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

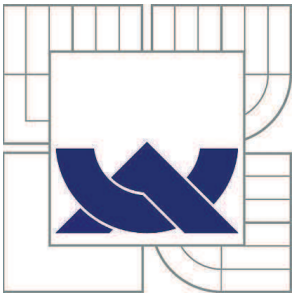
ZPRACOVÁNÍ SIGNÁLU UHF RFID ČTEČKY

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

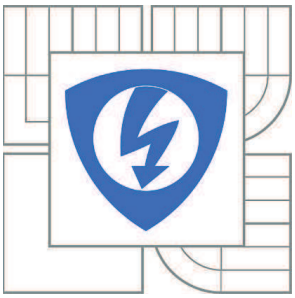
Bc. JAN NOVOTNÝ

BRNO 2015



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

ZPRACOVÁNÍ SIGNÁLU UHF RFID ČTEČKY

SIGNAL PROCESSING FOR UHF RFID READER

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

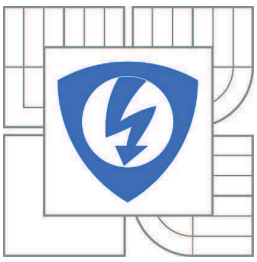
AUTOR PRÁCE — Bc. JAN NOVOTNÝ

AUTHOR

VEDOUCÍ PRÁCE — Ing. ALEŠ POVALAČ, Ph.D.

SUPERVISOR

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav radioelektroniky

Diplomová práce

magisterský navazující studijní obor
Elektronika a sdělovací technika

Student:	Bc. Jan Novotný	***ID:***	134575
Ročník:	2	***Akademický rok:***	2014/2015

NÁZEV TÉMATU:

Zpracování signálu UHF RFID čtečky

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s koncepcí front endu pro UHF RFID čtečku EXIN-1. Analyzujte dostupná komunikační rozhraní, zhodnoťte možnosti zpracování signálu v základním pásmu pomocí FPGA a mikrokontroléru s jádrem ARM Cortex-M4. Proveďte oživení základních funkcí front endu, navrhněte blokovou koncepci spojení front endu s řídicím systémem.

Navrhněte a realizujte prototyp UHF RFID čtečky, která umožňuje čtení RFID tagů standardu Gen2 a přečtené údaje zobrazuje na displeji. Vytvořte firmware pro mikrokontrolér, který bude řídit front end, vysílat a přijímat data v základním pásmu a zpracovávat základní příkazy Gen2 protokolu (Select, Query, QueryRep, Ack). Proveďte testování a měření parametrů vyvinuté čtečky.

DOPORUČENÁ LITERATURA:

[1] POVALAČ, A.; ŠEBESTA, J. Experimental Front End for UHF RFID Reader [online]. Elektrorevue - Internetový časopis (http://www.elektrorevue.cz), roč. 2011, č. 1, s. 55-59. Dostupné na www: http://goo.gl/z8uOg2.

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

Termín zadání:	9.2.2015	***Termín odevzdání:***	21.5.2015

Vedoucí práce: Ing. Aleš Povalač, Ph.D.
Konzultanti diplomové práce:

doc. Ing. Tomáš Kratochvíl, Ph.D.
Předseda oborové rady

ABSTRAKT

Tato diplomová práce se zabývá zpracováním signálu experimentální UHF RFID čtečky EXIN-1. V první části popisuje koncepci front endu čtečky EXIN-1, její oživení a dostupná komunikační rozhraní pro řízení čtečky a pro vysílání a příjem signálu v základním pásmu. V druhé části se zabývá zjednodušeným popisem standardu EPC Global Class-1 Generation-2 UHF RFID Protocol s důrazem na základní příkazy, používané modulace a kódování. V poslední části je navržena bloková koncepce připojení front endu k vývojové desce s mikrokontrolérem s jádrem ARM Cortex-M4, prostřednictvím kterého jsou generovány potřebné řídící i užitečné signály a který zachytává příchozí signály a zpracovává je tak, aby byla získána identifikační čísla RFID karet (tagů) nacházejících se v dosahu čtečky. Algoritmus zpracování je navržen v programu MATLAB a následně implementován do zvoleného mikrokontroléru. Získané identifikační údaje jsou jednak zobrazeny na LCD displeji a za druhé posílány do osobního počítače pomocí sériové komunikace.

KLÍČOVÁ SLOVA

UHF RFID čtečka, protokol EPC Global Class1 Gen2, mikrokontrolér STM32F429, ARM Cortex M4, vývojová deska STM32F4 Discovery, identifikace RFID tagů.

ABSTRACT

The master's thesis is focused on the UHF RFID reader EXIN-1 signal processing. The first part describes the concept of the EXIN-1 front end, its basic testing and possible communication interfaces for reader control and for receiving and transmitting baseband signals. The second part of this work is aimed to a simple description of EPCglobal Class-1 Generation-2 UHF RFID Protocol, especially to used modulations and codings. In the last part, a block connection between the front end and an ARM Cortex-M4 microcontroller discovery board is designed. The microcontroller is used for generating of all required signals and also for receiving incoming signals and processing them for identification numbers of RFID cards (tags), which are in the reading range of the reader. A decoding algorithm is designed in MATLAB software and implemented to the selected microcontroller. Obtained identification data are displayed on an LCD display and also sent to a PC through a serial communication.

KEYWORDS

UHF RFID reader, EPC Global Class1 Gen2 Protocol, STM32F429 microcontroller, ARM Cortex M4, STM32F4 Discovery board, identification of RFID tags.

NOVOTNÝ, J. *Zpracování signálu UHF RFID čtečky*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Ústav radioelektroniky, 2015. 59 s., 11 s. příloh. Diplomová práce. Vedoucí práce: Ing. Aleš Povalač, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma Zpracování signálu UHF RFID čtečky 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/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne 21. května 2015 .....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu diplomové práce Ing. Aleši Povalačovi, Ph.D. za účinnou metodickou a odbornou pomoc a především za cenné rady při zpracování mé diplomové práce.

V Brně dne 21. května 2015 .....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Radiofrekvenční identifikace (RFID) pracující v UHF pásmu je dnes široce nasazovanou technologií. Používá se např. k inventarizaci zboží ve skladech, identifikaci vozidel, k evidenci dálničních poplatků (mýtné brány) atd.

Rozsah používaných kmitočtů se pohybuje v rozmezí 860MHz až 960MHz, přičemž v tomto rozsahu jsou nejdůležitější dvě pásma, a to evropské (EU) a americké (US), ale existují ještě další. V evropském pásmu je rozsah kmitočtů 865,6MHz až 867,6MHz a toto pásmo je rozděleno na čtyři kanály po 200kHz (střední kmitočty 865,7MHz, 866,3MHz, 866,9MHz, 867,5MHz), v každém kanálu se může vysílat s maximálním výkonem 2W při šířce vyzařovacího svazku antény menším než 90°. Signály v RFID tedy musí plnit přísnou spektrální masku odpovídající evropským regulačním předpisům.

Oproti systémům pracujícím v pásmu LF a HF má identifikace v pásmu UHF výhodu v mnohem větším dosahu, protože pracuje na principu přenosu elektromagnetickými vlnami, zatímco v pásmech LF a HF se jedná o induktivní vazbu.

Cílem této diplomové práce je seznámení se z koncepcí front endu UHF čtečky EXIN-1, oživení základních funkcí front endu, zpracování analogového signálu v základním pásmu přijatého čtečkou do číslicové podoby a jeho vyhodnocení s cílem identifikovat tagy nacházející se v poli antény čtečky. K tomuto účelu je použit mikrokontrolér s jádrem ARM Cortex-M4, který také zajišťuje generování vysílaného signálu v podobě základních příkazů protokolu EPC Global Class-1 Generation-2 a řízení a nastavení front endu. Tím nahrazuje mikrokontrolér AVR, který původně sloužil pro základní ověření funkčnosti front endu popisované čtečky.

1 POPIS KONCEPCE FRONT ENDU PRO ČTEČKU EXIN-1

V následující kapitole je popsána koncepce front endu pro experimentální UHF RFID čtečku EXIN-1, která byla vyvinuta na Ústavu radioelektroniky FEKT VUT v Brně. Kapitola je rozčleněna do několika podkapitol, kdy každá z nich popisuje určitou část front endu. Celá tato kapitola čerpá z literatury [1].

1.1 Úvod, motivace

Na trhu nyní existuje mnoho RFID čteček, avšak tato zařízení většinou neumožňují plnou kontrolu všech parametrů, protože obvykle bývají založeny na specializovaném integrovaném obvodu. To také znesnadňuje další vývoj.

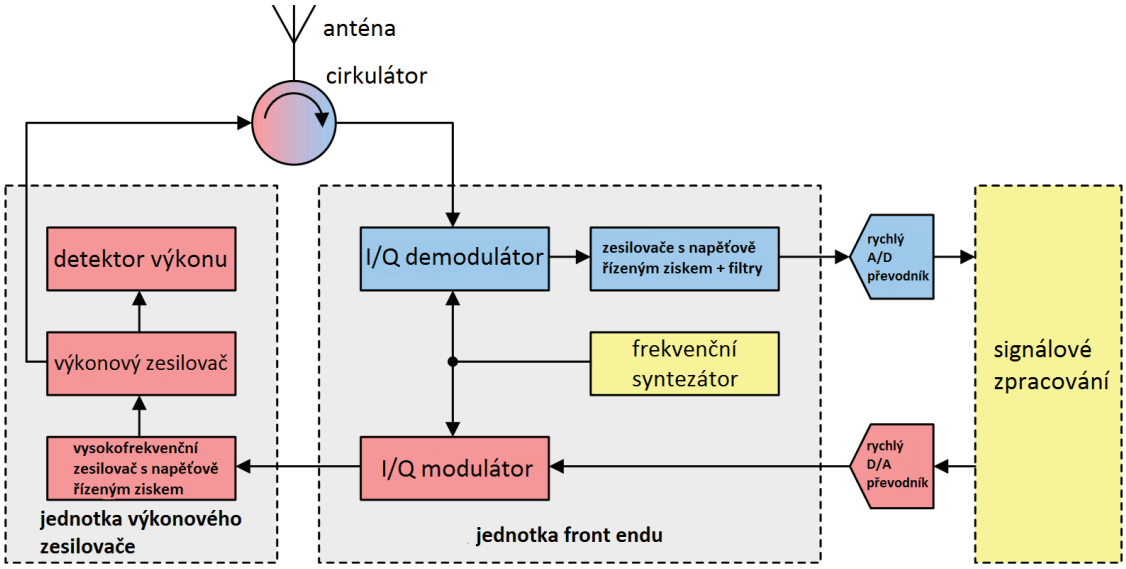
Naopak popisovaný front end dovoluje změnu všech nastavení a zpřístupňuje mnoho vnitřních signálů, které bývají u komerčních čteček nedostupné. Díky přímé I/Q konverzi mezi základním pásmem a radiofrekvenčním pásmem systém vykazuje vysokou linearitu a nízký šum. Protože systém je modulární, je velmi flexibilní a tato koncepce umožňuje snadnou výměnu či nahrazení libovolného bloku.

1.2 Bloková struktura front endu

Blokový diagram systému je na obrázku 1.1. Sestává ze čtyř hlavních částí, jimiž jsou: vysokofrekvenční část (modulátor, demodulátor, frekvenční syntéza atd.), výstupní výkonový zesilovač, cirkulátor a A/D a D/A převodníky následované jednotkou pro zpracování signálu.

Vstupně/výstupní díl má být schopen pracovat v kmitočtovém rozsahu od 860MHz do 960MHz. Výstupní výkon je nastavitelný v rozsahu 10dBm až 33dBm.

A/D a D/A převodníky slouží pro přípravu přijímaných a vysílaných signálů čtečky. Po vynechání D/A převodníku a připojení vstupu modulátoru přímo k mikrokontroléru lze však velice jednoduše generovat základní příkazy RFID čtečky i bez digitálně - analogového převodu.



Obr. 1.1 Blokový diagram UHF RFID front endu (obrázek převzat z [1]).

1.3 Vysokofrekvenční část a výkonový zesilovač

Vysokofrekvenční zpracování signálů zajišťuje integrovaný obvod Analog Devices ADF9010, který v sobě zahrnuje většinu potřebných bloků (viz literatura [1] a [10]). Převod mezi vysokofrekvenčním signálem a I/Q signály v základním pásmu se děje přímo bez konverze na mezifrekvenci, což zajišťuje vysokou linearitu a nízký šum v signálové cestě.

RFID čtečka EXIN-1 je klasifikována jako monostatická (společná anténa pro příjem i vysílání). Proto je pro oddělení přijímaného a vysílaného signálu použit cirkulátor. Parametry cirkulátoru pak spolu se zpětným odrazem antény s_{11} ovlivňují úroveň pronikání vysílaného signálu do vstupu přijímače.

