93 37 01 80 91 27 01 90 91 28 01 00 97
000006E0 A1 F4 80 91 37 01 90 91 38 01 85 30 91 05 69 F4
000006F0 81 E0 80 93 2C 01 80 91 30 01 88 23 E1 F3 10 92
00000700 30 01 10 92 38 01 10 92 37 01 8D 81 85 95 8D 83
00000710 8B 81 9C 81 01 96 9C 83 8B 83 8B 81 9C 81 88 30
00000720 91 05 0C F4 9F CF 25 96 0F B6 F8 94 DE BF 0F BE
00000730 CD BF DF 91 CF 91 08 95 CF 93 DF 93 CD B7 DE B7
00000740 26 97 0F B6 F8 94 DE BF 0F BE CD BF 9C 83 8B 83
00000750 7E 83 6D 83 6F E5 70 EA 8A E5 90 E0 9D DE E8 E2
00000760 F0 E0 10 82 84 E6 90 E0 ED DE 10 92 20 01 8F EF
00000770 90 E0 90 93 36 01 80 93 35 01 8F EF 90 E0 90 93
00000780 2B 01 80 93 2A 01 10 92 38 01 10 92 37 01 81 E0
00000790 90 E0 90 93 28 01 80 93 27 01 10 92 32 01 10 92
000007A0 31 01 1A 82 19 82 07 C0 8E E7 4E DF 89 81 9A 81
000007B0 01 96 9A 83 89 83 89 81 9A 81 85 32 91 05 A0 F3
000007C0 10 92 28 01 10 92 27 01 1A 82 19 82 0C C0 89 81
000007D0 9A 81 FC 01 E0 50 FF 4F 80 81 36 DF 89 81 9A 81
000007E0 01 96 9A 83 89 83 89 81 9A 81 87 31 91 05 78 F3
000007F0 1A 82 19 82 0E C0 29 81 3A 81 8B 81 9C 81 F9 01
00000800 E8 0F F9 1F 80 81 20 DF 89 81 9A 81 01 96 9A 83
00000810 89 83 29 81 3A 81 8D 81 9E 81 28 17 39 07 58 F3
00000820 81 E0 90 E0 90 93 32 01 80 93 31 01 20 91 35 01
00000830 30 91 36 01 8F EF 90 E0 82 27 93 27 90 93 36 01
00000840 80 93 35 01 20 91 2A 01 30 91 2B 01 8F EF 90 E0
00000850 82 27 93 27 90 93 2B 01 80 93 2A 01 80 91 35 01
00000860 90 91 36 01 F1 DE 80 91 2A 01 90 91 2B 01 EC DE
00000870 10 92 32 01 10 92 31 01 81 E0 90 E0 90 93 28 01
00000880 80 93 27 01 8E E7 E0 DE 8E E7 DE DE 8E E7 DC DE
00000890 8E E7 DA DE 10 92 2C 01 E8 E2 F0 E0 88 E0 80 83
000008A0 67 E5 70 EA 8A E5 90 E0 F7 DD 26 96 0F B6 F8 94
```

