VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV RADIOELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF RADIO ELECTRONICS

ANALYZÁTOR UHF RFID KOMUNIKACE ZALOŽENÝ NA SDR

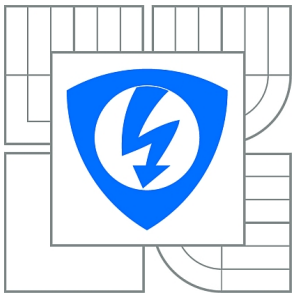
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. JOSEF VYCHODIL
AUTHOR

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV RADIOELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF RADIO ELECTRONICS

ANALYZÁTOR UHF RFID KOMUNIKACE ZALOŽENÝ NA SDR

SDR-BASED UHF RFID COMMUNICATION ANALYZER

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

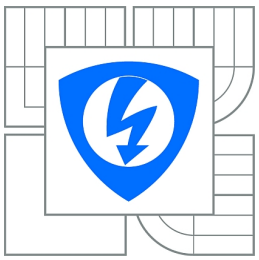
AUTOR PRÁCE — Bc. JOSEF VYCHODIL

AUTHOR

VEDOUCÍ PRÁCE — Ing. ALEŠ POVALAČ

SUPERVISOR

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav radioelektroniky

Diplomová práce

magisterský navazující studijní obor
Elektronika a sdělovací technika

Student: Bc. Josef Vychodil ***ID:*** 115311
Ročník: 2 ***Akademický rok:*** 2012/2013

NÁZEV TÉMATU:

Analyzátor UHF RFID komunikace založený na SDR

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte princip RFID komunikace pro pásmo UHF. Seznamte se s RFID protokolem Gen2, platformou softwarově definovaného rádia Ettus USRP-N200 a principem ovladačů UHD.

Navrhněte koncepci systému pro odposlech evropského UHF RFID pásma a jeho analýzu v reálném čase. Vyberte vhodné vývojové prostředí (Visual C++, LabVIEW, MATLAB nebo GNU Radio). Vytvořte software pro zachycení a zobrazení Gen2 komunikace na jednotlivých kanálech.

Rozšiřte vytvořený software o pokročilou analýzu protokolu Gen2, dekódování jednotlivých příkazů a odpovědí tagů, ověřování kontrolních součtů. Otestujte funkci a proveďte experimentální měření s vyvinutým systémem.

DOPORUČENÁ LITERATURA:

[1] DOBKIN, D. M. The RF in RFID: Passive UHF RFID in Practice. Burlington: Newnes, 2008.

[2] Class-1 Generation-2 UHF RFID Protocol for Communications at 860 MHz – 960 MHz. Version 1.2.0. EPCglobal Inc., 2008.

Termín zadání: 11.2.2013 ***Termín odevzdání:*** 24.5.2013

Vedoucí práce: Ing. Aleš Povalač
Konzultanti diplomové práce:

prof. Dr. Ing. Zbyněk Raida
Předseda oborové rady

ABSTRAKT

Tato práce se zabývá vývojem aplikace pro zachycení a analýzu komunikace RFID v pásmu UHF ve standardu EPC Class-1 Generation-2. Jsou popsány základní vlastnosti systému RFID, principy platformy SDR a představen grafický programovací jazyk LabVIEW. V hlavní části se práce zabývá popisem vlastního vyvinutého softwaru, použitými postupy a principy. V další části je ukázáno grafické prostředí a funkčnost vytvořeného programu. Poslední částí práce jsou ukázky různých zachycených a analyzovaných dějů v UHF RFID komunikaci.

KLÍČOVÁ SLOVA

RFID, SDR, Ettus, USRP, LabVIEW

ABSTRACT

Main topic of this thesis is development of software for capture and analysis of RFID communication in UHF band (standard EPC Class-1 Generation-2). One part of the work is focused on the basics of the RFID systems, software defined radio concept and graphical programming language LabVIEW. Main part of this thesis is discussing the developed software itself, its methods and principles. Next part is dedicated to present the graphical user interface of created application and its functionality. Last part of this thesis contains examples of captured and analysed processes in UHF RFID communication.

KEYWORDS

RFID, SDR, Ettus, USRP, LabVIEW

VYCHODIL, Josef *Analyzátor UHF RFID komunikace založený na SDR*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav radioelektroniky, 2013. 57 s. Vedoucí práce byl Ing. Aleš Povalač

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Analyzátor UHF RFID komunikace založený na SDR“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Aleši Povalačovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

(podpis autora)

Výzkum realizovaný v rámci této diplomové práce byl finančně podpořen projektem
CZ.1.07/2.3.00/20.0007 **Wireless Communication Teams**
operačního programu **Vzdělávání pro konkurenceschopnost**.

Finanční podpora byla poskytnuta Evropským sociálním fondem
a státním rozpočtem České republiky.

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Technologie RFID, což je zkratka z anglického *Radio Frequency Identification – identifikace na rádiové frekvenci*, v posledních letech zažívá nebývalý rozvoj. Čím dál častěji se s touto technologií setkáváme např. v obchodech, kde slouží (prozatím pouze) k zabránění odcizení dražších předmětů, při sportech – jak na rekreační úrovni (bezdrátové systémy např. pro odbavení na lyžařských vlecích), tak na vrcholové úrovni (např. identifikace závodníků při běžeckých závodech s hromadným startem) nebo v dopravě a logistice při identifikaci vozidel. Není vyloučené, že systémy RFID v budoucích dobách zcela nahradí v současnosti hojně používané čárové kódy, které slouží k podobným účelům.

Cílem této diplomové práce je navrhnout koncepci pro odposlech UHF RFID komunikace v evropských oblastech a vytvořit software pro její analýzu. Nejprve bude tedy stručně popsán účel a princip RFID systémů, použité frekvence, antény, způsob komunikace a samotný hardware – čtečky a tagy. V další části bude vysvětlen princip platformy SDR neboli *softwarově definovaného rádia*, na kterém je práce založena. V jednoduchosti budou popsány základní principy a metody SDR. Další částí bude seznámení s přístrojem Ettus USRP-N200, který je použit k vlastnímu zachytávání komunikace. Následně bude část práce věnována představení grafického programovacího jazyka firmy National Instruments LabVIEW, ve kterém je vytvořený program realizován.

V hlavní části tohoto textu jsou pak dopodrobna popsány postupy a principy při zpracování signálu. To zahrnuje vše od zachycení komunikace pomocí přístroje USRP, demodulace amplitudově klíčovaného signálu, detekce hran přes dekódování jednotlivých PIE kódovaných symbolů až po převedení jednotlivých příkazů do čitelné podoby.

V další kapitole se text věnuje vlastní stvořené aplikaci, zejména uživatelskému rozhraní a implementovaným funkcím. Jsou popsány všechny prvky GUI a podrobně okomentovány všechny dostupné panely.

Poslední část práce ukazuje některé případy UHF RFID komunikace zachycené a analyzované pomocí programu realizovaného v rámci této diplomové práce.

1 RFID

Technologie RFID (zkratka anglického *Radio Frequency Identification*) slouží k bezdrátové identifikaci např. zboží, motorových nebo kolejových vozidel, lidí apod. Její historie sahá do 40. let minulého století, kdy byl patentován čárový kód, který slouží k podobným účelům. Jednou z hlavních korporací, stojících za vývojem RFID je, stejně jako u čárových kódů, společnost WalMart.

Systém RFID předpokládá co nejlevnější a nejjednodušší transpondér (tag), který může být např. na každém jednotlivém kusu prodávaného zboží a složitější zařízení – čtečku, která slouží ke komunikaci s tagem. Některé vyrobené RFID tagy jsou na obr. č. 1.1. Aplikace mohou být např.

- identifikace zboží
- kontrola pohybu osob
- turnikety lyžařských vleků
- systémy mýtného
- a další ...

RFID systémy můžeme rozdělit podle pásma, ve kterém pracují, viz tabulka 1.1.

Pásmo	Frekvenční rozsah	Dosah	Rychlost přenosu
LF	120–150 kHz	10 cm	Nízká
HF	13,56 MHz	1 m	Nízká až střední
UHF	433 MHz	1-100 m	Střední
UHF (Evropa)	865-868 MHz	1-2 m	Střední až vysoká
UHF (Severní Amerika)	902-928 MHz	1-2 m	Střední až vysoká
SHF	2 450-5 800 MHz	1-2 m	Vysoká
SHF	3,1–10 GHz	200 m	Vysoká

Tab. 1.1: Pásma používaná pro RFID.

1.1 Tagy

Tagy lze v zásadě rozdělit na

- aktivní – Tag obsahuje vlastní zdroj energie (baterii), může nezávisle vysílat. Umožňuje spolehlivější komunikaci zejména v nepříznivých podmínkách.
- pasivní – Tag získává veškerou energii pouze z rádiových vln vysílaných čtečkou. Od toho se odvíjí i způsob komunikace. Namísto klasického způsobu rádiové komunikace, tedy použití oscilátorů, směšovačů apod., se používá tzv.

Obr. 1.1: Ukázky RFID tagů. Převzato z [4].

backscattering – v případě, že se po tagu očekává odpověď, čtečka vysílá konstantní vlnu (CW) a tag ji pouze moduluje změnou stavu své antény (např. přizpůsobení/zkrat). Tyto dvě skutečnosti bohužel značně omezují dosah pasivních tagů.

Tato práce se zabývá RFID systémy s pasivními tagy pracující v pásmu UHF, zejména pak v evropské regulaci, tedy v pásmu 865-868 MHz.

1.2 Čtečky

RFID čtečky jsou složitější elektronické zařízení, které slouží ke čtení, organizaci, případně i k zápisu do tagů, vyskytujících se v poli čtečky. V případě protokolu EPC Class-1 Generation-2 UHF RFID, kterým se zabývá tato práce, vždy zahajuje a obstarává komunikaci čtečka. V dnešní době existuje mnoho výrobců čteček (např. METRA BLANSKO a.s., Motorola a další ...) jak stacionárních (viz obr. č. 1.2), tak mobilních (viz obr. č. 1.3).

1.3 Antény pro UHF RFID

Pro frekvence používané pro UHF RFID je délka vlny $\lambda \approx 33\ cm$. Délka vlny je tedy srovnatelná s fyzickým rozměrem antén. V úvahu tedy připadají klasické dipólové antény a plošné antény (flíčkové, atd.).

Obr. 1.2: Ukázka stacionární RFID čtečky. Převzato z [6].

Obr. 1.3: Ukázka mobilní RFID čtečky. Převzato z [7].

V nejběžnějším případě, tedy pokud předpokládáme stacionární čtečku a tag umístněný na sledovaném objektu (např. pásová výroba, samoobslužný obchod) dochází k různému natočení rovin vysílacích (čtečka) a přijímacích (tag) antén. Je tedy nevhodné používat soustavu dvou lineárně polarizovaných antén (např. dipólů) viz tabulka č. 1.2.

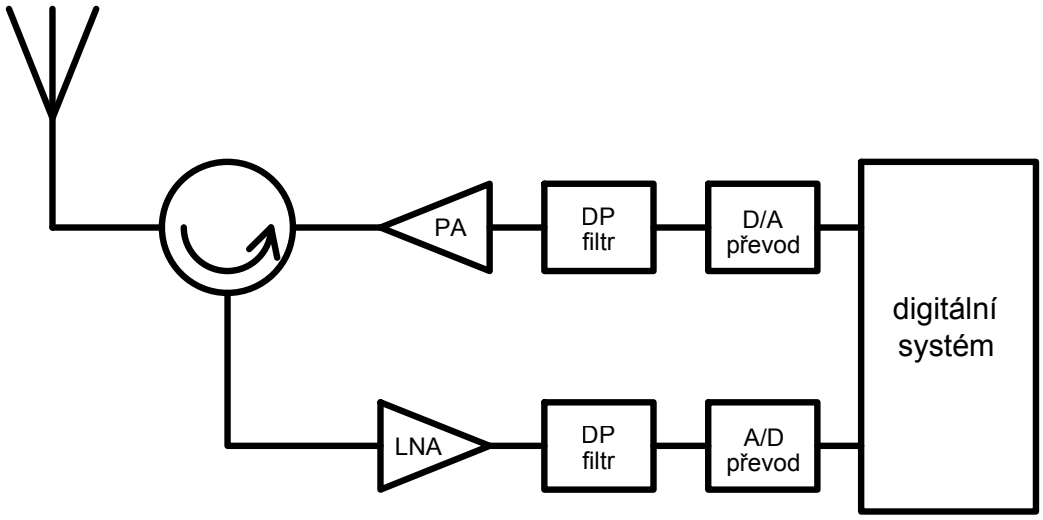
V praxi se používá kruhově polarizovaná (pravotočivá nebo levotočivá) anténa pro čtečku a jednoduchý dipól nebo jeho modifikace jako anténa tagu. Tím zajistíme, že velikost signálu přicházejícího od tagu bude nezávislá na rotaci tagu v rovině jeho dipólu, ovšem s tím, že úroveň bude o 3 dB menší, než v případě shodně natočených lineárních antén.

		Polarizace vysílací antény		
		Kruhová	Vertikální	Horizontální
Orientace přijímací antény	Vertikální	3 dB	0 dB	∞ dB
	Horizontální	3 dB	∞ dB	0 dB
	Šikmá	3 dB	3 dB	3 dB

Tab. 1.2: Útlum daný nepřízpusobením polarizace (při lineární přijímací anténě).

2 SOFTWAROVĚ DEFINOVANÉ RÁDIO

Softwarově definované rádio (SDR) je koncept moderního rádia, kde je většina (v případě ideálního SDR veškeré) zpracování signálu prováděno v digitální oblasti. Ideální softwarové rádio (obr. č. 2.1) obsahuje pouze analogové zesilovače a převodníky, zbylé části (modulátory, demodulátory, směšovače, filtry, detektory, atd . . .) jsou realizovány digitálně buďto v FPGA, DSP, případně jiném číslicovém systému (PC).

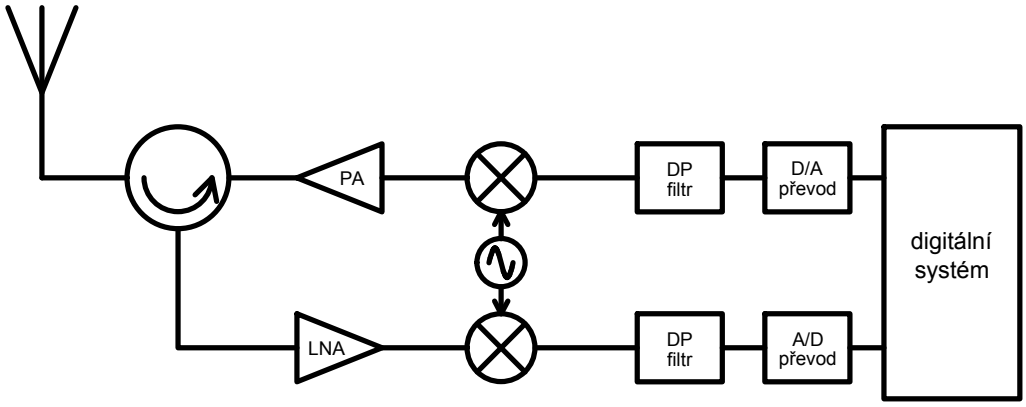


Obr. 2.1: Ideální softwarové rádio.

V dnešní době je tato koncepce zatím technicky problematicky realizovatelná – jsme omezeni rychlostí A/D a D/A převodníků, které v současné době dosahují vzorkovacích frekvencí maximálně stovky MHz, i obvody na zpracování signálu – je výpočetně náročné zpracovávat signál o vzorkovací frekvenci např. 1 GHz v reálném čase. Z těchto důvodu se používá digitální zpracování pouze v základním pásmu nebo na mezifrekvenci a dále se analogovým směšovačem směšuje do oblasti rádiových frekvencí viz obr. č. 2.2. Tímto (zejména pro úzkopásmové aplikace) zmenšíme nároky jak na A/D a D/A převodníky, tak na digitální obvody, které signál dále zpracovávají.