1.4 Bloky společné pro příjem i vysílání

Společnými bloky pro přijímací i vysílací část systému je distribuce hodinového signálu a syntezátor založený na smyčce fázového závěsu (PLL). Hlavním zdrojem hodinového signálu je teplotně kompenzovaný krystalový oscilátor (TCXO) o kmitočtu 40MHz. Použití jediného hodinového signálu zajišťuje koherenci všech hodinových signálů v systému.

Frekvenční syntezátor je součástí ADF9010 a jedná se o LC napětím řízený oscilátor se smyčkou fázového závěsu (PLL) s celočíselným poměrem. Parametry lze nalézt v literatuře [10].

1.5 Vysílací větev systému

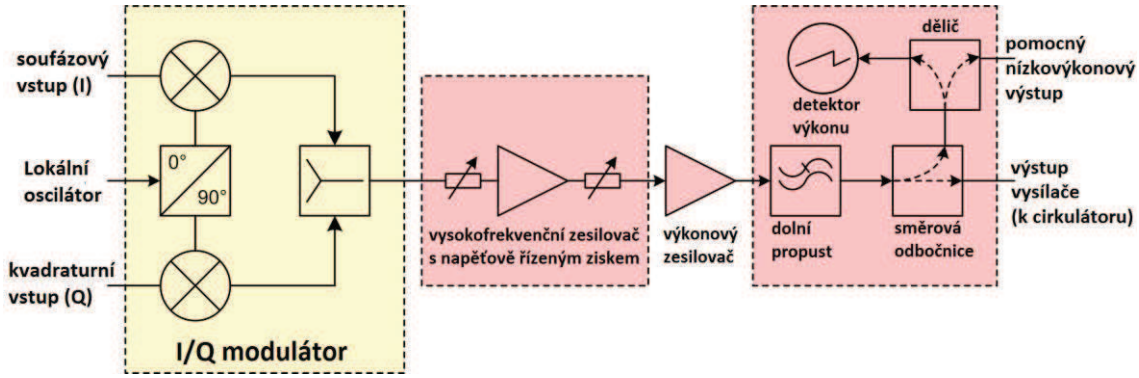
Blok vysílače začíná diferenčním kvadraturním modulátorem následovaným zesilovačem (součást ADF9010). Výstupní úroveň na výstupu zesilovače je 8dBm při

šířce pásma 20MHz.

Modulovaný signál je dále zesílen v zesilovači s proměnným ziskem (VGA), který pak budí výkonový zesilovač tvořený lineárním hybridním modulem PA1162.

Nežádoucí kmitočtové produkty vyšších řádů jsou následně filtrovány dolní propustí a signál je veden do směrové odbočnice a dále do cirkulátoru.

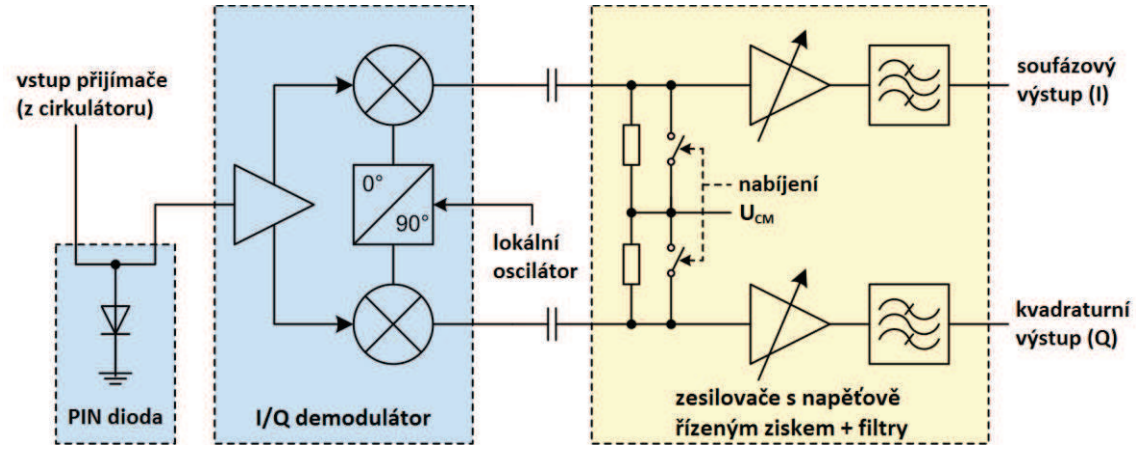
Signál v základním pásmu pro modulátor je obvykle generován vysokorychlostním D/A převodníkem. Pro testování je však možno přímo připojit některý digitální výstup mikrokontroléru na vstup modulátoru. Tento postup představuje jednobitovou konverzi. Takto generovaný signál nesplňuje regulační limity, nicméně pro experimentální měření je postačující. Blokové schéma vysílací části je na obrázku 1.2.



Obr. 1.2 Blokové schéma vysílací větve systému (obrázek převzat z [1]).

1.6 Přijímací větev sytému

Obrázek 1.3 ukazuje blokový diagram přijímací části systému. První součástkou zde je PIN dioda fungující jako omezovač vysokofrekvenčního výkonu.



Obr. 1.3 Blokové schéma přijímací větve systému (obrázek převzat z [1]).

Jako I/Q demodulátor je zvolen obvod ADL5382, vykazující vhodný parametr IP3. Další parametry obvodu lze nalézt v literatuře [11].

V soufázové i kvadraturní větvi se vlivem průniku vysílaného signálu do přijímací cesty objevují velká nenulová stejnosměrná napětí (stejnosměrné posuvy, ofsety). Tato napětí mohou být vyfiltrována použitím střídavé vazby prostřednictvím kondenzátorů. Nevýhodou tohoto řešení je, že při změně činnosti z vysílání na příjem nejsou vazební kondenzátory nabity na správné souhlasné napětí a nabíjejí se příliš pomalu.

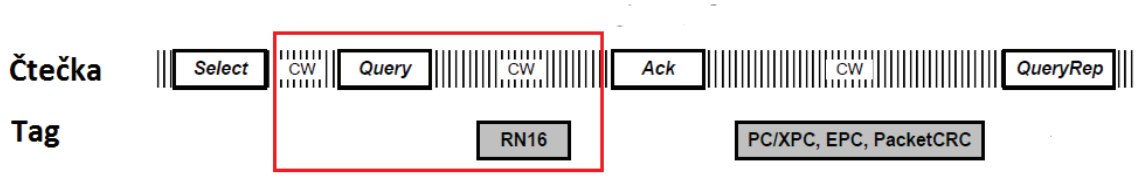
Řešení spočívá v tzv. *boost módu*, který rychle nabije vazební kondenzátory na požadované napětí. Spuštění *boost módu* je třeba provést ručně po každém vyslání signálu.

V základním pásmu je přijatý signál zpracováván dolními propustmi s nastavitelným mezním kmitočtem a zesilovačem s nastavitelným ziskem.

Předpokládá se, že zesílený a vyfiltrovaný signál bude veden na vysokorychlostní A/D převodník a dále zpracován pomocí FPGA nebo mikrokontroléru.

1.7 Testování front endu

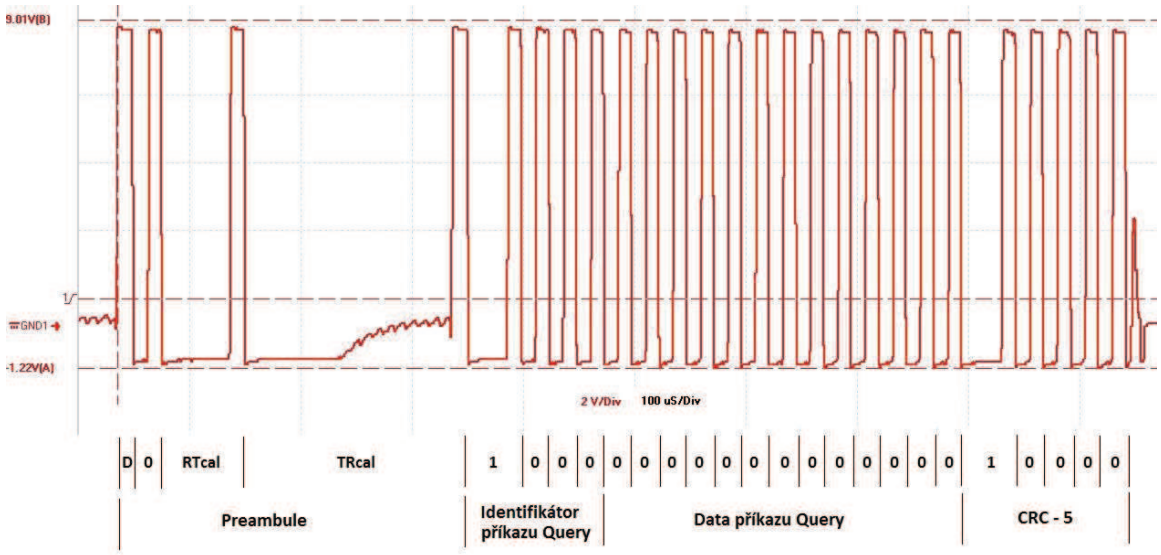
Součástí front endu jsou mimo jiné dva osmibitové mikrokontroléry Atmel ATmega8. Jeden slouží pro nastavení parametrů koncového zesilovače a druhý pro nastavení obvodu ADF9010 a také pro generování základních příkazů protokolu EPC Global Class-1 Gen-2 tak, aby bylo možno ověřit, že lze navázat komunikaci s tagem. Je zde použita již zmíněná jednobitová konverze, kdy digitální výstup mikrokontroléru je přímo připojen na soufázový vstup modulátoru. Testovací program pro mikrokontrolér tedy nastaví čtečku tak, aby na nosné 902MHz periodicky vysílala příkaz *Query*. Mezi jednotlivými vysíláními je časová prodleva, během které je očekáván příjem odpovědi tagu ve formě náhodného 16-bitového čísla *RN16* a která je důležitá také proto, aby vypršel v normě specifikovaný časový interval (*timeout*) a příkaz *Query* mohl být zopakován. Na obrázku 1.4 je tato opakující se sekvence označena červeným obdélníkem. Odpověď tagu je pak možno v základním pásmu za přijímačem zobrazit na osciloskopu. Protože výstup demodulátoru je pro soufázovou i kvadraturní složku diferenční, je pro každou větev předřazen diferenční zesilovač s desetinásobným zesílením, který převede diferenční výstup demodulátoru na výstup vztažený proti zemi. Na osciloskopu lze pak zobrazit dva signály, a to soufázovou i kvadraturní složku signálu, obě vztažené proti zemi systému. Nastavená konfigurace parametrů komunikace je v tabulce 1.1. Na obrázku 1.5 pak lze vidět příkaz *Query* získaný z přijatého signálu, protože vysílaný signál proniká přes cirkulátor na vstup přijímače. Odpověď tagu ve formě čísla *RN16* zobrazuje obrázek 1.6. Bližší výklad k protokolu je součástí kapitoly 2 této práce.



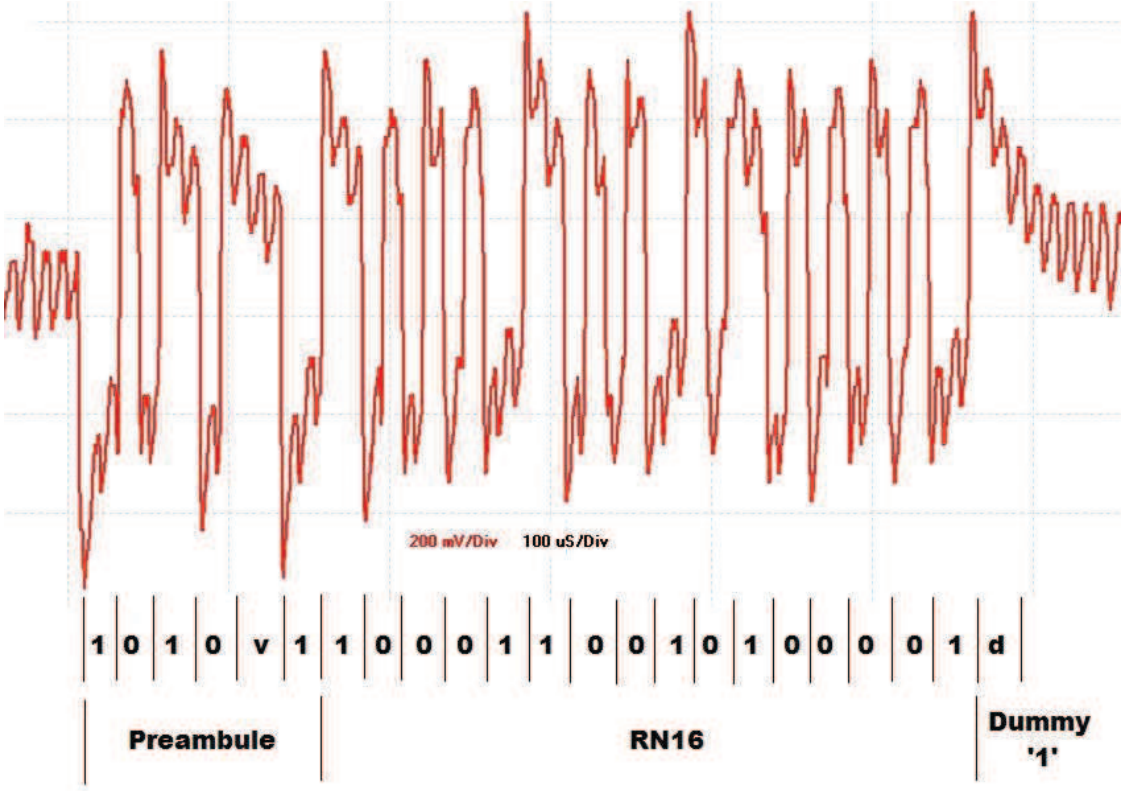
Obr. 1.4 EPC Class1 Gen2 komunikační proces s *Query* příkazem (obrázek převzat z [2]).