```
000008B0 DE BF 0F BE CD BF DF 91 CF 91 08 95 CF 93 DF 93
000008C0 CD B7 DE B7 21 97 0F B6 F8 94 DE BF 0F BE CD BF
000008D0 81 E0 80 93 2D 01 E7 E2 F0 E0 8C E0 80 83 E8 E2
000008E0 F0 E0 88 E0 80 83 E4 E2 F0 E0 8F E2 80 83 A5 E2
000008F0 B0 E0 E5 E2 F0 E0 80 81 85 60 8C 93 EA E2 F0 E0
00000900 82 E0 80 83 E5 EC F0 E0 10 82 E4 EC F0 E0 83 E1
00000910 80 83 A1 EC B0 E0 E1 EC F0 E0 80 81 88 61 8C 93
00000920 E2 EC F0 E0 86 E8 80 83 EC E4 F0 E0 81 E5 80 83
00000930 ED E4 F0 E0 80 81 89 83 EE E4 F0 E0 80 81 89 83
00000940 E1 EB F0 E0 84 E0 80 83 E3 EB F0 E0 8A E3 80 83
00000950 8A E3 80 93 2E 01 E0 E8 F0 E0 10 82 E1 E8 F0 E0
00000960 89 E0 80 83 E8 E8 F0 E0 80 E0 98 E2 91 83 80 83
00000970 E2 EB F0 E0 10 82 E4 E8 F0 E0 11 82 10 82 EF E6
00000980 F0 E0 82 E0 80 83 E0 E7 F0 E0 82 E0 80 83 78 94
00000990 21 96 0F B6 F8 94 DE BF 0F BE CD BF DF 91 CF 91
000009A0 08 95 CF 93 DF 93 CD B7 DE B7 22 97 0F B6 F8 94
000009B0 DE BF 0F BE CD BF 89 E3 91 E0 68 EC 70 E0 ED DC
000009C0 99 27 9A 83 89 83 89 81 9A 81 29 E3 31 E0 BC 01
000009D0 C9 01 B2 DE 22 96 0F B6 F8 94 DE BF 0F BE CD BF
000009E0 DF 91 CF 91 08 95 CF 93 DF 93 CD B7 DE B7 2A 97
000009F0 0F B6 F8 94 DE BF 0F BE CD BF 8A 87 CE 01 04 96
00000A00 66 E0 70 E0 CA DC 99 27 9B 83 8A 83 8A 81 9B 81
00000A10 86 30 91 05 AC F4 8A 81 89 83 0F C0 89 81 28 2F
00000A20 33 27 27 FD 30 95 CE 01 04 96 FC 01 E2 0F F3 1F
00000A30 80 E2 80 83 89 81 8F 5F 89 83 89 81 86 30 74 F3
00000A40 19 82 1F C0 89 81 48 2F 55 27 47 FD 50 95 89 81
00000A50 28 2F 33 27 27 FD 30 95 CE 01 04 96 FC 01 E2 0F
00000A60 F3 1F 80 81 99 27 87 FD 90 95 88 0F 99 1F 28 2F
00000A70 CE 01 04 96 FC 01 E4 0F F5 1F 20 83 89 81 8F 5F
00000A80 89 83 89 81 86 30 F4 F2 8A 85 99 27 87 FD 90 95
00000A90 9C 01 20 50 3F 4F CE 01 04 96 46 E0 50 E0 BC 01
00000AA0 C9 01 15 D2 2A 96 0F B6 F8 94 DE BF 0F BE CD BF
00000AB0 DF 91 CF 91 08 95 CF 93 DF 93 CD B7 DE B7 DF 91
00000AC0 CF 91 08 95 CF 93 DF 93 CD B7 DE B7 2A 97 0F B6
00000AD0 F8 94 DE BF 0F BE CD BF CE 01 03 96 65 E0 70 E0
00000AE0 5C DC 99 27 9A 83 89 83 8B 81 81 33 99 F5 81 E0
00000AF0 80 93 2D 01 ED E3 F0 E0 10 82 62 EE 77 E3 80 E0
00000B00 90 E0 CA DC E1 EB F0 E0 84 E0 80 83 E3 EB F0 E0
00000B10 8A E3 80 83 8A E3 80 93 2E 01 E0 E8 F0 E0 10 82
00000B20 E1 E8 F0 E0 89 E0 80 83 E8 E8 F0 E0 80 E0 98 E2
00000B30 91 83 80 83 E2 EB F0 E0 10 82 E4 E8 F0 E0 11 82
00000B40 10 82 EF E6 F0 E0 82 E0 80 83 E0 E7 F0 E0 82 E0
00000B50 80 83 2D C0 8B 81 89 33 A1 F4 82 E0 80 93 2D 01
00000B60 66 EC 77 E4 81 E8 90 E0 97 DC E0 E7 F0 E0 10 82
00000B70 E9 E6 F0 E0 88 E0 80 83 ED E3 F0 E0 82 E0 80 83
00000B80 16 C0 8B 81 83 33 99 F4 82 E0 80 93 2D 01 66 EC
00000B90 77 EA 80 E1 90 E0 80 DC E0 E7 F0 E0 10 82 E9 E6
00000BA0 F0 E0 88 E0 80 83 ED E3 F0 E0 82 E0 80 83 2A 96
00000BB0 0F B6 F8 94 DE BF 0F BE CD BF DF 91 CF 91 08 95
00000BC0 CF 93 DF 93 CD B7 DE B7 60 97 0F B6 F8 94 DE BF
00000BD0 0F BE CD BF CE 01 03 96 68 E0 70 E0 DE DB 99 27
00000BE0 9A 83 89 83 29 81 3A 81 CE 01 03 96 FC 01 E2 0F
00000BF0 F3 1F 10 82 CE 01 0D 96 9F 93 8F 93 87 E1 91 E0
00000C00 9F 93 8F 93 CE 01 03 96 9F 93 8F 93 6F D1 8D B7
```

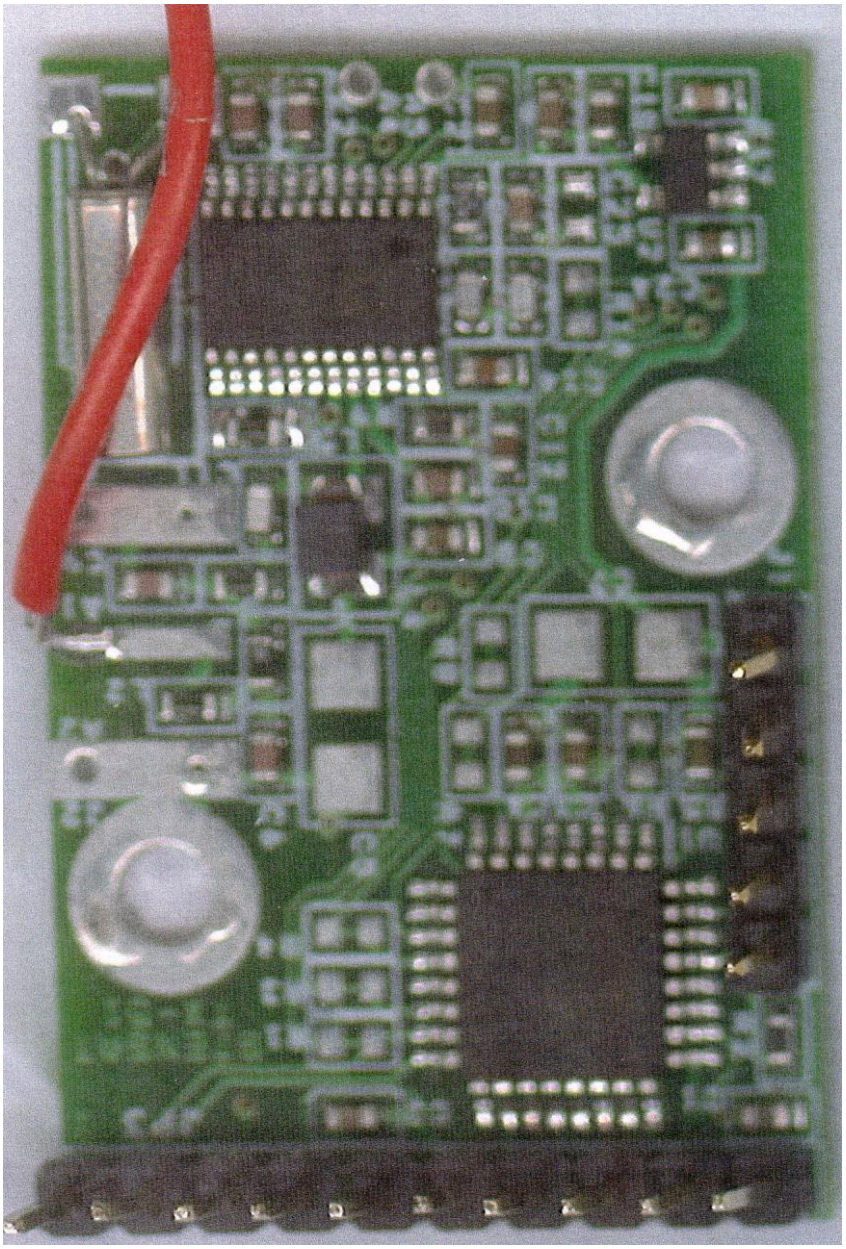