Výhody softwarového rádia jsou

- větší pružnost (možnost např. užití nových modulací nebo kódování pouhým stažením nové aplikace nebo firmwaru)
- větší integrace (z důvodu použití moderních CMOS obvodů)



Obr. 2.2: Softwarově definované rádio se směšováním do základního pásma.

- možnost adaptivní volby frekvence (díky přeladitelným směšovačům z/do základního pásma)
- nižší cena
- a další ...

2.1 Ettus USRP-N200

USRP neboli *Universal Software Radio Peripheral* je zařízení určené pro vývoj a testování SDR aplikací, případně pro jiné účely, které je koncipováno jako periferie k PC. Zařízení USRP-N200 je produkt firmy Ettus. Jedná se o modulární zařízení – základní deska obsahuje obvody pro práci v základním pásmu (FPGA, A/D a D/A převodníky apod ...) a k ní se připojují desky směšující signál do radiofrekvenčních pásem (vysílací, přijímací, případně kombinované). Základní parametry Ettus USRP-N200 jsou

- možnosti zpracování signálu od DC do 6 GHz
- dvojkanálový 100 MS/s 14-bitový A/D převodník
- dvojkanálový 400 MS/s 16-bitový D/A převodník
- DDC/DDU (přímá konverze nahoru/dolů) s rozlišením 25 mHz
- streamování přes Gigabit Ethernet do PC do rychlosti 50 MS/s
- 2 Gbps rozhraní pro rozšíření
- FPGA Xilinx Spartan 3A-DSP 1800
- 1 MB vysokorychlostní RAM
- 2,5 ppm TCXO frekvenční reference

Na výběr je velké množství rozšiřujících RF desek viz tabulka č. 2.1.

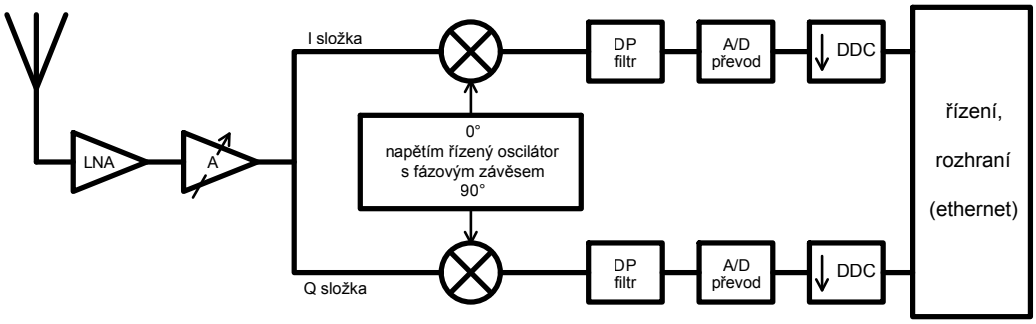
Označení	Pásmo [MHz]	Rx/Tx
BasicTX	1-250	Tx
BasicRX	1-250	Rx
LFTX	0-30	Tx
LFRX	0-30	Rx
TVRX2	50-860	2×Rx
DBSRX2	800-2 350	Rx
WBX	50-2 200	Rx/Tx
SBX	400-4 400	Rx/Tx
XCVR2450	2 400-2 500, 4 900-5 900	Rx/Tx
RFX900	750-1 050	Rx/Tx
RFX1200	1 150-1 450	Rx/Tx
RFX1800	1 500-2 100	Rx/Tx
RFX2400	2 300-2 900	Rx/Tx

Tab. 2.1: Rozšiřující desky pro Ettus USRP-N200.

V laboratoři školy jsou přístroje Ettus USRP-N200 vybaveny rozšiřujícími deskami WBX, umožňují tedy příjem i vysílání v rozsahu 50-2 200 MHz.

Na obrázku č. 2.3 lze spatřit zjednodušené schéma přijímací části Ettus USRP-N200. Princip příjmu je následující: signál přicházející z anténního konektoru je nejprve zesílen v nízkošumovém LNA zesilovači, poté následuje další zesilovací stupeň s proměnným zesílením. Takto upravený signál je směšován s harmonickým signálem s fází 0° (cosinus) a 90° (sinus) z laditelného oscilátoru. Tím získáme I (in-phase, soufázovou) a Q (quadrature, kvadraturní) složku signálu. Tyty složky jsou pak každá zvlášť vyfiltrovány a navzorkovány rychlostí (vždy) 100 MS/s. Dále se pak dle potřeby data zpracovávají v FPGA. Zejména se signál přeloží do základního pásma pomocí DDC, neboli *direct downconversion*, vyfiltruje se požadovaná šířka pásma a zdecimuje se na požadovanou vzorkovací rychlost. Zdigitalizovaná data se posléze streamují přes Gigabitový Ethernet do hostitelského počítače.

Je tedy patrné, že konstrukce přijímače dovoluje jak homodynní (v případě, že lokální oscilátor naladíme přímo na frekvenci vysokofrekvenční nosné vlny), tak superheterodynní příjem (pokud lokální oscilátor naladíme na frekvenci odlišnou od frekvence vf vlny).



Obr. 2.3: Zjednodušené blokové schéma přístroje Ettus USRP-N200 s rozšiřující deskou WBX (pouze přijímací část).

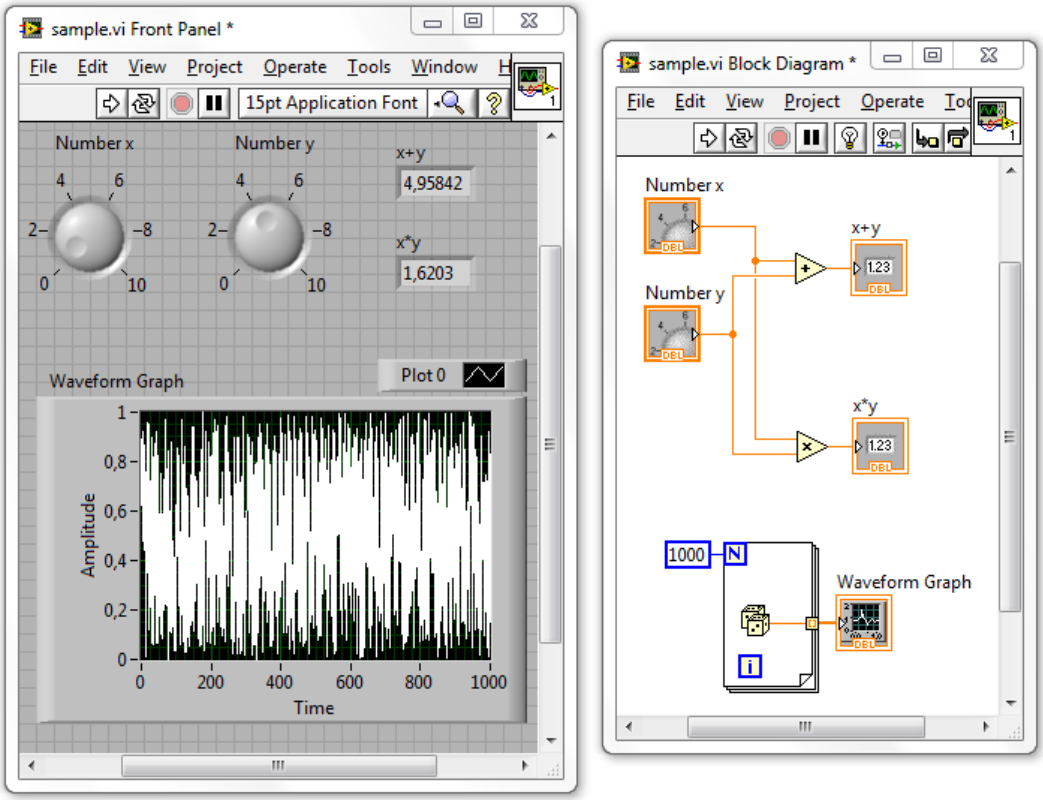
2.2 UHD

Jako rozhraní pro komunikaci se zařízením USRP se používají ovladače UHD. Zkratka UHD znamená *Universal software radio peripheral Hardware Driver*, jedná se tedy o ovladač a zároveň i API pro současné i budoucí produkty firmy Ettus. Lze ho použít přímo (např. ve Visual C++, Pythonu) nebo s aplikacemi třetích stran – GNU Radio, LabVIEW, Simulink a dalšími. Ovladače UHD jsou dostupné pro všechny nejrozšířenější platformy (Windows, Linux, Mac). Obsaženy jsou veškeré funkce jak pro komunikaci a přenos signálu, tak i pro ladění kanálů apod.

3 LABVIEW

LabVIEW je moderní grafický programovací jazyk vyvíjený firmou National Instruments. Jeho historie sahá do roku 1986, kdy byla vydána první verze určená pro Apple Macintosh. LabVIEW se používá především pro záznam/zpracování dat, ovládání přístrojů, automatizaci v průmyslu, různá měření apod.

Základní princip programování spočívá v tvoření tzv. „virtuálních instrumentů“. Analogie se skutečnými přístroji je zřejmá – stejně tak jako skutečný přístroj obsahuje i virtuální instrument vytvořený v prostředí LabVIEW přední panel (front panel) a určité zapojení obvodů – program (block diagram). Jednoduchý příklad programu, stvořený během několika minut, je na obrázku č. 3.1.



Obr. 3.1: Ukázka jednoduchého virtuálního instrumentu v LabVIEW.

Tento jednoduchý program nedělá nic jiného, než že sečte a vynásobí dvě vstupní čísla, zobrazí výsledek a zároveň vygeneruje pole 1 000 náhodných čísel a zobrazí je v grafu. To je jedna z výhod prostředí LabVIEW – poměrně jednoduchá implementace paralelního programování. Díky tomu můžeme velmi efektivně např. přijímat

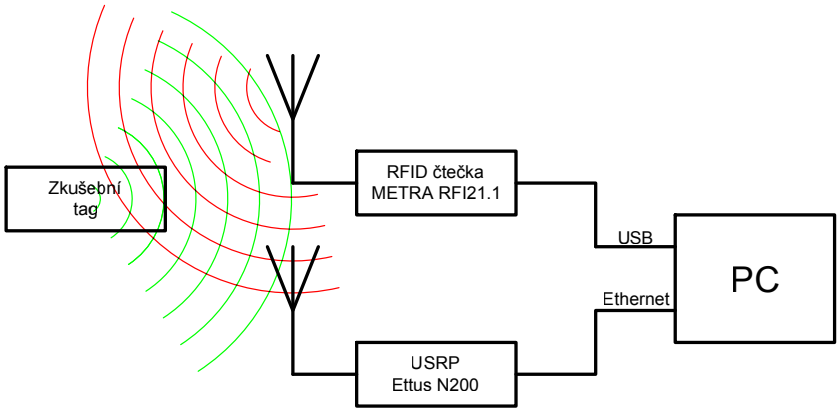
(v reálném čase) blok dat a paralelně s příjmem zpracovávat jiný blok dat, přijatý v předchozím cyklu. Další z mnoha výhod oproti tradičním programovacím jazykům (např. Visual C++) jsou např. velmi jednoduché zobrazení dat (viz graf v příkladu 3.1), jednoduché tvoření GUI, usnadněný přístup k zařízením určeným k záznamu/zpracování/odesílání dat (např. příjem/odesílání dat pomocí výše uvedeného přístroje Ettus USRP-N200 lze realizovat pomocí několika málo „krabiček“, jak se nesprávně přezdívá subVI – subVirtuálním instrumentům, které jsou obdobou funkcí klasických programovacích jazyků). Především z těchto důvodů je vlastní analyzátor realizován v prostředí LabVIEW.

4 PRINCIP ZACHYCENÍ A ANALÝZY UHF RFID KOMUNIKACE

Tato část textu se zabývá popisem použitých postupů a principů využitých ve vlastním vytvořeném programu.

4.1 Zachycení komunikace

Pro odposlech UHF RFID komunikace mezi čtečkou a tagem (případně tagy) se předpokládá prostorové uspořádání přístrojů a tagů dle obrázku č. 4.1.



Obr. 4.1: Uspořádání pracoviště pro odposlech UHF RFID komunikace.

Jako testovací UHF RFID čtečka je použita čtečka firmy METRA RFI21.1. Umožňuje provoz jak v oblastech spadající pod evropskou (865 až 868 MHz), tak i americkou (902 až 928 MHz) regulaci, různé nastavení komunikační rychlosti (Tari), použití vhodných modulací (DSB-ASK, PR-ASK) apod. Disponuje rozhraními USB 2.0, UART a GPIO [3]. Pro připojení k PC je použito nejvhodnější rozhraní, tedy USB.

Pro vlastní zachycení komunikace je použito zařízení Ettus USRP-N200. Více podrobností o tomto zařízení obsahuje kapitola 2.1. Z důvodů uvedených v kapitole 1.3 mají antény použité jak pro čtečku (vysílání/příjem) tak pro USRP (pouze příjem) kruhovou polarizaci. Antény tagu jsou lineárně polarizované.

Pro ukázku byly přístroje nastaveny dle následujících parametrů. Nastavení čtečky:

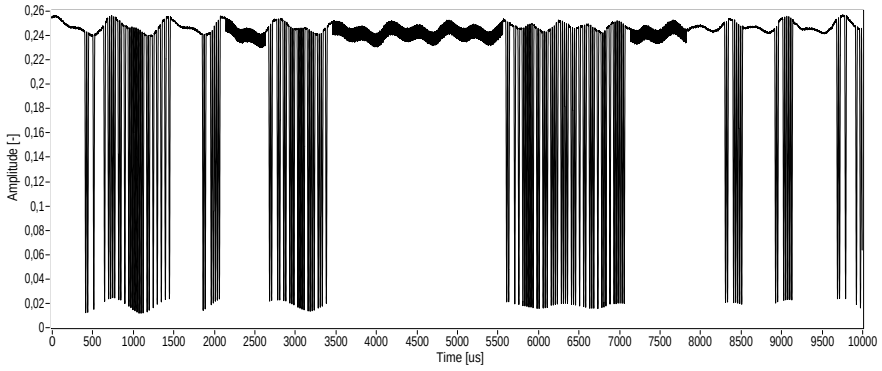
- protokol: EPC Class 1 Gen 2 [2]

- nosná frekvence: 865,7 MHz
- modulace: DSB-ASK
- Tari: 25 μs

Nastavení Ettus USRP-N200:

- nosná frekvence: 865,7 MHz
- výstupní vzorkovací rychlost (IQ rate): 4 MS/s
- kvantování: 16-bitové

Ukázka komunikace zachycené tímto způsobem je na obrázku č. 4.2.

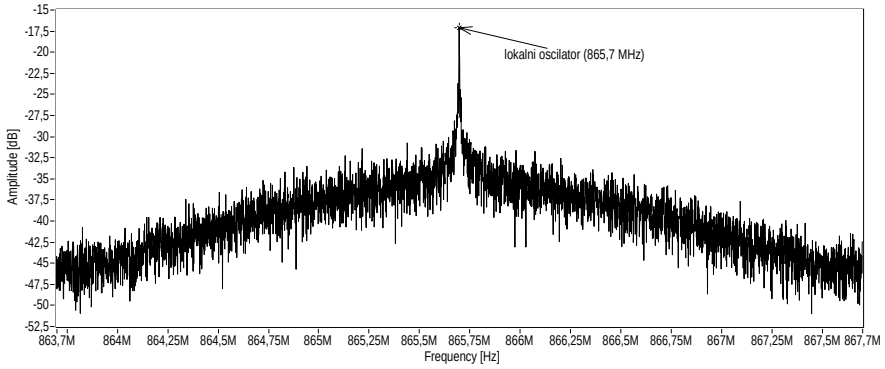


Obr. 4.2: Ukázka zachycené UHF RFID komunikace (časový průběh v absolutní hodnotě ze složek I a Q).