Tab. 1.1 Nastavení parametrů komunikace pomocí příkazu Query a jeho struktura.

Data	**Parametr**	**Popis**
Preambule	T_{ari}=25µs	Časová délka symbolu T_{ari}
	RT_{cal}=75 µs	R→T kalibrace
	TR_{cal}=200 µs	T→R kalibrace
1000	**Query**	Identifikátor příkazu
0	DR=8	BLF=40kHz (nastavuje se spolu s TR_{cal})
00	M=1	Kódování FM0
0	TR_{ext}=0	Bez pilotního tónu
00	Sel=All	Netestovat SL flag
00	Session=S0	Nastavit příznak S0 v populaci tagů A
0	Target=A	Vybrat populaci tagů A
0000	Q=1	Jen 1 slot v *Inventory round*
10000	CRC-5	CRC-5 nad příkazem Query



Obr. 1.5 Časový průběh příkazu Query – invertovaný.



Obr. 1.6 Časový průběh odezvy tagu RN16 – invertovaný.

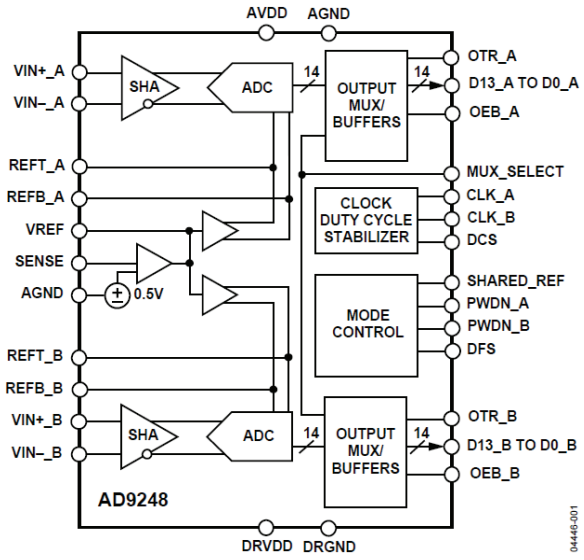
1.8 Možnosti A/D převodu

Součástí prototypu čtečky jsou také dvě desky s analogově-digitálními převodníky. V obou případech se jedná o 14-bitové převodníky se dvěma diferenčními analogovými vstupy a dvěma paralelními digitálními výstupy.

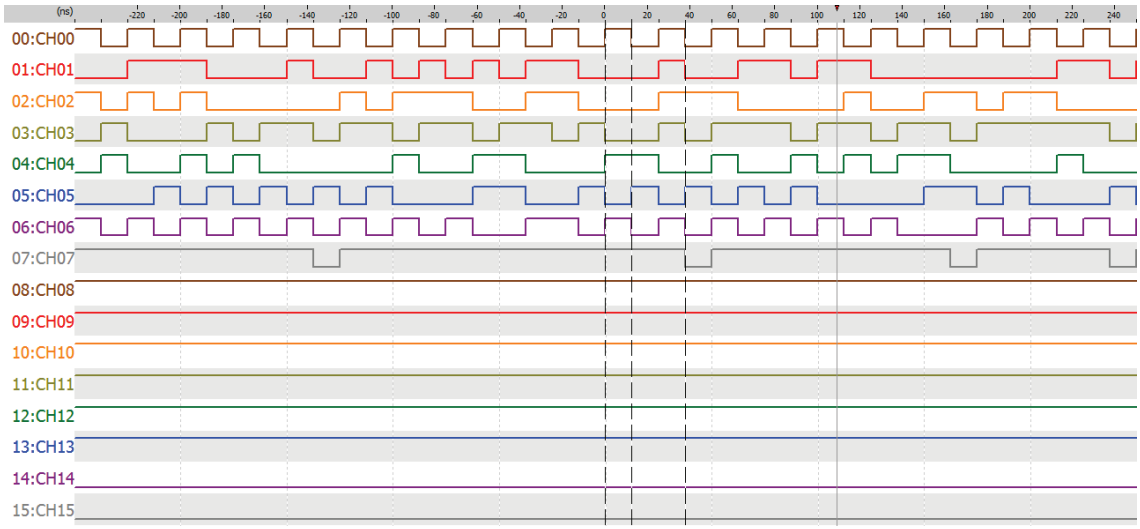
1.8.1 A/D převodník AD9248

Prvním ze zmíněných převodníků je analogově – číslicový převodník Analog Devices AD9248. Jeho analogová i digitální část je napájena napětím 3,3V. Taktován je z centrálního hodinového signálu 40MHz. Na výstupu je doplněn budiči 74LVC541, které mají oddělené napájení přivedené z vnějšku, aby byly logické úrovně přizpůsobeny navazujícímu mikrokontroléru nebo obvodu FPGA. Převodník má řídící vstup MUX_SELECT (viz obrázek 1.7) připojen přímo na hodinový signál, což má za následek, že výstup je prokládaný. To znamená, že na digitální výstup A jsou na nástupnou hranu hodinového signálu přivedena data vzniklá převedením analogového kanálu A a na sestupnou hranu jsou přivedena data vzniklá převedením analogového kanálu B. Zapojení řídícího vstupu pro výběr napěťové reference VREF je takové, že umožňuje zpracovávat vstupní signál (mezi diferenčními vstupy) o rozkmitu 1V. Poslední řídící vstup DFS je připojen na logickou úroveň „0“, takže výstupní data jsou v přímém offsetovém binárním formátu. Časový průběh výstupních dat pro nezapojený vstup (takže na vstupu je jen šum z okolí) je na obrázku 1.8. Kanál CH00 je hodinový signál, zbytek jsou jednotlivé datové bity v pořadí od LSB do MSB. Kanál CH15 je

signál značící překročení vstupního rozsahu převodníku (přetečení/podtečení).



Obr. 1.7 Blokové schéma vnitřní struktury převodníku AD9248 (obrázek převzat z [4]).



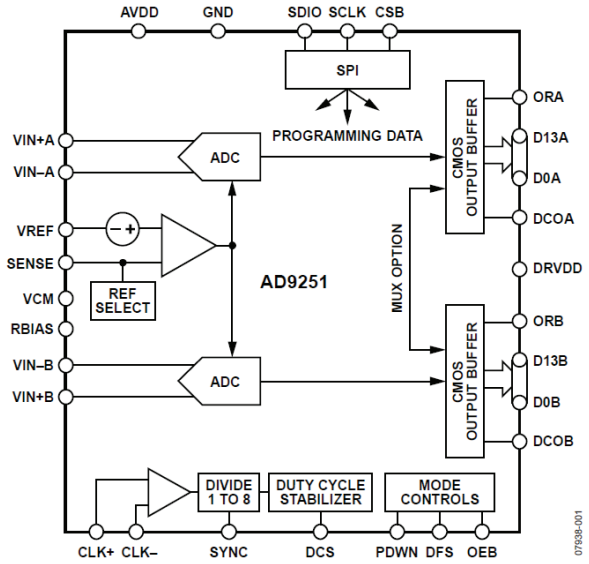
Obr. 1.8 Časové průběhy výstupních dat z převodníku AD9248.

1.8.2 A/D převodník AD9251

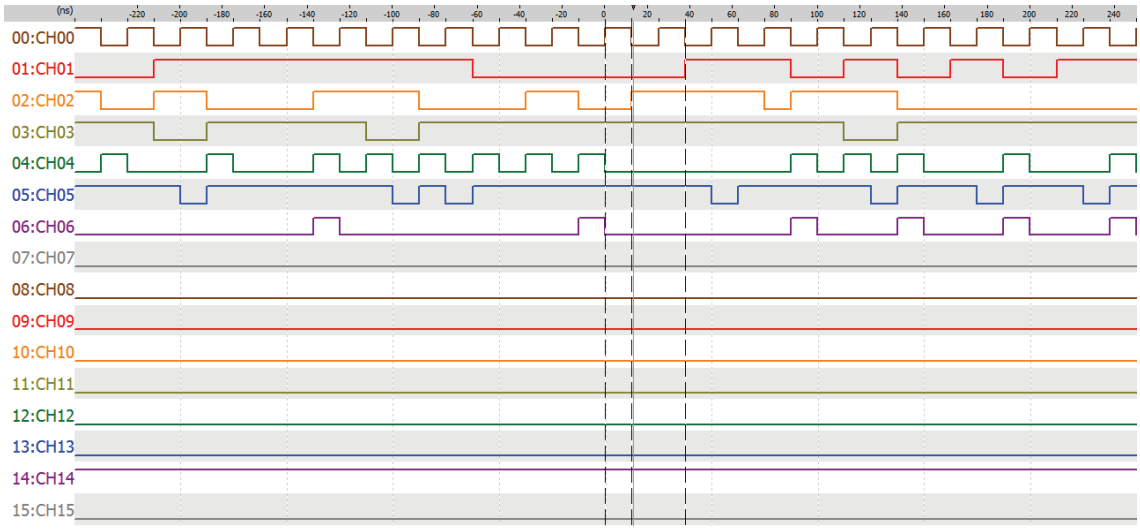
Druhým převodníkem je 14-bitový analogově – číslicový převodník Analog Devices AD9251. Je velmi podobný předchozímu převodníku, je i podobně zapojen. Rozdíl je v tom, že jeho analogová část je napájena napětím 1,8V. Napěťová reference je zapojena tak, že rozkmit vstupního napětí mezi diferenčními vstupy může být 2V. Výstup není ve výchozím nastavení prokládaný, takže na výstupu jsou k dispozici pouze data vzniklá převedením analogového kanálu A. Taktování je opět hodinovým signálem o kmitočtu 40MHz.

Hlavním rozdílem oproti převodníku AD9248 je, že tento převodník může být řízen po sběrnici SPI, která je mj. jediným způsobem, jak dosáhnout prokládání výstupních dat,

nebo jak změnit výstupní datový formát, který je na počátku vždy nastaven na přímý binární offsetový kód. Více informací lze nalézt v literatuře [5]. Na obrázku 1.9 je blokově naznačeno vnitřní uspořádání převodníku. Na obrázku 1.10 jsou pak časové průběhy výstupních dat v neprokládaném módu. Na vstupu převodníku je opět pouze šum pronikající z okolí. Organizace průběhů je opět taková, že kanál CH00 je hodinový signál, zbytek jsou jednotlivé datové bity v pořadí od LSB do MSB. Kanál CH15 je signál značící překročení vstupního rozsahu převodníku (přetečení/podtečení).



Obr. 1.9 Blokové schéma vnitřní struktury převodníku AD9251 (obrázek převzat z [5]).