```
00000C10 9E B7 06 96 0F B6 F8 94 9E BF 0F BE 8D BF 8D 85
00000C20 9E 85 AF 85 B8 89 BC 01 CD 01 36 DC 60 96 0F B6
00000C30 F8 94 DE BF 0F BE CD BF DF 91 CF 91 08 95 CF 93
00000C40 DF 93 CD B7 DE B7 23 97 0F B6 F8 94 DE BF 0F BE
00000C50 CD BF 89 E3 91 E0 D5 DB 89 83 6F E5 70 EA 8A E5
00000C60 90 E0 1A DC E8 E2 F0 E0 10 82 84 E6 90 E0 6A DC
00000C70 10 92 20 01 8F EF 90 E0 90 93 36 01 80 93 35 01
00000C80 8F EF 90 E0 90 93 2B 01 80 93 2A 01 10 92 38 01
00000C90 10 92 37 01 81 E0 90 E0 90 93 28 01 80 93 27 01
00000CA0 10 92 32 01 10 92 31 01 1B 82 1A 82 07 C0 8E E7
00000CB0 CB DC 8A 81 9B 81 01 96 9B 83 8A 83 8A 81 9B 81
00000CC0 85 32 91 05 A0 F3 10 92 28 01 10 92 27 01 1B 82
00000CD0 1A 82 0C C0 8A 81 9B 81 FC 01 E7 5C FE 4F 80 81
00000CE0 B3 DC 8A 81 9B 81 01 96 9B 83 8A 83 89 81 28 2F
00000CF0 33 27 8A 81 9B 81 82 17 93 07 60 F3 81 E0 90 E0
00000D00 90 93 32 01 80 93 31 01 20 91 35 01 30 91 36 01
00000D10 8F EF 90 E0 82 27 93 27 90 93 36 01 80 93 35 01
00000D20 20 91 2A 01 30 91 2B 01 8F EF 90 E0 82 27 93 27
00000D30 90 93 2B 01 80 93 2A 01 80 91 35 01 90 91 36 01
00000D40 83 DC 80 91 2A 01 90 91 2B 01 7E DC 10 92 32 01
00000D50 10 92 31 01 81 E0 90 E0 90 93 28 01 80 93 27 01
00000D60 8E E7 72 DC 8E E7 70 DC 8E E7 6E DC 8E E7 6C DC
00000D70 10 92 2C 01 E8 E2 F0 E0 88 E0 80 83 67 E5 70 EA
00000D80 8A E5 90 E0 89 DB 23 96 0F B6 F8 94 DE BF 0F BE
00000D90 CD BF DF 91 CF 91 08 95 C8 EF D4 E0 DE BF CD BF
00000DA0 8D DD E7 E2 F0 E0 81 E0 80 83 60 E1 7E E5 88 E0
00000DB0 92 E0 72 DB 61 E0 74 ED 88 E0 90 E0 6D DB 62 EE
00000DC0 77 E3 80 E0 90 E0 68 DB 67 E5 70 EA 8A E5 90 E0
00000DD0 63 DB B0 DA 89 83 89 81 28 2F 33 27 27 FD 30 95
00000DE0 3F 83 2E 83 8E 81 9F 81 8D 34 91 05 09 F4 6B C0
00000DF0 2E 81 3F 81 2E 34 31 05 EC F4 8E 81 9F 81 83 34
00000E00 91 05 C1 F1 2E 81 3F 81 24 34 31 05 3C F4 8E 81
00000E10 9F 81 82 34 91 05 09 F4 58 C0 DB CF 2E 81 3F 81
00000E20 24 34 31 05 51 F1 8E 81 9F 81 86 34 91 05 09 F4
00000E30 48 C0 CF CF 2E 81 3F 81 24 35 31 05 31 F1 8E 81
00000E40 9F 81 85 35 91 05 5C F4 2E 81 3F 81 20 35 31 05
00000E50 B1 F1 8E 81 9F 81 83 35 91 05 A9 F0 BA CF 2E 81
00000E60 3F 81 25 35 31 05 F1 F0 8E 81 9F 81 86 35 91 05
00000E70 39 F0 AF CF 87 E0 B7 DD AC CF 80 E0 B4 DD A9 CF
00000E80 8E E0 B1 DD A6 CF 8D DD A4 CF E0 E7 F0 E0 10 82
00000E90 6F E5 70 EA 8A E5 90 E0 FF DA E8 E2 F0 E0 84 E0
00000EA0 80 83 97 CF 67 E5 70 EA 8A E5 90 E0 F5 DA E8 E2
00000EB0 F0 E0 88 E0 80 83 E0 E7 F0 E0 82 E0 80 83 FB DD
00000EC0 88 CF 7E DE 86 CF FE DD 84 CF B9 DE 82 CF FB 01
00000ED0 DC 01 41 50 50 40 48 F0 01 90 0D 92 00 20 C9 F7
00000EE0 01 C0 1D 92 41 50 50 40 E0 F7 08 95 A0 E1 B0 E0
00000EF0 EB E7 F7 E0 22 C3 85 E0 8E 83 8D 89 9E 89 9C 83
00000F00 8B 83 AE 01 47 5E 5F 4F 5A 83 49 83 6F 89 78 8D
00000F10 CE 01 03 96 03 D0 E2 E0 60 96 2B C3 A1 E1 B0 E0
00000F20 E3 E9 F7 E0 FA C2 66 24 77 24 43 01 1C 01 79 87
00000F30 68 87 5B 87 4A 87 FC 01 17 82 16 82 83 81 80 FD
00000F40 05 C0 2F EF 3F EF 3C 83 2B 83 42 C2 1E 82 1D 82
00000F50 4F EF 5F EF 5C 83 4B 83 44 24 55 24 1A 82 19 82
00000F60 FF 24 06 C2 AF 2C BB 24 A0 FE CA C1 15 32 19 F4
```

