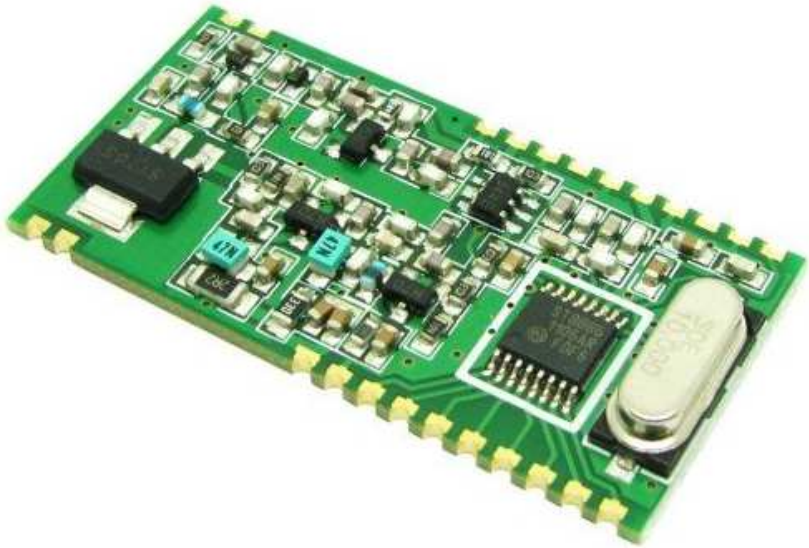
Příručka pro nastavení a obsluhu radiového modulu RFM12BP



Tato příručka slouží jako náhrada za prakticky nepoužitelnou originální dokumentaci výrobce HOPE RF. Autor nenese žádnou odpovědnost za správnost informací v této příručce. V případě pochybností se obraťte přímo na výrobce (http://www.hoperf.com) případně na originální dokumentaci.
Originál tohoto dokumentu je ke stažení na serveru ulozto.cz. Případné dotazy můžete konzultovat na fóru http://forum.mcontrollers.com/

Upozornění:
Vzhledem k možnostem radiového modulu se před prvním použitím seznamte s všeobecným oprávněním Českého Telekomunikačního Úřadu číslo: VO-R/10/04.2012-7 k využívání radiových kmitočtů a k provozování zařízení krátkého dosahu. Autor upozorňuje, že informace zde uvedené jsou poplatné době vzniku tohoto dokumentu (březen 2015) a nenese žádnou odpovědnost za případné škody vzniklé nedodržením aktuálně platných nařízení ČTÚ.

Příručka slouží jako zdroj informací výhradně ke studijním soukromým účelům.

Adam Deštěnský
e-mail
Únor 2016
verze dokumentu: 2

Obsah

1. Historie změn dokumentu

verze	datum	popis změn
1.	03/2015	původní verze dokumentu
2.	02/2016	Opraveny chyby v zapojení, opravena procedura inicializace modulu, přidán popis vysílání a příjmu dat včetně příkladů v jazyku C. První publikovaná verze.

2. Technické informace modulu RFM12BP

Radiové parametry:

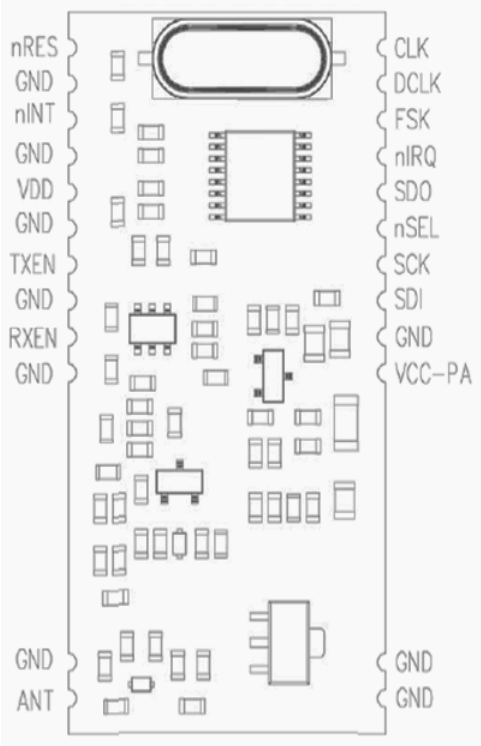
Kmitočtová pásma modulu:	433/868/915 MHz
Modulace:	FSK
Přenosová rychlost:	max. 115,2 kbps (s interním digitálním filtrem) max. 256 kbps (s externím RC analogovým filtrem)
Výstupní výkon:	500 mW (regulovatelný v 8 krocích)
Citlivost:	- 115 dBm (BER=10^{-3}, BW=134 kHz, BR=1,2 kbps)
Šířka pásma přijímače:	67, 134, 200, 270, 350, 400 kHz
Frekvenční zdvih vysílače:	15 - 240 kHz (s krokem nastavení 15 kHz)
Impedance antény:	50 Ω

Elektrické parametry:

Napájecí napětí:	logika: 2,2 - 3,8 V výkonový zesilovač: 12V
Stand-by proudový odběr:	max. 1,2 mA (při zapnutém integrovaném krystalu)
Proudový odběr v režimu příjmu (Rx):	typ. 20 - 25 mA
Proudový odběr v režimu vysílání (Tx):	typ. 185 - 200 mA

3. Zapojení modulu RFM12BP

Modul RFM12BP je zhotoven na desce plošných spojů určené pro povrchovou montáž. Vývody nemají standardní DIP rozteč a tak je nutné (pro použití např. v nepájivém poli) zhotovit adaptér. Modul dále nemá vyvedenou anténu, kterou je nutné také zhotovit.

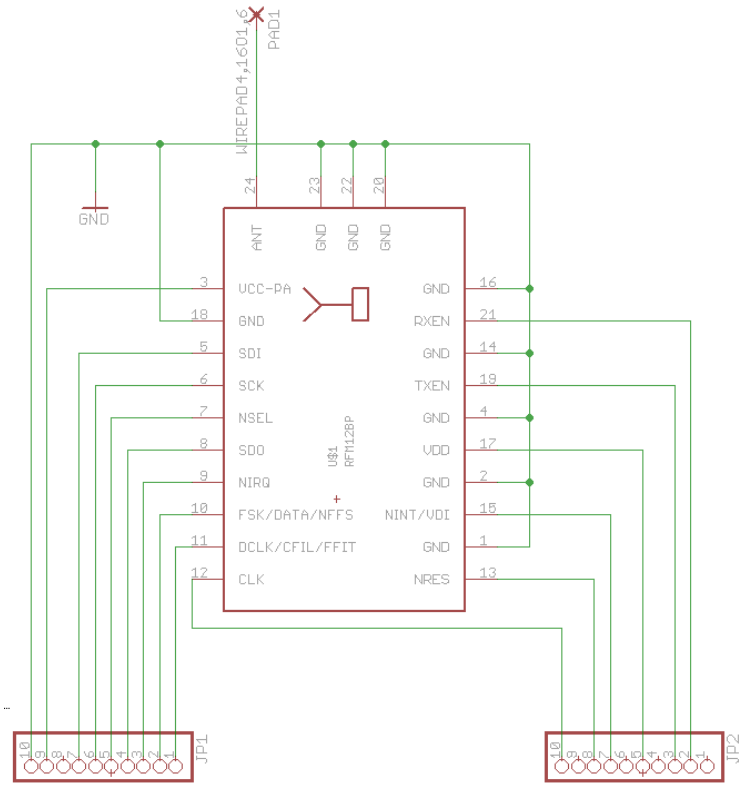


Obr. 1: Vývody modulu RFM12BP

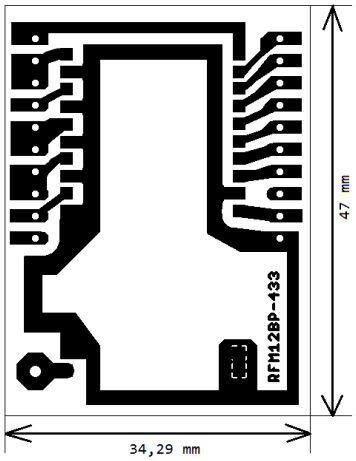
nRES = hardwarový reset modulu (když je log.0, tak proběhne RESET)
nINT/VDI – defaultně je nastaveno VDI
nINT = požadavek na přerušení z MCU do RFM12 (MCU žádá o přerušení)
VDI= potvrzuje správnost přijatých dat na základě různých mechanismů
VDD = napájecí napětí logiky (2,2 – 3,8 V)
TXEN = zapnutí vysílače (log. 1 = zapnuto)
RXEN = zapnutí přijímače (log. 1 = zapnuto)
CLK = výstup osazeného krystalu (dá se vypnout) – může sloužit jako zdroj hodin pro MCU (a ten pak nepotřebuje vlastní krystal)
DCLK = vývod pro připojení externího RC filtru (pro přenosové rychlosti nad 115 kbps)
FSK = přímý vstup dat, pokud nepoužíváme FIFO registr (standardně je přes odpor 10 kΩ připojen na napájecí napětí 3,3 V)
nIRQ = požadavek na přerušení z RFM12 do MCU (IRQ je aktivní při log. 0). Dále jsou popsány události, které aktivují IRQ (tj. nIRQ = 0).
SDO = serial data output – na straně MCU propojit na MISO (SPI data)
nSEL = signál slave select – log. 0 povoluje komunikaci s MCU po SPI (musí být držena celou dobu komunikace s MCU) – řídí MCU
SCK = hodiny SPI
SDI = serial data input – na straně MCU propojit na MOSI (SPI data)
VCC-PA = napájení zesilovače 0,5W (napětí 6 až 12V)

4. Adaptér na RFM12BP pro použití v nepájivém poli

Jak bylo popsáno výše, modul nemá standardní rozteč vývodů 2,54 mm a pro použití v nepájivém poli je nutné zhotovit redukci (adaptér). Návrh je vytvořen v SW Eagle, do kterého je možné na internetu zdarma stáhnout knihovnu **RFMxx.lbr**, která mimo jiné obsahuje i obrazec pro RFM12BP.
Modul se osazuje ze strany spojů ! Radiový modul osazený na adaptéru v nepájivém poli je vidět na obrázku číslo 4.



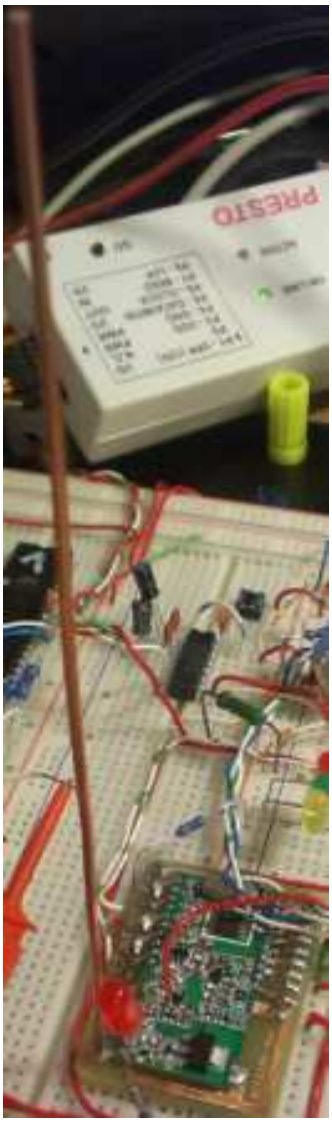
Obr. 2: Schéma adaptéru



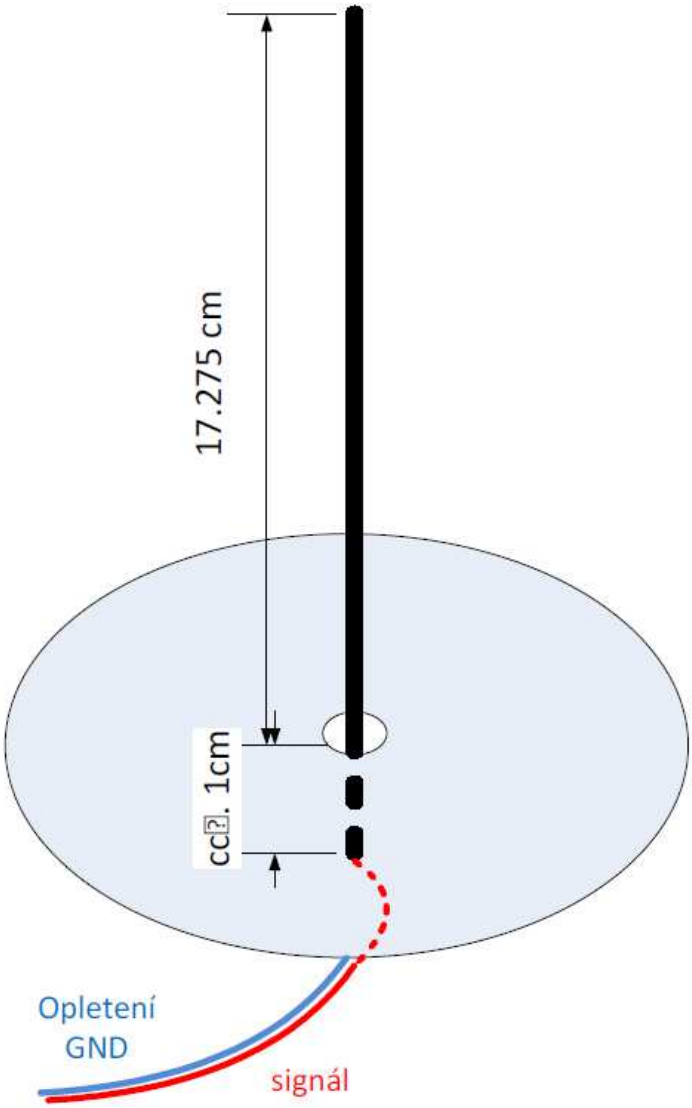
Obr. 3: Tištěný spoj adaptéru

5. Anténa pro RFM12BP

Antén je možné použít celou řadu a typů. Nejjednodušší varianta je 1/4 vlnný dipól (prutová anténa) zhotovený z elektrikářského 1,5 mm^2 Cu drátu napájeného přímo na RFM12BP adaptér (obrázek č. 4). Anténa má délku 17,275 cm, což odpovídá naladění antény na kmitočet 433,92 MHz. Takto vyhotovená anténa nemá protiváhu a může se objevit problém, že při vyšším napájecím napětí výkonového zesilovače než cca. 6V se modul při vysílání nebo příjmu dat restartuje.
Řešením je použít 1/4 vlnný dipól s plechovou protiváhou, který se k modulu RFM12BP připojí pomocí koaxiálního kabelu s impedancí 50 Ω. Řešení takové antény je ukázáno na obrázku č. 5.
Další variantou je použití samotného 50 Ω koaxiálního kabelu, kdy aktivní délku antény (posledních 17,275 cm kabelu) zbavíme opletení. Tato varianta se často vyskytuje v průmyslově vyráběných výrobcích, kdy je následně kabel vložen, nebo přilepen k plastovému šasi hotového výrobku. Výhodou této varianty je, že anténa je poddajná a je možno ji schovat.