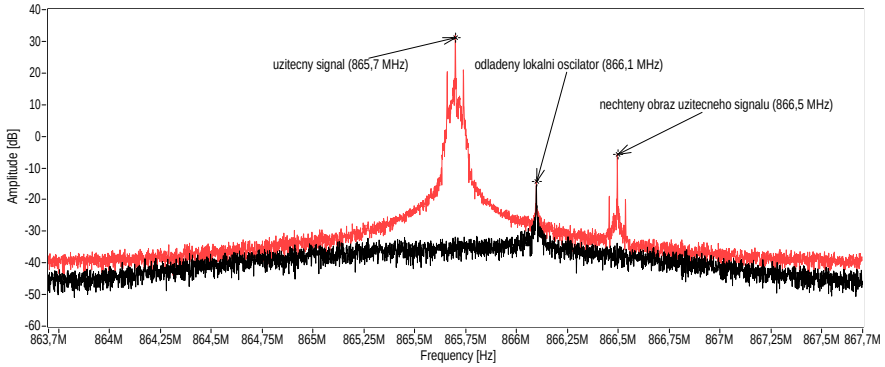
Na tomto příkladu lze vidět jak vysílání čtečky (s velkou amlitudou) i odpovědi tagu (s malou amplitudou). Toho je však docíleno ideálním uspořádáním měřicího pracoviště. Ve skutečnosti je zachycení odpovědi tagu velmi problematické (kvůli způsobu komunikace, tzv. backscatteringu, viz kapitola 1.1). Z tohoto důvodu se tato práce a výsledný program zaobírá pouze dekódováním a analýzou příkazů čtečky. Dále si lze povšimnout, že signál je lehce zarušen. To je způsobeno nastavením a konstrukcí USRP přijímače (viz kapitola 2.1). V implicitním nastavení se přístroj Ettus USRP-N200 chová jako standardní homodynní přijímač, tzn. frekvence lokálního oscilátoru se nastaví na požadovanou nosnou frekvenci a směšuje se přímo do základního pásma, kde dochází k navzorkování a dalšímu zpracování signálu.

Na obrázku č. 4.3 je zachyceno frekvenční spektrum přijímaného signálu při odpojené anténě, jedná se tedy o vlastní šum přijímače. Nastavení přijímače je stejné jako u předchozího příkladu, tedy nosná frekvence: 865,7 MHz a výstupní vzorkovací rychlost: 4 MS/s. Lze vidět, že do užitečného signálu prostupuje i signál lokálního oscilátoru, který leží v oblasti stejnosměrné složky.

Konstrukce přijímače (viz kapitola 2.1) umožňuje superheterodynní příjem, tedy nejprve přeložení (směšování) signálu do mezifrekvence a posléze (digitální) směšování do základního pásma. Na obrázku č. 4.4 je pro ilustraci zobrazena tato situace



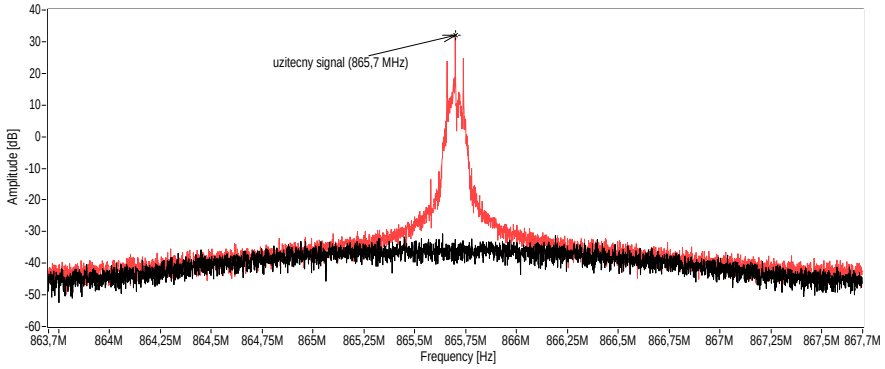
Obr. 4.3: Vlastní šum USRP přijímače při implicitním nastavení. Lokální oscilátor osciluje na frekvenci vysokofrekvenční nosné, je tedy ve středu pásma našeho zájmu.



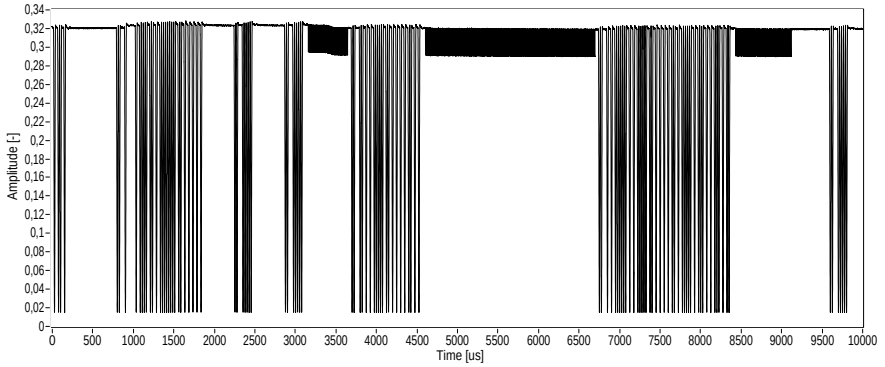
Obr. 4.4: Vlastní šum USRP přijímače (černý průběh) při lokálním oscilátoru kmitajícím na frekvenci o 0,4 MHz vyšší než vf nosná. Červený průběh zobrazuje příjem vysílání RFID čtečky se stejnými parametry USRP přístroje.

při odladění lokálního oscilátoru o 0,4 MHz nad vysokofrekvenční nosnou. Černý průběh ukazuje vlastní šum přijímače při tomto nastavení – lze pozorovat pouze lokální oscilátor na frekvenci 866,1 MHz. Při příjmu signálu (červený průběh) ovšem nastává nechtěný jev – dochází k vytvoření obrazu užitečného signálu na frekvenci $f_{obraz} = f_{LO} + (f_{LO} - f_{VF}) = 866,1 + (866,1 - 865,7) = 866,5$ MHz. Při takto odladěném lokálním oscilátoru, kdy nedochází k překrytí spekter užitečného signálu a nechtěného obrazu, dochází pouze ke zvýšení šumu. Pokud je ovšem lokální oscilátor naladěn blízko vf nosné (jako v příkladu č. 4.2 a 4.3), dochází k překrývání spekter a tím k zarušení a znehodnocení užitečného signálu, jak lze pozorovat na časovém průběhu grafu č. 4.2.

K eliminaci těchto jevů je potřeba odladit lokální oscilátor dostatečně daleko od vf nosné. Vzhledem ke konstrukci a způsobu zpracování signálu v USRP přijí-

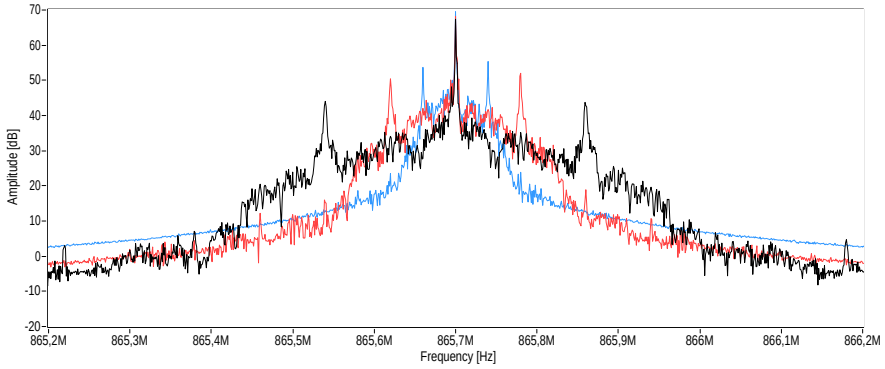


Obr. 4.5: Vlastní šum USRP přijímače (černý průběh) při lokálním oscilátoru kmitajícím na frekvenci o 20 MHz nižší než vf nosná. Červený průběh zobrazuje příjem vysílání RFID čtečky se stejnými parametry USRP přístroje.

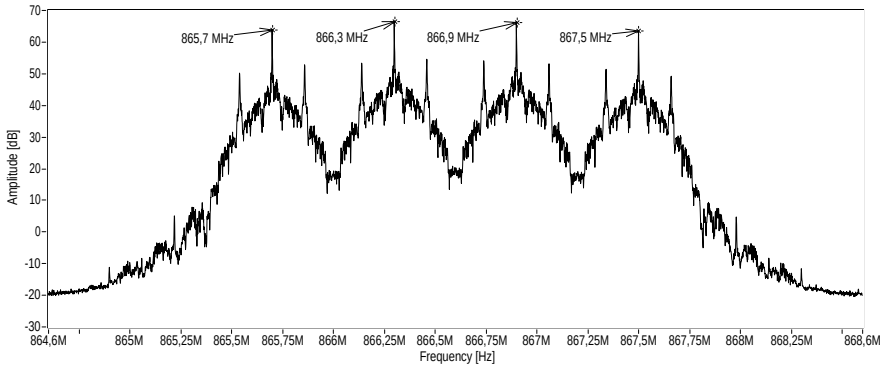


Obr. 4.6: Ukázka zachycené UHF RFID komunikace (časový průběh v absolutní hodnotě ze složek I a Q) při lokálním oscilátoru odladěném na frekvenci o 20 MHz nižší, než frekvence vf nosné vlny.

mači (přijímaný signál se vždy vzorkuje rychlostí 100 MS/s a teprve poté se v FPGA dále zpracovává na požadovanou vzorkovací rychlost, viz kapitola 2.1) je možné lokální oscilátor odladit tak, že ani signál lokálního oscilátoru ani nechtěné obrazy nezasahují do pásma našeho zájmu. Pro ilustraci je na grafu č. 4.5 vyobrazeno spektrum vlastního šumu přijímače (černě) i spektrum při příjmu vysílání RFID čtečky (červeně) při lokálním oscilátoru naladěném na frekvenci o 20 MHz nižší než frekvence vf nosné vlny. Časový průběh signálu čtečky i odpovědi tagu (tedy stejné situace jako na obrázku č. 4.2) zachycené tímto způsobem je vyobrazen na obrázku č. 4.6



Obr. 4.7: Frekvenční spektrum signálu UHF RFID čteček s různou komunikační rychlostí Tari (černě – 6,25 μs, červeně – 12,5 μs a modře – 25 μs).



Obr. 4.8: Frekvenční spektrum pásma vyhrazeného pro evropský UHF RFID provoz při současném vysílání čtyř čteček s komunikační rychlostí Tari=6,25 μs.

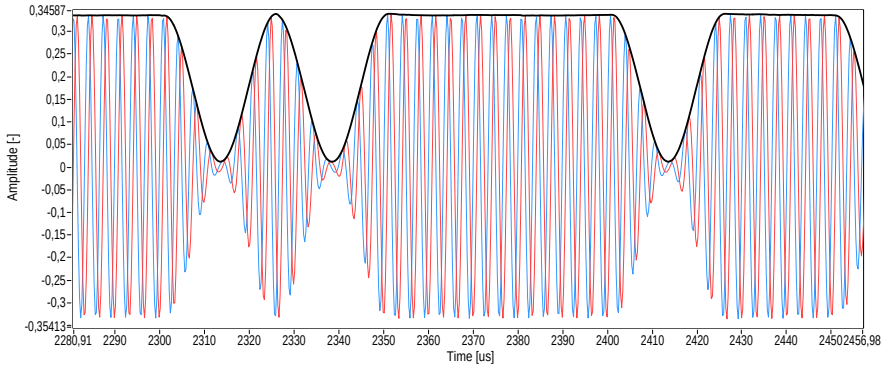
4.1.1 Volba vzorkovací frekvence

Standard EPC Class 1 Gen 2 umožňuje různé komunikační rychlosti čtečky, které jsou definovány pomocí časové jednotky Tari. Tari může nabývat libovolných hodnot v rozmezí 6,25 až 25 μs. Od této rychlosti se odvíjí šířka pásma zabíraná RFID čtečkou – se zvyšující se komunikační rychlostí (s klesající časovou konstantou Tari) se šířka pásma zvětšuje, jak lze vidět na obrázku č. 4.7. Dle standardu je šířka kanálu definovaná na 500 kHz, teoreticky tedy stačí vzorkovat rychlostí 500 kS/s, při vyšších hodnotách Tari i méně. V praxi je potřeba počítat s určitou rezervou a vzorkovat s větší rychlostí. Evropská regulace (ETSI) definuje čtyři frekvence, na kterých mohou UHF RFID čtečky vysílat, a to 865,7 MHz, 866,3 MHz, 866,9 MHz a 867,5 MHz. Při započtení šířky pásma jednoho kanálu 500 kHz je tedy pásmo vyhrazené pro UHF RFID provoz široké $867,5 - 865,7 + 0,5 = 2,3$ MHz. Pokud tedy budeme vzorkovat signál rychlostí 4 MS/s, obsáhneme i s rezervou celé poža-

dované pásmo. Nejhorší případ, tedy vysílání čtyř čteček s komunikační rychlostí Tari=6,25 μs na všech čtyřech kanálech, je vyobrazen na obrázku č. 4.8. Při zachytávání těchto dat bylo zařízení Ettus USRP-N200 nastaveno následovně: nosná frekvence: 866,6 MHz (střed evropského UHF RFID pásma), frekvence lokálního oscilátoru: 846,6 MHz (odladění o -20 MHz) a vzorkovací rychlost 4 MS/s. Lze vidět, že toto nastavení umožňuje univerzální příjem veškerého provozu v evropském UHF RFID pásmu na kterémkoliv kanále s libovolnou přenosovou rychlostí. Přestože program analyzuje a dekóduje příkazy pouze jedné čtečky (v reálném čase), je toto nastavení výhodné, jelikož čtečka může pracovat např. v režimu frequency hopping, kdy dochází k rychlé změně kanálů během vysílání.

4.2 Demodulace ASK signálu

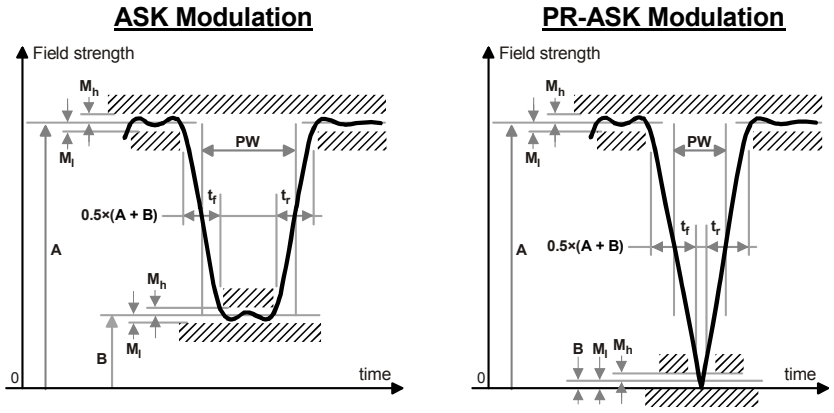
Standard EPC Class 1 Gen 2 používá pro modulaci signálu vysílaného čtečkou jednoduchých amplitudových modulací, a to: DSB-ASK (*Double-SideBand Amplitude Shift Keying – amplitudová modulace s dvěma postranními pásmy*), SSB-ASK (*Single-SideBand Amplitude Shift Keying – amplitudová modulace s jedním postranním pásmem*) nebo PR-ASK (*Phase Reversal Amplitude Shift Keying – amplitudová modulace s otáčením fáze*).



Obr. 4.9: Demodulace ASK klíčovaného signálu. Červený průběh představuje I složku, modrý Q složku signálu, černě vyznačený průběh je obálka signálu, nebo-li demodulovaný signál vypočítaný jako $s = \sqrt{I^2 + Q^2}$.

V analogové technice lze pro demodulaci amplitudově modulovaného signálu použít např. primitivního diodového detektoru, skládajícího se pouze ze dvou součástek – diody a kondenzátoru. V případě digitálního zpracování signálu v podobě IQ dat je základní řešení taktéž jednoduché – aktuální amplituda přijímaného signálu se spočítá jako absolutní velikost komplexního (IQ) vzorku, tedy $s = \sqrt{I^2 + Q^2}$. Ukázka takto zpracovávaného signálu je na obrázku č. 4.9.