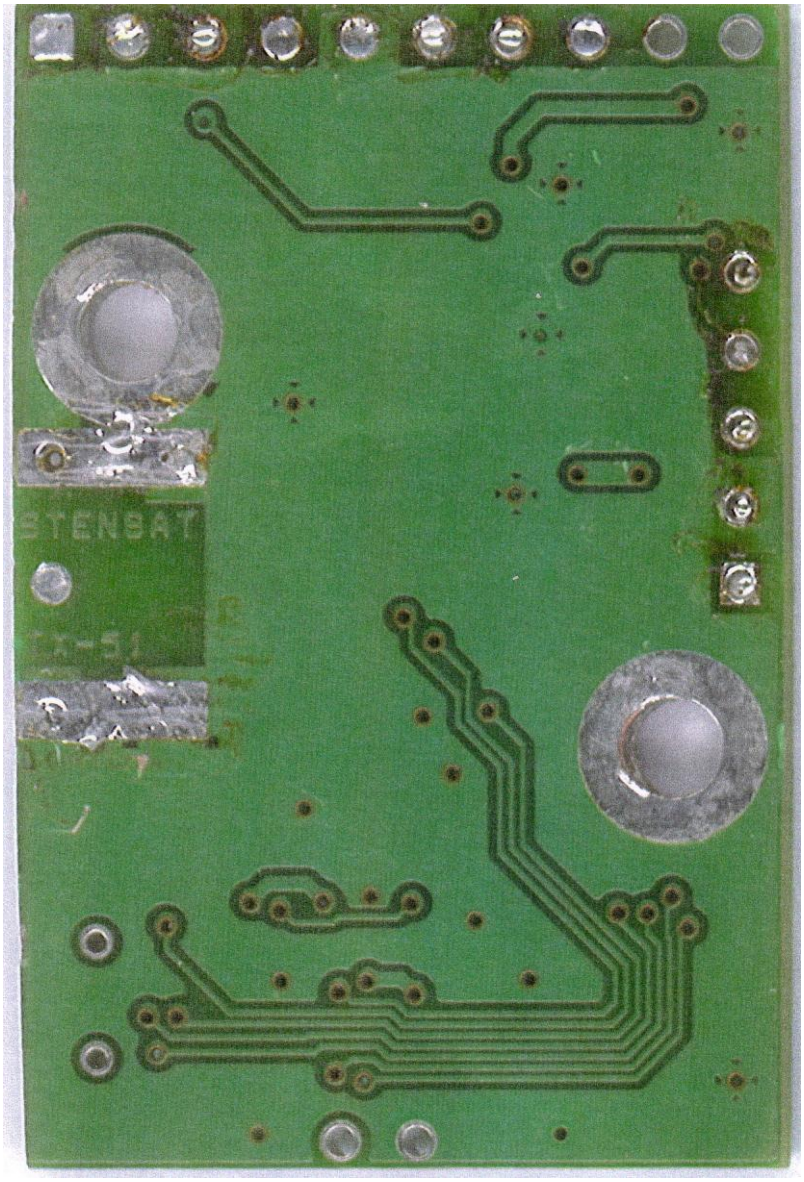
```
00000F70 5E EF F5 22 EA C1 91 2F 90 53 9A 30 60 F4 8F E7
00000F80 E8 16 11 F4 80 E0 04 C0 8A E0 E8 9E 80 2D 11 24
00000F90 E8 2E E9 0E E7 C1 81 2F 99 27 87 FD 90 95 8B D2
00000FA0 08 2F 8E 36 09 F4 7D C1 8F 36 AC F4 88 36 09 F4
00000FB0 D9 C1 89 36 4C F4 83 36 39 F1 84 36 09 F4 95 C0
00000FC0 8A 32 09 F0 7F C1 1A C0 89 36 09 F4 8E C0 8C 36
00000FD0 09 F0 78 C1 16 C0 83 37 09 F4 40 C0 84 37 3C F4
00000FE0 8F 36 09 F4 79 C0 80 37 09 F0 6C C1 7A C0 85 37
00000FF0 09 F4 79 C0 88 37 09 F0 65 C1 73 C0 90 E1 F9 2A
00001000 B1 C1 A4 E0 FA 2A AE C1 20 E1 C2 2E D1 2C CA 20
00001010 DB 20 A4 FC 07 C0 EA 85 FB 85 60 80 71 80 32 96
00001020 FB 87 EA 87 FF E7 EF 16 11 F4 EE 24 E3 94 53 01
00001030 0E C0 C1 01 D2 D1 2C 01 2F EF 8F 3F 92 07 09 F4
00001040 A5 C1 C1 14 D1 04 19 F4 D5 01 8D 93 5D 01 EA 94
00001050 8E 2D 8F 5F 18 16 6C F3 35 01 34 C1 90 E1 C9 2E
00001060 D1 2C CA 20 DB 20 A4 FC 07 C0 EA 85 FB 85 60 80
00001070 71 80 32 96 FB 87 EA 87 C1 01 AF D1 5C 01 0C D2
00001080 89 2B D1 F7 25 01 FF EF AF 16 FF EF BF 06 09 F4
00001090 7D C1 53 01 15 C0 C2 01 FF D1 89 2B 29 F0 35 01
000010A0 B1 01 C2 01 DD D1 12 C0 C1 14 D1 04 19 F4 D5 01
000010B0 4D 92 5D 01 C1 01 91 D1 2C 01 8F 5F 9F 4F 29 F0
000010C0 EA 94 8E 2D 8F 5F 18 16 34 F3 35 01 CD 28 09 F0
000010D0 F9 C0 F3 01 10 82 F6 C0 E2 E0 FE 2A F8 E0 FF 83
000010E0 04 C0 20 E1 2F 83 32 E0 F3 2A C1 01 76 D1 2C 01
000010F0 D3 D1 89 2B D1 F7 4F EF 44 16 4F EF 54 06 09 F4
00001100 45 C1 84 2D 8D 32 11 F0 8B 32 89 F4 EA 94 1E 14
00001110 0C F0 3C C1 8D 32 11 F4 58 E0 F5 2A C1 01 5D D1
00001120 2C 01 8F EF 48 16 8F EF 58 06 09 F4 2F C1 90 E3
00001130 49 16 61 F5 66 24 77 24 43 01 EA 94 1E 14 0C F0
00001140 79 C0 C1 01 4A D1 2C 01 AF EF 8F 3F 9A 07 09 F4
00001150 71 C0 B1 D1 88 37 B1 F4 0F 36 21 F0 04 36 11 F0
00001160 05 37 21 F4 B1 01 C2 01 7B D1 64 C0 C1 01 35 D1
00001170 2C 01 8F 5F 9F 4F 19 F4 E0 E1 EF 83 5B C0 F0 E1
00001180 FF 83 04 C0 09 36 11 F4 28 E0 2F 83 AA 24 BB 24
00001190 65 01 4F 81 55 27 59 8B 48 8B 5F 81 65 2E 77 24
000011A0 88 24 99 24 C2 01 87 D1 9C 01 20 53 30 40 2A 30
000011B0 31 05 34 F0 C9 01 C1 97 52 F0 27 52 30 40 02 C0
000011C0 37 FD 05 C0 88 89 99 89 28 17 39 07 34 F0 35 01
000011D0 46 01 B1 01 C2 01 44 D1 24 C0 C9 01 AA 27 97 FD
000011E0 A0 95 BA 2F 8C 87 9D 87 AE 87 BF 87 C6 01 B5 01
000011F0 A4 01 93 01 73 D1 5B 01 6C 01 2C 85 3D 85 4E 85
00001200 5F 85 A2 0E B3 1E C4 1E D5 1E EA 94 1E 14 3C F4
00001210 C1 01 E3 D0 2C 01 8F 5F 9F 4F 09 F0 C3 CF 35 01
00001220 46 01 F3 FE 07 C0 66 24 77 24 43 01 6A 18 7B 08
00001230 8C 08 9D 08 2F 2D 33 27 24 FD 44 C0 C9 01 86 70
00001240 90 70 06 97 51 F4 EA 85 FB 85 A0 81 B1 81 6D 92
00001250 7D 92 8D 92 9C 92 13 97 08 C0 21 FF 0A C0 EA 85
00001260 FB 85 A0 81 B1 81 6D 92 7C 92 32 96 FB 87 EA 87
00001270 29 C0 8A 85 9B 85 02 96 22 FF 0A C0 EA 85 FB 85
00001280 A0 81 B1 81 6D 92 7D 92 8D 92 9C 92 13 97 06 C0
00001290 EA 85 FB 85 A0 81 B1 81 6D 92 7C 92 9B 87 8A 87
000012A0 11 C0 A4 FC 0F C0 EA 85 FB 85 A0 81 B1 81 F1 01
000012B0 86 81 97 81 8D 93 9C 93 2A 85 3B 85 2E 5F 3F 4F
000012C0 3B 87 2A 87 F1 01 86 81 97 81 29 81 3A 81 28 17
```