Obr. 4: anténa přímo napájená na adaptér RFM12BP



Obr. 5: prutová anténa s plechovou protiváhou

6. Schéma propojení na MCU (Atmel MEGA 32A)

Varianta zapojení využívající systému přerušení (nIRQ)

RFM12	MCU	popis
nIRQ	PD2 (INT0)	požadavek na přerušení ze strany RFM12 - v Rx režimu = nově příchozí data - v Tx režimu = požadavek na poslání dalšího bajtu
SDO	MISO	data z RFM12 do MCU
nSEL	SS	Slave Select - výběr obvodu pro SPI komunikaci
SCK	SCK	Hodiny SPI
SDI	MOSI	data z MCU do RFM12
RXEN	PD7	ovládání přijímače
TXEN	PD6	ovládání vysílače
nINT	PD5	požadavek na přerušení ze strany MCU
nRES	PD4	hardwarové resetování radiového modulu

Tab. 1: Popis jednotlivých vodičů potřebných pro obsluhu radiového modulu

Na propojení MCU s RFM12 v této variantě potřebujeme **minimálně 5 vodičů** + optimálně 4 vodiče.

varianta zapojení	Připojeno na MCU	Připojeno na napájecí napětí 3,3V přes odpor 10 kΩ
Minimální zapojení	nIRQ, SDO, SDI, nSEL, SCK	RXEN, TXEN, nRES, nINT
Doporučené zapojení	nIRQ, SDO, SDI, nSEL, SCK, RXEN, TXEN, nRES, nINT	-

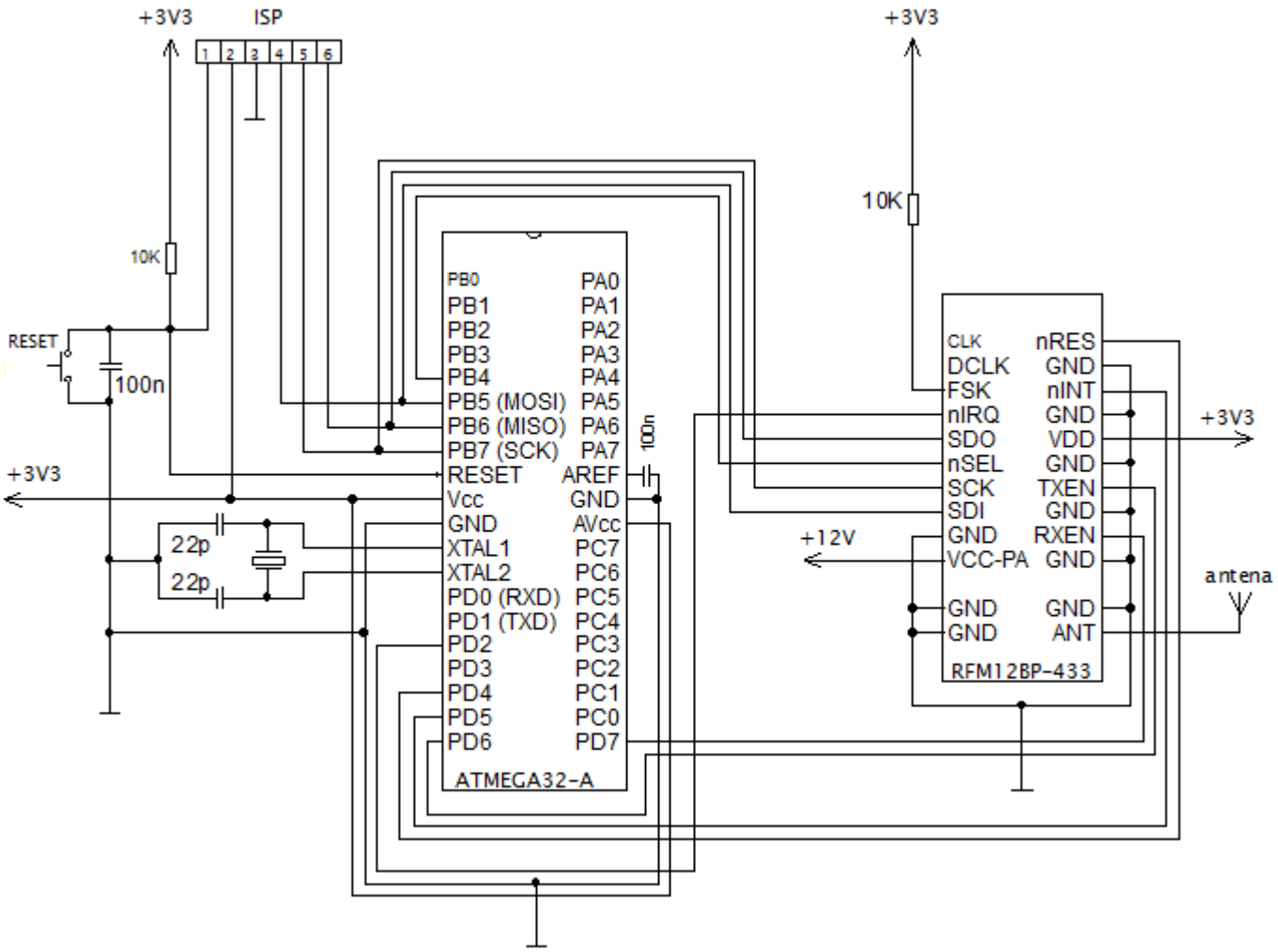
Tab. 2: Propojení radiového modulu RFM12BP na MCU

Jediným speciálním požadavkem je, aby byl vodič nIRQ připojen k takovému pinu MCU, který podporuje externí přerušení. U Atmel Mega 32A se jedná o pin PD2, který má implementovánu funkci externího přerušení (EXT_INT0) s vektorem číslo 2 a tudíž po reset vektrou s druhou nejvyšší prioritou.
U minimálního zapojení musíme dbát na připojení signálů RXEN, TXEN, nRES, nINT na napájecí napětí, kdy log. 1 u RXEN a TXEN způsobí trvalé hardwarové povolení těchto funkcí a u signálu nRES, nINT zaručí neaktivitu těchto funkcí (HW reset a přerušení ze strany MCU). Oba signály jsou aktivní až při logické 0. Vývod FSK pak nemá pro běžné aplikace význam, tak jej připojíme přes odpor 10 kΩ k napájecímu napětí.

Praktické zkušenosti:

- ✓ Zapojit nRES na vývod MCU je velmi výhodné. Díky tomu máme možnost provést pohodlně restart modulu, což je minimálně při inicializaci (po ustálení napájecího napětí) výhodné.
- ✓ Při vysílání na maximální výkon se koncový stupeň zesilovače silně zahřívá a hrozí při zapomenutém vypnutí jeho zničení. Je tak vhodné propojit s MCU i signály TXEN a RXEN, což usnadní hardwarové řízení především vysílače a zpřehlední kód programu (jasně víme, kdy se vysílá a kdy přijímá).
- ✓ Pokud nastávají problémy se signálem nIRQ, doporučují někteří autoři připojení pull-up rezistoru 4,7 kΩ, vzhledem k tomu, že se jedná o výstup typu otevřený kolektor (pozn. I v mých experimentech se použití pull-up rezistoru jednoznačně vyplatilo - linka je vždy v jasně definovaném stavu).
- ✓ Pro fázi ladění je ideální doplnit zapojení o 3 ks kontrolních LED diod. Jednu na linku TXEN, druhou na RXEN a třetí na nIRQ (diodu vždy zapojit do série s odporem 1 kΩ proti zemi). Tím získáme vizuální kontrolu stavu těchto linek.
- ✓ Připojit nINT k vývodu MCU nemá příliš valný význam, protože pro základní použití jej stejně ponecháváme trvale v log. 1. Stejnou funkci dosáhneme i tím, že připojíme nINT na napájecí napětí přes odpor 10 kΩ. Nicméně pro budoucí použití a pokud máme volný pin MCU to může být výhodné.
- ✓ Pozor na volbu krystalu MCU. Vzhledem k napájecímu napětí logiky 3,3V není např. u Atmelu Mega 32A možno použít max. rychlost 16 MHz, ale jen cca. 10 MHz. Informace typu „Speed Grades“ je možné dohledat v datasheetu konkrétního MCU.

Schéma zapojení:



Obr. 6: Schéma propojení radiového modulu RFM12BP s MCU Atmel Mega 32A

7. Komunikace s MCU po rozhraní SPI a problematika přenosové rychlosti

Komunikace je vedena po standardním SPI rozhraní. Rychlost SPI by se měla odvíjet od použité komunikační rychlosti radiového kanálu a to tak, abychom dokázali dostatečně rychle zásobovat novými daty (a nebo je vyčítat) radiový modul i při nevyšší uvažované radio-komunikační rychlosti.
Existuje zde ale jedno omezení. Čtení Rx FIFO registru nesmí být rychlejší než $f_{ref}/4$, kde f_{ref} je kmitočet integrovaného krystalu RFM12 (10 MHz). **Z toho vyplívá maximální rychlost SPI 2,5 MHz.**

Poznámka:
Pozor při experimentech v nepájivém poli. Rychlost 2,5 MHz je již poměrně vysoká dost na to, aby nastávaly problémy s komunikací. Trochu si pomůžeme, když ISP patici posuneme co nejdál od MCU a např. programátor Asix Presto umožňuje programování na rychlostech 750 kHz – 5,5 MHz. Postupně zkoušíme MCU programovat na různé rychlosti a (obvykle) zjistíme, že musíme mít nastavenou nejnižší rychlost, aby byla data úspěšně přenesena. Vzhledem k tomu, že ISP programování využívá SPI, můžeme si z toho odvodit, jakou rychlost SPI nastavit, aby byla data spolehlivě přenášena do/z radiového modulu.

Před volbou komunikační rychlosti si také musíme ujasnit, jak bude vypadat radiový paket. Je potřeba si uvědomit, že nepřenášíme pouze užitečná data, ale také dva bajty „Preamble“, jež má v binární podobě formát 10101010, která slouží k synchronizaci přenosové rychlosti přijímače na vysílač a dále standardně dva bajty synchronizace, díky které bude přijímač reagovat jen na odpovídající vysílače. Pro alespoň elementární bezpečnost by pak měla být data ochráněna kontrolním součtem (např. CRC-8). Z popisu (a tab. 3) je patrné, že pro přenos jednoho bajtu dat budeme potřebovat minimálně dalších 5 bajtů „režijních“ informací.
Pro odvysílání 1 Bajtu po rádiu potřebujeme přenést celkem 6 Bajtů po SPI

Preamble	Preamble	Synchronizace		data	CRC
0xAA	0xAA	0x2D	0xD4	1 Byte	1 Byte

Tab. 3: Struktura doporučeného radiového paketu

Poznámka:

Preamble	Synchronizace	Data
0xAA	0xD4	1 Byte

Tab. 4: Struktura minimálního radiového paketu

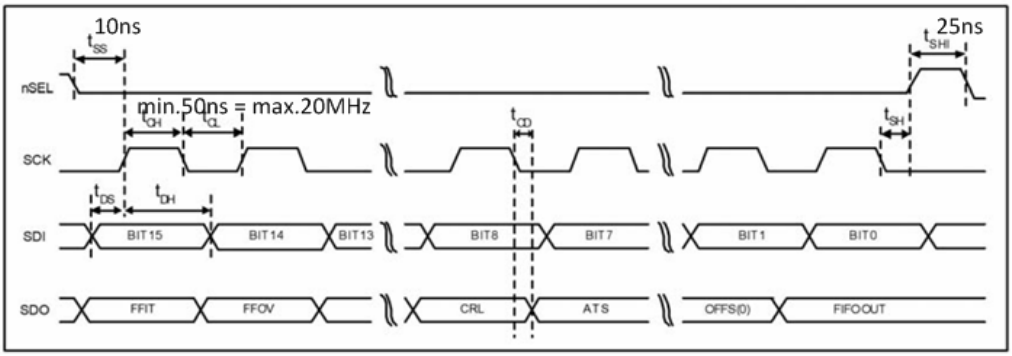
Příklad v tabulce 3 ukazuje nejméně efektivní způsob přenosu, kdy celou režii opakujeme pořád dokola pro každý přenášený bajt zvlášť. Prakticky pak ale budeme přenášet radiové pakety větší délky (20 bajtů), čímž se poměr režijních bajtů vůči užitečným datům vylepší. Pro další úvahy o přenosové rychlosti ale budeme uvažovat neefektivní způsob naznačený v tabulce 3.

Příklad:
MCU XTAL = 9,216 MHz, požadovaná radiová rychlost = 115,2 kbps. Jak zvolit prescaler v MCU pro SPI ?

115 200 bps = 14 400 B/s (115 200 / 8) – což odpovídá 115 200 Bd/s
přenos 1 Bajtu radiovým kanálem trvá = 1 / 14 400 = 69,44 µs

Pokud použijeme prescaler /8
=> **takt SPI = 1,152 MHz** => doba přenosu 1 bitu = 0,868 µs => doba přenosu 1 Bajtu = 6,944 µs. Vidíme, že takto nastavená rychlost SPI bohatě stačí na požadovanou radiovou přenosovou rychlost.

- SS je na straně RFM označen jako nSEL a musí být celou dobu komunikace v log. 0
- Bity jsou přenášeny vždy od MSB po LSB



Problematika přenosové rychlosti

Musíme si ale uvědomit, že díky veliké režii nám klesne přenosová rychlost užitečných dat na 1/6 radiové přenosové rychlosti. Takže v uvažovaném příkladu můžeme např. do MCU dodávat užitečná data maximální rychlostí 2,4 kB/s, což odpovídá např. u RS-232 modulační rychlosti 19200 Bd.
U často používané modulační rychlosti sériové linky 9600 Bd (což je 1,2 kB/s) nám postačí rádiová rychlost 6x vyšší, tj. 57,6 kbps, což radiový modul i SPI linka bez problémů zvládnou.