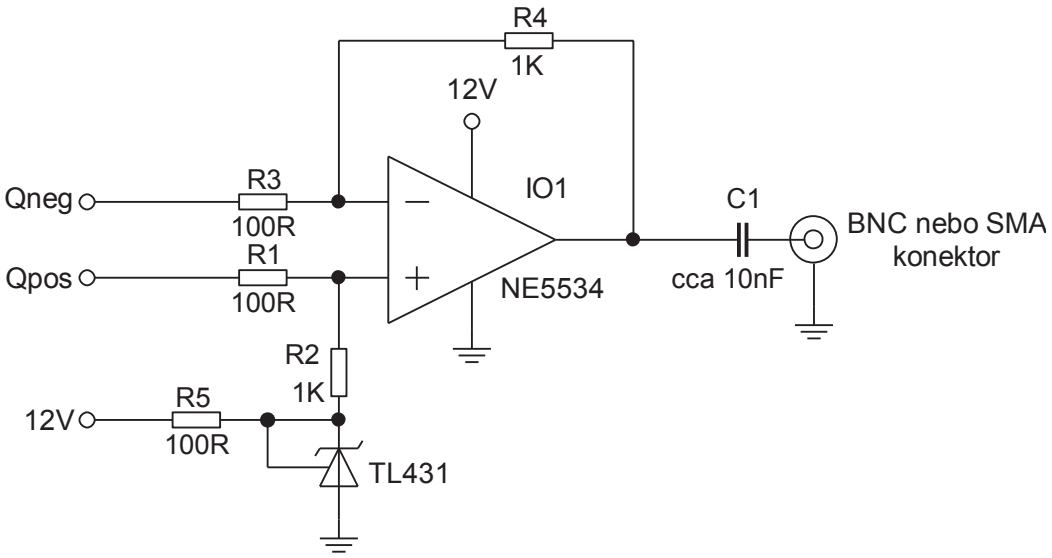


Obr. 1.10 Časové průběhy výstupních dat z převodníku AD9251.

1.8.3 Vnitřní A/D převodník mikrokontroléru STM32F429

Poslední zvažovanou možností analogově–digitálního převodu je využití vnitřního 12-bitového A/D převodníku mikrokontroléru STM32F429. Tento mikrokontrolér má

celkem tři tyto převodníky, takže je možné využít dva z nich, jeden pro složku soufázovou a druhý pro složku kvadraturní. Nevýhodou je, že tyto převodníky nemají diferenční vstupy. Protože však výstupem demodulátoru jsou signály I_P a I_N pro soufázovou složku a signály Q_P a Q_N pro složku kvadraturní, je třeba si vždy dva signály pro danou složku sloučit do signálu jednoho, který bude vztažený proti zemi systému. To lze udělat např. rozdílovým zesilovačem dle obrázku 1.11. Tyto zesilovače zde musejí být dva, každý pro jednu složku. Reference TL431 zvyšuje souhlasné napětí vstupních signálů na přibližně 2,5V, aby tyto signály, které mohou jinak nabývat kladné i záporné polarity, mohly být bez zkreslení zpracovány, aniž by bylo nutné použít pro zesilovač symetrické napájení. Posun stejnosměrné úrovně signálů je nutný také proto, aby výstupní signál zesilovače mohl být zpracován vnitřním A/D převodníkem navazujícího mikrokontroléru, který je schopen zpracovávat pouze napětí od 0V do napájecího napětí mikrokontroléru a není tudíž schopen zpracovat záporné napětí.



Obr. 1.11 Diferenční zesilovač pro převod diferenčního signálu na signál vztažený proti zemi (zde pro kvadraturní složku).

1.9 Volba obvodu pro zpracování signálu

Jednou ze základních otázek je, zda pro zpracování signálu v základním pásmu použít mikrokontrolér nebo obvod FPGA. Po zvážení obou možností bylo vybráno použití mikrokontroléru. Konkrétněji byla zvolena vývojová deska s mikrokontrolérem s jádrem ARM Cortex-M4, protože se ukázalo, že jeho výpočetní výkon by měl být pro danou aplikaci dostačující a navíc byla zvolená vývojová deska snadno dostupná.

Obvod FPGA má naopak nevýhodu v tom, že práce s ním je složitější a vývojové desky s FPGA jsou hůře dostupné. Navíc běžně dostupné obvody FPGA nemívají integrovaný analogově-digitální převodník, takže by bylo třeba použít převodník externí, čímž by se řešení zadání práce zkomplikovalo. Výhodou obvodů FPGA je na druhou stranu jejich vyšší výpočetní výkon, protože celý kód (program) je vykonáván prakticky současně na rozdíl od sekvenčního vykonávání u mikrokontrolérů.

2 STANDARD EPC GLOBAL CLASS-1 GENERATION-2

Tento standard specifikuje fyzickou interakci (signalizaci a komunikaci) čtečky a RFID etikety (tagu) v UHF pásmu 860MHz – 960MHz, jejich provozní procedury a příkazy a také antikolizní schémata používaná pro identifikaci konkrétního tagu v prostředí s mnoha tagy. Celá kapitola čerpá z literatury [2] a [3], kde je také možno nastudovat podrobnější informace.

Čtečka posílá informace tagům modulované na vysokofrekvenční nosné s použitím amplitudového klíčování s oběma postranními pásmy (DSB-ASK), amplitudového klíčování s jedním postranním pásmem (SSB-ASK) nebo amplitudového klíčování s reverzí fáze (PR-ASK), a to s užitím pulzně-intervalového kódování (PIE). Tagy získávají veškerou energii pro svou činnost právě z tohoto vysokofrekvenčního modulovaného signálu.

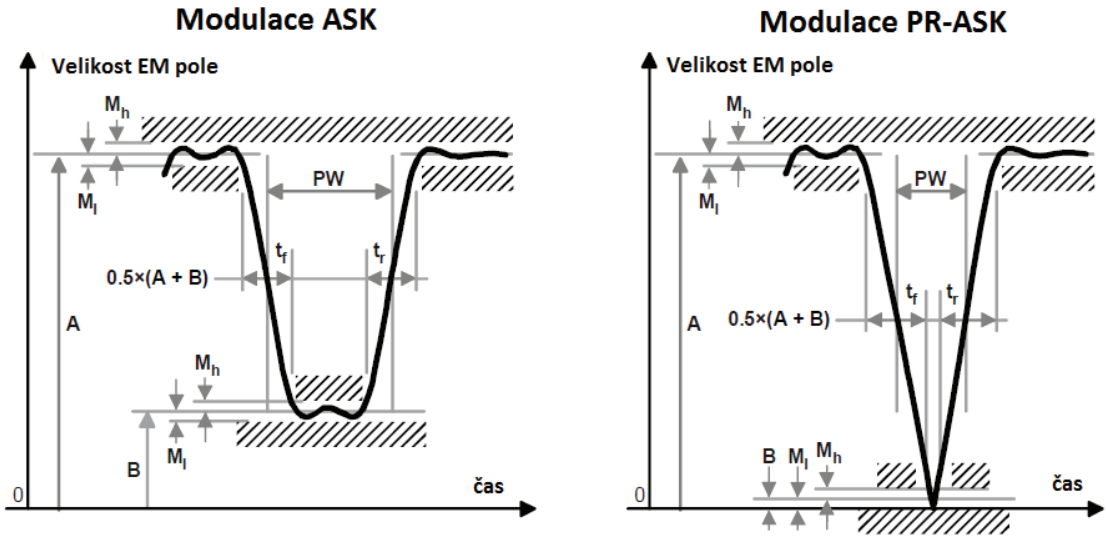
Příjem informací od tagu je řešen tak, že čtečka vysílá nemodulovanou nosnou a naslouchá, zda přijde odražená modulovaná vlna. Tag totiž používá amplitudovou a/nebo fázovou modulaci řešenou přizpůsobováním své antény. Pokud je anténa nepřizpůsobená, většina energie nemodulované vlny vyslané čtečkou se odrazí ke čtečce zpět, zatímco v téměř přizpůsobeném stavu antény se odráží jen menší část energie vlny. Data vysílaná tagem jsou kódována pomocí kódování FM0 nebo pomocí Millerovými funkcemi modulované subnosné (*Miller-modulated subcarrier*). Komunikace mezi tagem a čtečkou a opačně probíhá v polovičním duplexu, takže buď vysílá pouze tag (tagy) nebo pouze čtečka. Tag proto nemusí reagovat na příkazy, které přijdou v době, kdy tag vysílá.

2.1 Komunikace čtečka→tag

Tato podkapitola se zabývá požadavky na signály vysílané ve směru od čtečky k tagu. Jedná se především o požadavky na modulaci, kódování dat v základním pásmu a frekvenční spektrum signálů vysílaných čtečkou.

2.1.1 Modulace

Čtečka komunikuje s tagem (resp. tagy) pomocí vysokofrekvenční nosné s modulací DSB-ASK, SSB-ASK nebo PR-ASK a s pulsně-intervalovým kódováním (PIE). V rámci jedné „inventarizační obrátky“ (*Inventory round*) používá čtečka stejnou modulaci a datovou rychlost. Datová rychlost se nastavuje pomocí časových intervalů v preambuli na začátku obrátky. Na obrázku 2.1 vidíme tvar modulační obálky pro modulaci ASK i PR-ASK. Význam jednotlivých parametrů na obrázku je vysvětlen v tabulce 2.1. Úroveň A koresponduje s maximální amplitudou vysokofrekvenční nosné vysílané čtečkou, úroveň B odpovídá minimální amplitudě (utlumené vlně). Přesnost kmitočtu nosné musí být lepší než ±0,01‰ (±10ppm) v nominálním teplotním rozsahu (-25°C až 40°C), pokud místní regulační předpisy nestanoví přísnější požadavek.



Obr. 2.1 Modulační obálky pro modulaci ASK a PR-ASK (obrázek převzat z [2]).

Tab. 2.1 Parametry modulačních obálek (převzato z [2]).

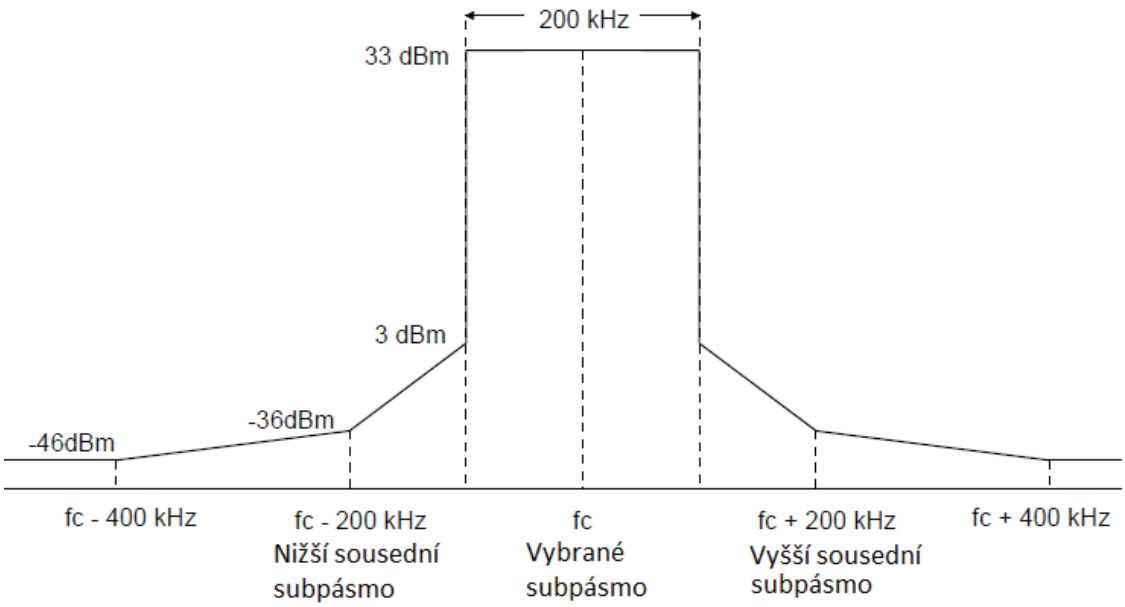
Tari	Parametr	Označení	Minimální hodnota	Nominální hodnota	Maximální hodnota	Jednotka
6,25µs až 25µs	Modulační hloubka	(A-B)/A	80	90	100	%
	Zvlnění obálky	M_h=M_I	0		0,05(A-B)	V/m nebo A/m
	Doba nástupné hrany obálky	$t_{r,10\text{-}90\%}$	0		0,33Tari	µs
	Doba sestupné hrany obálky	$t_{f,10\text{-}90\%}$	0		0,33Tari	µs
	Šířka pulzu	PW	0,265Tari nebo 2µs, dle toho co je větší		0,525Tari	µs

2.1.2 Spektrální maska

Aby čtečka splňovala regulační požadavky na hospodaření s kmitočtovým spektrem, musí vysílané signály splňovat spektrální masku, která říká, jaká může být maximální úroveň výkonu pronikajícího z kanálu, ve kterém čtečka vysílá, do kanálů ostatních.