```
000012D0 39 07 64 F4 F4 FC 06 C0 4D 81 5E 81 4F 5F 5F 4F
000012E0 5E 83 4D 83 1C 82 1B 82 FF 24 3C C0 0E 36 09 F0
000012F0 4D C0 5F EF 45 16 5F EF 55 06 09 F4 47 C0 F4 CF
00001300 15 32 69 F4 F1 01 06 80 F7 81 E0 2D FA 83 E9 83
00001310 FA E0 FF 83 8F E7 E8 2E FF 24 F3 94 23 C0 81 2F
00001320 99 27 87 FD 90 95 B8 D0 89 2B 79 F0 C1 01 55 D0
00001330 8C 01 B2 D0 89 2B D1 F7 28 01 2F EF 0F 3F 12 07
00001340 29 F1 B1 01 C8 01 8C D0 0D C0 C1 01 46 D0 2C 01
00001350 8F 5F 9F 4F D9 F0 81 2F 99 27 87 FD 90 95 48 16
00001360 59 06 A1 F4 48 85 59 85 4F 5F 5F 4F 59 87 48 87
00001370 F1 01 83 81 83 FF 04 C0 E8 85 F9 85 14 91 03 C0
00001380 A8 85 B9 85 1C 91 11 23 09 F0 EC CD 2F 2D 33 27
00001390 20 FF 11 C0 F1 01 86 81 97 81 49 81 5A 81 48 17
000013A0 59 07 4C F4 24 FD 05 C0 8D 81 9E 81 01 96 9E 83
000013B0 8D 83 1C 82 1B 82 9F EF 49 16 9F EF 59 06 21 F4
000013C0 AD 81 BE 81 AB 2B 21 F0 ED 81 FE 81 FC 83 EB 83
000013D0 8B 81 9C 81 E2 E1 61 96 BC C0 CF 93 DF 93 EC 01
000013E0 2B 81 82 2F 99 27 80 FF 15 C0 86 FF 0A C0 2F 7B
000013F0 2B 83 8E 81 9F 81 01 96 9F 83 8E 83 8A 81 99 27
00001400 2C C0 82 FF 12 C0 E8 81 F9 81 80 81 88 23 29 F4
00001410 20 62 2B 83 8F EF 9F EF 20 C0 28 2F 33 27 27 FD
00001420 30 95 31 96 F9 83 E8 83 11 C0 EA 85 FB 85 CE 01
00001430 09 95 9C 01 97 FF 0A C0 9B 81 2F 5F 3F 4F 11 F4
00001440 80 E1 01 C0 80 E2 89 2B 8B 83 E4 CF 8E 81 9F 81