Nyní budeme uvažovat nový radiový paket

Preamble	Preamble	Synchronizace		2nd Synchro	service Byte	DATA	CRC	rezerva	rezerva
0xAA	0xAA	0x2D	0xD4	0xCA	1 Byte	16x Byte	1 Byte	1 Byte	1 Byte

Tab. 5: Struktura radiového paketu o délce 25 Bajtů

Tento nový radiový paket má přidanou další synchronizaci (2nd Synchro), aby byl vyloučen vliv jiných zařízení se stejným radiovým modulem. Dále obsahuje servisní bajt, který popisuje vlastnosti paketu (zabezpečení proti chybám, šifrování apod.). Samotná data jsou pak organizována do bloku 16 bajtů, což je výhodné např. pro šifrovací algoritmus AES 128bit. Nechybí kontrolní součet CRC8 a dva přídavné bajty pro další použití.
Paket má celkovou délku 25 bajtů, z toho je 16 bajtů užitečných dat. Poměr užitečných dat je tak oproti předchozímu případu (1/6 = 0,1666) zvětšen na 16/25 = 0,64 což je 3,84x lepší.

Pokud nastavíme radiovou přenosovou rychlost na 115,2 kbps (= 14,4 kB/s), tak nám užitečná datová rychlost spadne na 16/25 (=0,64), což bude 9,216 kB/s a odpovídá to modulační rychlosti sériové linky 73728 Bd. Takovou modulační rychlost ale v normálním PC nenastavíme, tak se musíme spokojit s nejbližší nižší nastavitelnou hodnotou, což je 57600 Bd (= 7,2 kB/s).
U běžné modulační rychlosti sériové linky 9600 Bd (= 1,2 kB/s) nám postačí radiová přenosová rychlost násobená převrácenou hodnotou 1/0,64 (= 1,5625), což je 15 kbps (= 1,875 kB/s).

Poměry mezi užitečnou datovou rychlostí sériové linky a radiovou rychlostí včetně režie u uvažovaného radiového paketu ukazuje následující tabulka:

sériová linka RS-232 (PC)		**Potřebná radiová rychlost**		**Potřebná SPI rychlost (x2)**
Bd	kB/s	kB/s	kbps	MHz
75	0,009375	0,015	0,120	0,002
110	0,01375	0,022	0,176	0,002
134	0,01675	0,027	0,216	0,002
150	0,01875	0,030	0,240	0,002
300	0,0375	0,059	0,472	0,002
600	0,075	0,118	0,944	0,002
1200	0,150	0,235	1,880	0,004
1800	0,225	0,352	2,816	0,006
2400	0,3	0,469	3,752	0,008
4800	0,6	0,938	7,504	0,016
7200	0,9	1,407	11,256	0,024
9600	1,2	1,875	15	0,030
14400	1,8	2,813	22,504	0,046
19200	2,4	3,750	30	0,060
38400	4,8	7,500	60	0,120
57600	7,2	11,25	90	0,180

Tab. 6: Vzájemný vztah přenosových rychlostí pro 16/25 paket

SPI rychlost musíme násobit x2, protože se do RFM12 před každým bajtem paketu přenáší i příkaz k plnění vysílacího FIFO registru.

Za tabulky číslo 6 vidíme, že nejužší hrdlo komunikačního řetězce:
PC – RS232 – MCU – SPI – RFM12 – radiový kanál – RFM12 – protistrana

Vzniká v radiovém kanále, který omezuje maximální použitelnou rychlost sériové linky na 50%. Naopak největší rezervu má SPI rozhraní, které je využito pouze na necelých 7% svého maximálního výkonu. Naproti tomu rádiový kanál je zatížen z 78,125%.

8. Instrukční sada RFM12B

Ještě před tím, než začneme RFM12 používat pro přenos dat, tak musíme provést konfiguraci pomocí 16 bitových konfiguračních slov.

Seznam konfiguračních slov:

- **Configuration Setting Cmd.** (kmitočtové pásmo, kapacita krystalu, povolení práce s dat. registry a FIFO)
- **Power Management Cmd.** (zapnutí přijímače, vysílače, krystalu, low-battery, wake-up timer,CLK výstup)
- **Frequency Set Cmd.** (nastavení nosného kmitočtu)
- **Data Rate Cmd.** (nastavení datové přenosové rychlosti)
- **Receiver Cntr. Cmd.** (funkce nINT/VDI, VDI odezva, Rx BW, LNA zisk, příjmová úroveň RSSI)
- **Data Filter Cmd.** (Clock recovery, Clock recovery control, volba filtru (Analog – Digital), DQD práh)
- **FIFO and Reset mode Cmd.** (počet přijatých bitů pro nIRQ, délka synchronizace, plnění FIFO, "citlivý" reset)
- **Synchro Pattern Cmd.** (nastavení druhého synchronizačního Bajtu)
- **AFC Cmd.** (měření Tx kmitočtu a výpočet případného offsetu)
- **Tx conf. Cmd.** (polarita modulace, kmitočtový zdvih modulace, výstupní výkon)
- **PLL Set Cmd.** (parametry PLL smyčky - zpoždění, dithering, šířka pásma)
- **Low Bat. Det. and MCU CLK Divider Cmd.** (nastavení CLK kmitočtu pro MCU a low-battery napětí)
- **Wake-up Timmer Cmd.** (perioda probouzení rád. modulu)
- **Low Duty-Cycle Cmd.** (nastavení low duty-cycle - poměrného času, kdy bude modul v provozu)

- **Reset Cmd.*** (resetuje modulu do stavu POR - všechny registry se uvedou do defaultního nastavení (POR))

** Nejedná se o konfigurační slovo v pravém slova smyslu, ale o příkaz, kterým SW cestou resetujeme rádiový modul (všechny registry uvedeme do stavu POR). Tato funkcionalita ale není 100% prokázána!*

Seznam příkazů pro čtení/zápis dat:

- **Status Read Command** (načtení STATUS registru)
- **Receiver FIFO Read Command** (čtení přijatých dat z Rx FIFO registru)
- **Transmitter Register Write Command** (zápis dat do Tx FIFO registru)

Nyní budou probrány jednotlivé konfigurační slova podrobně. **Tučně** vyznačené hodnoty jednotlivých bitů jsou **doporučené** pro základní oživení modulů a sestavení radiové komunikace. V momentě kdy moduly navážou komunikaci a díky tomu známe funkční konfiguraci, můžeme začít experimentovat s nastavením jednotlivých hodnot.
Pro snadnější orientaci v konfiguračních slovech doporučuji on-line „konfigurační kalkulátor“ [7].

Configuration Setting Command 0x80

		7	6	5	4	3	2	1	0	POR
1000	0000	el	ef	b1	b0	x3	x2	x1	x0	0x8008

- Povoluje přístupu k interním datovým Tx registrům (v tom případě musí být pin FSK připojen na log. 1) *(el)*
- Povoluje práci s Rx FIFO registrem (registr pro příjem dat) *(ef)*
- Nastavuje pásmo, ve kterém modul pracuje (433, 868, 915 MHz) *(b1 - b0)*
- Nastavuje kapacitu integrovaného krystalu (v rozmezí 8,5 - 16 pF) *(x3 - x0)*

Význam jednotlivých bitů:

el	význam
0	přístup k Tx regitrům zakázán
1	**přístup k Tx registrům povolen**

ef	význam
0	Rx FIFO zakázán*
1	**Rx FIFO povolen**

** v tom případě je pro přenos dat a hodin využito pinů FSK (DATA) a DCLK*

b1	b0	význam (pásmo)
0	0	rezervováno
0	1	pásmo 433 MHz
1	0	pásmo 868 MHz
1	1	pásmo 915 MHz

x3	x2	x1	x0	význam (C krystalu)
0	0	0	0	8,5 pF
0	0	0	1	9 pF
0	0	1	0	9,5 pF
0	0	1	1	10 pF
0	1	0	0	10,5 pF
0	1	0	1	11 pF
0	1	1	0	11,5 pF
0	1	1	1	12 pF
1	**0**	**0**	**0**	**12,5 pF**
1	0	0	1	13 pF
1	0	1	0	13,5 pF
1	0	1	1	14 pF
1	1	0	0	14,5 pF
1	1	0	1	15 pF
1	1	1	0	15,5 pF
1	1	1	1	16 pF

Výchozí nastavení registru (POR):

- Přístup k datovým registrům zakázán
- FIFO zakázáno
- Pásmo nenastaveno
- Kapacita krystalu 12,5pF

Doporučené nastavení registru:

- Registry povoleny, FIFO povolen, pásmo: 433MHz, kapacita: 12,5pF **0x80D8**

Power Management Command 0x82

		7	6	5	4	3	2	1	0	POR
1000	0010	er	ebb	et	es	ex	eb	ew	dc	0x8208

- Zapíná celý přijímací řetězec *(er)*
- Zapíná pásmovou propusť (jednu z částí příjmového řetězce) *(ebb)*
- Zapíná PLL smyčku, výkonový zesilovač a zapíná vysílání *(et)*
- Zapíná syntetizátor *(es)*
- Zapíná integrovaný krystal (oscilátor) *(ex)*
- Zapíná detekci nízkého stavu baterie *(eb)*
- Zapíná wake-up timer *(ew)*
- Vypíná výstup integrovaného krystalu na výstup CLK *(dc)*

Význam jednotlivých bitů:

er	**význam**
0	přijímací řetězec vypnut
1	přijímací řetězec zapnut (11 mA)

ebb	**význam**
0	pásmová propusť přijímače vypnuta
1	**pásmová propusť přijímače zapnuta***

* *pro správnou funkci přijímače (Rx) postačí zapnout bit **er**. Bit **ebb** být zapnutý nemusí, ale zkracuje dobu náběhu přijímače. Pro správnou funkci musí být zapnut syntetizátor bit **es = 1**.*

et	**význam**
0	vysílač vypnut
1	vysílač zapnut (22 mA)

es	**význam**
0	syntetizátor vypnut
1	**syntetizátor zapnut****

** *Pro správnou funkci syntetizátoru musí být zapnut oscilátor, bit **ex = 1**.*

ex	**význam**
0	integrovaný oscilátor vypnut
1	**integrovaný oscilátor zapnut (3 mA)**

eb	**význam**
0	**detekce nízkého stavu baterie vypnuta**
1	detekce nízkého stavu baterie zapnuta***

*** *detailní informace o konfiguraci viz. registr Low Battery Detector and MCU Clock Divider Command*

ew	**význam**
0	**wake-up timer vypnut**
1	wake-up timer zapnut****

**** *detailní informace o konfiguraci viz. registr Wake-Up Timer Command*

dc	**význam**
0	na výstupu CLK **je** signál integrovaného krystalu
1	**na výstupu CLK není signál integrovaného krystalu**

Výchozí nastavení registru (POR):

- Vysílač i přijímač vypnuty
- Integrovaný krystal zapnut
- Detekce nízkého stavu baterie vypnuta
- Wake-up timer vypnut
- Na výstupu CLK je signál z integrovaného krystalu

Doporučené nastavení registru:

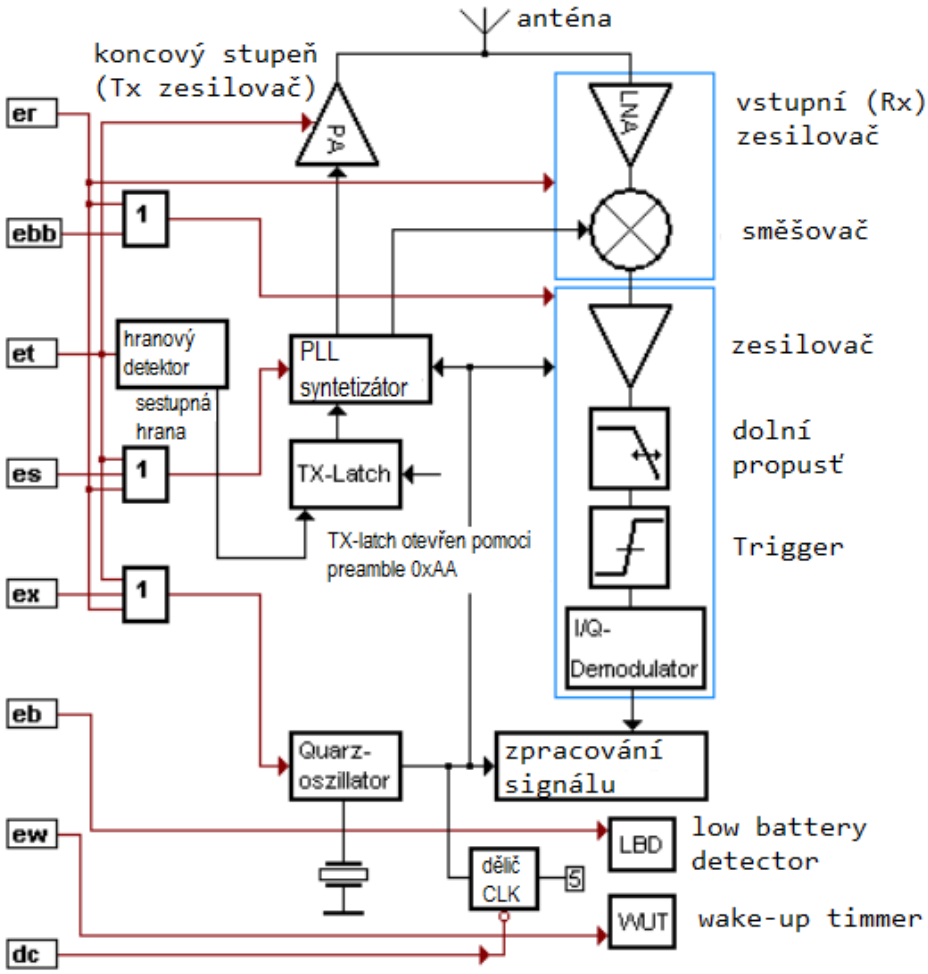
- **Vypnutí přijímače a vysílače**, krystal zapnut, detekce bat., wake-up a výstup na CLK vypnut **0x8209**
- **Zapnutí přijímače**, integrovaný krystal zapnut, detekce baterie, wake-up a výstup na CLK vypnut **0x82C9**
- **Zapnutí vysílače**, integrovaný krystal zapnut, detekce baterie, wake-up a výstup na CLK vypnut **0x8239**

Poznámky:

- *Pokud bude současně zapnut přijímač (**er=1**) i vysílač (**et=1**), modul bude v režimu příjmu Rx*
- *Největší vliv na rychlost náběhu má bit **ex** (interní krystal) - nechat neustále zapnutý (=1), protože doba náběhu krystalu je téměř 1ms.*
- *Následující tabulka ukazuje jednotlivé časy přechodů mezi různými režimy:*

parametr	podmínky	doba
Zapnutí vysílače	syntetizátor vypnut, integrovaný krystal zapnut	**250 µs**
Zapnutí přijímače	syntetizátor vypnut, integrovaný krystal zapnut	**250 µs**
Přechod Tx - Rx	syntetizátor zapnut, integrovaný krystal zapnut	**150 µs**
Přechod Rx - Tx	syntetizátor zapnut, integrovaný krystal zapnut	**150 µs**

Schéma ovládání modulu RFM12 pomocí registru Power Management Cmd.