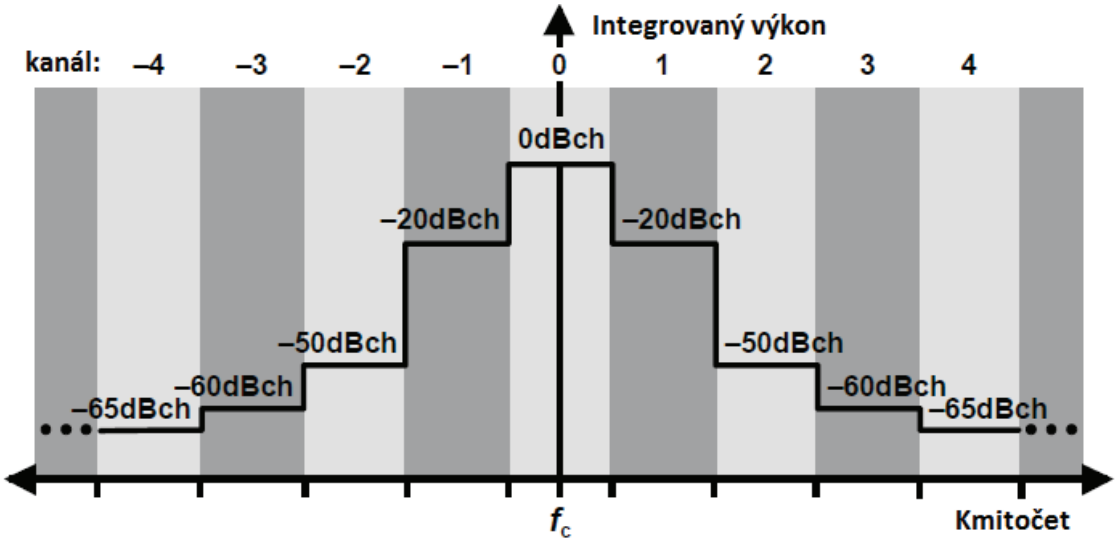
V první řadě je třeba dodržet požadavky normy ETSI EN 302 208, která udává spektrální masku dle obrázku 2.2 a která stanovuje povolené úrovně výkonů pronikajících z kanálu, ve kterém čtečka vysílá do kanálů sousedních. Požadované výkonové úrovně jsou patrné z obrázku a jedná se o efektivně vyzářený výkon (*Effective Radiated Power*-ERP). Tato norma také udává dovolené vysílací výkony pro kanál, ve kterém čtečka pracuje. Jsou podmíněny šířkou hlavního laloku vyzařovacího svazku antény a jsou následující:

- méně nebo rovno 500mW ERP bez omezení šířky svazku.
- 500mW ERP až 1000mW ERP včetně pro šířku svazku menší nebo rovnu 180°.
- do 2000mW ERP včetně pro šířku svazku menší nebo rovnu 90°.

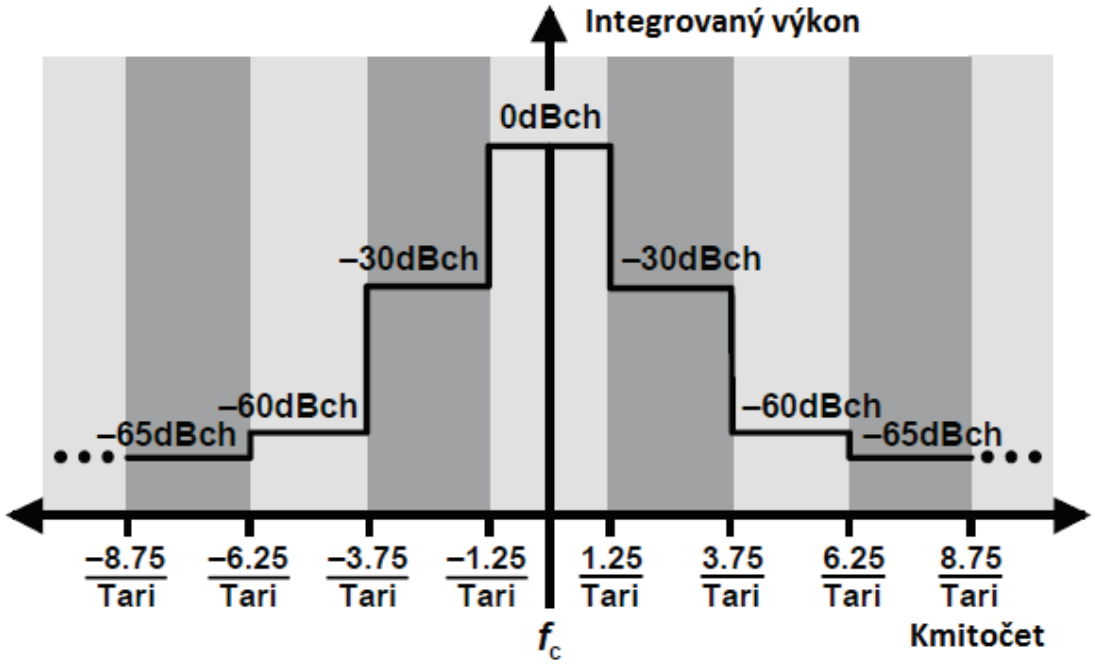


Obr. 2.2 Spektrální maska pro modulované signály (převzato z [11]).

Standard EPC Global Class-1 Gen-2 pak specifikuje dva případy spektrální masky. Prvním případem je maska pro prostředí s více čtečkami (*Multiple-Interrogator Transmit Mask*, prostředí, kde je více čteček, avšak méně než je dostupných kanálů), která je zobrazena na obrázku 2.3. Druhým případem je maska pro prostředí s plným obsazením kanálů (*Dense-Interrogator Transmit Mask*, prostředí, kde je stejný počet čteček jako dostupných kanálů), která je zobrazena na obrázku 2.4. Jednotka dBch je decibelová jednotka, kde je úroveň výkonu v kanálu S vztažena k úrovni výkonu v referenčním kanálu R.



Obr. 2.3 Spektrální maska pro prostředí s více čtečkami (obrázek převzat z [2]).

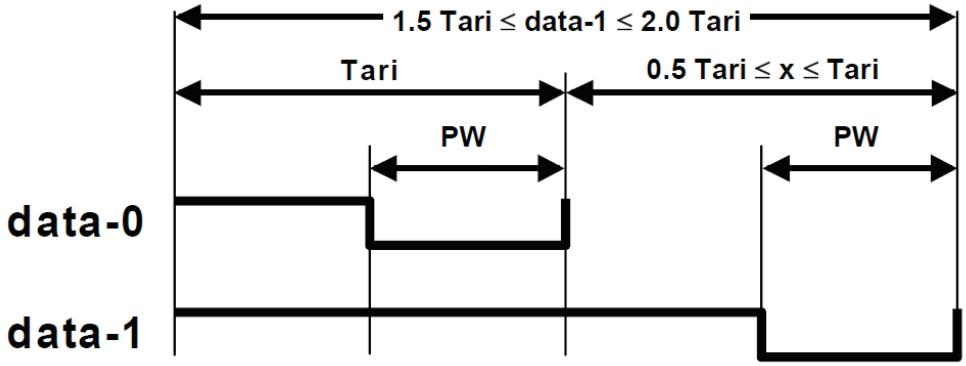


Obr. 2.4 Spektrální maska pro prostředí s plným obsazením kanálů (obrázek převzat z [2]).

Výkonové úrovně, které je třeba dodržet, jsou opět patrné z obrázků. Veličina *T*ari ve druhém obrázku je referenční časový interval (*Type A Reference Interval*).

2.1.3 Kódování dat

Ve směru vysílání od čtečky k tagu se používá pulzně-intervalové kódování dle obrázku 2.5, kde je vidět vyjádření datového symbolu „0“ a datového symbolu „1“. Základní časovou jednotkou je *T*ari, což je délka symbolu „0“ a může být nastavena v rozsahu 6,25µs až 25µs. Na obrázku 2.5 vysoká úroveň reprezentuje vysílání netlumené vysokofrekvenční nosné, nízká úroveň reprezentuje utlumenou nosnou.



Obr. 2.5 Symboly pro pulzně-intervalové kódování-PIE (obrázek převzat z [2]).

Synchronizace časových parametrů datového přenosu se děje prostřednictvím R→T preambule nebo synchronizačního rámce (viz obrázky 2.6 a 2.7). Preambule se vysílá na začátku inventarizační obrátky před příkazem *Query*, synchronizační rámec před ostatními příkazy.

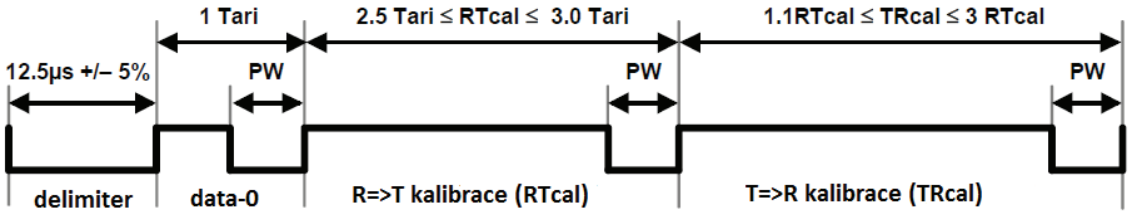
Preambule začíná delimiterem o fixní délce 12,5µs a datovým symbolem „0“ a pokračuje kalibračním symbolem *RTcal*, jehož délka je rovna součtu délek datových symbolů „0“ a „1“. Tag si změří délku *RTcal* a vydělí ji dvěma. Vše, co je kratší než *RTcal*/2, pak považuje za datový symbol „0“ a vše, co je delší než *RTcal*/2, za datový symbol „1“. Na konci preambule je kalibrační symbol *TRcal*, pomocí jehož délky a údaje o dělicím poměru zapsaném v příkazu *Query* je specifikována BLF (*Backscatter Link Frequency*, datová rychlost ve směru od tagu ke čtečce) dle rovnice:

$$BLF = \frac{DR}{TRcal} \qquad [\text{Hz; - , s}] \quad (2.1)$$

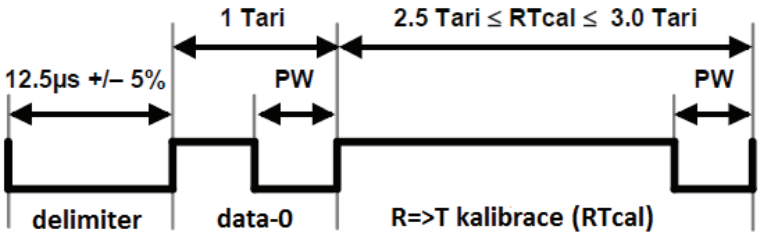
kde DR je dělicí poměr (viz tabulka 2.2) a kde délka *TRcal* je:

$$1{,}1 \cdot RTcal \le TRcal \le 3 \cdot RTcal \qquad [\mu\text{s}] \quad (2.2)$$

Synchronizační rámec má strukturu stejnou, pouze chybí kalibrační symbol *TRcal*.



Obr. 2.6 Preambule pro komunikaci ve směru od čtečky k tagu (obrázek převzat z [2]).



Obr. 2.7 Synchronizační rámec pro komunikaci ve směru od čtečky k tagu (obrázek převzat z [2]).

2.2 Komunikace tag→čtečka

Tag komunikuje se čtečkou pomocí modulace odražené vlny, což je způsob, při kterém tag přepíná činitel odrazu své antény mezi dvěma stavy. Nízká úroveň v obrázcích 2.8 až 2.12 a v obrázcích 2.14 až 2.16 je reprezentována stavem, kdy tag pohlcuje nemodulovanou nosnou vlnu vysílanou čtečkou, vysoká úroveň je reprezentována stavem, kdy tag nosnou vlnu odráží.

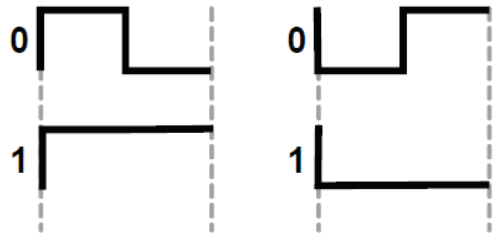
2.2.1 Modulace

Tag používá buď amplitudové klíčování (ASK) nebo fázové klíčování (PSK). Typ modulace určuje výrobce tagu a čtečka musí umět demodulovat oba typy.

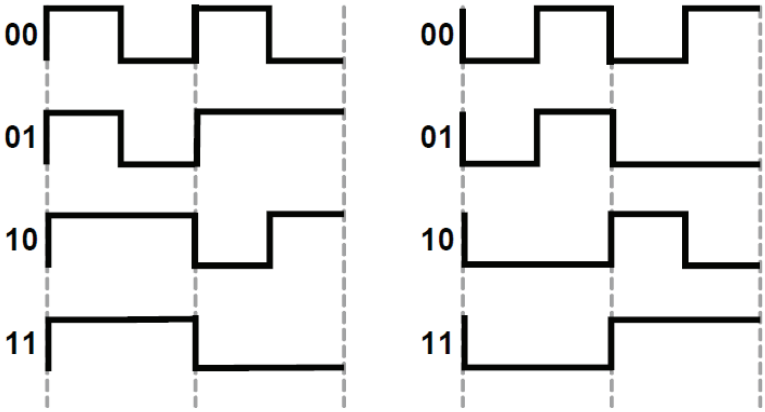
2.2.2 Kódování dat

Data vysílaná tagem jsou kódována buď pomocí kódu FM0, nebo pomocí Millerovými funkcemi modulované subnosné (*Miller-modulated subcarrier*). O formátu dat rozhoduje čtečka.