4.3 Detekce hran

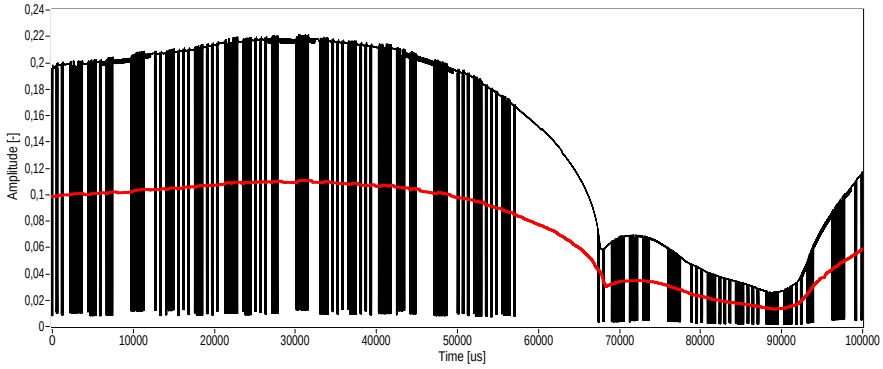


Obr. 4.10: Definice RF pulsů vysílaných čtečkou. Převzato z [2].

Parametr	Symbol	Minimum	Maximum	Jednotka
Hloubka modulace	(A–B)/A	80	100	%
Zvlnění RF obálky	$M_h = M_l$	0	0,05(A–B)	V/m, A/m
Sest. hrana RF obálky	$t_{r,10-90\%}$	0	0,33Tari	μs
Vzes. hrana RF obálky	$t_{f,10-90\%}$	0	0,33Tari	μs
Šířka RF pulsu	PW	Max(0,265Tari, 2)	0,525Tari	μs

Tab. 4.1: Význam veličin pro 4.10. Převzato z [2].

Pro další zpracování je nyní potřeba v amplitudově klíčovaném demodulovaném signálu určit hrany. Hlavním problémem při detekci hran je stanovení vhodného rozhodovacího prahu. Při příjmu se může amplituda signálu vlivem různých okolností (zejména při pohybu čtečky či přijímače, případně při nevhodné manipulaci s anténami) nečekaně měnit a práh je potřeba tomuto okamžitě uzpůsobovat. Na obrázku č. 4.10 a v tabulce č. 4.1 lze spatřit poměrně striktní definici vf pulsů vysílaných čtečkou. Rovněž je zde naznačeno určení prahu – vcelku triviálně jako průměr mezi maximální a minimální hodnotou. Také je zadefinována velikost hloubky modulace, která se musí pohybovat v rozmezí 80-100%. Toho lze využít pro zjednodušení určování prahu pro detekci hran – namísto sledování maxim a minim v přijímaném signálu stačí vyhledávat maxima, minimum lze považovat vždy za nulové, chyba není ani při hloubce modulace 80% nijak vysoká. Výsledný práh tedy bude polovina maximální hodnoty signálu.



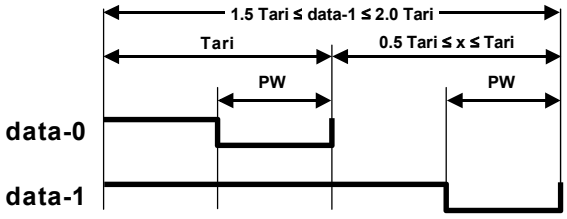
Obr. 4.11: Ukázka určování prahu pro detekci hran. Černý průběh značí zpracovávaný signál, červeně je naznačen určený práh.

Jako řešení se nabízí sledovat maximální hodnotu signálu za posledních několik uplynulých mikrosekund. Vzhledem k povaze signálu lze považovat za rozumnou hodnotu cca 500 μs. To by ovšem znamenalo při vzorkovací rychlosti 4 MS/s nutnost určovat každých $\frac{1}{4\,000\,000} = 0,25\mu s$ maximální hodnotu z $500\mu \cdot 4M = 2\,000$ vzorků, což je výpočetně velmi náročné a stěží se dá realizovat na běžném osobním počítači v reálném čase. Pro zmenšení výpočetních nároku lze například odebírat vzorky každých 25 μs a určovat maximum z posledních dvaceti takto získaných vzorků. Vzhledem k repetitivnímu charakteru signálu není tato metoda nejvhodnější, neboť by mohlo docházet k tomu, že budeme odebírat vzorky takovým způsobem, že neobdržíme skutečné maximum. Proto je potřeba, aby vzdálenost mezi odebíranými vzorky měla náhodný charakter. Praktická realizace v programu používá uniformní rozložení v rozmezí 0 až 50 μs (střední hodnota je tedy 25 μs). Zásobník obsahuje 20 vzorků, statisticky je tedy výsledná hodnota maxima rovna maximu za posledních $20 \cdot 25 = 500$ μs. Ukázka prahu určeného touto metodou je na obrázku č. 4.11.

Nyní stačí již pouze detekovat průchod signálu přes tento práh a podle toho, zda byl signál před průchodem větší (respektive menší) než daný práh určit, zda je v původním signálu sestupná či vzestupná hrana. Při detekci hrany se uloží časový okamžik se znaménkem značícím typ hrany (kladná reálná čísla – vzestupné hrany, záporná reálná čísla – sestupné hrany) do pole pro další zpracování. Ve velmi ojedinělých případech může nastat situace, kdy tag vysílá odpověď, ale čtečka ji nezachytí a začne vysílat. V takovém případě dojde k superpozici těchto dvou signálů a může se stát, že vznikne zákmit přímo na rozhodovací úrovni a k chybné detekci hrany. Řešením je definovat minimální vzdálenost mezi hranami na např. 1,5μs, pak je algoritmus imunní vůči tomuto jevu.

4.4 Dekódování PIE

Data vyslaná čtečkou jsou kódována pomocí PIE neboli *pulzně intervalového kódování* – to znamená, že jednotlivé symboly (binární – „0“ a „1“) jsou zakódovány v délce trvání horní úrovně, po které následuje nízká úroveň konstantní délky, viz obrázek č. 4.12. Referenční časová délka (zmiňovaná již v kapitole 4.1) se nazývá Tari a je rovna délce symbolu „0“. Může nabývat libovolných hodnot v rozmezí 6,25 až 25 μs. Tento způsob kódování, kdy je po většinu času vysílána vysoká úroveň vf vlny, zajistí, že tag bude mít vždy dostatek energie pro vlastní obvody.



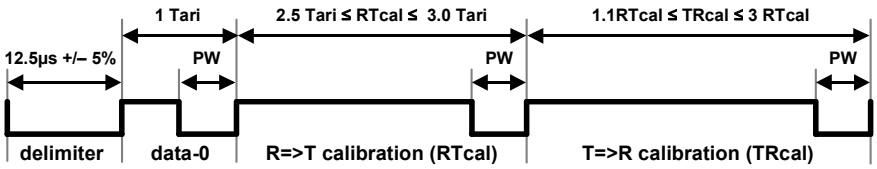
Obr. 4.12: Kódování PIE. Převzato z [2].

Samotnému příkazu vždy přechází tzv. preambule nebo frame-sync. Preambule je zobrazena na obr. č. 4.13. Obsahuje několik částí:

- oddělovač (delimiter) pevné délky 12,5 μs ±5 %
- symbol „0“
- symbol kalibrace Reader=>Tag RTCal: jeho délka je dána délkou symbolu „0“ a „1“. Pomocí něj se spočítá tzv. pivot jako polovina délky symbolu RTCal. Symboly kratší než pivot jsou poté vyhodnoceny jako „0“ a symboly delší než pivot jako „1“. Symboly delší než čtyřnásobek symbolu RTCal by měly být vyhodnoceny jako neplatné.
- symbol kalibrace Tag=>Reader TRCal: čtečka definuje modulační rychlost tagu (*Backscatter Link Frequency*, BLF) pomocí TRCal a DR podle vztahu $BLF = \frac{DR}{TRCal}$, kde DR (*Divide Ratio*) může nabývat hodnot 8 nebo 64/3 (určeno bitem v příkazu **Query**) a TRCal musí být v rozsahu 1,1·RTCal až 3·RTCal.

Preambule předchází pouze příkazu **Query**, před ostatními příkazy se vysílá frame-sync, který je totožný s preambulí s tím rozdílem, že neobsahuje symbol TRCal.

Pro rychlou detekci a zpracování takto kódovaných dat (které jsou nyní ve formě časových okamžiků vzestupných a sestupných hran viz kapitola č. 4.3) je nejvhodnější použít stavový automat, který bude zpracovávat jednotlivé hrany okamžitě po jejich přijetí, případně ihned dekódovat bity příkazů. Zjednodušený vývojový



Obr. 4.13: Preambule vysílání. Převzato z [2].

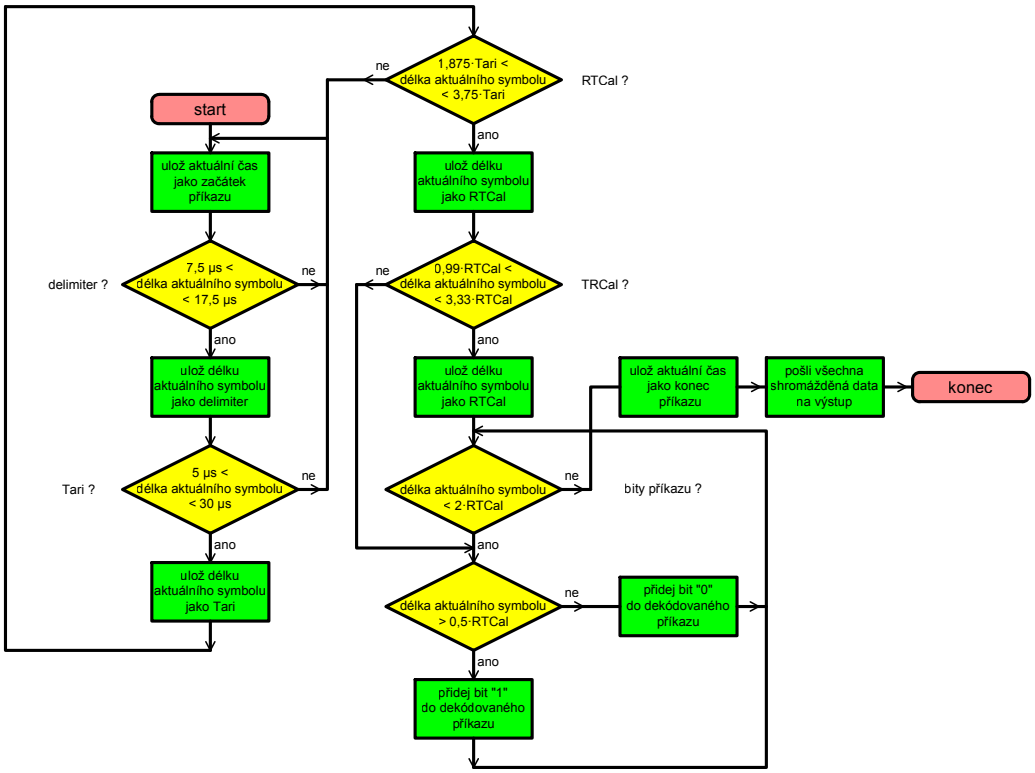
diagram je vyobrazen na obrázku č. 4.14. Z principu činnosti tohoto podprogramu vyplývá, že kromě dekódování PIE symbolů provádí i segmentaci příkazů – při detekci symbolu delimiteru označí začátek příkazu a po dekódování jednotlivých bitů označí konec. Veškeré zjištěné informace (tedy délky symbolů delimiter, RTCal, TRCal a jednotlivé dekódované bity) jsou poté uloženy a poslány na další zpracování.

4.5 Dekódování jednotlivých příkazů

Příkaz	Kód	Délka (bity)	Povinný?	Ochrana
QueryRep	00	4	Ano	Unikátní délka příkazu
ACK	01	18	Ano	Unikátní délka příkazu
Query	1000	22	Ano	Unikátní délka a CRC-5
QueryAdjust	1001	9	Ano	Unikátní délka příkazu
Select	1010	> 44	Ano	CRC-16
Vyhrazeno	1011	–	–	–
NAK	11000000	8	Ano	Unikátní délka příkazu
Req_RN	11000001	40	Ano	CRC-16
Read	11000010	> 57	Ano	CRC-16
Write	11000011	> 58	Ano	CRC-16
Kill	11000100	59	Ano	CRC-16
Lock	11000101	60	Ano	CRC-16
Access	11000110	56	Ne	CRC-16
BlockWrite	11000111	> 57	Ne	CRC-16
BlockErase	11001000	> 57	Ne	CRC-16
BlockPermalock	11001001	> 66	Ne	CRC-16

Tab. 4.2: Příkazy čtečky. [2].

Posledním bodem základní analýzy příkazů čtečky je převést nasbíraná data do podoby, která je lidsky čitelná. Znamená to tedy převést binární vyjádření příkazů



Obr. 4.14: Zjednodušený vývojový diagram rutiny pro dekódování PIE symbolů a segmentaci příkazů.

na srozumitelný textový formát. V tabulce č. 4.2 je vypsán přehled všech příkazů, které jsou definované normou EPC Class 1 Gen 2. Norma definuje jedenáct příkazů, které musí každý vyrobený tag zvládnout zpracovat, jsou to: **Select**, **Query**, **QueryAdjust**, **QueryRep**, **ACK**, **NAK**, **Req_RN**, **Read**, **Write**, **Kill** a **Lock**, které slouží k základní organizaci, manipulaci, čtení, zápisu, případně uzamčení a zničení tagů. Nepovinné přikazy **Access**, **BlockWrite**, **BlockErase** a **BlockPermalock** slouží k čtení a zápisu větších bloků dat, případně k pokročilé manipulaci s pamětí tagu (uzamykání apod.).

Každý příkaz má svůj unikátní kód, dlouhý 2–8 bitů, který ho jednoznačně identifikuje. Kromě definovaných příkazů jsou určité bitové kombinace vyhrazeny pro budoucí použití, případně pro vlastní příkazy, definované výrobcem. Za těmito bity následují bity, které mají význam parametrů příkazů – mohou jimi být např. data, která se mají zapsat, adresa, ze které se má číst apod. Tyto parametry jsou po-

	Příkaz	Session	Odpověď Tagu
Počet bitů	2	2	16
Popis	00	00: S0 01: S1 10: S2 11: S3	RN16

Tab. 4.3: Definice příkazu **QueryRep**. [2].

drobně pro každý příkaz popsány v [2]. Příklad takového popisu je v tabulce č. 4.3. Kromě názvu příkazu a parametrů je také zadefinováno, zda má na daný příkaz tag reagovat, případně jakým způsobem. Veškeré příkazy jsou také chráněny, aby nemohlo dojít k záměně – buďto jeho unikátní délkou, 5-ti bitovým nebo 16-ti bitovým cyklickým redundantním součtem, případně kombinací. Mimo to jsou kódy příkazů zvoleny tak, že žádný z příkazů s delším kódovým označením nezačíná jako některý z příkazů kratších.

Vlastní převod do čitelné podoby je realizován vyhledáváním binárního vyjádření příkazů v předdefinované tabulce, vyčtením řetězce názvu a očekávané odpovědi tagu a případnou extrakcí všech parametrů daného příkazu.