Frequency Setting Command 0xA

	11	10	9	8	7	6	5	4	3	2	1	0	POR
1010	f11	f10	f9	f8	f7	f6	f5	f4	f3	f2	f1	f0	0xA680

- Nastavuje pracovní kmitočet nosné modulu *(f11 - f0)*

Nastavení kmitočtu se realizuje dle vztahu:

- Pro RFM 433 MHz kmitočet = 430 + F . 0,0025
- Pro RFM 868 MHz kmitočet = 860 + F . 0,0050
- Pro RFM 915 MHz kmitočet = 900 + F . 0,0075

Kde **F** je dekadická hodnota čísla složeného z bitů f0 - f11. Jeho rozsah by měl být: 96 < F < 3903

Výchozí nastavení registru (POR):

- Pro RFM 433MHz platí: F = 104 => kmitočet = 430,26 MHz

Doporučené nastavení registru:

- Nastavení kmitočtu 433,920 MHz (F = 1568) **0xA620**

Band [MHz]	C1	C2	f_{min} [MHz]	f_{max} [MHz]	Δf [kHz]	Vorgabe [MHz]	*f* berechnen [MHz]	F berechnen (*f* in MHz)	ISM-Bereich [MHz]
315	1	31	310,24	319,7575	2,5	314,16	310 + F/400	(*f* - 310) * 400	*(beim RFM12B nicht vorhanden)*
433		43	430,24	439,7575		434,16	430 + F/400	(*f* - 430) * 400	433,05 .. 434,79 (Europa, BW = 1,74)
868	2		860,48	879,515	5	868,32	860 + F/200	(*f* - 860) * 200	868 .. 870 (kein ISM! BW = 2)
915	3	30	900,72	929,2725	7,5	912,48	900 + 3*F/400	(*f* - 900) * 133,$\overline{3}$	902 .. 928 (Amerika, BW = 26)

Poznámky:

- *Pro ČR platí všeobecné oprávnění č. VO-R/10/04.2012-7 k využívání radiových kmitočtů a k provozování zařízení krátkého dosahu (viz. https://www.ctu.cz/cs/download/oop/rok_2012/vo-r_10-04_2012-07.pdf) podle kterého platí pro pásmo 433 MHz:*
- *Min. kmitočet: 433,050 MHz*
- *Max. kmitočet: 434,790 MHz*
- *Max. výkon: 10 mW (což je u RFM12 pro vysílací výkon položka -18dB) - pokud je klíčovací poměr 100%, tak kanálové rozteč smí být max. 25 kHz*
  1 mW (pro širokopásmové kanály pro BW > 250 kHz)
- *Pozor na střední kmitočet 433,920 MHz. Jedná se o velmi často používaný kmitočet pro různé bezdrátové zvonky, pokojové termostaty apod. Existují i zařízení, která pracují s moduly RFM12. Abychom zamezili příjmu falešného signálu (neustále a "bez zjevné příčiny" je linka nIRQ nastavována do log. 0), tak nastavíme práci se dvěma synchronizačními bajty (registr: **FIFO and Reset Mode Command 0xCA**, bajt **sp = 0**) a druhý bajt změníme z defaultního 0xD4 na nějaký jiný (registr: **Synchron Pattern Command 0xCE**). Máme tak šanci 1 : 256 že se trefíme do stejné synchro skupiny a přerušení bude aktivováno na základě dat z jiného vysílače.*

Data Rate Command 0xC6

		7	6	5	4	3	2	1	0	POR
1100	0110	cs	r6	r5	r4	r3	r2	r1	r0	0xC623

- Nastavuje bitovou rychlost radiového přenosu *(r6 - r0)*
- Umožňuje nastavit prescaler 1/8, kterým můžeme lépe donastavit přesně požadovanou bitovou rychlost *(cs)*

Význam jednotlivých bitů:

cs	význam
0	prescaler 1/8 vypnut
1	prescaler 1/8 zapnut

Nastavení přenosové rychlosti se realizuje dle vztahu:

BR = 10 000 / 29 / (R + 1) / (1 + cs.7)

Kde **BR** je přenosová rychlost v kbps, **R** je dekadická hodnota čísla složeného z bitů r0 - r6 (1 - 128) a ***cs*** je bit nastavení prescaleru (hodnota 0 => prescaler vypnut; hodnota 1 => prescaler zapnut).

Pokud známe přenosovou rychlost, tak můžeme vypočítat hodnotu R pro správné nastavení:

R = (10 000 / 29 / (1 + cs.7) / BR) - 1

Výchozí nastavení registru (POR):

- Nastavená bitová přenosová rychlost: 9,579 kbps

Doporučené nastavení registru:

- Nastavení rychlosti 15,674 kbps **0xC615**

Poznámky:

- *Je vhodné (nutné), aby byly vysílač a přijímač nastaveny pokud možno totožně kvůli eliminaci špatné synchronizace.*
- *Minimální dosažitelná hodnota přenosové rychlosti je pro **cs = 1** a **R = 127** => 0,336 kbps*

Receiver Control Command 0x9

	11	10	9	8	7	6	5	4	3	2	1	0	POR
1001	0	p16	d1	d0	i2	i1	i0	g1	g0	r2	r1	r0	0x9080

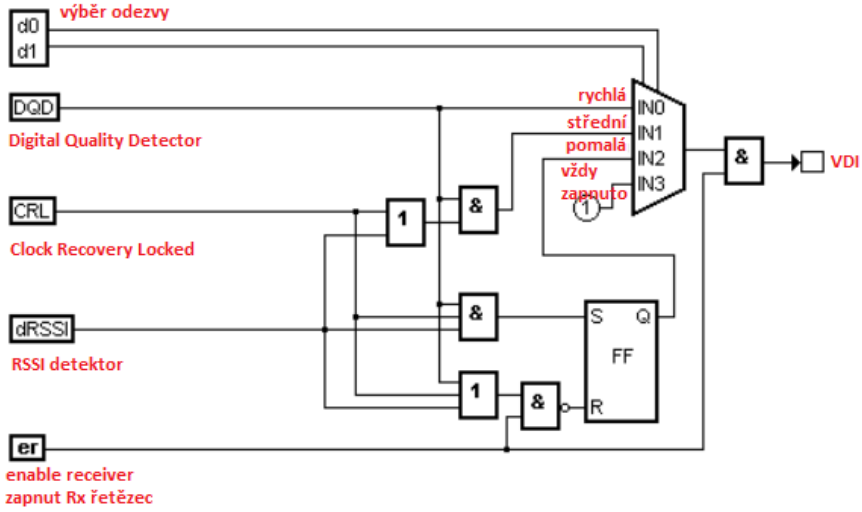
- Nastavuje funkci pinu nINT/VDI *(p16)*
- Nastavuje odezvu VDI *(d1 - d0)*
- Nastavuje šířku pásma přijímače Rx BW *(i2 - i0)*
- Nastavuje zisk vstupního zesilovače LNA (Low Noise Amplifier) *(g1 - g0)*
- Nastavuje práh úrovně RSSI *(r2 - r0)*

Význam jednotlivých bitů:

p16	význam
0	**zvolena funkce nINT**
1	zvolena funkce VDI

d1	d0	význam
0	**0**	**rychlá VDI odezva**
0	1	střední VDI odezva
1	0	pomalá VDI odezva
1	1	VDI vždy zapnuto

Podle toho, jaká varianta je zvolena se rozhoduje o platnosti přijímaného signálu:



- Výstup VDI bude aktivní pouze když je zapnut Rx řetězec.

*- **Rychlá odezva** pracuje pouze s DQD detektorem.*

*- **Střední odezva** pracuje s DQD detektorem a dále musí být splněno Clock Recovery Locked nebo úroveň RSSI.*

*- Pro **pomalou odezvu** musím mít splněny všechny podmínky. Tj. musím mít DQD, Clock Recovery Locked a úroveň RSSI současně a zároveň musí být zapnut Rx řetězec.*

Z popisu vyplívá, že pro zabezpečení nejrobustnějšího spojení zvolím pomalou odezvu (musí být splněny všechny podmínky vyhodnocení přijímaného signálu) a pokud mi nevadí občasné chyby ale zato vyžaduji maximální rychlost, tak zvolím rychlou odezvu.
Poslední možností je VDI zapnout vždy => nehledí na kvalitu přijímaných dat.

i2	i1	i0	význam (Rx BW)
0	0	0	rezervováno
0	0	1	400 kHz
0	1	0	340 kHz
0	1	1	270 kHz
1	0	0	200 kHz
1	0	1	134 kHz
1	**1**	**0**	**67 kHz**
1	1	1	rezervováno

g1	g0	význam (LNA zisk)
0	0	0 dB
0	1	-6 dB (zeslabení 4x)
1	**0**	**-14 dB (zeslabení 25x)**
1	1	-20 dB (zeslabení 100x)

r2	r1	r0	význam (RSSI)
0	0	0	-103 dBm (0,050 nW)
0	0	1	-97 dBm (0,199 nW)
0	**1**	**0**	**-91 dBm (0,794 nW)**
0	1	1	-85 dBm (3,162 nW)
1	0	0	-79 dBm (0,012 µW)
1	0	1	-73 dBm (0,050 µW)
1	1	0	rezervováno
1	1	1	rezervováno

Výchozí nastavení registru (POR):

- Pin nINT/VDI je nastaven na nINT
- VDI nastaveno na rychlou odezvu
- Šířka pásma 200 kHz
- Zisk LNA nastaven na 0 dB (žádné zeslabení)
- RSSI nastaveno na -103 dBm

Doporučené nastavení registru:

- Zapnuto nINT, rychlá odezva, BW=67kHz, LNA=-14dB a RSSI=-91dBm **0x90D2**

Poznámky:

- *Parametry RSSI a LNA spolu souvisejí. RSSI je úroveň přijímaného signálu, která když je překročena (přijímaný signál je větší než nastavené RSSI), tak je to indikováno bitem 8 STATUS registru. Pokud nastavíme RSSI např. na - 103 dBm a LNA na -20dB, tak RSSI je indikováno pokud má signál úroveň alespoň -83 dBm.*
- *Mezi šířkou pásma přijímače (Rx BW), přenosovou rychlostí a frekvenčním zdvihem vysílače existuje souvislost. Tyto 3 parametry nastavujeme společně s ohledem na to, jakou budeme využívat přenosovou rychlost. Nevhodně nastavená šířka pásma nebo modulační zdvih nám způsobí nárůst SNR (příliš velká BW) nebo naopak se symboly nepřenesou správně (příliš úzká BW).*
- *Nastavení Rx BW a frekvenčního zdvihu v závislosti na přenosové rychlosti ukazuje následující tabulka:*

přenosová rychlost	Rx BW	frekvenční zdvih	bw0 (0xCC)*
1200 bps	67 kHz	45 kHz	0
2400 bps	67 kHz	45 kHz	0
4800 bps	67 kHz	45 kHz	0
9600 bps	67 kHz	45 kHz	0
19 200 bps	67 kHz	45 kHz	0
38 400 bps	134 kHz	90 kHz	0
57 600 bps	145 kHz	90 kHz	0
115 200 bps	200 kHz	120 kHz	1
250 000 bps	400 kHz	240 kHz	1

* *Bit, který nastavuje šířku pásma PLL smyčky v registru PLL Setting Command 0xCC*

Data Filter Command 0xC2

		7	6	5	4	3	2	1	0	POR
1100	0010	al	ml	1	s	1	f2	f1	f0	0xC22C

- Nastavuje automatické ovládání clock recovery *(al)*
- Nastavuje režim clock recovery v případě manuálního ovládání - má význam pouze pro digitální filtr (*s=0*) v manuálním režimu (*al=0*) *(ml)*
- Volba typu filtru *(s)*
- Nastavení DQD prahu *(f2 - f0)*

Význam jednotlivých bitů:

al	význam
0	manuální ovládání clock recovery
1	**automatické ovládání clock recovery***

* *Clock recovery se nejprve spustí v rychlém módu a poté automaticky přepne do pomalého módu*

ml	význam
0	**pomalý mód clock recovery****
1	rychlý mód clock recovery**

** *pomalý mód je vhodný pro 12 - 16 bitové preamble. Vyžaduje přesné časování, za to je odolnější vůči šumu. rychlý mód je vhodný pro preamble 6 - 8 bitů dlouhé (preambule předchází synchronizaci a je určeno pro přijímač, aby si dokázal odvodit přenosovou rychlost (takt) - viz. schéma radiového paketu)*

s	význam
0	**digitální filtr*****
1	analogový RC filtr

*** *pro rychlosti do 115 kbps. Pro přenos až 256 kbps je nutné zvolit externí analogový RC filtr*

f2	f1	f0	význam (DQD)
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	**0**	**0**	**4**
1	0	1	5
1	1	0	6
1	1	1	7

Výchozí nastavení registru (POR):

- Manuální režim clock recovery v pomalém módu
- Digitální filtr
- DQD=4

Doporučené nastavení registru:

- Automatický režim, digitální filtr, DQD=4 **0xC2AC**

Poznámky:

- *Při použití externího analogového RC filtru se clock recovery a FIFO registr nedají použít*
- *RC filtr se připojí na pin DCLK. Hodnota rezistoru R = 10 kΩ. Hodnota kondenzátoru C se odvíjí od použité přenosové rychlosti, jak ukazuje následující tabulka:*

1,2 kbps	2,4 kbps	4,8 kbps	9,6 kbps	19,2 kbps	38,4 kbps	57,6 kbps	115,2 kbps	256 kbps
12 nF	8,2 nF	6,8 nF	3,3 nF	1,5 pF	680 pF	270 pF	150 pF	100

- *DQD je založeno na počítání špiček přijatého nefiltrovaných signálu a pomocí něj určujeme "kvalitu" přijatých dat. Ve většině případů postačí nastavení na 4. Pokud nastavíme menší hodnotu, může být šum detekován jako platný signál => (větší hodnota = přísnější detekce, ale může se stát, že se nám nepodaří přijmout žádaný signál).*