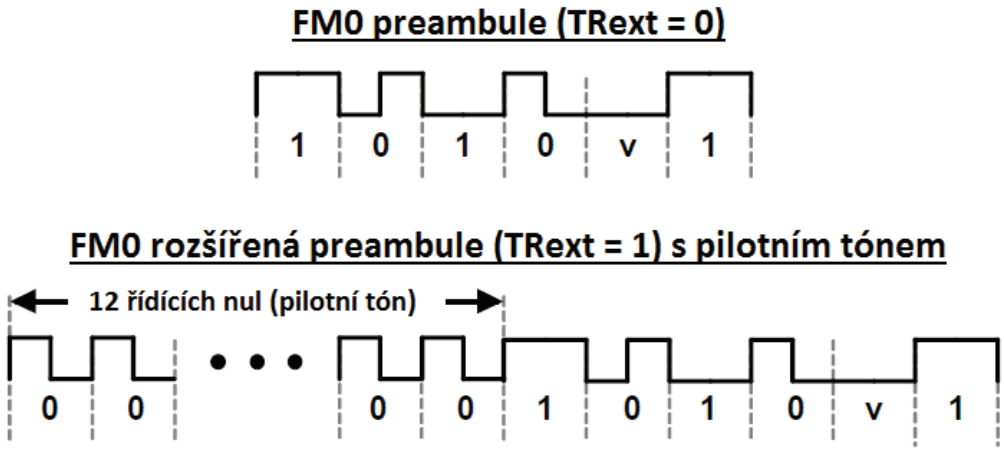
Nejjednodušším kódováním je kódování FM0, které využívá takové vyjádření datových symbolů, při kterém je symbol „0“ reprezentován přechodem mezi úrovněmi uprostřed symbolové periody. U symbolu „1“ dochází k přechodu mezi úrovněmi na konci symbolové periody. Symboly jsou čtyři možné (viz obrázek 2.8). To, který ze dvou symbolů pro „1“ nebo „0“ se použije, závisí na tom, v jaké úrovni skončil předchozí symbol, což vysvětluje obrázek 2.9. Komunikace tagu se čtečkou s kódováním FM0 začíná jednou ze dvou možných preambulí z obrázku 2.10. Výběr záleží na parametru *TRext* v příkazu *Query* vysílaném čtečkou na začátku inventarizační obrátky. V případě, že *TRext*=0, se nepoužívá pilotní tón a použije se první verze preambule, v případě, že *TRext*=1, se pilotní tón použije a preambule je druhého typu. Speciální symbol „v“ v preambuli má význam fázové chyby, protože správně by mělo dojít ke změně fáze signálu, což se ale nestane. Komunikace končí symbolem *„dummy 1“,* tedy jedničkou navíc, jak zachycuje obrázek 2.11.



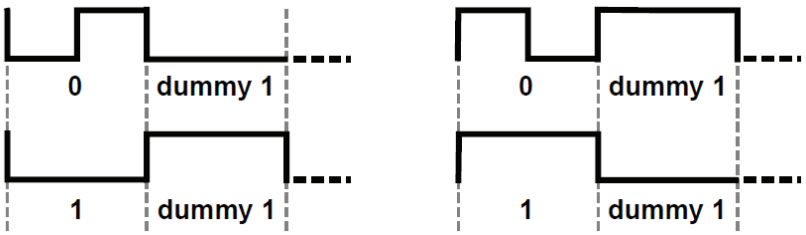
Obr. 2.8 Možné symboly kódu FM0 (obrázek převzat z [2]).



Obr. 2.9 Sekvence symbolů kódu FM0 (obrázek převzat z [2]).

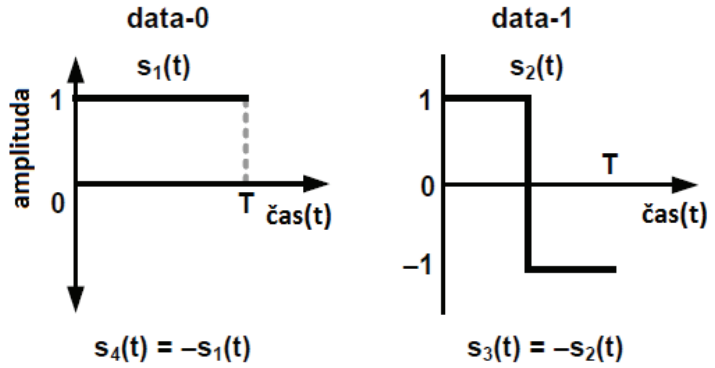


Obr. 2.10 Preambule používané při kódování FM0 (obrázek převzat z [2]).

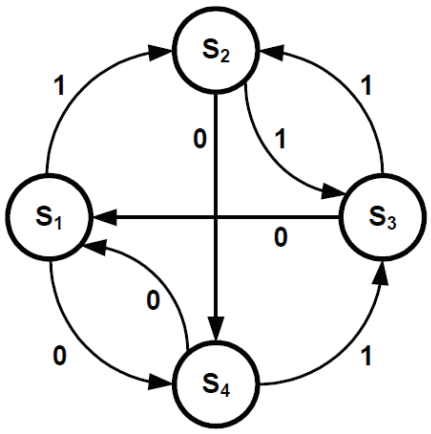


Obr. 2.11 Konec signalizace v kódování FM0 (obrázek převzat z [2]).

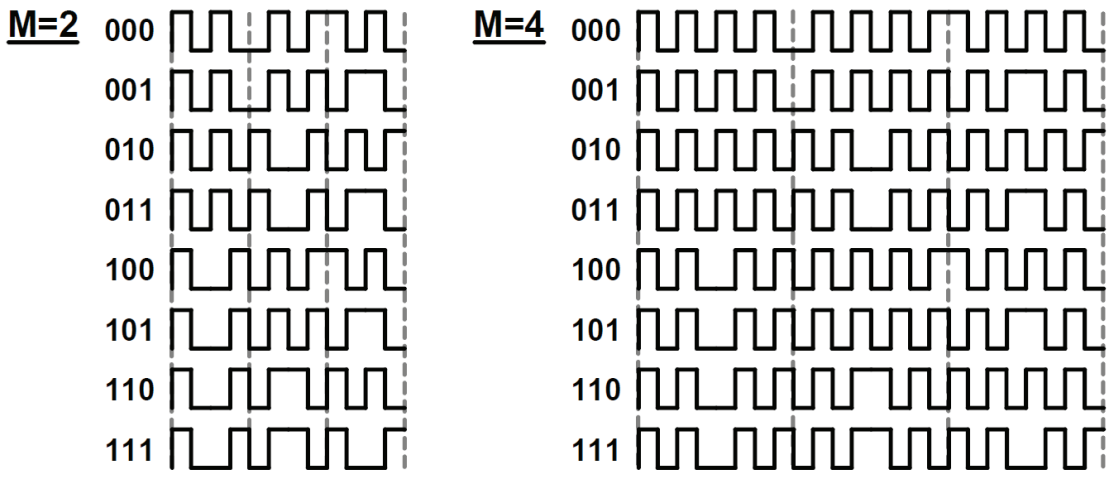
Při kódování pomocí Millerovými funkcemi modulované subnosné opět existují dvě základní Millerovy funkce pro symboly „0“ a „1“ a další dvě, které jsou jejich inverzí (viz obrázek 2.12). Při přechodu mezi dvěma symboly „0“ se mění fáze symbolů. U symbolů „1“ dochází ke změně fáze uprostřed symbolové periody. Tyto základní Millerovy funkce jsou následně vynásobeny obdélníkovou vlnou (subnosnou) o M násobku symbolové rychlosti, a tak vzniká výsledná vysílaná posloupnost. Ze stavového diagramu na obrázku 2.13 je patrné, že přechody mezi některými stavy nejsou možné, protože by docházelo ke změně fáze na přechodu mezi symboly „0“ a „1“ a mezi symboly „1“ a „0“. Obrázek 2.14 ukazuje sekvence Millerovými funkcemi modulované subnosné pro dva a čtyři cykly subnosné na jeden Millerův datový symbol. Podobně jako u kódování FM0 jsou na výběr dvě možnosti preambule (bez pilotního tónu/ s pilotním tónem – viz obrázek 2.15). Rozdíl je samozřejmě v jiném formátu, kdy preambule je uvozena čtyřmi (bez pilotního tónu) nebo šestnácti (s pilotním tónem) symbolovými periodami, v rámci kterých je vysílána pouze subnosná. Datová část preambule pak nese jiná data (jiné datové symboly). Konec komunikace je opět signalizován symbolem *„dummy 1“* (viz obrázek 2.16).



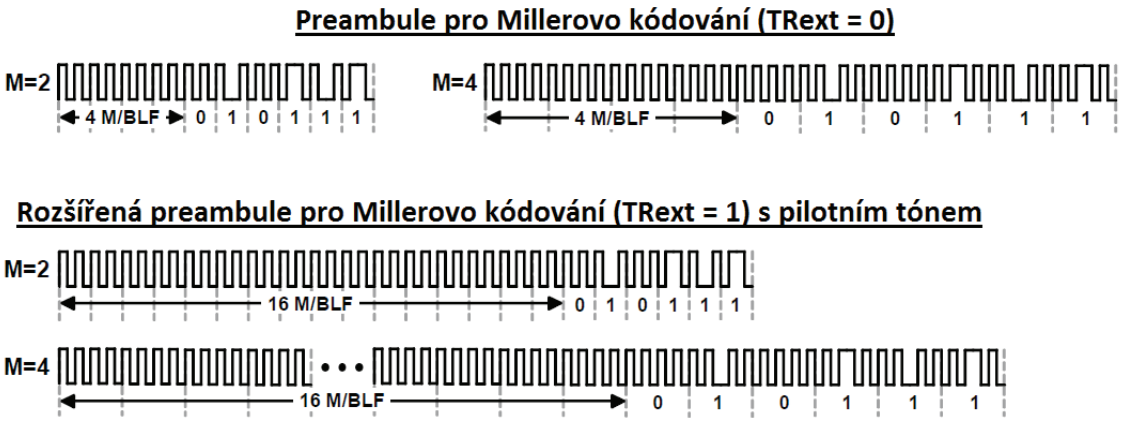
Obr. 2.12 Základní Millerovy funkce (obrázek převzat z [2]).



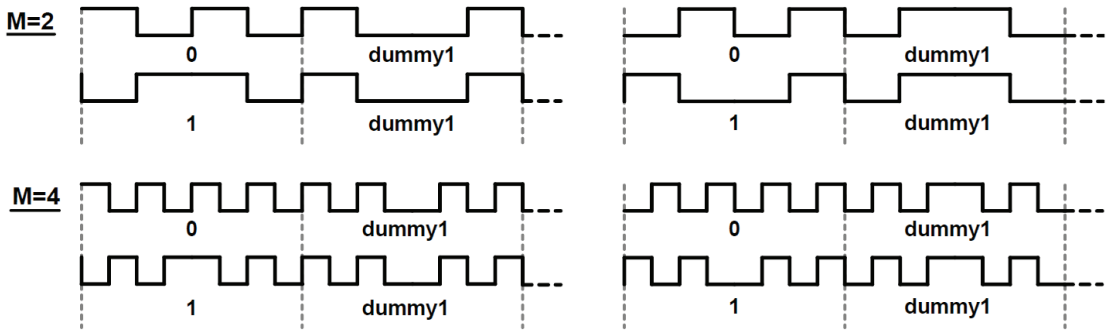
Obr. 2.13 Stavový diagram kódování pomocí Millerových funkcí (obrázek převzat z [2]).



Obr. 2.14 Sekvence Millerovými funkcemi modulované subnosné pro různé posloupnosti dat a pro dva různé počty period subnosné na jeden datový symbol (obrázek převzat z [2]).



Obr. 2.15 Preambule pro kódování pomocí Millerovými funkcemi modulované subnosné pro M=2 a 4 (obrázek převzat z [2]).



Obr. 2.16 Konec signalizace pro kódování pomocí Millerovými funkcemi modulované subnosné M=2 a 4 (obrázek převzat z [2]).

Co se týče pořadí, ve kterém jsou přenášeny jednotlivé bity, je třeba poznamenat, že nejvýznamnější bit (MSB) je přenášen první, a to platí uvnitř každé zprávy i uvnitř

každého datového slova. Tabulka 2.2 pak zachycuje příklady možných komunikačních rychlostí pro směr tag→čtečka (BLF) a příslušnou délku intervalu *TRcal* a dělicí poměry pro jejich nastavení.