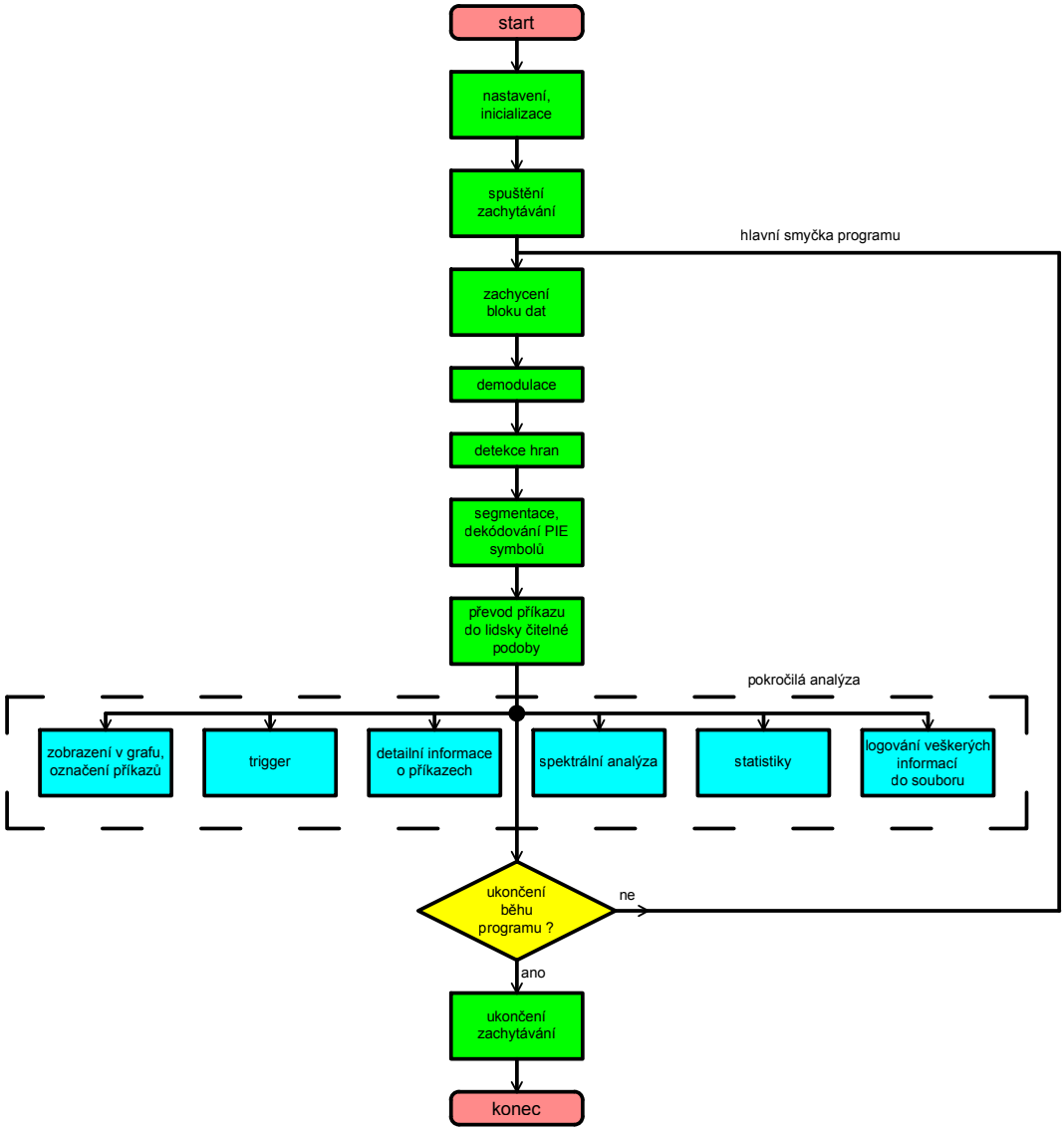
4.5.1 EBV

V parametrech příkazů určených pro manipulaci s pamětí tagu (čtení, zápis, zamykání, mazání atd.) se používá pro adresaci a délku masek formátu EBV-8. Tento formát umožňuje vyjádřit libovolně velké číslo následujícím způsobem: data jsou uspořádána v blocích, v případě EBV-8 o velikosti 8 bitů. První bit bloku (zleva, MSB) je tzv. rozšiřující bit – v případě, že je roven jedné, následuje další blok dat, pokud je roven nule, EBV po tomto bloku končí. Datová hodnota EBV se zjistí jednoduše čtením bitů zleva doprava s ignorováním rozšiřujících bitů. Hodnoty, které lze vyjádřit pomocí jednoblokového EBV-8, jsou tedy 0 až $2^7 - 1 = 127$. Příklady čísel uložených v EBV-8 formátu jsou v tabulce č. 4.4.

	0	0	0000000				
	1	0	0000001				
$2^7 - 1$	127	0	1111111				
2^7	128	1	0000001	0	0000000		
$2^{14} - 1$	16 383	1	1111111	0	1111111		
2^{14}	16 384	1	0000001	1	0000000	0	0000000

Tab. 4.4: Příklady čísel vyjádřených v EBV-8 formátu. [2].

5 ANALYZÁTOR UHF RFID KOMUNIKACE



Obr. 5.1: Schematické znázornění běhu programu.

V této části bude popsán samotný realizovaný software, který byl vytvořen s použitím postupů a principů uvedených v kapitole 4 za pomoci vývojového prostředí LabVIEW, které bylo představeno v kapitole 3. Výsledný program se podařilo re-

alizovat s takovou výpočetní efektivitou, že umožňuje zpracování a analýzu UHF RFID provozu v reálném čase s minimálním zpožděním.

Základní vývojový diagram běhu programu je zobrazen na obrázku č. 5.1. Probíhá tedy následovně: po spuštění programu je nejprve načtena konfigurace, proběhne inicializace proměnných a zařízení Ettus USRP-N200 (drobnými modifikacemi programu by měla být možná spolupráce s libovolným USRP zařízením). Poté je spuštěno samotné zachytávání. Od této chvíle je potřeba neustále zpracovávat přijímaná data s takovou rychlostí, aby nedošlo k přetečení zásobníku USRP přístroje. Následuje tedy hlavní smyčka programu, ve které dochází k samotné analýze přijímaného signálu.

Program tedy zachytí a následně demoduluje blok dat, ve kterém se ihned detekují hrany. V následující fázi se z těchto hran obnoví PIE symboly a dojde k segmentaci jednotlivých příkazů. Nyní již jsou k dispozici binární vyjádření jednotlivých příkazů, takže je lze pomocí předdefinované tabulky převést na textové vyjádření, které je srozumitelné pro uživatele – určí se, o který příkaz se jedná a dojde k popisu veškerých jeho parametrů.

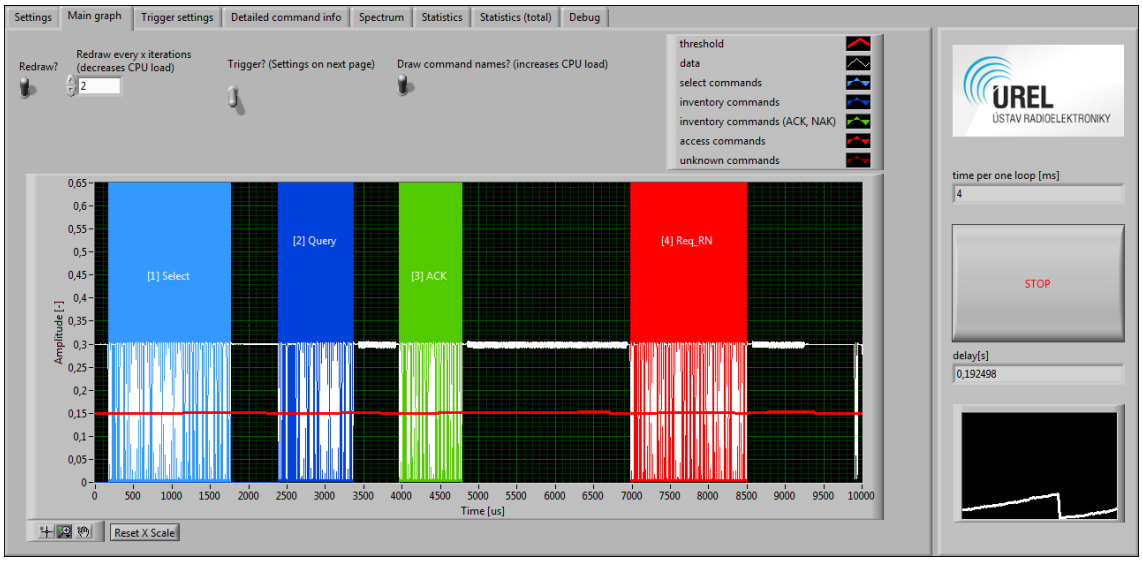
Následují bloky pokročilé analýzy. Podle nastavení se např. zobrazí časový průběh signálu s vyznačením jednotlivých příkazů a jejich názvů nebo dojde k zastavení vykreslování, pokud signál obsahuje některý z námi zvolených příkazů (obdoba funkce trigger u digitálních osciloskopů). Dále je možnost zobrazit si detailní informace o dekódovaném příkazu čtečky – jeho název, veškeré popsané parametry a zda-li je očekávána odpověď tagu. Dalšími bloky jsou sledování statistik jednotlivých příkazů a zápis všech informací do logovacího souboru.

Po skončení zpracování jednoho bloku dojde k otestování, zda-li si uživatel nepřeje ukončit běh programu. Pokud ano, dojde k zastavení zachytávání, ukončení relace s USRP zařízením a zastavení běhu aplikace. V opačném případě se pokračuje zachycením a analýzou dalšího bloku signálu.

5.1 Popis uživatelského prostředí programu

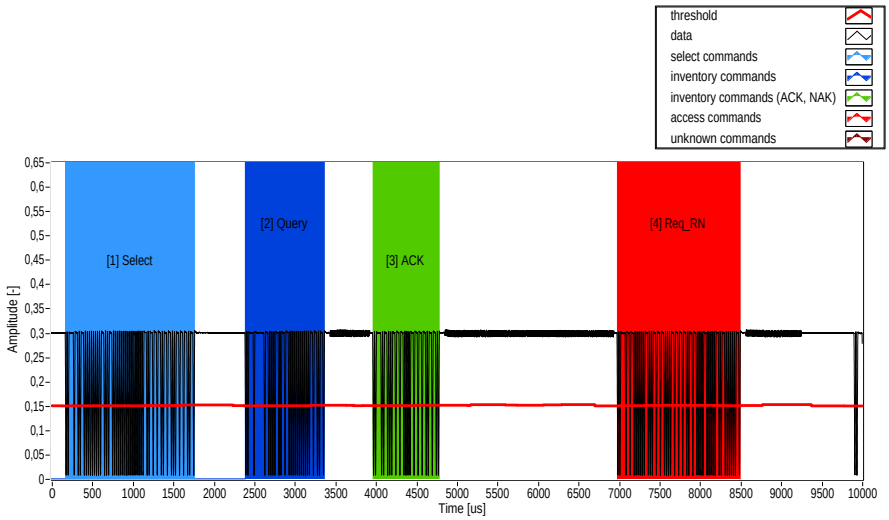
5.1.1 Hlavní panel (Main panel)

Na obrázku č. 5.2 lze vidět základní panel vytvořené aplikace. Obrazovka je rozdělená na dvě části – převážná část (vlevo) je věnována aktuální zvolené funkci, vpravo se nachází indikátory a prvky společné pro všechny funkce. Jedná se o: ukazatele zobrazující, jaký časový interval v milisekundách je potřeba ke zpracování jednoho bloku dat. Přijatelná hodnota záleží na aktuálním nastavení, viz dále. Dále je zde tlačítko sloužící k ukončení běhu programu a ukazatel, který zobrazuje aktuální hodnotu zpoždění (v sekundách) mezi zpracovávaným signálem a reálným časem.



Obr. 5.2: Základní panel programu.

Ideálně by se tato hodnota měla blížit k nule, v praxi se pohybuje v hodnotách řádově desítky milisekund. Zároveň je tato hodnota vykreslována do grafu, z něhož lze letmým pohledem zjistit, zda program stíhá zpracovávat signál v reálném čase – pokud hodnota zpoždění vytrvale stoupá, je potřeba určitým způsobem (např. snížením frekvence vykreslování, viz dále) snížit výpočetní náročnost.



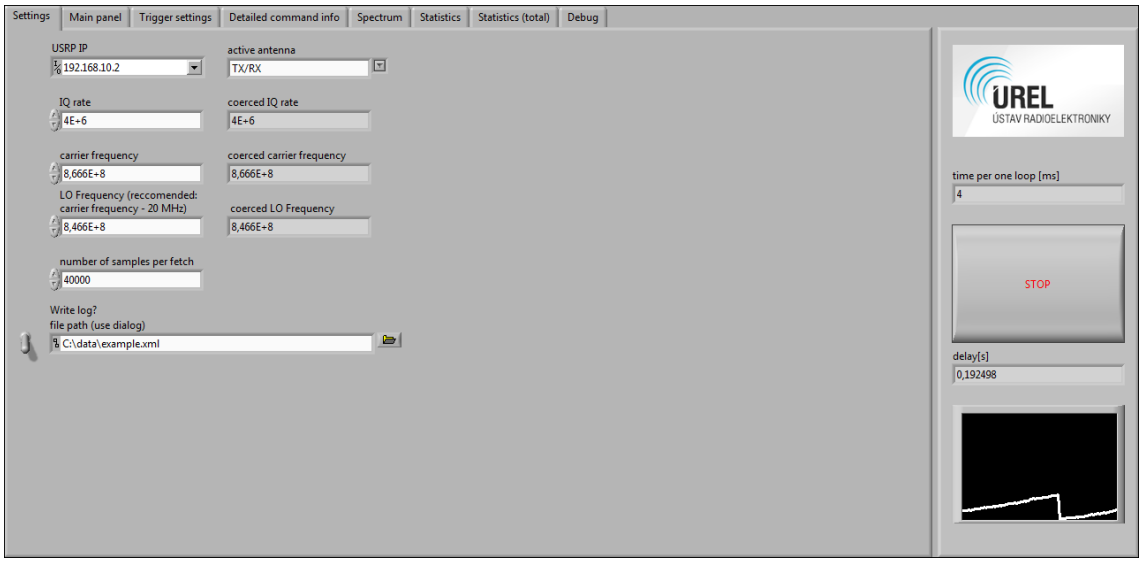
Obr. 5.3: Graf z obrázku č. 5.2 vyobrazený v barvách, vhodnějších pro tisk.

Dále je zde k vidění otevřený základní panel – převážnou část tohoto panelu zabírá graf, ve kterém je průběžně vykreslován demodulovaný časový průběh přijímaného signálu a určená rozhodovací úroveň pro detekci hran. Dále jsou zde vy-

značeny detekované příkazy čtečky, které jsou od sebe barevně odlišeny podle typu příkazu – příkaz **Select** je vyznačen světle modrou barvou, příkazy **ACK** a **NAK** jsou vyznačeny zeleně a ostatní příkazy spadající do skupiny Inventory příkazů (tedy **Query**, **QueryAdjust** a **QueryRep**) jsou označeny tmavě modrou barvou. Zbývající příkazy – tedy příkazy kategorie Access (tzn. **Req_RN**, **Read**, **Write**, **Kill**, **Lock**, **Access**, **BlockWrite**, **BlockErase** a **BlockPermalock**) jsou označeny červeně. Příkazy, které se nepodařilo rozpoznat jsou vyznačeny hnědou barvou. Dále je každý příkaz opatřen (pokud je tato funkce povolena) textovým popisem s názvem a pořadím příkazu. Tento graf je pro lepší čitelnost vyobrazen v barvách vhodnějších pro tisk na obrázku č. 5.3.

Tento panel obsahuje i několik ovládacích prvků – přepínač sloužící k zastavení vykreslování průběhu (např. za účelem detailního prozkoumání dat), číselný údaj, který udává, jak často se má překreslovat hlavní okno – zvětšením této hodnoty se rapidně snižuje zatížení procesoru a spínač, který povoluje/zakazuje zobrazování textových popisků v grafu – vypnutím této funkce se taktéž snižuje výpočetní náročnost. Posledním nejmenovaným prvkem je spínač povolující funkci trigger (viz kapitola 5.1.3).

5.1.2 Nastavení (Settings)



Obr. 5.4: Panel sloužící ke konfiguraci programu.

Na obrázku č. 5.4 je vyobrazen panel sloužící k nastavení parametrů zachycení (tedy konfiguraci USRP přístroje) i samotné analýzy. Je zde možnost zvolit IP adresu USRP zařízení, aktivní anténu (v případě Ettus USRP-N200 jsou na výběr

dvě možnosti - TX/RX a RX2), vzorkovací kmitočet (IQ rate, implicitně 4 MS/s), nosnou frekvenci (implicitně 866,6 MHz), frekvenci lokálního oscilátoru (implicitně 846,6 MHz) a počet vzorků v jednom bloku dat (implicitně 40 000). Časová délka jednoho bloku se rovná $t = \frac{\text{počet vzorků v jednom bloku}}{\text{vzorkovací frekvence}}$, při standardním nastavení tedy $t = \frac{40\,000}{4\,000\,000} = 10$ ms. Lze také povolit zápis veškerých dekódovaných informací do souboru ve formátu XML a případně určit název tohoto souboru. Tato nastavení nelze měnit během běhu programu, v případě nutnosti je tedy potřeba program ukončit, změnit parametry a aplikaci spustit znovu.