FIFO and Reset Mode Command 0xCA

		7	6	5	4	3	2	1	0	POR
1000	1010	f3	f2	f1	f0	sp	al	ff	dr	0xCA80

- Nastavuje, po kolika datových bitech (mimo synchronizaci) dojde k přerušení - signál nIRQ *(f3 - f0)*
- Volba délky synchronizačního řetězce *(sp)*
- Podmínky, za kterých se začne plnit FIFO *(al)*
- Povolení plnit FIFO *(ff)*
- Nastavení "citlivého" resetu *(dr)*

Význam jednotlivých bitů:

f3	f2	f1	f0	význam (počet bitů)
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	**0**	**0**	**0**	**8**
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

sp	význam
0	**2 synchronizační bajty***
1	1 synchronizační bajt*

* *sp=0 => synchronizace 0x2D D4*
sp=1 => synchronizace 0xD4
Synchronizační Bajt 0xD4 lze programově změnit na libovolný jiný pomocí registru Synchron Pattern Cmd. 0xCE

al	význam
0	**FIFO plněno po synchronizačním řetězci**
1	FIFO plněno vždy

ff	význam
0	plnění FIFO registru zakázáno
1	**plnění FIFO registru povoleno**

dr	význam
0	**citlivý reset**
1	citlivý reset vypnut

Výchozí nastavení registru (POR):

- Přerušení po 8 datových bitech
- 2 synchronizační bajty (výchozí hodnota 0x2DD4)
- Plnění FIFO je zakázáno a po povolení bude plněno po synchronizačním řetězci
- Citlivý reset

Doporučené nastavení registru:

- 8 datových registrů, 2 synchro bajty, FIFO povoleno po synchronizaci a vypnut citlivý reset **0xCA83**

Synchron Pattern Command 0xCE

		7	6	5	4	3	2	1	0	POR
1100	1110	b7	b6	b5	b4	b3	b2	b1	b0	0xCED4

- Nastavuje druhý bajt synchronizace – tento bajt je použit vždy i při nastavení ***sp=1*** (FIFO and Rst. Mode Cmd. 0xCA) *(b7 – b0)*

Nastavení synchronizačního bajtu se děje přímo pomocí bitů b0 – b7. Je vhodné se vyhnout kombinacím:

- 0x00 (0b00000000)
- 0xFF (0b11111111)

Je možné použít výchozí hodnotu 0xD4 (ale pozor na rušení od okolních zařízení s modulem RFM12B !!)

Výchozí nastavení registru (POR):

- Synchronizační byt je nastaven jako: **0xD4 = 0b11010100**

Poznámky:

- *Je užitečné si nastavit druhý bajt synchronizace na vlastní hodnotu, abychom předešli náhodnému spouštění přerušení nIRQ na základě příjmu dat z jiného, než našeho zařízení (zvonky, pokojové termostaty apod.)*

AFC Command 0xC4

		7	6	5	4	3	2	1	0	POR
1100	0100	a1	a0	rl1	rl0	st	fi	oe	en	0xC4F7

- Nastavuje režim měření kmitočtu *(a1 – a0)*
- Nastavuje maximální odchylky pracovního kmitočtu *(rl1 – rl0)*
- Povoluje uložení offsetu do STATUS registru *(st)*
- Nastavuje režim vysoké přesnosti měření *(fi)*
- Povoluje použití offset registru (spodní 4 bity STATUS registru) *(oe)*
- Povoluje výpočet kmitočtového offsetu (odchylky) *(en)*

Význam jednotlivých bitů:

a1	a0	význam (režim měření kmitočtu)
0	**0**	**AFC vypnuto**
0	1	pouze jedno měření po zapnutí napájení
1	0	pouze když je VDI log. 1
1	1	měření bez ohledu na VDI

* *a1=0, a0=1 : vhodné pro již odladěné, hotové aplikace. Toto nastavení dovoluje dosáhnout maximální komunikační vzdálenosti*
a1=1, a0=0 : vhodné, pokud jeden přijímač přijímá signály od více vysílačů.
a1=1, a0=1 : vhodné pro spojení jeden vysílač - jeden přijímač.
Více informací viz. datasheet

rl1	rl0	význam (max. odchylka)
0	0	odchylka se nezjišťuje
0	1	+15f ... -16f**
1	0	+7f ... -8f**
1	**1**	**+3f ... -4f****

** *pro modul 433 MHz je f = 2,5 kHz*
pro modul 868 MHz je f = 5 kHz
pro modul 915 MHz je f = 7,5 kHz

st	význam
0	**neukládá offset hodnotu do STATUS reg.**
1	ukládá offset hodnotu do STATUS reg.

fi	význam
0	zakáže režim vysoké přesnosti
1	**povoluje režim vysoké přesnosti*** **

*** *režim vysoké přesnosti měří kmitočet s cca. 2x větší přesností, ale 2x tak dlouho*

oe	význam
0	zakázáno použití offset registru
1	**povoleno použití offset registru**

en	význam
0	zakáže výpočet kmit. offsetu
1	**povolí výpočet kmit. offsetu**

Výchozí nastavení registru (POR):

- měření AFC bez ohledu na VDI
- Odchylky nastaveny na nejpřísnější (+3f ... -4f)
- Hodnota kmitočtového offsetu není ukládána do STATUS registru
- Režim vysoké přesnosti
- Výpočet kmitočtového offsetu povolen

Doporučené nastavení registru:

- Stejně jako POR **0xC4F7**

Tx Configuration Control Command 0x9

	11	10	9	8	7	6	5	4	3	2	1	0	POR
1001	1	0	0	mp	m3	m2	m1	m0	0	p2	p1	p0	0x9800

- Nastavuje polaritu modulace *(mp)*
- Nastavuje modulační kmitočtový zdvih *(m3 – m0)*
- Nastavuje výstupní výkon zesilovače *(p2 – p0)*

Význam jednotlivých bitů:

mp	význam (polarita modulace)
0	**positivní polarita modulace (1 = vyšší f)**
1	negativní polarita modulace (1 = nižší f)

m3	m2	m1	m0	význam (Tx mod. zdvih)
0	0	0	0	15 kHz
0	0	0	1	30 kHz
0	**0**	**1**	**0**	**45 kHz**
0	0	1	1	60 kHz
0	1	0	0	75 kHz
0	1	0	1	90 kHz
0	1	1	0	105 kHz
0	1	1	1	120 kHz
1	0	0	0	135 kHz
1	0	0	1	150 kHz
1	0	1	0	165 kHz
1	0	1	1	180 kHz
1	1	0	0	195 kHz
1	1	0	1	210 kHz
1	1	1	0	225 kHz
1	1	1	1	240 kHz

p2	p1	p0	význam (výstupní výkon)
0	0	0	0 (500 mW)
0	0	1	-3 dB (250 mW)
0	1	0	-6 dB (125 mW)
0	1	1	-9 dB (62,5 mW)
1	0	0	-12 dB (31,25 mW)
1	0	1	-15 dB (15,63 mW)
1	**1**	**0**	**-18 dB (7,81 mW)**
1	1	1	-21 dB (3,91 mW)

Výchozí nastavení registru (POR):

- Positivní polarita modulace
- Modulační zdvih 15kHz
- Výkon 500 mW

Doporučené nastavení registru:

- Positivní polarita modulace, modulační zdvih: 45 kHz, výkon: 7,81 mW **0x9826**

Poznámky:

- *Vzhledem k přesycení pásma 433 MHz (různé bezdrátové zvonky, termostaty apod.) důrazně doporučuji řídit se všeobecným oprávnění č. VO-R/10/04.2012-7 k využívání radiových kmitočtů a k provozování zařízení krátkého dosahu (viz. https://www.ctu.cz/cs/download/oop/rok_2012/vo-r_10-04_2012-07.pdf) hlavně z hlediska použitých výkonů, kdy RFM12 umožňuje tento výkon výrazně překročit !*
- *Kombinace nastavení parametrů vysílače, přijímače (Receiver Command Control) a přenosové rychlosti (Data Rate Command) zde uvedených funguje bezproblémů.*

PLL Setting Command 0xCC

		7	6	5	4	3	2	1	0	POR
1100	1100	0	ob1	ob0	lpx	ddy	ddit	1	bw0	0xCC77/0xCC67

- Nastavuje strmost nástupné a sestupné hrany signálů pro MCU, pokud je připojen na CLK vývod *(ob1 - ob0)*
- Nastavuje low power režim pro integrovaný krystal *(lpx)*
- Povoluje zpoždění fázového detektoru *(ddy)*
- Povoluje dithering z PLL (dithering zvětšuje možné rušení ?) *(ddit)*
- Nastavuje šířku pásma PLL smyčky *(bw0)*

Význam jednotlivých bitů:

ob1	ob0	význam (takt MCU z CLK)
0	**0**	**5 nebo 10 MHz (doporučeno)**
0	1	3,3 MHz
1	x	≤ 2,5 MHz

lpx	význam
0	doba náběhu = 1 ms (I = 620 µA)
1	**doba náběhu = 2 ms (I = 460 µA)**

ddy	význam
0	**zpoždění zakázáno**
1	zpoždění povoleno

ddit	význam
0	dithering zapnut
1	**dithering vypnut**

bw0	význam
0	max. bit rate = 86,2 kbps
1	**max. bit rate = 256 kbps**

Výchozí nastavení registru (POR):

- Strmost hran nastavená pro 2,5 MHz
- Doba náběhu integrovaného krystalu 2 ms
- Zpoždění fázového detektoru zakázáno
- dithering vypnut
- Max. bitrate až 256 kbps

Doporučené nastavení registru:

- Stejně jako POR **0xCC77**

Poznámky:

- *Pro verzi A0 RFM12 je POR hodnota 0xCC67 (doba náběhu integrovaného krystalu 1 ms). Je nutné ji v programu změnit na 0xCC77 (doba náběhu integrovaného krystalu 2 ms).*
- *Pro základní funkčnost modulu není potřeba PLL nijak nastavovat (viz. kapitola „Inicializace RFM12“).*

Low Battery Detector and MCU Clock Divider Command 0xC0

		7	6	5	4	3	2	1	0	POR
1100	0000	d2	d1	d0	0	v3	v2	v1	v0	0xC000

- Nastavuje dělič kmitočtu integrovaného krystalu na výstupu CLK pro MCU *(d2 – d0)*
- Nastavuje práh napětí pro detekci nízkého stavu baterie *(v3 – v0)*

Význam jednotlivých bitů:

d2	d1	d0	význam (kmitočet CLK pro MCU)
0	0	0	1 MHz
0	0	1	1,25 MHz
0	1	0	1,66 MHz
0	1	1	2 MHz
1	0	0	2,5 MHz
1	0	1	3,33 MHz
1	1	0	5 MHz
1	**1**	**1**	**10 MHz**

Pokud nebudeme výstup CLK používat jako externí zdroj hodin pro mcu, vypneme tuto funkci (dc = 1) registru power management command 0x82

Nastavení rozhodovací úrovně napětí pro detekci vybité baterie:

$$U_{bat} = 2{,}2 + V \cdot 0{,}1 \text{ [V]}$$

Kde V je dekadická hodnota složená z bitů v0 – v3. Minimální hodnota (V = 0b0000 => 2,2V), maximální hodnota (V = 0b1111 => 3,7V).

Výchozí nastavení registru (POR):

- Na výstup CLK je přiveden kmitočet 1 MHz
- Napětí baterie nastaveno na 2,2 V

Doporučené nastavení registru:

- Na výstup CLK kmitočet 10 MHz, napětí baterie 3,7 V **0xC0EF**

Poznámky:

- *Pokud chceme funkci hlídání napětí využívat, musí být povolena v power management command registru (0x82), bit **eb = 1**. Pokud napětí klesne pod nastavenou úroveň, jsme o tom vyrozuměni pomocí bitu LBD ve STATUS registru.*
- *Některé verze RFM12B mají možnost na 4. bitové pozici nastavit ještě bit v4 => maximální hodnota (V = 0b11111 => 5,3V).*
- *Pozor při použití různých typu baterií (Li-ion, Li-pol, NiMh apod.) na správné nastavení prahového napětí, aby nedošlo k hlubokému vybití baterie a jejího zničení (platí hlavně u lithiových baterií).*
- *Pro základní funkčnost modulu není potřeba nijak nastavovat (viz. kapitola „Inicializace RFM12“).*

Wake-up Timmer Command 0xE

	11	10	9	8	7	6	5	4	3	2	1	0	POR
111 r4	r3	r2	r1	r0	m7	m6	m5	m4	m3	m2	m1	m0	0xE196

- Nastavení exponentu **R** do rovnice *(r4 - r0)*
- Nastavení násobitele **M** do rovnice *(m7 - m0)*

Nastavení periody wake-up:

$$T = M \cdot 2^R \text{ [ms]}$$

Kde ***M*** je dekadická hodnota čísla složená z bitů ***m0 - m7*** a ***R*** je dekadická hodnota čísla složená z bitů ***r0 - r4***.

Výchozí nastavení registru (POR):

- Hodnota R = 1
- Hodnota M = 150
- => $T = 150 \cdot 2^1 = 300$ ms

Poznámky:

- *Minimální hodnota kterou lze nastavit je pro R=0 a M=1 => T = 1 ms.*
- *Maximální hodnota, kterou lze nastavit je R=29 a M=256 => T = 4,34 let*
- *Pro zajištění budoucí kompatibility by měla být hodnota **R** v rozmezí 0 až 29.*
- *Po každém probuzení musí být bit **ew** v registru **Power Management Command 0x82** nastaven znovu na 1 (tj. wake-up timmer povolen).*
- *Po každém probuzení se znovu provede kalibrace integrovaného krystalu. To zabere až 7ms.*
- *Wake-up lze rovněž použít i pro probuzení MCU.*
- *Pro základní funkčnost modulu není potřeba nijak nastavovat (viz. kapitola „Inicializace RFM12“).*

Low Duty-Cycle Command 0xC8

		7	6	5	4	3	2	1	0	POR
1100	1000	d6	d5	d4	d3	d2	d1	d0	en	0xC80E

- Nastavení low duty - cycle *(d6 - d0)*
- Povoluje režim Low Duty-Cycle Command *(en)*

Význam jednotlivých bitů:

en	význam
0	**Low Duty-Cycle vypnut**
1	Low Duty-Cycle zapnut

Výpočet Duty-cycle:

$$Duty = (d \cdot 2 + 1) / M \cdot 100\ [\%]$$

Kde ***d*** je dekadická hodnota čísla složená z bitů ***d0 - d6*** a ***M*** je hodnota násobitele z registru ***Wake-up Timmer Cmd.***

Příklad:
Perioda wake-up je nastavena na 300 ms. Duty cycle je nastaven na0x0E. Jak dlouho tedy bude přijímač aktivní?