00001450 01 96 9F 83 8E 83 82 2F 99 27 DF 91 CF 91 08 95
00001460 AC 01 FB 01 23 81 82 2F 99 27 80 FF 12 C0 86 FD
00001470 10 C0 8F EF 4F 3F 58 07 61 F0 42 83 20 64 2F 7D
00001480 23 83 86 81 97 81 01 97 97 83 86 83 84 2F 99 27
00001490 08 95 8F EF 9F EF 08 95 91 11 13 C0 80 32 51 F0
000014A0 8A 30 41 F0 8C 30 31 F0 8D 30 21 F0 89 30 11 F0
000014B0 8B 30 99 F7 08 95 8F 93 09 D0 8F 91 09 F0 80 62
000014C0 08 95 99 27 88 27 08 95 85 FD FB CF 80 62 91 11
000014D0 F8 CF 81 36 EC F3 8B 37 DC F7 08 95 62 9F D0 01
000014E0 73 9F F0 01 82 9F E0 0D F1 1D 64 9F E0 0D F1 1D
000014F0 92 9F F0 0D 83 9F F0 0D 74 9F F0 0D 65 9F F0 0D
00001500 99 27 72 9F B0 0D E1 1D F9 1F 63 9F B0 0D E1 1D
00001510 F9 1F BD 01 CF 01 11 24 08 95 2F 92 3F 92 4F 92
00001520 5F 92 6F 92 7F 92 8F 92 9F 92 AF 92 BF 92 CF 92
00001530 DF 92 EF 92 FF 92 0F 93 1F 93 CF 93 DF 93 CD B7
00001540 DE B7 CA 1B DB 0B 0F B6 F8 94 DE BF 0F BE CD BF
00001550 09 94 2A 88 39 88 48 88 5F 84 6E 84 7D 84 8C 84
00001560 9B 84 AA 84 B9 84 C8 84 DF 80 EE 80 FD 80 0C 81
00001570 1B 81 AA 81 B9 81 CE 0F D1 1D 0F B6 F8 94 DE BF
00001580 0F BE CD BF ED 01 08 95 FF CF 86 A2 40 40 40 40
00001590 60 96 88 68 90 84 9E 62 A8 8A 98 8A 9A 40 61 03
000015A0 F0 25 6C 78 00 00 FF FF FF FF FF FF FF FF FF FF
```

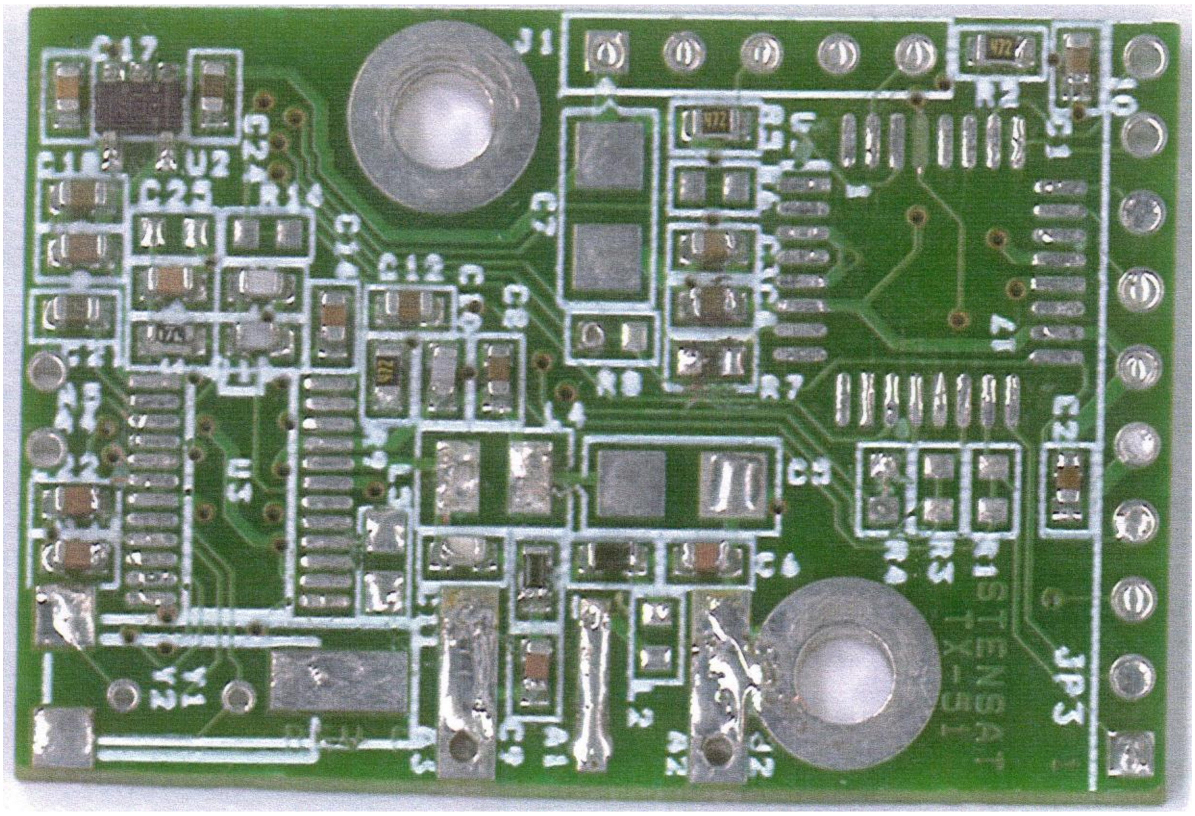
Dodatek 3 – PCB vysílače Stensat (Pratt Hobbies)

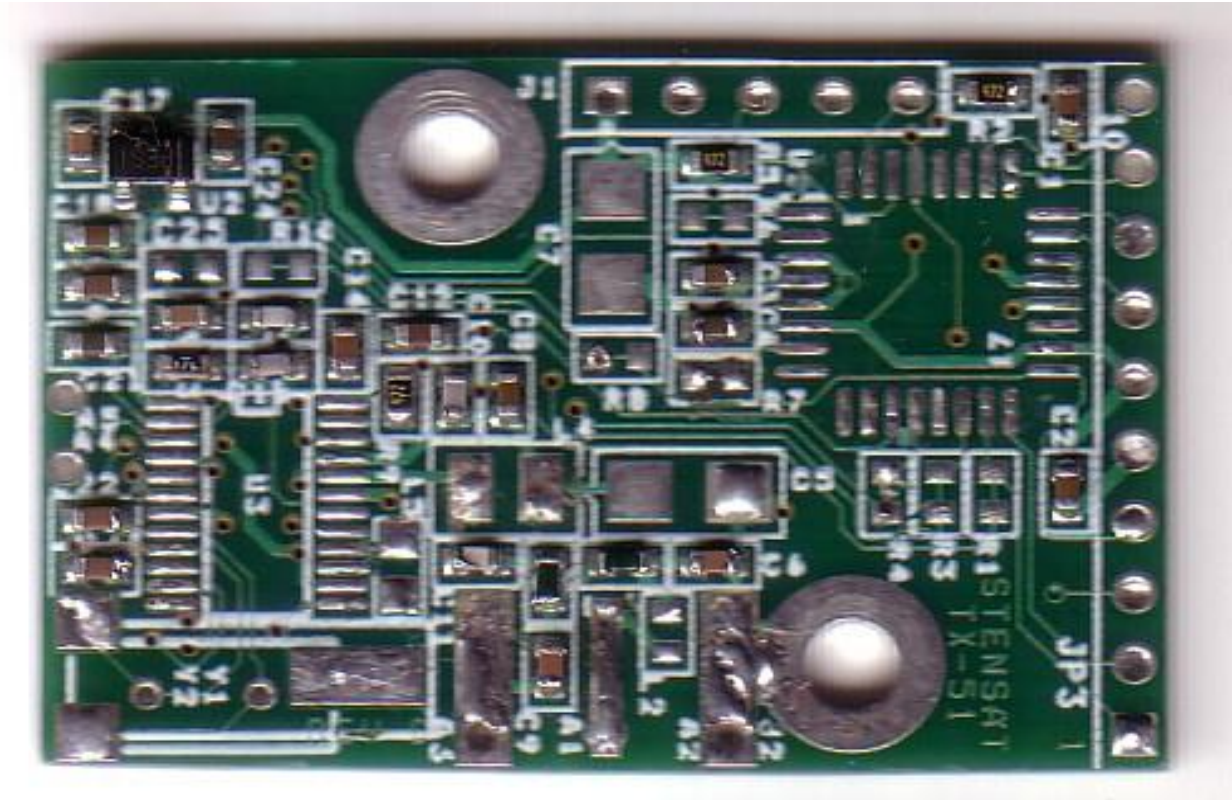