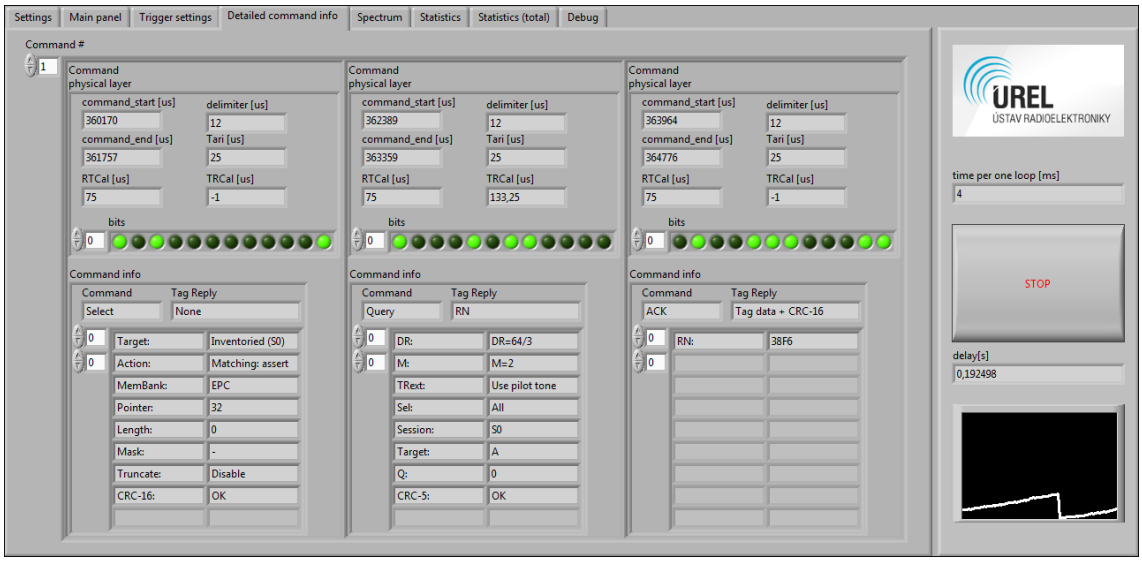
5.1.3 Nastavení triggeru (Trigger settings)

Obr. 5.5: Panel sloužící ke konfiguraci funkce trigger.

V tomto okně se provádí konfigurace funkce trigger, která funguje na stejném principu, jako stejnojmenná funkce u digitálních osciloskopů – pokud se v signálu vyskytuje pro nás zajímavý nebo důležitý děj, dojde k zastavení vykreslování průběhu v hlavním grafu (a v dalších s ním souvisejících panelech, viz dále) a je možné detailně prostudovat danou situaci. Lze zvolit jeden nebo více příkazů, při kterých k tomuto dojde.

5.1.4 Detailní informace o příkazech (Detailed command info)

Na tomto panelu jsou k dispozici detailní informace o příkazech zobrazovaných v hlavním okně. Překreslování tohoto panelu je spjato s vykreslováním hlavního



Obr. 5.6: Panel zobrazující detailní informace o příkazech.

grafu, číslování jednotlivých příkazů tedy vždy koresponduje s očíslováním v základním panelu.

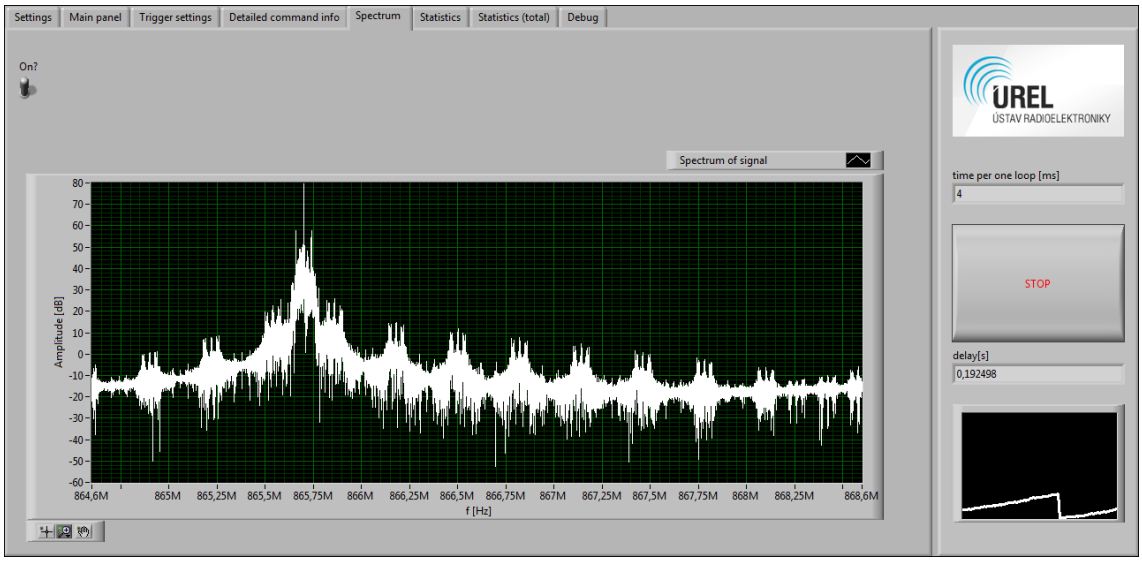
Pro každý dekódovaný příkaz se zobrazí jak informace o přenosové vrstvě (tedy začátek a konec vysílání, délky symbolů delimiter, Tari, RTCal, případně TRCal), tak bitové vyjádření příkazu i jeho název, očekávaná odpověď tagu a jednotlivé přeložené parametry s popisem. Samozřejmostí je ověřování kontrolních součtů a zobrazení výsledku této operace (OK/Bad).

5.1.5 Spektrum (Spectrum)

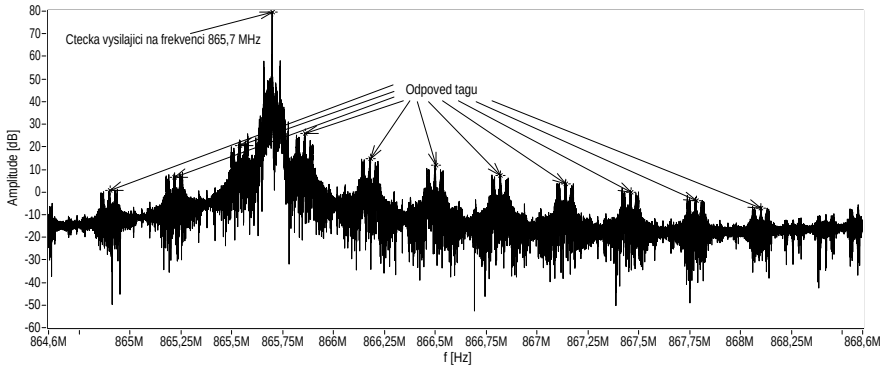
V tomto okně je vyobrazeno (pokud je tato funkce zapnuta – opět lze vypnout pro úsporu procesorového času) frekvenční spektrum bloku signálu, zobrazeného v hlavním okně. Ukázka je na obrázku č. 5.7. Graf je pro lepší čitelnost zobrazen v barvách vhodnějších pro tisk i na obrázku č. 5.8. V tomto spektru lze pozorovat jak vysílání čtečky (poměrně úzkopásmové) na frekvenci 865,7 MHz, tak i odpověď tagu. Z důvodů uvedených v kapitole 1.1 (tag komunikuje tzv. backscatteringem bez jakékoliv filtrace) tato odpověď obsahuje velké množství vyšších harmonických složek.

5.1.6 Statistiky (Statistics)

V tomto panelu jsou zobrazovány průběžné statistiky četností výskytů jednotlivých příkazů za zadaný počet zpracovávaných bloků dat. Jsou zobrazeny jak číselné hodnoty (počty) výskytů, tak i grafické vyjádření poměrného zastoupení jednotlivých



Obr. 5.7: Panel zobrazující spektrální analýzu bloku signálu.

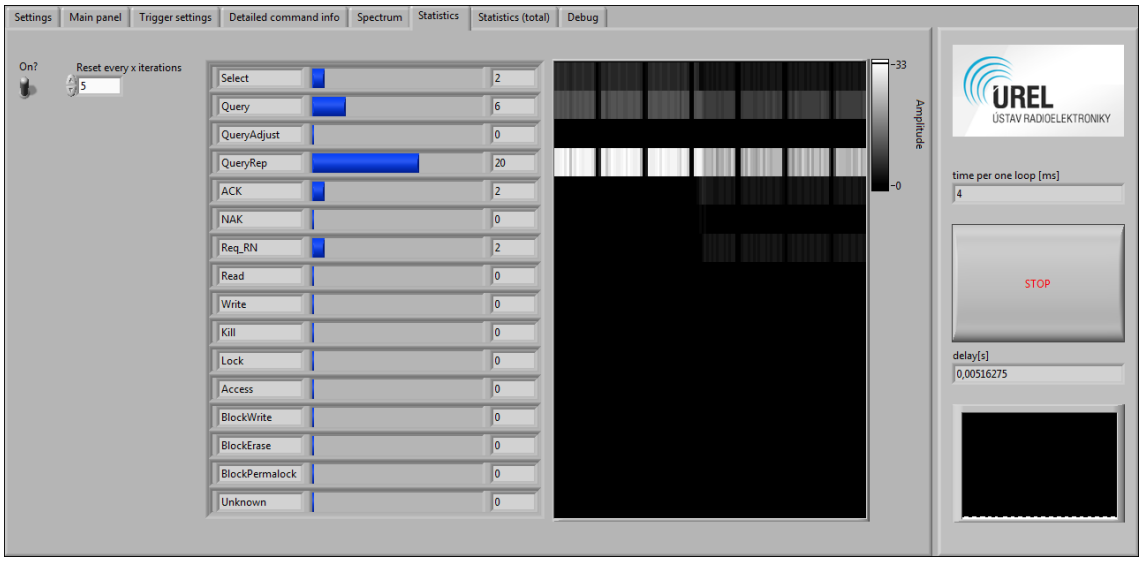


Obr. 5.8: Graf z obrázku č. 5.7 v podobě vhodnější pro tisk.

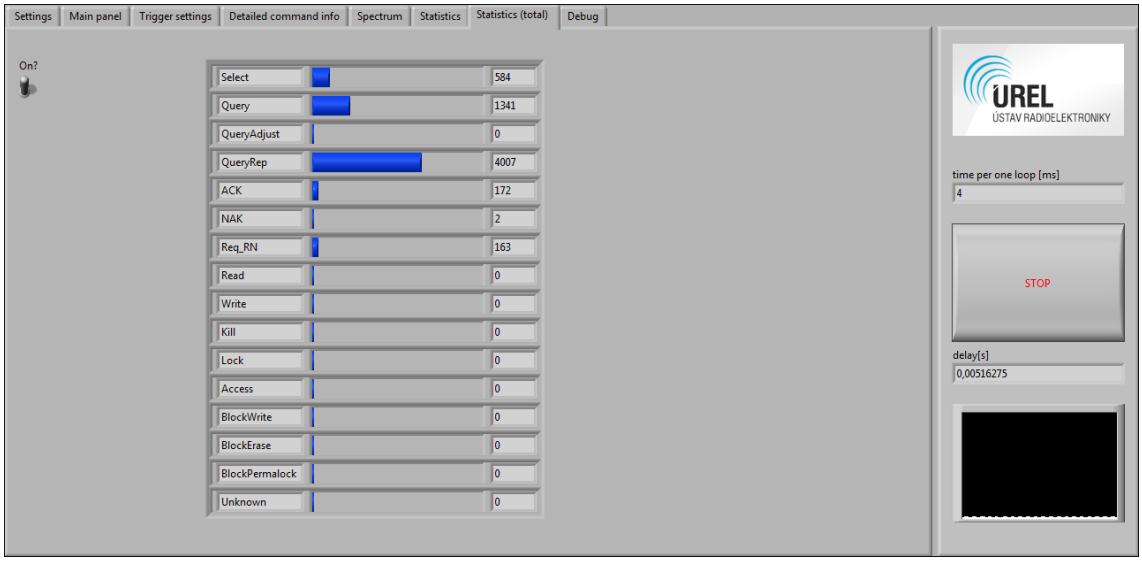
příkazů. Dále jsou tato data vykreslována do grafu takovým způsobem, aby bylo možné pozorovat změny výskytů v čase. Ukázka je na obrázku č. 5.9. Zde lze pozorovat četnosti příkazů čtečky ve dvou různých situacích – nejdříve se v poli čtečky nevyskytoval žádný tag (levá polovina grafu, jsou zastoupeny pouze příkazy **Select**, **Query** a **QueryRep**) a poté byl do pole čtečky vložen tag (pravá polovina grafu, kromě příkazů **Select**, **Query** a **QueryRep** lze pozorovat i příkazy potvrzující přítomnost tagu – **ACK**, **NAK** a **Req_RN**).

5.1.7 Celkové statistiky (Statistics – total)

Tento panel je prakticky totožný jako předchozí (statistiky), ovšem s tím rozdílem, že zobrazuje četnosti výskytů příkazů za celý běh programu.



Obr. 5.9: Panel zobrazující průběžné statistiky příkazů.



Obr. 5.10: Panel zobrazující celkové statistiky příkazů.

5.1.8 Ladění (Debug)

V tomto panelu jsou různé prvky, které sloužily k ladění a vývoji programu. Z uživatelského hlediska jsou bezvýznamné.

6 PRAKTICKÉ UKÁZKY ZACHYCENÝCH DĚJŮ

V této kapitole budou ukázány některé situace v UHF RFID komunikaci zachycené (zejména s pomocí funkce trigger) a analyzované aplikací vytvořenou v rámci této diplomové práce.

Grafy jsou zobrazeny přímo tak, jako v programu, pouze v barvách vhodnějších pro tisk.

6.1 Kontinuální čtení populace tagů

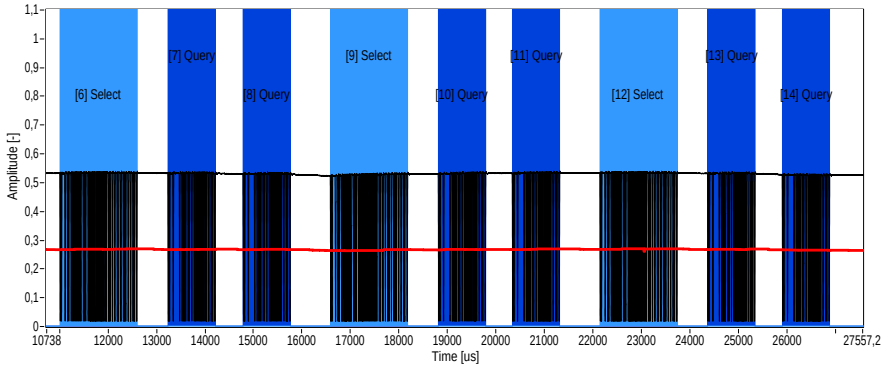
Zde budou ukázány průběhy z nejběžnějšího použití UHF RFID systémů, a to průběžného čtení populace tagů. V tomto případě se uplatí antikolizní princip protokolu EPC Class-1 Generation-2, který funguje následovně: tag po přijetí příkazu **Query** (který má mimo jiné důležitý parametr Q vyjadřující pravděpodobnost odpovědi tagu) nebo **QueryAdjust** (slouží k upravování hodnoty Q) vygeneruje do své paměti náhodné (respektive pseudonáhodné) číslo v rozsahu 0 až $2^Q - 1$ (včetně). Pokud je vygenerována hodnota 0, daný tag začne vysílat, při jiných hodnotách vyčkává. Při přijetí příkazu **QueryRep** se tato hodnota, uložená v paměti tagu, sníží o jedničku. Pokud se nyní toto číslo rovná nule, tag začne vysílat. Vysílání příkazu **QueryRep** je možné libovolně opakovat (prakticky nemá význam toto opakovat více než $2^Q - 1$, kdy i v případě, pokud by tag vygeneroval nejvyšší možnou hodnotu, tedy $2^Q - 1$, by postupným odčítáním došel k nulové hodnotě). Vyčerpávající informace o celém protokolu jsou v [2].