Tab. 2.2 Tabulka možných komunikačních rychlostí BLF (převzato z [3]).

Dělicí poměr	**TRcal [µs]**	**BLF [kHz]**	**T_{pri} = 1/BLF [µs]**
64/3	33	640	1,6
	67	320	3,1
	83	256	3,9
	225	95	10,5
8	17	465	2,1
	25	320	3,1
	50	160	6,3
	200	40	25

2.2.3 Zabezpečení dat proti chybám

Po kontrolu správnosti přijatých dat se používá cyklický redundantní kód (*Cyclic Redundancy Check – CRC*), konkrétně o délce 16 bitů pro data přenášená ve směru od tagu ke čtečce a pět bitů pro data přenášená ve směru od čtečky k tagu.

CRC-16 je popsán polynomem:

$$x^{16} + x^{12} + x^5 + 1 \quad [-] \quad (2.3)$$

CRC-5 je popsán polynomem:

$$x^5 + x^3 + 1 \quad [-] \quad (2.4)$$

2.3 Výběr, inventarizace a přístup k tagu

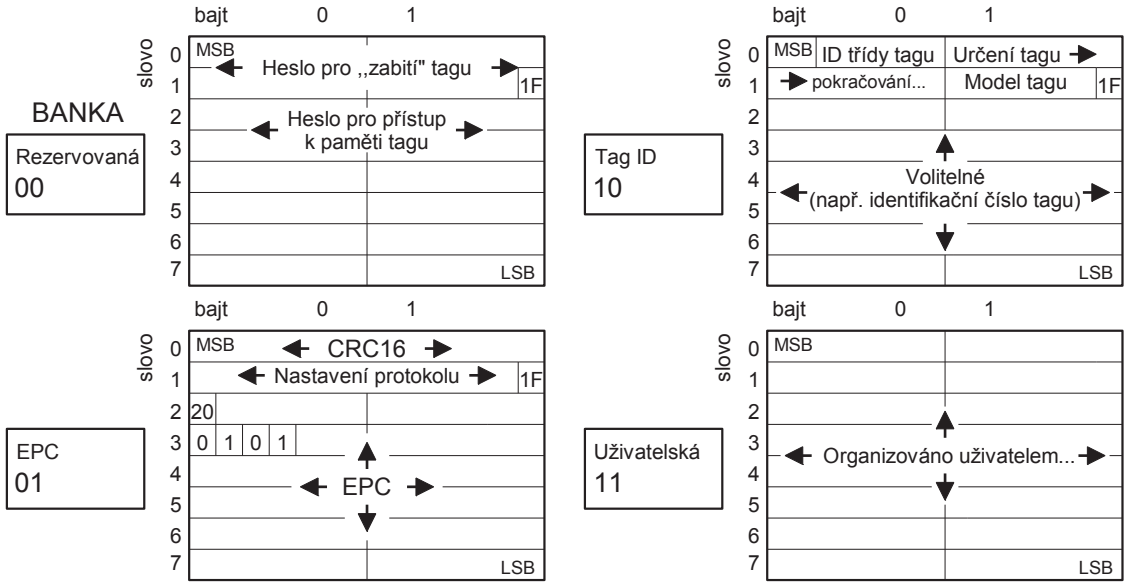
Výběr, inventarizaci a přístup k tagu lze chápat jako velmi jednoduchý příklad linkové vrstvy ve vrstvovém modelu komunikační sítě.

2.3.1 Organizace paměti tagu

Paměť tagu je logicky rozdělena do čtyř částí (bank), jak je naznačeno na obrázku 2.17. Tyto banky jsou číslovány dvoumístnými binárními čísly a jsou to:

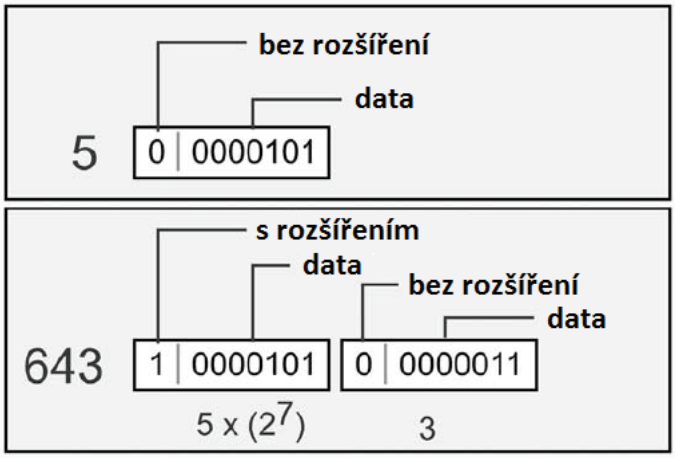
- Rezervovaná banka (00_b) – obsahuje přinejmenším 32-bitová hesla pro „zabití“ tagu (KILL) a pro přístup k informacím obsaženým v tagu (ACCESS).

- EPC banka (01_b) – obsahuje *EPC* tagu (identifikační číslo tagu), 32-bitové slovo pro kontrolu protokolu obsahující délku *EPC* (*PC*), různé volitelné informace a *CRC-16* pro kontrolu hodnoty *EPC*.
- TID banka (10_b) – obsahuje identifikační informace související s tagem, lišící se např. podle toho na jaký druh výrobku je tag připevněn. Lze to využít také např. k identifikaci výrobce tagu apod.
- Uživatelská banka (11_b) – nemá specifikovanou strukturu až na číslování bajtů a slov. Je určena pro aplikačně specifické informace.



Obr. 2.17 Organizace paměti tagu (obrázek převzat z [3]).

Velikost jednotlivých bank je velice flexibilní díky adresování pomocí vektorů s rozšiřujícím bitem (*Extension Bit Vector* – EBV). V tomto schématu má každý bajt adresy jeden rozšiřující bit a sedm bitů dat. Pokud je rozšiřující bit roven „0“, datová část obsahuje celou adresu. Pokud je tento bit roven „1“, pak následuje nejméně ještě jeden bajt, kde datová část obsahuje dalších sedm bitů adresy. Protože i tento bajt má rozšiřující bit, pak adresa může mít téměř libovolnou délku a omezení spočívá pouze ve fyzické velikosti paměti (samozřejmě nelze adresovat paměť, která neexistuje). Cenou za libovolné rozšíření adresy je, že 1/8 bitů je vyhrazena pro rozšiřující bity. Dva příklady tohoto adresování i s výslednou adresou jsou na obrázku 2.18.



Obr. 2.18 Příklady adresování EBV (obrázek převzat z [3]).

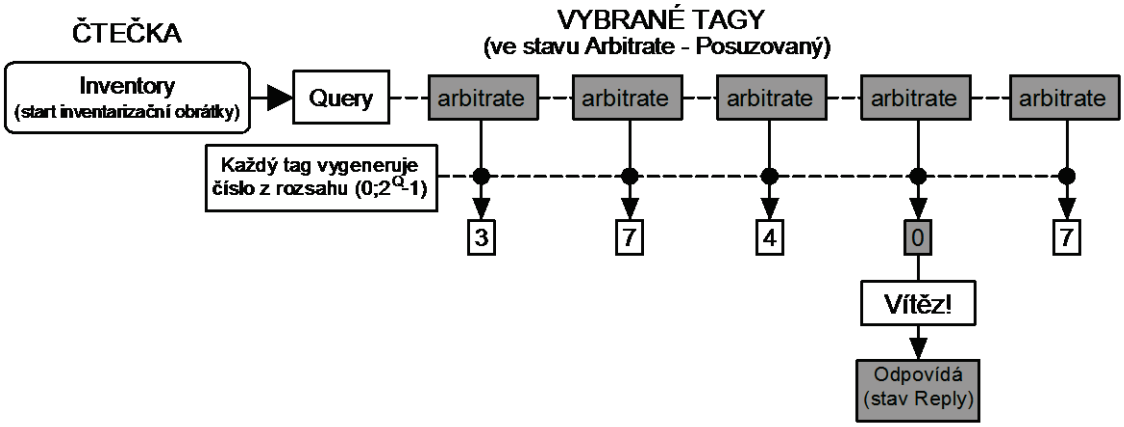
Stav určitého tagu v inventarizační obrátce je určen hodnotami čtyř jednobitových příznaků (*Session flags*) S0 až S3 a jednobitového příznaku výběru SL (*Selection flag*). Tyto příznaky se nastavují příkazem *Select* a slouží k omezení počtu tagů obsluhovaných v rámci jedné inventarizační obrátky.

2.3.2 Kontrola přístupu k přenosovému médiu

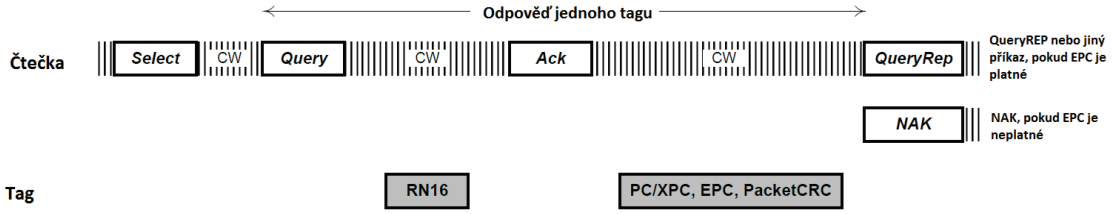
Pro přístup k přenosovému médiu se používá varianta náhodného přístupu odvozená od přístupu *slotted Aloha* (zařízení odpovídají náhodně, avšak pouze v určitých časových okamžicích – slotech), která se nazývá *Q-protokol*.

Q-protokol funguje tak, že čtečka pošle k vybraným tagům příkaz *Query*, který mj. obsahuje čtyřbitové číslo Q (odtud *Q-protokol*). Každý tag si na základě tohoto čísla vygeneruje náhodné číslo, které leží v rozsahu 0 až (2^Q-1) a říká, ve kterém slotu v pořadí bude tag odpovídat, přičemž 2^Q určuje počet slotů v nadcházející inventarizační obrátce. Slotů musí být aspoň tolik, kolik je tagů ve čtecí zóně. Teoretický počet tagů ve čtecí zóně je $2^{15} = 32768$. Tag si vygenerované číslo uloží do čítače slotu.

Pokud číslo vygenerované tagem je 0, tag odpoví čtečce (viz obrázek 2.19) paketem obsahujícím preambuli a náhodné 16-bitové číslo *RN16*. Paket neobsahuje kontrolu *CRC*, protože pokud budou data poškozena, bude chybná i potvrzovací odpověď čtečky *ACK* a potvrzení jednoduše selže. Pokud však čtečka přijme *RN16* nepoškozené, odpoví příkazem *ACK* obsahujícím stejné *RN16*. Pokud tag přijme tento příkaz a *RN16* je to, které vyslal, odpoví čtečce paketem, který obsahuje jeho identifikační číslo *EPC*, délku tohoto čísla a kontrolu *CRC-16*. Celý tento proces je naznačen na obrázku 2.20. Nyní tedy čtečka zná *EPC* tagu i specifické *RN16* a může k tagu přistoupit, zapisovat do jeho paměti nebo z ní číst a může tak vytvořit unikátní logické spojení s konkrétním tagem, i když naslouchá mnoho tagů. K vytvoření tohoto logického spojení stačí čtečce pouze specifické *RN16* tagu.



Obr. 2.19 Příklad začátku inventarizační obrátky s příkazem Query pro Q = 3. Tagy začínají ve stavu *Arbitrate* (obrázek převzat z [3]).



Obr. 2.20 Komunikace čtečky s jedním tagem (obrázek převzat z [2]).

Pokud však jediná informace, kterou chce čtečka získat, je *EPC* tagu, pak čtečka může vyslat další příkaz *Query* nebo lépe příkaz *QueryRep* (je kratší). Tag, který už je ve stavu Potvrzeno (*Acknowledged*, tedy přijal platný příkaz *ACK*), přepne svůj příznak spojení (*Session flag*) a čeká na další inventarizační obrátku. Ostatní tagy sníží svůj čítač slotu o jedničku a pokud čítač některého tagu dosáhne nuly, pak tag odpoví číslem *RN16*. Pokud neodpoví žádný tag, čtečka pošle *QueryRep* znovu. Pokud teď některý tag odpoví, čtečka pošle *ACK*, daný tag odešle své *EPC* a následně pošle čtečka opět *QueryRep* a proces se opakuje.