Uvedeme si několik snímků, umožňující návrh PCB clonu vysílače Pratt Hobbies.

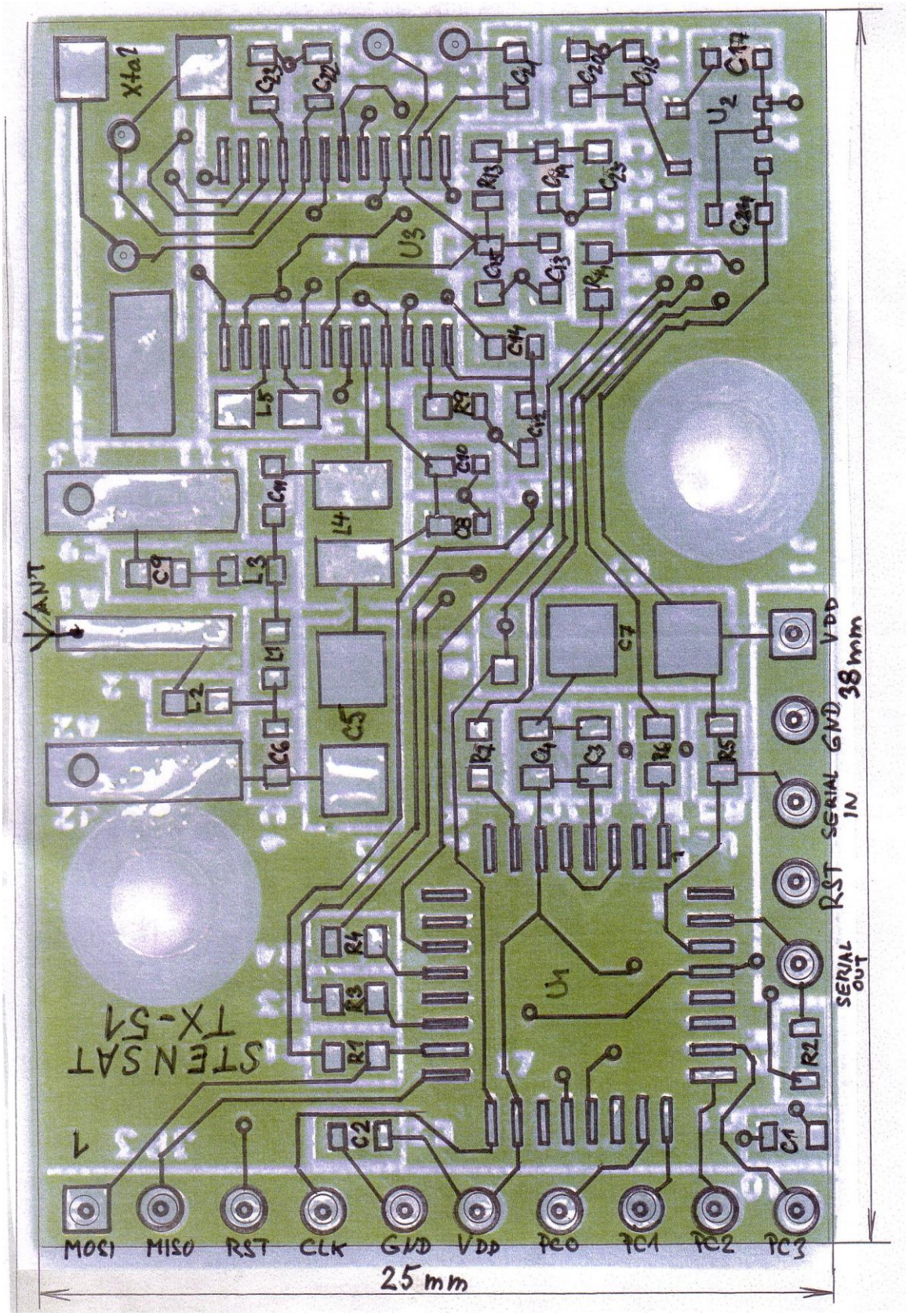


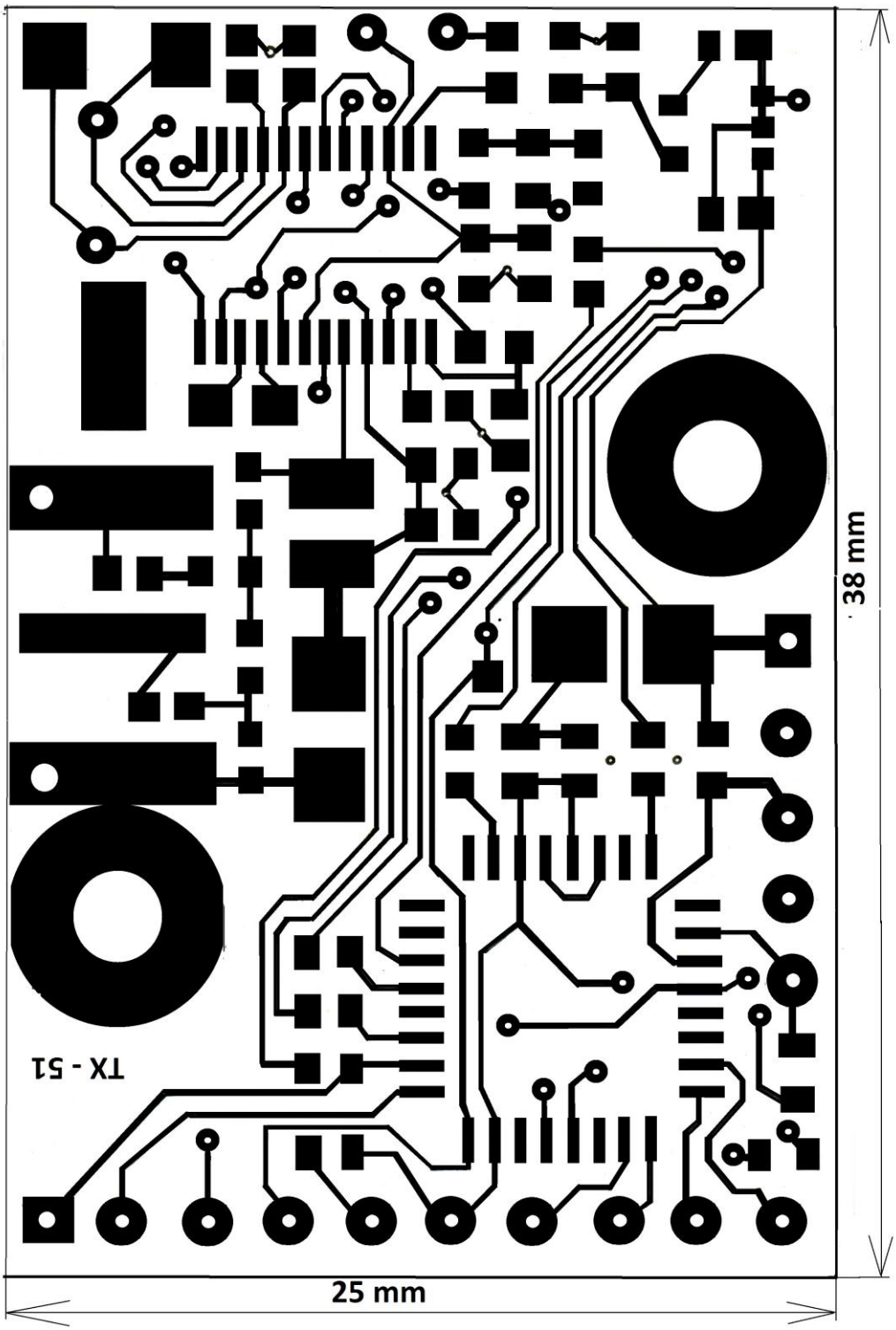


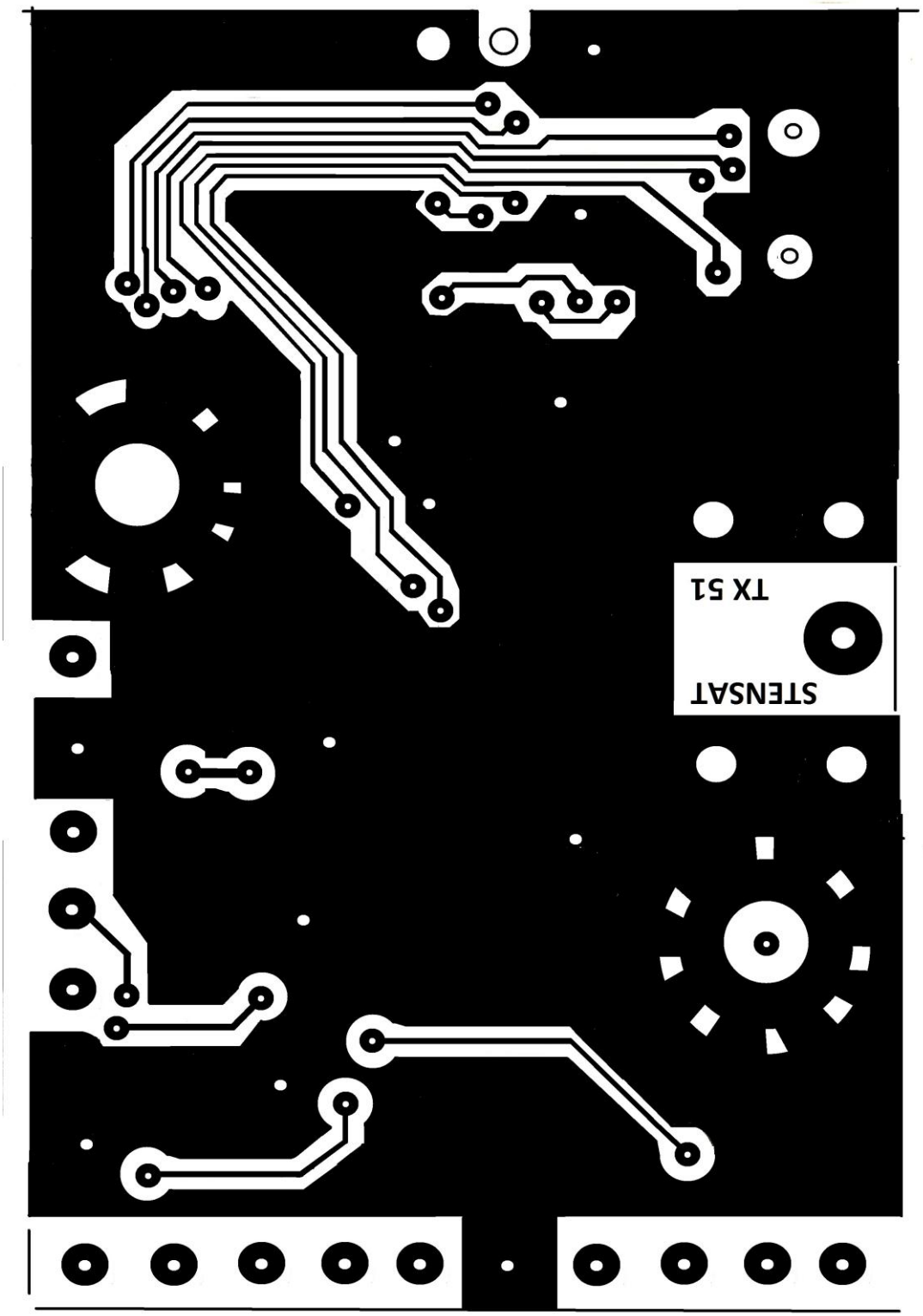
A dále desku vysílače bez konektorů a obou integrovaných obvodů:






Xtal
U2
U3
U1
ANT
L1
L2
L3
L4
L5
C5
C7
C9
C17
C18
C19
C20
C21
C22
C23
C24
R1
R2
R3
R4
R5
R6
R7
R8
VDD
GND
SERIAL IN
RST
SERIAL OUT
38 mm
STENSAT TX-S1
MOSI
MISO
RST
CLK
GND
VDD
PC0
PC1
PC2
PC3
25 mm


38 mm
25 mm
TX - 51



Dodatek 4 – inicializace ADF7012

my remarks: *CanSat Book for Students* – part.1 2011

V projektu Vesmírná odysea http://2010.instruktori.cz/tym použil Mgr. Šimon Řeřucha <res@isibrno.cz> z Fakulty Informatiky VUT Brno obvod ADF7012 spolu s PIC 16F629 + 25AA1024 ve vysílači pro 150,3 MHz a poskytl o tomto Txu informace:

Pracovní frekvence byla v pásmu 150 - 151 MHz. Mezi OSC1 a OSC2 byl krystal 4,096MHz. Řada nastavení se odvozuje právě od této hodnoty + nastavení ruzných násobičů (PLL). Byl mírně problém trefit správnou hodnotu cívky mezi L1 a L2.

Níže jsou výtržky z kódu, vysílač jsem používal v režimu OOK (on-off keying) a nebo pro FSK (frequency shift). Registry F a R jsem měl nastavené jednotně, M bylo různe pro FSK a OOK, v registru N byla akorát frekvence v kHz bitově posunuta o jedna doleva, a poslednií bit byl nastaven na jednčcku. Uvedená funkce si zapínala OOK a vyslala krátky puls.