6.1.1 Q=0

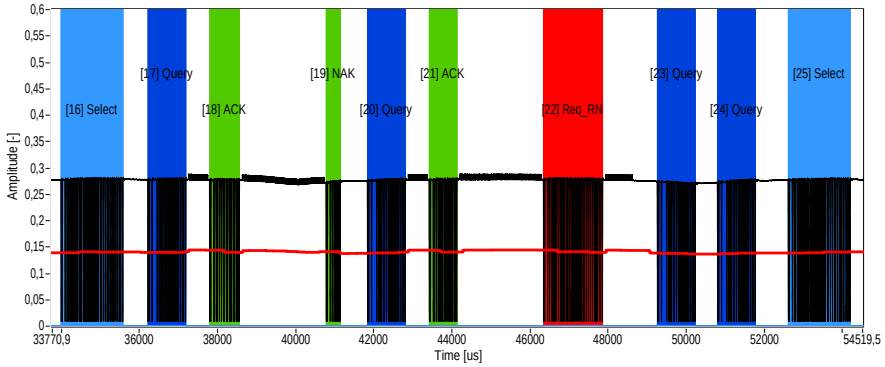
Pro demonstraci funkce antikolizního protokolu byl nejprve nastaven ve čtečce parametr Q na hodnotu 0. Tato hodnota v konečném důsledku znamená, že tag odpoví se stoprocentní pravděpodobností, tedy vždy.

Na obrázku č. 6.1 je zobrazena situace, kdy nejsou v poli čtečky žádné tagy. Čtečka vysílá příkaz **Select** (s parametry nastavenými tak, aby reagoval každý tag) a příkaz **Query** (se zmíněným parametrem Q=0). Pokud ani po druhém vysílání příkazu **Query** čtečka neobdrží žádnou odpověď, zahájí nové kolo inventarizace příkazem **Select** (toto chování záleží na druhu použité čtečky a na jejím nastavení).

Situaci po vložení jednoho tagu do pole čtečky lze vidět na obrázku č. 6.2. Lze pozorovat, že za prvním příkazem **Query** následuje odpověď tagu – náhodně vygenerované 16-ti bitové číslo. Na tuto odpověď čtečka reaguje potvrzením (příkazem



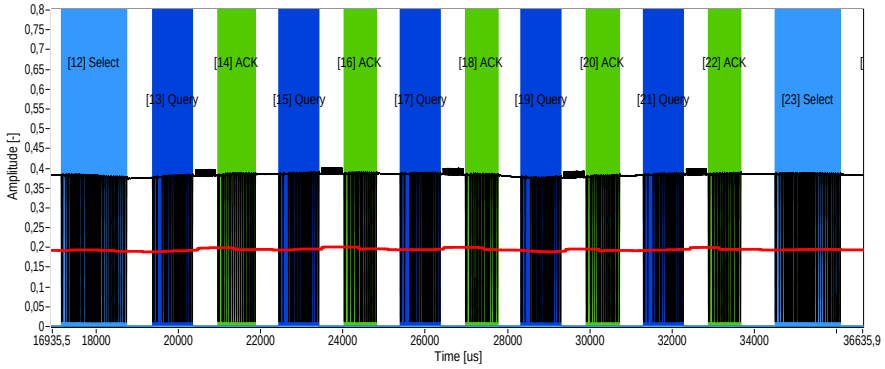
Obr. 6.1: Ukázka komunikace při nastaveném Q=0, pokud se v poli čtečky nevyskytuje žádný tag.



Obr. 6.2: Ukázka komunikace při nastaveném Q=0, pokud se v poli čtečky vyskytuje jeden tag.

ACK s přijatým 16-ti bitovým číslem jako parametrem), načež tag vyšle svůj EPC chráněný cyklickým redundantním součtem. Tato odpověď se ovšem nepovedla čtečkou dekódovat (cyklický redundantní součet nesouhlasil) a proto byl vyslán příkaz **NAK**. Druhý pokus o přečtení EPC je již úspěšný a po jeho odvysílání čtečka reaguje příkazem **Req_RN** (s parametrem dříve přijatého náhodného čísla jako identifikátoru tagu), který slouží k navázání bližší komunikace s konkrétním tagem – tag na tento příkaz reaguje vysláním dalšího 16-ti bitového čísla (chráněného pomocí CRC-16), které od této chvíle slouží jako autentifikační kód pro případné další příkazy (z kategorie Access příkazů), dále zvaný „handle“. Následují dva neúspěšné příkazy **Query** (tag nereaguje, neboť je v tomto kole již přečten) a poté zahájení nového kola inventarizace příkazem **Select**.

Případ, kdy je v poli čtečky více než jeden tag, lze vidět na obrázku č. 6.3. Je vidět, že po prvním příkazu **Query** následuje odpověď – náhodně vygenerované

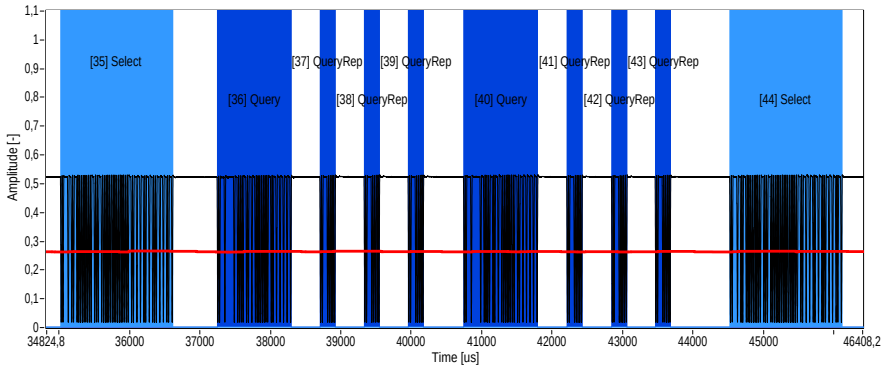


Obr. 6.3: Ukázka komunikace při nastaveném Q=0, pokud se v poli čtečky vyskytuje více než jeden tag.

16-ti bitové číslo. Jelikož je pravděpodobnost odpovědi tagu stoprocentní, je tato odpověď ovšem kolidovaná – je superpozicí vysílání všech tagů v poli. Čtečka dále vyšle příkaz **ACK** s parametrem (chybně) přijatého náhodného čísla, na který žádný z tagů nereaguje, neboť se čísla neshodují. Čtečka tedy znovu vyšle příkaz **Query**, ale z důvodů uvedených výše se situace opět opakuje – celkem pětkrát (opět záleží na nastavení čtečky) než je zahájeno nové kolo inventarizace příkazem **Select**.

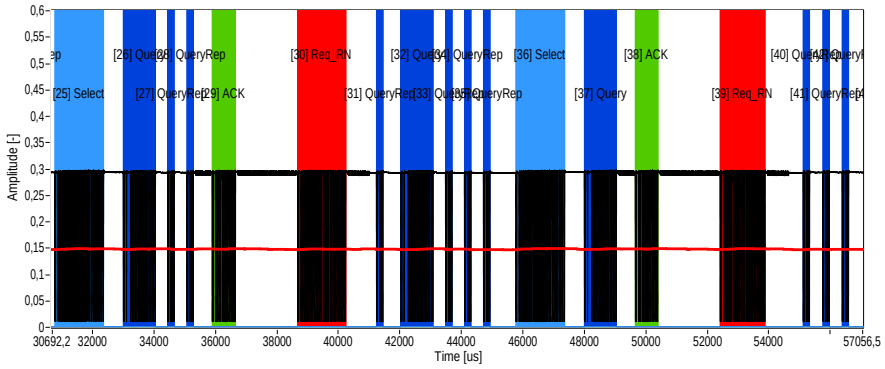
Lze pozorovat, že hodnota Q=0 znemožňuje fungování antikolizního protokolu, proto se v praxi samozřejmě používá větší hodnota, např. Q=2 (implicitní nastavení čtečky METRA RFI21.1).

6.1.2 Q=2



Obr. 6.4: Ukázka komunikace při nastaveném Q=2, pokud se v poli čtečky nevyskytuje žádný tag.

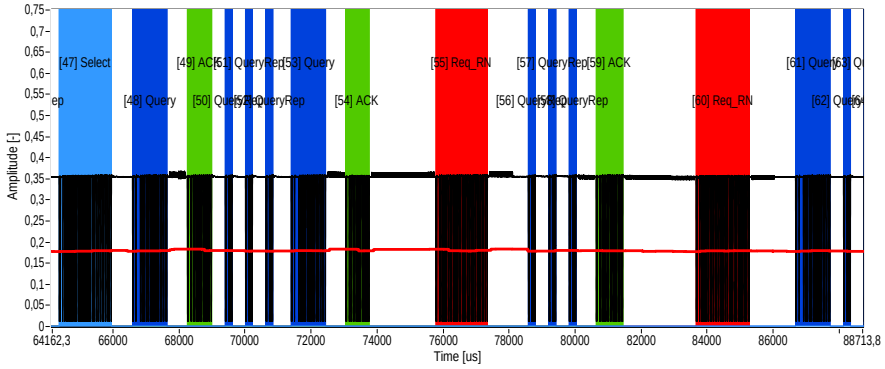
Na obrázku č. 6.4 lze vidět zachycený průběh vysílání čtečky s nastaveným parametrem Q=2, pokud se v blízkosti antény čtečky nenachází žádný tag. Čtečka zahájí kolo čtení vysláním příkazu **Select**, po něm následuje příkaz **Query** se zminovaným parametrem Q=2. Maximální hodnota, kterou tag může vygenerovat je tedy $2^Q - 1 = 2^2 - 1 = 3$. Následují tedy tři opakování příkazu **QueryRep**. Žádny tag se neozve ani po druhém opakování sekvence (**Query** a tři opakováni **QueryRep**) a proto je zahájeno další kolo inventarizace příkazem **Select**.



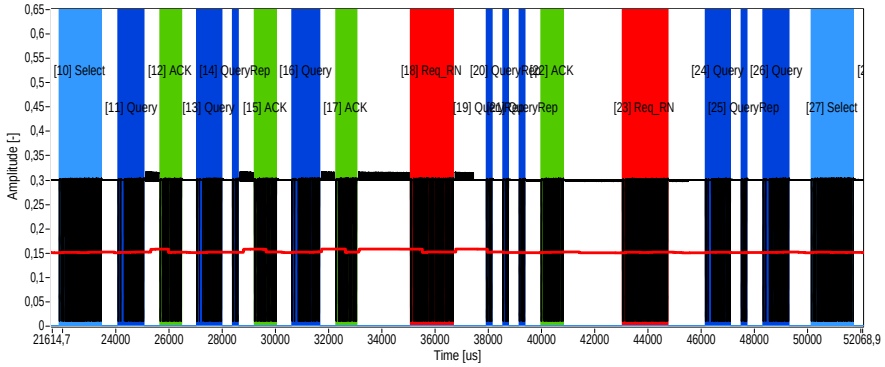
Obr. 6.5: Ukázka komunikace při nastaveném Q=2, pokud se v poli čtečky vyskytuje jeden tag.

Případ, kdy se v blízkosti antény čtečky nachází jediný tag, zachycuje obrázek č. 6.5. Lze vidět, že tag v prvním případě odpoví po druhém vysílání příkazu **QueryRep**, evidentně tedy náhodně vygeneroval číslo dva. Následuje známá sekvence potvrzující zachycení odpovědi tagu čtečkou – příkazy **ACK** a **Req_RN**. Poté je zopakována sekvence (opět **Query** a tři opakováni **QueryRep** jako v předchozím příkladu), kdy tag neodpoví, jelikož už je přečten. Následuje další kolo inventarizace – zde si lze všimnout, že tag vysílá ihned po příkazu **Query**, náhodně tedy vygeneroval číslo nula.

Na obrázku č. 6.6 lze pozorovat obdobnou situaci zachycující dva tagy v poli čtečky. Lze si povšimnout, že po prvním příkazu **Query** došlo ke kolizi – oba dva tagy vygenerovaly číslo nula. Následuje tedy **ACK** (s parametrem chybně dekódovaného přijatého čísla), na který žádný z tagů neodpoví. Posléze proběhnou tři opakování příkazu **QueryRep**, které jsou rovněž bez odezvy, jelikož tagy už v tomto kole vyslaly svou odpověď. Po dalším příkazu **Query** jsou již tagy bez obtíží přečteny, jelikož vygenerovaly jiná náhodná čísla (konkrétně nula a tři).



Obr. 6.6: Ukázka komunikace při nastaveném Q=2, pokud se v poli čtečky vyskytují dva tagy.



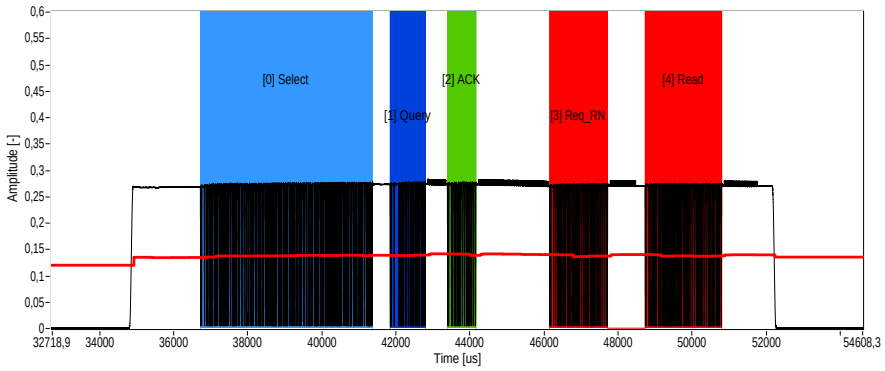
Obr. 6.7: Ukázka komunikace při variabilním Q, pokud se v poli čtečky vyskytují dva tagy.

6.1.3 Variabilní Q

Další možností nastavení čtečky, místo zadané pevné hodnoty Q, je tuto hodnotu variabilně měnit dle situace. V případě, že se v blízkosti antény čtečky nenachází žádný tag, je průběh stejný, jako by byla nastavena hodnota Q=0, tedy tak, jako na obrázku č. 6.1 – dochází k opakovanému vysílání příkazů **Select** a **Query** s parametrem Q=0. Rovněž pro jeden tag v poli je situace obdobná jako na obrázku č. 6.2 – dochází k bezproblémovému čtení onoho jediného tagu pomocí příkazu **Query** s parametrem Q=0. Zajímavý průběh nastane až v případě, kdy jsou v poli čtečky dva a více tagů, viz obrázek č. 6.7. Po prvním vysílání příkazu **Query** s parametrem Q=0 dojde samozřejmě ke kolizi vysílání dvou tagů. Toto čtečka detekuje a proto má další pokyn **Query** parametr nastaven na Q=1 (stejného efektu by se dalo dosáhnout i za pomoci příkazu **QueryAdjust**). I v tomto případě ovšem dojde ke kolizi, jelikož tagy vygenerovaly stejnou náhodnou hodnotu (na vyobrazeném případu jedničku).

Po třetím příkazu **Query** již oba tagy vygenerovaly odlišnou náhodnou hodnotu (nula a tři) a proto došlo k bezproblémovému přečtení obou tagů.