Duty = (14 . 2 + 1) / 150 . 100 = 19,33 %
=> Přijímač bude aktivní 300 . 0,1933 = 57,99 ms

Výchozí nastavení registru (POR):
- Hodnota d = 14 => duty = 19,33 %
- Low Duty-Cycle zakázáno

Poznámky:
- *Low Duty-Cycle se používá jako další rozšíření wake-up režimu, pokud chci aktivovat pouze modul RFM12. Ten příjme data a pokud je vyhodnotí jako správné (nastaví VDI), pak teprve probudí i MCU. Toto opatření umožňuje úsporu energie.*
- *Přijímač je aktivní dokud je DQD v log. 1. Po rozeznání synchronizační skupiny je naplněno FIFO a vyvoláno přerušení pro MCU.*
- *Vysílač musí vysílat pakety po celou dobu, kdy předpokládá že je přijímač v provozu, aby s dostatečnou jistotou došlo k předání dat. Je výhodou, pokud známe přibližné časy, kdy je přijímač aktivní, protože to umožní šetřit energií i na straně vysílače.*
- *Vyšší přesnosti dosáhneme, pokud nebudeme řídit duty - cycle pomocí integrovaného 32 kHz Quartz oscilátoru, ale pomocí čítače v MCU (Timer 2 pro AT-MEGA).*
- *Pro základní funkčnost modulu není potřeba nijak nastavovat (viz. kapitola „Inicializace RFM12“).*

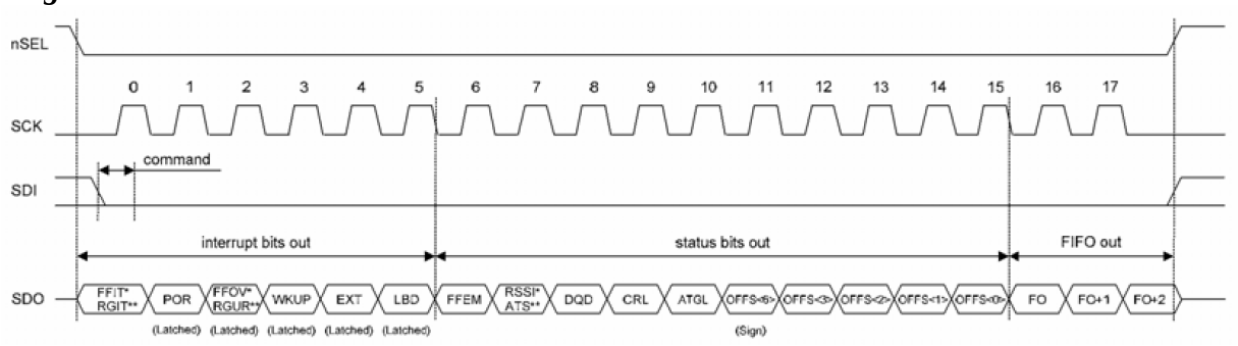
Reset Command 0xFE00
Existuje možnost, jak resetovat RFM12 softwarově. Tato možnost není v datasheetu popsána. Podmínkou je, že musí být nastaven citlivý reset (bit ***dr*** registru ***FIFO and Reset Mode Command0xCA***).

POZOR – tato funkcionalita není spolehlivě otestována !!!

9. STATUS registr

<u>Pro čtení status registru vyšleme do RFM12 příkaz 0x0000</u>. Jedná se o jediný příkaz, který začíná nulou a tak jej RFM12 ihned rozezná a na výstup SDO (u MCU MISO) začne vracet obsah STATUS registru.

Časový diagram komunikace:



Obr. 10: Schéma vyčítání STATUS registru

Poznámky:

- *Čtení STATUS registru po zapnutí napájení způsobí, že se linka nIRQ nastaví do log. 1 (tj. IRQ není aktivní). Je to proto, že příznak bit POR ze skupiny interupt bits out je nastaven a čtením STATUS registru jej vymažeme, čímž se nIRQ stane neaktivní.*

Tabulka určující význam jednotlivých bitů STATUS registru:

bit	Tx (er=0)	Rx (er=1)	nazev	význam
15 (MSB)	**RGIT**	**FFIT**	register interrupt FIFO interrupt	RGIT = 1 (Tx registr je připraven přijmout další Byte) bit je vynulován: - vypnutím vysílače (et=0) - vypnutím přístupu k registrům (el=0) FFIT = 1 (Počet přijatých data bitů v FIFO Rx registru dosáhl nastavené úrovně (reg. FIFO and rst. mode cmd 0xCA) bit je vynulován: - vypnutím přijímače (er=0) - vypnutím přístupu k FIFO (ef=0)
14	**POR**		power on reset	POR = 1 (všechny registry jsou v default hodnotě) bit je vynulován: - samotným příkazem k čtení STATUS reg.
13	**RGUR**	**FFOV**	register underrun FIFO overflow	RGUR = 1 (Data nejsou do Tx registru dodávána dostatečně rychle, nebo došlo k jejich přepsání) bit je vynulován: - vypnutím vysílače (et=0) - vypnutím přístupu k registrům (el=0) FFOV = 1 (Rx FIFO registr přetekl - data nejsou čtena dost rychle) bit je vynulován: - vypnutím přijímače (er=0) - vypnutím přístupu k FIFO (ef=0)
12	**WKUP**		wakeup	WKUP = 1 (wake-up timmer vypršel (dojde k probuzení ?)) bit je vynulován: - samotným příkazem k čtení STATUS reg. - vypnutím povolení wake-up (ew=0)
11	**EXT**		external interrupt	EXT = 1 (na pinu nINT došlo k žádosti o přerušení ze strany MCU - linka byla nastavena na log. 0) bit je vynulován: - samotným příkazem k čtení STATUS reg. - vypnutím funkce nINT (p16=1) - na lince nINT MCU opět nastaví log. 1
10	**LBD**		low battery detect	LBD = 1 (napájecí napětí je nižší než nastavený limit) bit je vynulován: - zvýšením napájecího U nad nastavenou mez - zvýšením meze pro detekci nízkého nap. U - vypnutím low batt. detektoru (eb=0)
9	FFEM		FIFO empty	FFEM = 1 (FIFO Rx registr je prázdný)
8	ATS	RSSI	antenna tuning: strong received signal strenght	RSSI = 1 (síla přijímaného signálu je větší než nastavený limit) ATS = 1 (obvod antény detekoval silný signál - může znamenat nesprávně přizpůsobenou anténu ?)
7	DQD		data quality detector	DQD = 1 (přijímaná data odpovídají nastavené požadované kvalitě DQD)
6	CRL		clock recovery locked	CRL = 1 (obnovení taktu přijímaných dat v pořádku)
5	ATGL		AFC toggle	
4	OFSS6		sign of offset	
3	OFFS3		offset	
2	OFFS2		offset	
1	OFFS1		offset	
0 (LSB)	OFFS0		offset	

Tab. 7: Popis bitů STATUS registru

10. Události, které způsobí přerušení nIRQ

Linka nIRQ slouží k přerušení ze strany modulu RFM12 na MCU. **Když je přerušení aktivní, je linka ve stavu log. 0**. Tam se dostane po určitých definovaných událostech:

- **Tx registr je připraven přijmout další byte** (bit RGIT ve STATUS registru). Díky systému přerušení se nemusíme starat o přesné časování s ohledem na zvolenou přenosovou rychlost radiového spojení. Modul dává sám MCU najevo, kdy požaduje další byte, aby byla přenosová rychlost přesně dodržena.
- **FIFO Rx registr přijal nastavený počet bitů** (bit FFIT ve STATUS registru). Nemusíme v programu MCU neustále hlídat, jestli nejsou přítomna ve FIFO nová data, ale vhodně využijeme systému externího přerušení a obsluha modulu ze strany MCU bude provedena až tehdy, jsou-li přítomna nová data. Zde si ale musíme velmi pečlivě hlídat, aby nám nedocházelo ke spouštění přerušení od jiných vysílačů. Je vhodné změnit druhý synchronizační bajt z defaultní hodnoty 0xD4 na jinou a v případě potřeby i zvýšit útlum LNA a snížit citlivost RSSI.
- **Power-on Reset** (bit POR ve STATUS registru). Po zapnutí nebo resetu modulu máme aktivní přerušení. To "vynulujeme" prostým přečtením STATUS registru (popsáno v kapitole 10).
- **Wake-up Timmer timeout** (bit WKUP ve STATUS registru). V momentě kdy dojde k probuzení modulu pomocí wake-up timeru je nastaveno přerušení, které dovolí také probudit připojený MCU.
- **Žádost o přerušení ze strany MCU** (bit EXT ve STATUS registru). Pokud máme na pinu nINT/VDI povolenu funkci nINT (bit ***p16 = 0*** registru ***Receiver Control Cmd. 0x9***) a MCU nastaví stav linky do log. 0, znamená to, že MCU spustil v modulu přerušení. Po vykonání zadaných operací a poté co MCU nastaví linku nINT zpět do log. 1 je přerušení ukončeno a modul pokračuje v normální činnosti.
- **Napájecí napětí kleslo pod nastavený práh** (bit LBD ve STATUS registru). Je využito, pokud nastavíme bit ***eb = 1*** v registru ***Power Management Cmd. 0x82*** a zároveň nastavíme práh napájecího napětí v registru ***Low Batt. and MCU Clock Divider Cmd. 0xC0***.

11. Čtení STATUS registru RFM12B

Čtení STATUS registru probíhá, jak bylo naznačeno v kapitole 9. Praktická ukázka procedury **RFM12_ReadStatus()** v jazyce C (IDE: CodeVision):

```
unsigned char RFM_STATUS_HIGH = 0x00;          // Globalni registr pro ulozeni hornich 8 bitu STATUS registru
unsigned char RFM_STATUS_LOW= 0x00;            // Globalni registr pro ulozeni spodnich 8 bitu STATUS registru

void RFM12_ReadStatus()
{
   RFM_STATUS_HIGH = SPI_Data(0x00,1);         // RFM12 nacteni status registru
   RFM_STATUS_LOW = SPI_Data(0x00,0);
   return;
}
// FUNKCE SPI_data pro komunikaci po SPI sbernici je posana v kapitole cislo 15
```

12. Hardwarový reset RFM12B

Pokud máme vyvedenou linku **nRES** na univerzální I/O pin MCU, tak se jedná o velmi jednoduchou operaci. **V klidovém režimu musí být linka v log. 1 !** (neopomeneme nastavit při inicializaci MCU).
Samotný reset se provede tak, že shodíme linku nRES do log. 0 na 10ms a následně opět nastavíme log. 1 a počkáme 50ms.

Příklad v jazyce C (IDE: CodeVision)

```
#define         nRES            PORTD.4         // reset RFM12 - OUTPUT
#include        <delay.h>                       // hlavickovy soubor pro zpozdeni

void RFM12_HW_Reset()
{
   nRES = 0;                                    // RFM12 HW reset
   delay_ms(10);
   nRES = 1;
   delay_ms(50);
   return;
}
```

13. Power on Reset test (práce s bitem „POR“ registru STATUS)

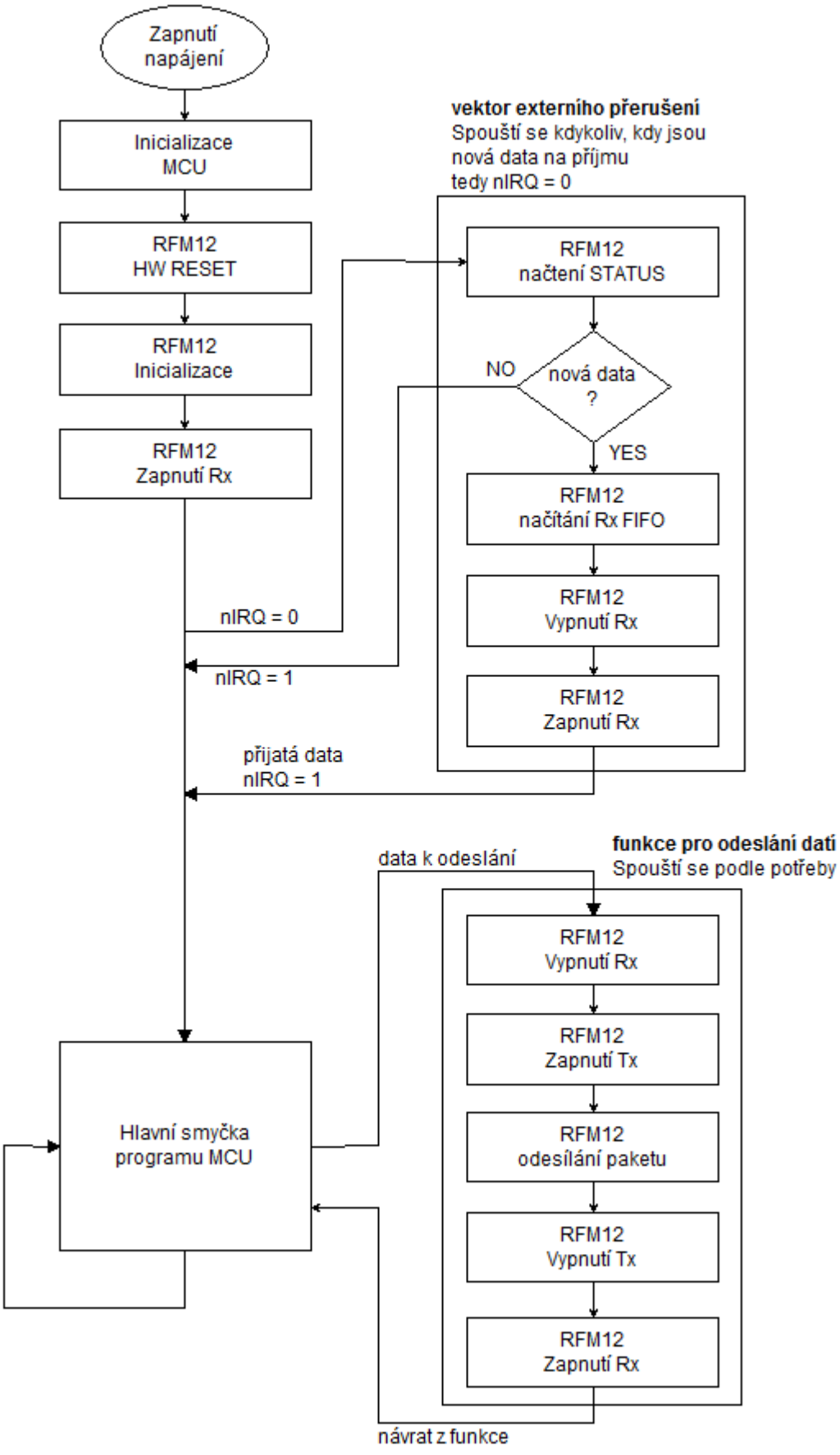
Ihned po přivedení napájecího napětí je linka nIRQ ve stavu log. 0 (přerušení aktivní) a to proto, že ve STATUS registru je nastaven bit POR do log. 1.
Přečtením STATUS registru a vyhodnocením bitu POR můžeme provést „power-on self test“, kdy si ověříme hned při spuštění modulu jeho funkčnost. Navíc samotným přečtením nastavíme nIRQ do log. 1 (přerušení neaktivní).