```
#define RF_REGF 0x0071A31F                          // reg F : OOK i FSK
#define RF_REGR 0x04A02000                          // reg R: OOK i FSK
#define RF_REGM 0x01BFFBCE                          // reg M: OOK max vykon
#define RF_FSK_M 0x00003BE2                         // reg M: FSK


void gtag_beep(int len){

    long int frtest = ((150300 ) << 2) + 1;

    int i;

    SIG_RF_KEY = 0;
    SIG_RF_CS = 1;

    SH_write(RF_REGR);
    SH_write(RF_REGM);
    SH_write(RF_REGF);


    SH_write(frtest);

    DelayMs(10);        // ustaleni PLL vysilace
    SIG_RF_KEY = 1;

    DelayMsLong(len);        // delka pulsu

    SIG_RF_KEY = 0;
    DelayMs(1);

    SIG_RF_CS = 0;
    SIG_RF_KEY = 1;
}
```

Piny CLK (.11), DATA(.12) a LE (.13) jsou pouzívané pro sériový přenos do shift registru, k tomu jsem měl funkci SH_write, viz níž. Tím, že to je obecná knihovnička, tak je to napohled trošku hloupě nadvakrát provázané.

SIG_RF_CS je provázaný na CE (.14) tím se to celé zapíná (po každém zapnutí je třeba znovu nastavit registry), pro kontinuální provoz stačí mít to v stále v hodnotě jedna.

Tím SIG_RF_KEY (na TxDATA) se řídí modulace ... např. v OOK se tím zapíná / vypíná nosná vlna, ve FSK se tím přepíná LO / HI a podobně.

```
/**
*  @file - shift.c Shift register driver
*
*  @ingroup mod_tag
*  @author Simon Rerucha <res@isibrno.cz>
*/

// ADF7012 - shift register
#define SH_data         SIG_RF_SHDATA
#define SH_clk          SIG_RF_SHCLK
#define SH_ld           SIG_RF_SHENA

// ADF7012 - signals
#define SIG_RF_CS RC4       // chip enable , pin .14
#define SIG_RF_KEY RC3      // txdata / keying .4

#define SIG_RF_SHDATA RA0   // . 12
#define SIG_RF_SHCLK RC7    //  .11
#define SIG_RF_SHENA RC5    //  .13


#ifndef SH_data
       #error "SH_data not defined!"
#endif

#ifndef SH_clk
       #error "SH_clk not defined!"
#endif

#ifndef SH_ld
       #error "SH_ld not defined!"
#endif

//#define SHdelay for(c = 0; c > 250;c++)
#define SHdelay DelayUs(2);

/*
* SH_write() - writes unsigned long in MSB first order
*/

void SH_write(unsigned long chr){
   //int mask = 1;
   unsigned char i,j;
   unsigned char data;

   SH_clk = 0;
   SH_ld  = 0;
```

```
   for (j = 4; j > 0; j-- ){

       data = (unsigned char) (chr >> (j-1)*8);

       for (i = 8; i > 0; i-- ){

           SH_data = ((data & 0x80) != 0);
           SHdelay;
           SH_clk = 1;
           SHdelay;
           SH_clk = 0;

           data = ((data) << 1);
       }
   }

   SH_data = 0;
   SH_ld = 1;
   SHdelay;
   SH_ld = 0;
}
```

moje pozn. : Mgr.Š.Řeřucha použil následující fintu:

150,3 MHz je 150300 kHz dekadicky, binárně je 100100101100011100
posun o 2 místa doleva je 10010010110001110000
a jestě přičist 1 tj máme 10010010110001110001
tj N_{fract} = 101100011100 což je 2844 dek
N_{INT} = 00100100 což je 36 dekadicky
Pokud zkontroluje pomocí vzorce z datesheetu ADF7012
$f_{OUT} = f_{FPD} (N_{int} + N_{FRACT} / 2^{12}) =$ 4,096 . (36 + 2844/4096) = 4,096 . 36,69433 = 150,3 MHz
dostaneme opravdu kmitočet 150,3 MHz