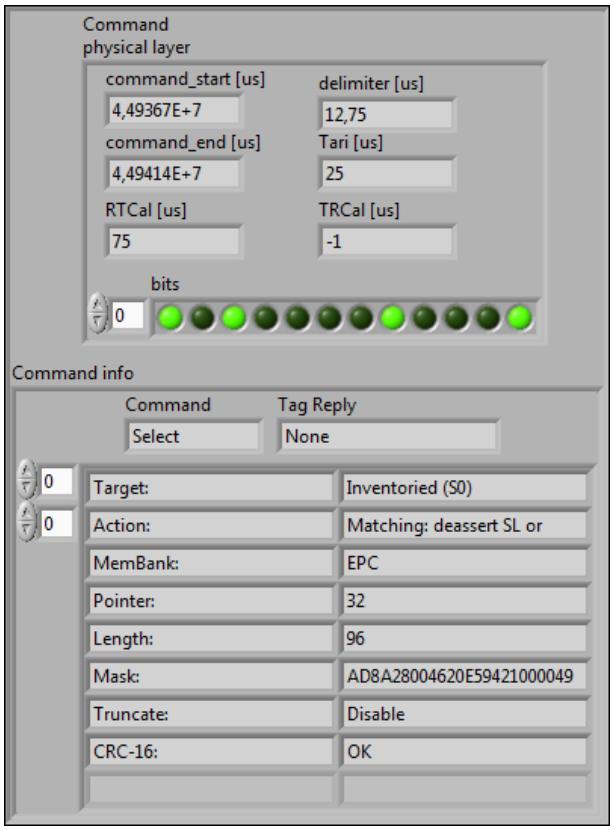
6.2 Čtení paměti konkrétního tagu



Obr. 6.8: Ukázka čtení paměti tagu.

Na obrázku č. 6.8 lze vidět průběh přečtení bloku dat jednoho konkrétního tagu. Jedná se o jednorázový děj – je vidět, že před i po skončení dané operace je úroveň signálu nulová. Paměťová banka EPC čteného tagu obsahuje (od adresy 0) tato data: „C3 3B 30 00 AD 8A 28 00 46 20 E5 94 21 00 00 49“. Organizace paměti této banky je následující: první slovo (16 bitů) obsahuje tzv. StoredCRC, což je kontrolní součet, který se vypočte pokaždé při startu tagu, tedy při jeho vložení do pole čtečky, a to z následujícího slova (StoredPC) a EPC. Další slovo je zmíněný StoredPC (Protocol Control), který obsahuje různé informace týkající se paměti tagu. Nejdůležitějších je prvních pět bitů, které vyjadřují délku vlastního identifikátoru EPC. V tomto případě je délka EPC $00110_2 = 6_{10}$, tedy šest slov nebo-li $6 \cdot 16 = 96$ bitů. Následující slova pak už vyjadřují přímo EPC.

Celý proces začíná příkazem **Select** – v tomto případě s takovými parametry, aby odpovídal pouze požadovaný tag. Tyto parametry lze vidět na obrázku č. 6.9. Zajímavé jsou především tyto: MemBank (EPC) – definuje paměťovou banku, ve které má tag provádět srovnání dat, Pointer (32) – začátek srovnávaných dat, Length (96) – délka srovnávaných dat a Mask („AD 8A 28 00 46 20 E5 94 21 00 00 49“) – vlastní data. Hledá se tedy tag s konkrétním EPC, viz výše. Při porovnání dat uložených v EPC se zadanou hodnotou od určené adresy (offset je 32 bitů, začíná se tedy 5. bajtem, kde začíná samotné EPC) vidíme, že data jsou skutečně shodná, proto tag po přijetí příkazu **Query** (s parametrem Q=0) odpoví, načež čtečka reaguje potvrzením (**ACK**). Následuje příkaz **Req_RN**, kterým čtečka získá „handle“ nutný

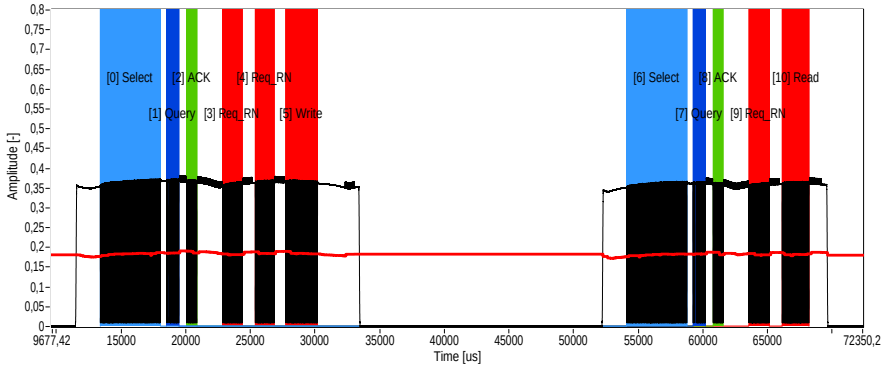


Obr. 6.9: Parametry příkazu **Select** pro výběr pouze jednoho konkrétního tagu.

k provedení příkazů ze skupiny Access. Poté už následuje příkaz **Read**, na který tag reaguje odvysíláním požadovaných dat.

6.3 Zápis do paměti konkrétního tagu

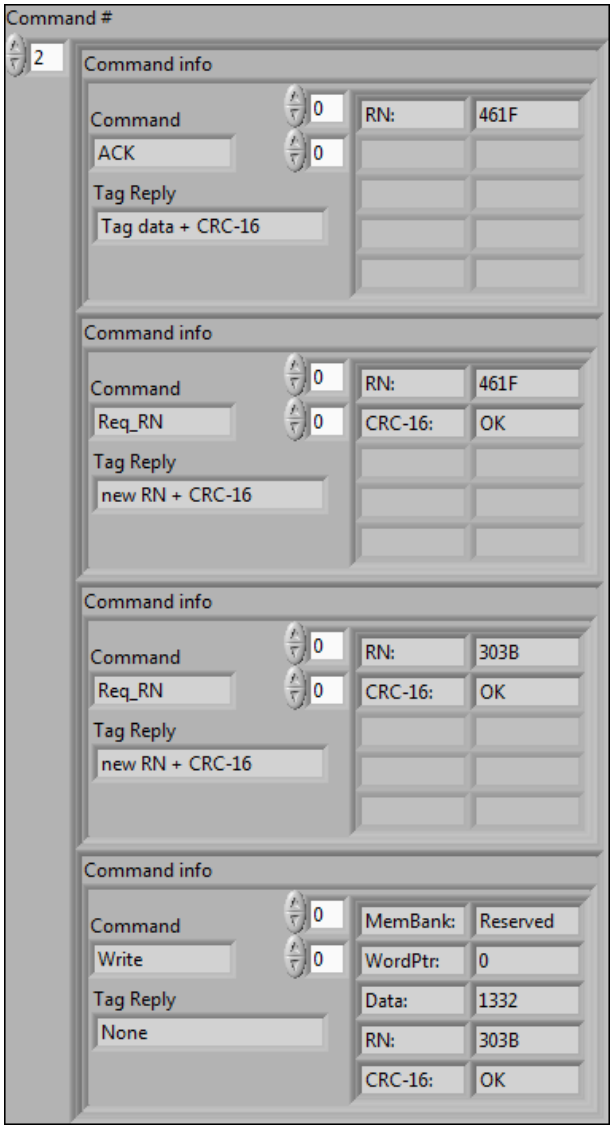
Na obrázku č. 6.10 lze vidět příklad zápisu do paměti stejného tagu, který byl použit v ukázce čtení. Vyslané příkazy **Select** a **Query** jsou totožné jako v předchozím příkladu. Tag poté tedy přirozeně reaguje vysláním náhodného 16-ti bitového čísla, které je potvrzeno příkazem **ACK**. Následuje opět příkaz **Req_RN**, kterým čtečka získá „handle“ k provedení dalších příkazů. Zápis pak probíhá takto: nejprve je vyslán další příkaz **Req_RN**, kterým čtečka získá nové náhodné 16-ti bitové číslo, se kterým XORuje zapisovaná data a teprve poté je vyslán příkaz **Write**, kterým se uskuteční samotný zápis do paměti tagu. Tento postup je určitým druhem zabezpečení – bez znalosti zmíněného 16-ti bitového čísla nelze zjistit, jaká konkrétní data byla zapsána. Toto číslo bylo ovšem vysíláno pouze tagem, jehož signál se dá zachytit na vzdálenost maximálně několika metrů – naproti tomu vysílání čtečky lze odposlechnout na velké vzdálenosti v řádech kilometrů. Pro lepší ilustraci jsou



Obr. 6.10: Ukázka zápisu do paměti tagu.

parametry některých příkazů v této sekvenci zobrazeny na obrázku č. 6.11. Dále si lze povšimnout delší odmlky mezi příkazem **Write** a následným vysíláním tagu – to je způsobeno větší časovou a energetickou náročností operace zápisu do paměti. Čtečka proto po odvysílání příkazu určeného pro zápis vysílá nemodulovanou vlnu, aby měl tag dostatek energie k provedení zápisu. Po obdržení potvrzení od tagu je vysílání ukončeno.

Procedura zápisu do paměti tagu je dokončena ověřovacím přečtením právě zapsaných dat prakticky stejnou sekvencí příkazů, jako v předchozím příkladě.



Obr. 6.11: Parametry vybraných příkazů při zápisu dat do tagu.

7 ZÁVĚR

Cílem této diplomové práce bylo navrhnout koncepci odposlechu a analýzy evropského UHF RFID provozu a následně provést realizaci. Tento cíl byl splněn vytvořením aplikace, která pracuje v reálném čase s minimálním zpožděním.

Tento program umožňuje pomocí zařízení USRP a běžného osobního počítače sledovat a analyzovat příkazy jedné čtečky (i v režimu frequency hopping). Jsou implementovány tyto funkce: nastavení parametrů zachytávání a analýzy, zobrazení demodulovaného časového průběhu s označením a popsáním jednotlivých příkazů, vykreslení frekvenčního spektra, vypsání všech příkazů a jejich dekódovaných parametrů včetně výpočtů kontrolních součtů, sledování statistik, zastavení vykreslování v případě výskytu určených příkazů (trigger) a zápis dekódovaných dat do souboru XML.

Činnost analyzátoru byla experimentálně vyzkoušena za různých podmínek. Příjem různě modulovaných (DSB-ASK, SSB-ASK i PR-ASK) signálů ani dat s různou přenosovou rychlostí (Tari 6,25 až 25 μs) vysílaných čtečkou nečiní problémy.

Aplikace může mít praktické využití při vývoji a testování různých RFID zařízení – jak čteček, tak i tagů nebo při hledání problémů a zlepšování efektivity v zaběhnutých provozech využívajících RFID systémy. Taktéž může být využito ve výuce pro názorné pozorování dějů probíhajících při RFID komunikaci.

Následujícím rozšířením programu by mohlo být analyzování vysílání na více kanaléch současně. To by obnášelo vyfiltrovat jednotlivé kanály a dále provádět demodulaci, detekci hran atd. pro každý kanál zvlášť. K tomu by se dal vhodně využít obvod FPGA v přístroji Ettus USRP-N200, případně Ettus USRP-N210, který je totožný s N200 s tím rozdílem, že má výkonnější FPGA (Spartan 3A-DSP 3400 FPGA oproti Spartan 3A-DSP 1800 FPGA u N200). Dále by vývoj mohl pokračovat implementací dekódování odpovědi tagů – toto nebylo realizováno z toho důvodu, že vysílání tagu se podaří zřídkakdy zachytit.

Kompletní zdrojové kódy programu jsou k dispozici na přiloženém kompaktním disku. Minimální verze vývojového prostředí LabVIEW, nutná k otevření a spuštění aplikace je 2010 (10.0f2).

LITERATURA

[1] DOBKIN, D. M. *The RF in RFID: Passive UHF RFID in Practice*. Burlington: Newnes, 2008.

[2] EPCglobal Inc. *Class-1 Generation-2 UHF RFID Protocol for Communications at 860 MHz – 960 MHz. Version 1.2.0* [online]. 2008, [cit. 10. 4. 2012]. Dostupné z URL: <http://www.gs1.org/gsmp/kc/epcglobal/uhfc1g2/uhfc1g2_1_2_0-standard-20080511.pdf>.

[3] Metra Blansko a. s. *UHF RFID Compact Reader 865 – 868 MHz, 902 – 928 MHz* [online]. 2010, [cit. 1. 5. 2012]. Dostupné z URL: <http://www.asicentrum.cz/data/pdf/datasheet_rfi21.pdf>.

[4] IEEE RFID 2012 *Advanced UHF RFID Tag Antenna Design* [online]. 2010, [cit. 1. 5. 2012]. Dostupné z URL: <http://2012.ieee-rfid.org/>.

[5] Ettus Research *USRP N200/N210 networked series* [online]. 2010, [cit. 1. 5. 2012]. Dostupné z URL: <http://www.ettus.com/content/files/2987_Ettus_N200-210_DS_FINAL_1.27.12.pdf>.

[6] Direct Industry *IDTRONIC GmbH* [online]. 2011, [cit. 8. 12. 2012]. Dostupné z URL: <http://www.directindustry.com/prod/idtronic-gmbh/rfid-reader-writers-88127-845563.html>.

[7] Atlanta Communications Company *Mobile RFID Readers* [online]. 2011, [cit. 8. 12. 2012]. Dostupné z URL: <http://www.atlantacomm.com/Templates/Mobile%20RFID%20Readers.dwt>.

[8] LabVIEW *In: Wikipedia: the free encyclopedia* [online]. 2012, [cit. 8. 12. 2012]. Dostupné z URL: <http://en.wikipedia.org/wiki/LabVIEW>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

BLF Backscatter Link Frequency – modulační rychlost odpovědi tagu

DSB-ASK Double-SideBand Amplitude Shift Keying – digitální amplitudová modulace s oběma postranními pásmy

DDC Direct downconversion – způsob přeložení signálu z mezifrekvence do základního pásma, realizuje se zejména v FPGA

DSP Digital Signal Processor – digitální signálový procesor, procesor určený zejména pro zpracování signálu

EBV Extensible Bit Vector – rozšířitelný bitový vektor, způsob vyjádření celočíselného čísla s proměnným rozsahem

EPC Electronic Product Code – jednoznačný unikátní kód produktu uložený v paměti tagu

ETSI European Telecommunications Standards Institute – evropský institut telekomunikačních standardů, nezávislá organizace spravující telekomunikační standardy v Evropě

FCC Federal Communications Commission – federální komunikační komise, nezávislá organizace spravující telekomunikační standardy ve Spojených státech amerických

FPGA Field Programmable Gate Array – programovatelné logické pole, číslicové integrované obvody, které lze naprogramovat pro různé použití

GPIO General Purpose Input/Output – vývod na procesoru, jehož chování lze uživatelsky naprogramovat

GUI Graphical User Interface – grafické uživatelské prostředí

LabVIEW Laboratory Virtual Instrumentation Engineering Workbench – moderní grafický programovací jazyk a vývojové prostředí firmy National Instruments

LNA Low Noise Amplifier – nízkošumový zesilovač

PA Power Amplifier – výkonový zesilovač

PIE Pulse Interval Encoding – technika kódování symbolů pomocí délky intervalu

PR-ASK Phase Reversal Amplitude Shift Keying – digitální amplitudová modulace s otáčením fáze

RFID Radio Frequency Identification – identifikace na rádiové frekvenci, generace identifikátorů navržených (nejen) k identifikaci zboží

SDR Software Defined Radio – softwarově definované rádio, rádiový systém, v němž je zpracování signálu prováděno převážně softwarově

SSB-ASK Single-SideBand Amplitude Shift Keying – digitální amplitudová modulace s jedním postranním pásmem

UART Universal Asynchronous Receiver/Transmitter – zařízení pro sériovou komunikaci

UHD Universal software radio peripheral Hardware Driver – ovladač pro USRP, je dostupný pro všechny nejrozšířenější platformy (Windows, Linux, Max)

UHF Ultra High Frequency – ultra vysoké frekvence, pásmo elektromagnetických vln v rozsahu 300 MHz až 3 GHz

USB Universal Serial Bus – moderní rozšířené PC rozhraní sloužící k připojení různých periferií

USRP Universal Software Radio Peripheral – univerzální zařízení pro softwarové rádio, periferie určená k připojení k PC, sloužící jako nástroj pro testování a vývoj SDR aplikací

XML Extensible Markup Language – rozšiřitelný značkovací jazyk, obecný standardizovaný značkovací jazyk pro různé typy dat.