Příklad v jazyce C (IDE: CodeVision)

```
#define         nIRQ            PIND.2          // pozadavek na preruseni od RFM12BP (EXT_INT0) – INPUT

void RFM12_POR_STATUS()                         // POR test
{
   if (nIRQ == 0){
      RFM12_ReadStatus();
      if (RFM_STATUS_HIGH == 0x40){
         putsf("RFM12 Power on Reset [OK]");    // RFM12 POR Ready
      }else{
         putsf("RFM12 Power on Reset [FAIL]");  // RFM12 POR Fail
      }
   }else{
      putsf("RFM12 Power on Reset [nIRQ high]"); // nIRQ line high
   }
   return;
}
```

14. Filosofie práce s radiovým modulem

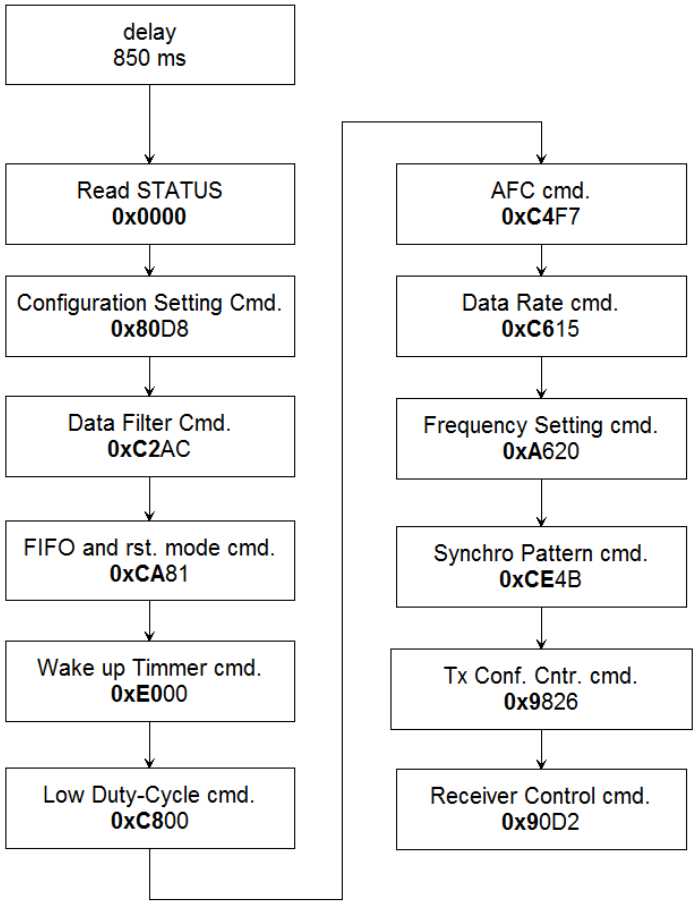
Základní princip spočívá v tom, že nejprve provedeme HW restart modulu a následně jej inicializujeme (viz. kapitola č. 15), tj. nastavíme hlavní parametry. V dalším kroku poté zapneme přijímač a povolíme přístup k Rx FIFO registrům. Obecně platí, že pokud není radiová stanice v režimu vysílání, tak je zbytek času na příjmu, protože nikdy dopředu nemůžeme vědět, kdy protistrana zahájí komunikaci. Obecný princip radiostanice ukazuje následující blokové schéma:



Obr. 11: Obecné schéma činnosti radiostanice

15. Inicializace radiového modulu RFM12B

Postup inicializace:
Pořadí jednotlivých kroků vychází z datasheetu modulu RFM12BP, "programing guide" RF12B verze 1.1 a zdrojů z internetu. Toto konkrétní uspořádání je v praxi odzkoušené, nicméně to nevylučuje i jiný, efektivnější způsob inicializace.



Obr. 12: Schéma inicializace radiového modulu RFM12

Modul inicializovaný dle schéma na obrázku číslo 8 bude mít následující parametry:

- Pásmo 433 MHz; XTAL cap = 12,5pF; FIFO Tx a FIFO Rx registry povoleny
- Digitální filtr; Automatický a pomalý mód obnovy hodinového taktu; Práh DQD = 4
- Plnění Rx FIFO až po synchronizaci; spouštění přerušení po 8 přijatých bitech v Rx FIFO; 2 synchro bajty; Reset s nízkou citlivostí; Přístup k Rx FIFO zakázán
- Wake up Timmer neaktivní
- Low Duty-Cycle neaktivní
- Měření AFC je bez ohledu na VDI; Odchylka je nastavena +3f ...-4f; Hodnota kmitočtového offsetu není ukládána do STATUS registru; Režim vysoké přesnosti aktivní; Výpočet kmitočtového offsetu povolen
- Přenosová rychlost: 15,674 kbps
- Střední hodnota kmitočtu: 433,9200 MHz
- Druhý synchronizační bajt má hodnotu 0x4B
- Positivní modulace; Kmitočtový zdvih Tx = 45 kHz; Výstupní výkon zesilovače = -18dB
- Funkce VDI/nINT zvolena na nINT; rychlé VDI; LNA útlum = -14dB; Šířka pásma přijímače = 67 kHz a citlivost = -91dBm

poznámka:

- *Přijímač a vysílač (ovládány příkazem Power Management cmd. 0x82) jsou vypnuty. Je zapnut pouze interní krystal.*
- *Radiový modul má nastavené potřebné parametry pro provoz a nyní již bude stačit ovládat přepínání přijímače a vysílače.*

Příklad inicializace v jazyce C (IDE: CodeVision)

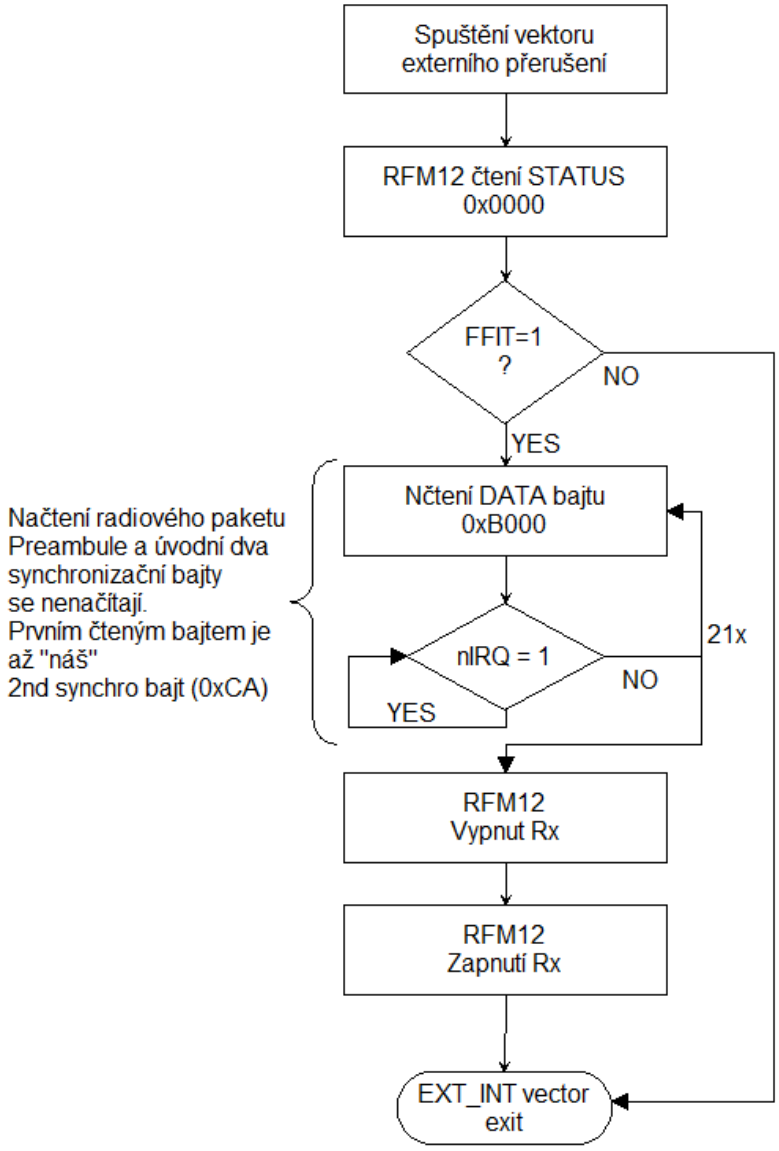
```
#define          SS              PORTB.4        // Slave Select (nSEL na RFM12BP) – OUTPUT


/*FUNKCE SPI_data pro komunikaci po SPI sbernici. Slouzi pro odesilani a prijem dat po SPI
dva mody zapisu - pro 8 bitove a 16 bitove cislo
  - IS_16BIT_NUM = 0 (plati pro 8 bitova cisla)
  - IS_16BIT_NUM = 1 (plati pro 16 bitova cisla)*/
unsigned char SPI_Data(unsigned char DATA_TO_SPI, unsigned char IS_16BIT_NUM)
{
   unsigned char DATA_FROM_SPI = 0x00;
   SS = 0;
   SPDR = DATA_TO_SPI;                          // start transmission from Master to Slave
   while (!(SPSR & (1<<SPIF)));                 // Wait for transmission complete
   if (IS_16BIT_NUM == 0){                      // Pokud bude nacitat 16bit cislo, tak drz Slave Select=0
      SS = 1;}                                  // pro dalsi byte
   DATA_FROM_SPI = SPDR;
   delay_us(1);
   return DATA_FROM_SPI;
}



void RFM12_Init()
{
delay_ms(850);                                  // cekej dokud se neustali napajeci napeti
// nastaveni RFM12
SPI_Data(0x00,1);                               // nulovy byte - zacatek konfigurace (cteni STATUS registru !!!)
SPI_Data(0x00,0);
SPI_Data(0x00,1);                               // RFM12 Configuration Setting Cmd. 0x80
SPI_Data(0xD8,0);
SPI_Data(0xC2,1);                               // RFM12 Data Filter Cmd. 0xC2
SPI_Data(0xAC,0);
SPI_Data(0xCA,1);                               // RFM12 FIFO and Reset Mode Cmd. 0xCA
SPI_Data(0x81,0);
SPI_Data(0xE0,1);                               // RFM12 Wake Up Timmer Cmd. 0xE0
SPI_Data(0x00,0);
SPI_Data(0xC8,1);                               // RFM12 Low Duty Cycle Cmd. 0xC8
SPI_Data(0x00,0);
SPI_Data(0xC4,1);                               // RFM12 AFC Cmd. 0xC4
SPI_Data(0xF7,0);
SPI_Data(0xC6,1);                               // RFM12 Data Rate Cmd. 0xC6
SPI_Data(0x15,0);
SPI_Data(0xA6,1);                               // RFM12 Frequency Setting Cmd. 0xA (SRAM Buffer)
SPI_Data(0x20,0);
SPI_Data(0xCE,1);                               // RFM12 Synchron Pattern Cmd. 0xCE
SPI_Data(0x4B,0);
SPI_Data(0x98,1);                               // RFM12 Tx Configuration Control Cmd. 0x98
SPI_Data(0x26,0);
SPI_Data(0x90,1);                               // RFM12 Receiver Control Cmd.
SPI_Data(0xD2,0);
return;
}
```

16. Čtení dat z RFM12B (příjem)

Příjem dat můžeme realizovat v zásadě dvěma způsoby:

- Stálým dotazováním na stav linky nIRQ a v případě, že linka spadne z log. 1 do log. 0 (IRQ aktivní), tak nejprve vyčteme STATUS registr a pokud bude nastaven bit FFIT provedeme čtení Rx FIFO registru.
- Pomocí externího přerušení, kdy spuštěný vektor přerušení provede výše popsané kroky až v okamžiku, kdy to bude potřeba. Tato metoda je podstatně efektivnější, obzvláště tehdy pokud námi vybraný typ MCU podporuje externí přerušení.

Další věcí, kterou musíme mít na paměti je nastavení přístupu k FIFO a skutečnost, kdy se spustí přerušení (příkaz: ***FIFO and Reset mode command 0xCA..***). V našem případě máme nastaveno, že se Rx FIFO začne plnit až po dvou úvodních synchronizačních bajtech (0x2D a 0xD4) a přerušení nIRQ spustí, až bude v Rx FIFO přijato 8 bitů, tedy celý 1. bajt přenášeného paketu. V příkladu budeme uvažovat paket 16/25, uvedený v kapitole 7.



Obr. 13: Vývojový diagram funkce pro čtení přijatých dat z RFM12B

Zvláštností Rx procedury je, že na konci příjmu, až jsme si jisti, že jsme přijali poslední bajt paketu (proto je výhodné používat pakety fixní délky) musíme přijímač vypnout a znovu zapnout, protože jinak se nám bude neustále aktivovat přerušení nIRQ (viz. tabulka 7 a popis jak se nuluje bit FFIT)
Procedura zapnutí přijímače spočívá v tom, že musíme zapnout samotnou jednotku přijímače (příkaz: ***Power Managemet Command 0x82..***) a následně i povolit přístup k Rx FIFO registrů (příkaz: ***FIFO and Reset Mode Command 0xCA..***)

Příklad funkce pro zapnutí přijímače v jazyce C (IDE: CodeVision)

```
#define     RFM12_TXEN      PORTD.6     // Tx ENABLE (RFM12BP) - OUTPUT
#define     RFM12_RXEN      PORTD.7     // Rx ENABLE (RFM12BP) - OUTPUT

unsigned char DATA_INPUT[21]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
unsigned char *DATA_IN_POINTER = DATA_INPUT;


void RFM12_Receiver_On ()                   // Procedura zapnuti prijimace
{
// zapnuti prijimace
SPI_Data(0x82,1);                           // RFM12 Power Management Cmd. 0x82
SPI_Data(0xC9,0);                           // zapnuti prijmace 0xC9
SPI_Data(0xCA,1);                           // RFM12 FIFO and Reset Mode Command 0xCA
SPI_Data(0x83,0);                           // Povoleni pristupu k Rx FIFO registru 0x83
RFM12_RXEN = 1;
while (nIRQ == 0){ }                        // ceka dokud je nIRQ=0 => dokonceni vnitrnich operaci RFM12
return;
}
```

Příklad funkce pro načítání přijatých dat v jazyce C (IDE: CodeVision)

```
void RFM12_Data_Received (unsigned char *INPUT_DATA)
{
unsigned char i = 0;

for (i=0;i<21;i++){
   SPI_Data(0xB0,1);                        // RFM12 cteni Rx FIFO registru 0xB0
   INPUT_DATA[i] = SPI_Data(0x00,0);        // SPI clk 0x00
   while (nIRQ == 1){ }                     // Cekej na dalsi nIRQ
}
// vypnuti prijmace
SPI_Data(0x82,1);                           // RFM12 Power Management Cmd. 0x82
SPI_Data(0x08,0);                           // 0x08 (Rx i Tx vypnut)
RFM12_RXEN = 0;
// zapnuti prijimace
RFM12_Receiver_On();
return;
}


interrupt [EXT_INT0] void nIRQ_Request (void)   // Vektor externiho preruseni
{
unsigned char MODE = 0x00;

RFM12_ReadStatus();                         // funkce pro nacteni STATUS registru
MODE = (RFM_STATUS_HIGH & 0b10000000);      // premaskovani bitu RGIT/FFIT
switch (MODE){
   case 0x80:{                              // RGIT/FFIT = 1
      if (RFM12_RXEN == 1){                 // RFM12 v rezimu prijmu
         RFM12_Data_Received(DATA_IN_POINTER);
         RADIO_STATUS_REGISTER |= (1<<7);   // nastaveni priznaku, ze doslo k prijmu dat (pro potreby „main“)
      }
   }break;
}
return;
}
```

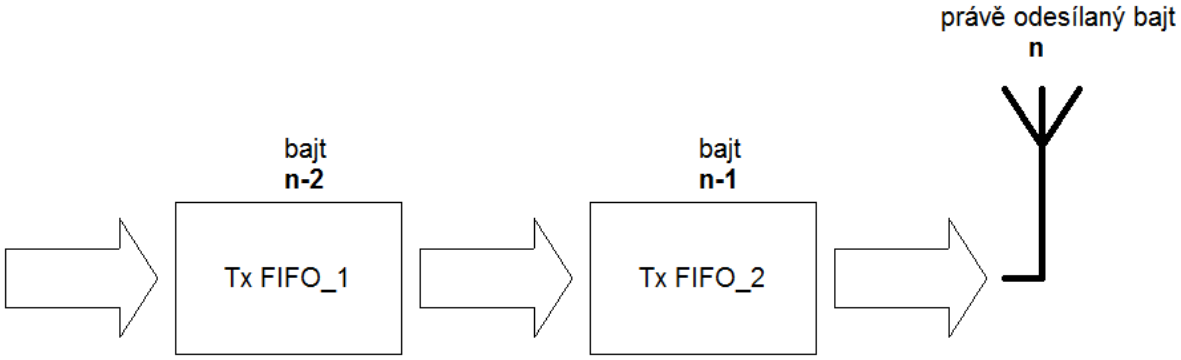
17. Zápis dat do RFM12B (vysílání)

Odesílání dat se děje postupným zápisem bajtů do vysílacího FIFO registru. Důležité je dodržet časování odesílaných bajtů, k čemuž vhodně použijeme vlastnosti **nIRQ** linky, která tím, že přejde ze stavu log. 1 (neaktivní) do stavu log. 0 (aktivní) dává najevo, že očekává další bajt.
Při samotném vysílání musíme mít na paměti strukturu radiového paketu (viz. kapitola 7: problematika přenosové rychlosti), že v paketu nejsou pouze samotná data, ale ještě minimálně 2 bajty preambule (0xAA), která slouží pro synchronizaci přijímače na vysílač, 1 až 2 bajty „datové synchronizace“. V následujícím příkladu budeme uvažovat paket 16/25 z kapitoly 7.

Preamble	Preamble	Synchronizace		2nd Synchro	service Byte	DATA	CRC	rezerva	rezerva
0xAA	0xAA	0x2D	0xD4	0xCA	1 Byte	16x Byte	1 Byte	1 Byte	1 Byte

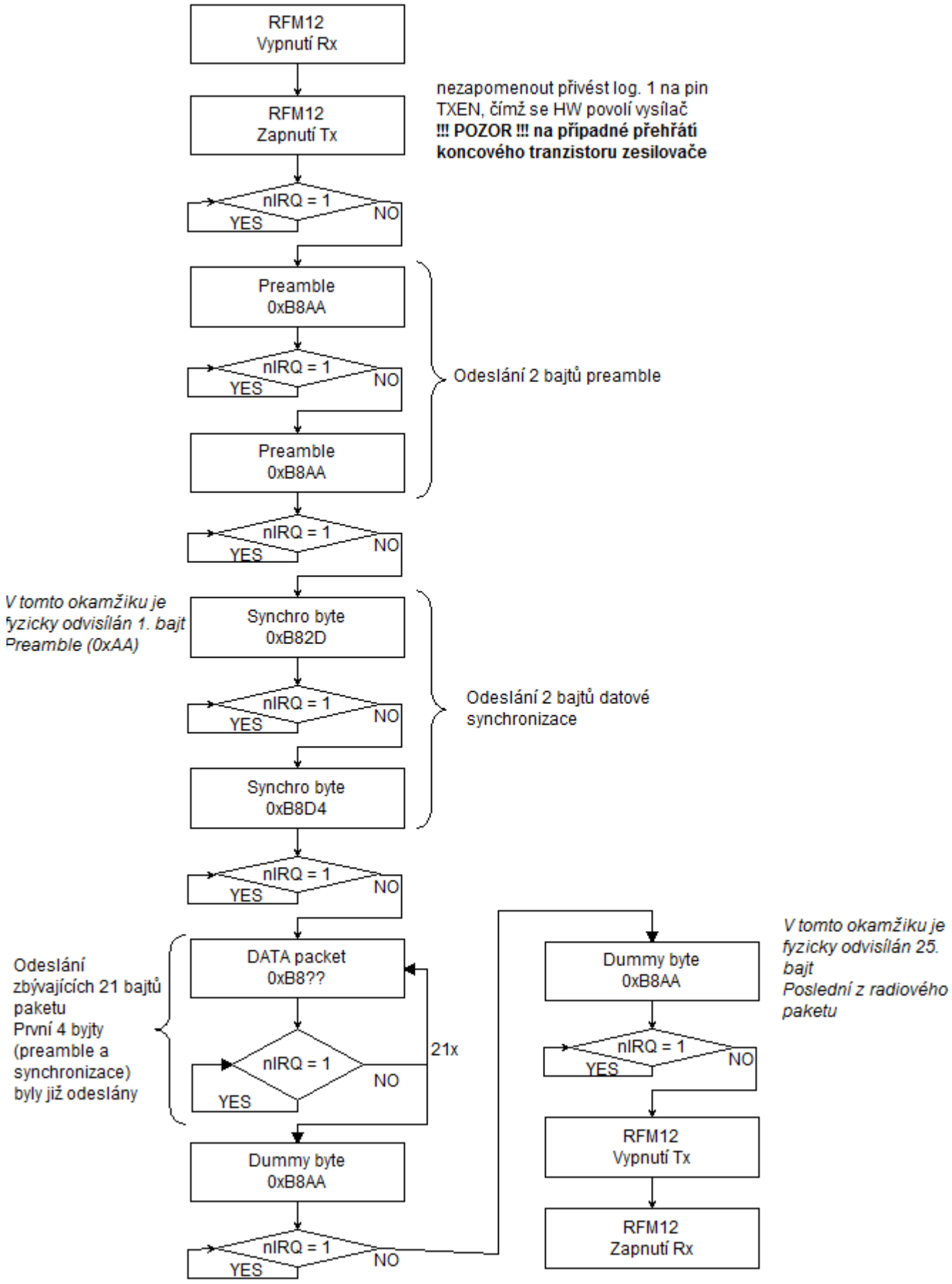
Hlavní smyčka programu bude pracovat se servisním bajtem, 16 datovými bajty s CRC a dvěma rezervními bajty. Preambule, a tři bajty synchronizace jsou pro všechny odesílané pakety neměnné.

Blokové schéma vysílače z pohledu dat ukazuje následující schéma:



Obr. 14: Blokové schéma vysílače

Jak ze schématu vyplívá, tak pokud odešleme do RFM12 bajt k odvysílání (příkaz: **Transmitter Register Write Command 0xB8..**), tak nedojde k odvysílání dat do antény okamžitě, ale bajt se přesune do registru Tx FIFO_1. Poté musíme odeslat druhý bajt, čímž zaujme místo 1. bajtu v registru Tx FIFO_1 a původní 1. bajt se přesune do registru Tx FIFO_2. Až teprve odesláním 3. bajtu se konečně 1. bajt dostane „do antény“ a je odvysílán. 2. bajt se přesune do registru Tx FIFO_2 a 3. bajt je umístěn do registru Tx FIFO_1 a tak dále...
Tento princip musíme mít na paměti především u konce radiového paketu, kdy pro korektní odvysílání posledního bajtu musíme zapsat ještě dva bajty „navíc“. Ty mohou být ale libovolné, protože stejně zůstanou v Tx registrech a až do dalšího vysílání se nedostanou ven. Je proto výhodné poslední dva bajty použít stejné jako preambuli (0xAA), díky čemuž při dalším vysílání dáme přijímači 2x více vzorků pro synchronizaci na vysílač.



Obr. 15: Vývojový diagram funkce pro vysílání dat z RFM12B

Příklad funkce pro vysílání radiového paketu v jazyce C (IDE: CodeVision)

```
unsigned char DATA_OUTPUT[21]={0xCA,0x41,0x48,0x4F,0x4A,0x20,0x4A,0x41,0x20,0x4A,0x53,0x45,0x4D,0x20,0x56,0x65,0x6E,0x64,0x61,0x01,0x00};
unsigned char *DATA_OUT_POINTER = DATA_OUTPUT;


void RFM12_Send_Data (unsigned char *DATA_TO_SEND){
unsigned char i = 0;
// vypnuti prijimace
SPI_Data(0x82,1);                       // RFM12 Power Management Cmd. 0x82
SPI_Data(0x08,0);                       // 0x08 (Rx i Tx vypnut)
RFM12_RXEN = 0;
// Zapnuti vysilace
SPI_Data(0x82,1);                       // RFM12 Power Management Cmd. 0x82
SPI_Data(0x39,0);                       // 0x39 (Tx zapnut)
RFM12_TXEN = 1;
while (nIRQ == 1){ }                    // ceka dokud neni nIRQ=0 => zada o preruseni (dalsi Tx data)
// Preamble
SPI_Data(0xB8,1);                       // RFM12 zapis do Tx registru 0xB8
SPI_Data(0xAA,0);                       // Preamble 0xAA
while (nIRQ == 1){ }                    // ceka dokud neni nIRQ=0 => zada o preruseni (dalsi Tx data)
SPI_Data(0xB8,1);                       // RFM12 zapis do Tx registru 0xB8
SPI_Data(0xAA,0);                       // Preamble 0xAA
while (nIRQ == 1){ }                    // ceka dokud neni nIRQ=0 => zada o preruseni (dalsi Tx data)
// Synchro
SPI_Data(0xB8,1);                       // RFM12 zapis do Tx registru 0xB8
SPI_Data(0x2D,0);                       // 1st synchro pattern 0x2D
while (nIRQ == 1){ }                    // ceka dokud neni nIRQ=0 => zada o preruseni (dalsi Tx data)
SPI_Data(0xB8,1);                       // RFM12 zapis do Tx registru 0xB8
SPI_Data(0x4B,0);                       // 2nd synchro pattern (0x4B)
while (nIRQ == 1){ }                    // ceka dokud neni nIRQ=0 => zada o preruseni (dalsi Tx data)
// DATA K PRENOSU
for (i=0;i<21;i++){                     // 21 opakovani ve smycce
SPI_Data(0xB8,1);                       // RFM12 zapis do Tx registru 0xB8
SPI_Data(DATA_TO_SEND[i],0);            // DATA K PRENOSU
while (nIRQ == 1){ }                    // ceka dokud neni nIRQ=0 => zada o preruseni (dalsi Tx data)
}
// Dummy Bytes
SPI_Data(0xB8,1);                       // RFM12 zapis do Tx registru 0xB8
SPI_Data(0xAA,0);                       // 1st Dummy Byte 0xAA
while (nIRQ == 1){ }                    // ceka dokud neni nIRQ=0 => zada o preruseni (dalsi Tx data)
SPI_Data(0xB8,1);                       // RFM12 zapis do Tx registru 0xB8
SPI_Data(0xAA,0);                       // 2nd Dummy Byte 0xAA
while (nIRQ == 1){ }                    // ceka dokud neni nIRQ=0 => zada o preruseni (dalsi Tx data)
// vypnuti vysilace
SPI_Data(0x82,1);                       // RFM12 Power Management Cmd. 0x82
SPI_Data(0x08,0);                        // 0x08 (Rx i Tx vypnut)
RFM12_TXEN = 0;
// zapnuti prijimace
RFM12_Receiver_On();                    // Funkce pro zapnuti Rx
return;}


void main (){
RFM12_Send_Data(DATA_OUT_POINTER);      // Funkce pro odeslani radioveho paketu
return;}
```

18. Použité zdroje

[1] Universal ISM band FSK transceiver module with 500mW output power RFM12BP; Hope RF datasheet; anglicky

[2] RFM12B Universal ISM Band FSK Transceiver; Hope RF datasheet; anglicky

[3] RF12B Universal ISM Band FSK Transceiver v1.1; Hope RF datasheet; anglicky

[4] http://www.mikrocontroller.net/articles/RFM12; Wiki stránka o RFM12; německy

[5] http://gobotronics.wordpress.com/2010/10/07/rfm12-programming/; návody k obsluze RFM12; anglicky

[6] http://blog.strobotics.com.au/2008/01/08/rfm12-tutorial-part1/; popis a zkušenosti s RFM12; anglicky

[7] http://tools.jeelabs.org/rfm12b.html; Kalkulátor konfiguračních slov RFM12; anglicky