Může se stát, že ve slotu odpoví víc tagů a dojde ke kolizi. Čtečka může (ale také nemusí) být schopna odlišit kolizi od prázdného slotu. Je dobré si uvědomit, že preambule při odesílání *RN16* je pro všechny tagy stejná, takže zůstane čitelná, zato *RN16* tagů jsou rozdílné a výsledek bude nesmyslný. Pro čtečku je výhodné rozeznat kolizi od prázdných slotů a platných odpovědí, protože po 2^Q slotech (jedné inventarizační obrátce) se může rozhodnout pro jednu z pěti možností:

1) Zvýšit číslo Q o 1 a udělat další inventarizační obrátku.
2) Nechat Q stejné a udělat další inventarizační obrátku.
3) Snížit číslo Q o 1 a udělat další inventarizační obrátku.
4) Vyslat příkaz *Query* s novým číslem Q.
5) Ukončit inventarizaci.

První tři možnosti se dějí prostřednictvím příkazu *QueryAdjust*. Čtečka předem neví, kolik tagů je ve čtecí zóně, a tedy kolik slotů v inventarizační obrátce je třeba

(z toho pak vyplývá volba čísla Q). Pokud je většina slotů prázdných, Q může být pravděpodobně sníženo. Pokud většina slotů obsahuje kolize, Q je pravděpodobně příliš malé. Q je optimální, pokud 30% až 50% slotů obsahuje platné odpovědi. Možnost pět je zvolena, pokud jsou sloty prázdné a přitom Q už je velmi nízké (např. 1 nebo 2).

Rozpoznání kolizních slotů je také důležité kvůli propustnosti, protože v případě, že čtečka pošle *ACK* s neplatným *RN16*, musí čekat na odpověď *EPC*, ta nepřijde, a tak musí poslat *NAK* příkaz všem tagům, aby věděly, že žádný tag nebyl načten. To trvá o mnoho déle než *QueryRep* po prázdném slotu.

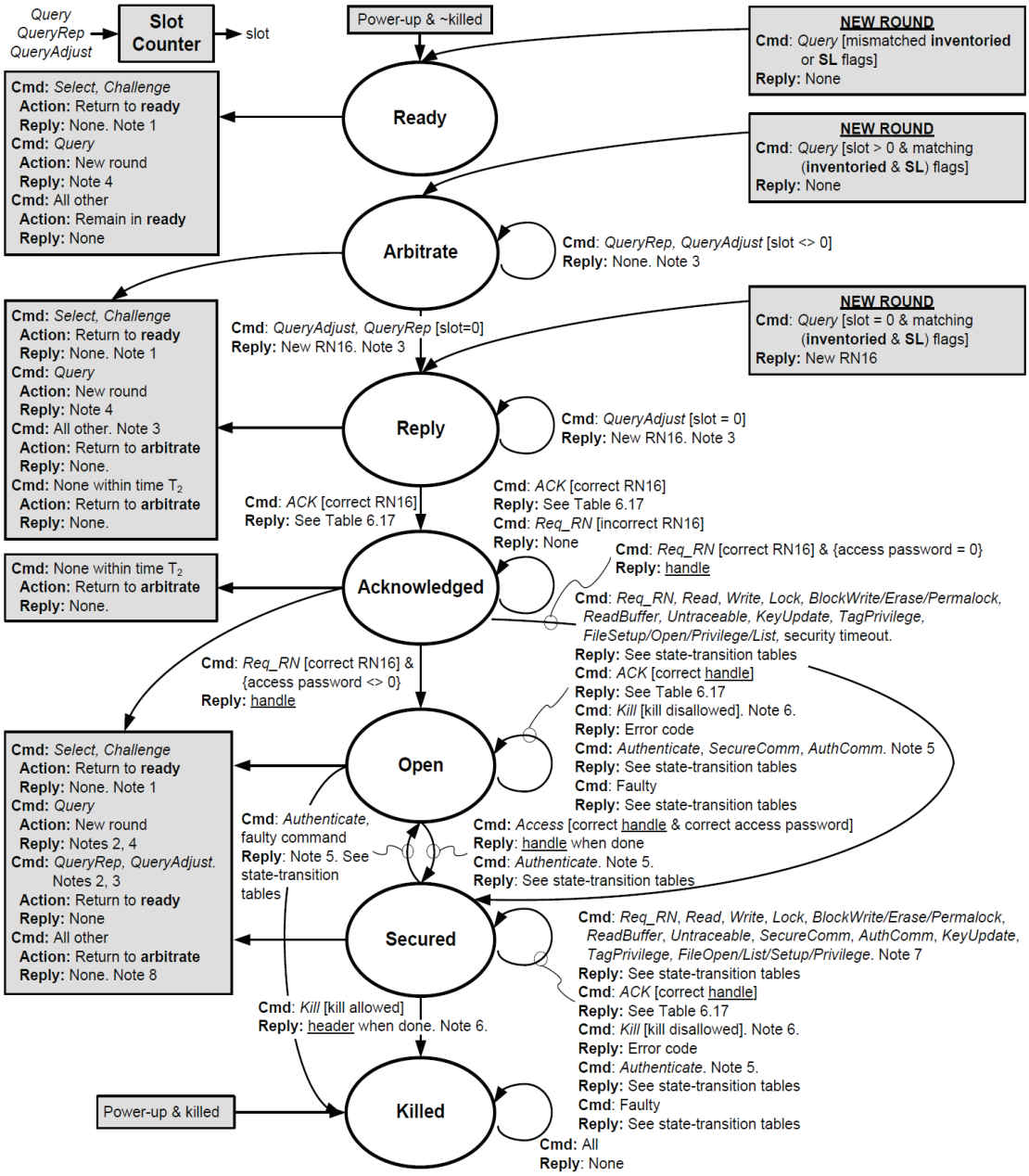
Pro zjištění, zda všechny tagy byly zaznamenány, lze použít následující proceduru:

1) Spočítat (a zaznamenat) všechny tagy, jejichž inventarizační příznak je ve stavu A, a přepnout jej do stavu B.
2) Udělat to stejné s tagy ve stavu B a přepnout je do stavu A.
3) Pokud jsou nalezeny stejné tagy, byly všechny tagy zaznamenány, jinak je třeba celý postup zopakovat.

Tato metoda bohužel nepočítá s různými náhodnými jevy, jako jsou například hluchá místa v pokrytí elektromagnetickým polem vysílaným čtečkou (viz podkapitola 2.3.4).

2.3.3 Stavy a příkazy

Stavový diagram, zachycující možné stavy tagu a přechody mezi nimi, je na obrázku 2.21.



Obr. 2.21 Stavový diagram tagu (obrázek převzat z [2]).

Tag se tedy může nacházet v jednom z následujících stavů:

- Připravený (*Ready*)
- Posuzovaný (*Arbitrate*)
- Odpovídající (*Reply*)
- Potvrzený (*Acknowledged*)
- Otevřený (*Open*)
- Zabezpečený (*Secured*)
- Zabitý (*Killed*)

Mezi stavy lze přecházet pomocí množství příkazů, které se dělí na povinné a nepovinné. Z povinných příkazů lze považovat za nejdůležitější ty, které slouží k inventarizaci tagů nacházejících se v poli působnosti čtečky. Jsou to následující příkazy:

- *Query*: startuje inventarizační obrátku, jeho kód je 1000_b, má délku 22 bitů (bez preambule a popř. pilotního tónu). Jeho parametry jsou v následující tabulce:

Tab. 2.3 Parametry příkazu Query (převzato z [2]).

	Číslo příkazu	**Dělicí poměr**	**Kódování**	**TRext**	**Výběr**	**Příznak spojení**	**Cíl**	**Q**	**CRC**
Počet bitů:	4	1	2	1	2	2	1	4	5
Popis:	1000	0:DR=8 1:DR=64/3	00:M=1 01:M=2 10:M=4 11:M=8	0: Bez pilotního tónu 1: S pilotním tónem	00:Vše 01:Vše 10:~SL 11:SL	00:S0 01:S1 10:S2 11:S3	0:A 1:B	0 až 15	CRC-5

- *QueryRep*: Nejkratší příkaz o délce pouze čtyři bity, číslo příkazu je 00_b, další dva bity jsou příznakem spojení (*Session flag*), jak ukazuje následující tabulka:

Tab. 2.4 Parametry příkazu QueryRep (převzato z [2]).

	Číslo příkazu	**Příznak spojení**
Počet bitů	2	2
Popis:	00	00:S0 01:S1 10:S2 11:S3

- *QueryAdjust*: startuje novou inventarizační obrátku. Jeho číslo je 1001_b a má dva parametry – viz tabulka 2.5:

Tab. 2.5 Parametry příkazu QueryAdjust (převzato z [2]).

	Číslo příkazu	**Příznak spojení**	**UpDn**
Počet bitů:	4	2	3
Popis.	1001	00:S0 01:S1 10:S2 11:S3	110: Q=Q+1 000: Neměnit Q 011: Q=Q-1

- *ACK*: potvrzení přijatého *RN16* čtečkou. Číslo příkazu je 10_b, příkaz obsahuje *RN16*, kontrola chyb (*CRC*) se neprovádí. Struktura příkazu je vidět v tabulce 2.6.

Tab. 2.6 Parametry příkazu ACK (převzato z [2]).

	Číslo příkazu	**RN**
Počet bitů	2	16
Popis:	01	Potvrzované RN16 nebo handle

- *NAK*: Oznamuje všem tagům, že po *ACK* nebylo přijato žádné *EPC*. Struktura je vidět v tabulce 2.7.

Tab. 2.7 Parametry příkazu NAK (převzato z [2]).

	Číslo příkazu
Počet bitů:	8
Popis:	11000000

Kromě výše uvedených příkazů existují ještě další povinné příkazy, které slouží např. k zápisu a čtení do paměti tagu, k zabití tagu, k výběru populace tagů, které mají být obslouženy apod. Dále existuje množství příkazů nepovinných. Jsou to např. příkazy pro zamykání paměti po jednotlivých slovech i blocích, příkazy pro blokové čtení a zápis apod. Více informací ke všem povinným i nepovinným příkazům lze získat v literatuře [2] a [3].

2.3.4 Příznaky spojení

Čtyři příznaky spojení (*session flag*) principiálně umožňují kvazi-simultánní inventarizace od různých čteček, prakticky to však nejde, protože každý tag má jen jeden čítač slotů. Proto je na kompetentním uživateli, aby vytvořil pravidla, podle kterých budou příznaky přiděleny jednotlivým čtečkám.

V této souvislosti je také důležitá perzistence dle tabulky 2.8. Perzistence říká, jak dlouho po ztrátě napájení je tag schopen udržet informace o nastavení svých příznaků (např. příznak výběru, inventarizační příznak atd.). To je důležité např. v situaci, kdy tag již byl zaznamenán, ale dostal se na krátký okamžik do hluchého místa, kde elektromagnetické pole je příliš slabé pro napájení tagu. Bez perzistence by se choval jako vyresetovaný a nezaznamenaný.

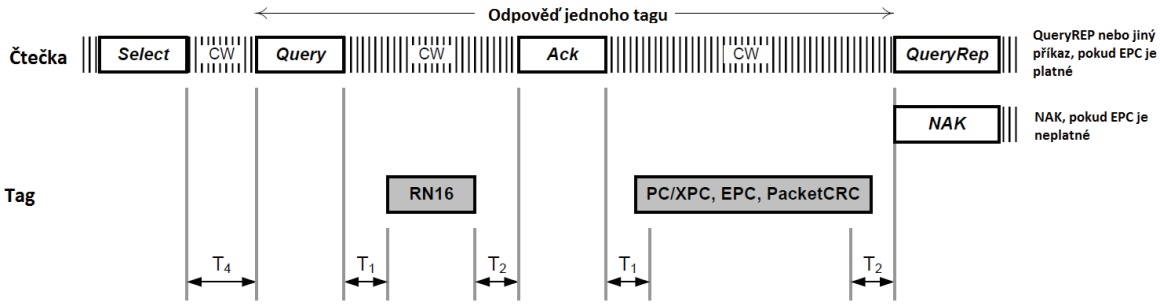
Tab. 2.8 Perzistence pro různé nastavení výběrových příznaků (převzato z [2]).

Příznak	**Požadovaná perzistence**
Příznak spojení S0	Tag napájen: nekonečná Tag nenapájen: žádná

Příznak spojení S1	Tag napájen: 500ms<persistence<5s Tag nenapájen: 500ms<persistence<5s
Příznak spojení S2	Tag napájen: nekonečná Tag nenapájen: 2s<persistence
Příznak spojení S3	Tag napájen: nekonečná Tag nenapájen: 2s<persistence
Příznak výběru SL	Tag napájen: nekonečná Tag nenapájen: 2s<persistence

2.3.5 Požadavky na časování

Na obrázku 2.22 vidíme požadavky na povolená zpoždění a potřebné prodlevy mezi příkazy. Význam a hodnoty jednotlivých časů jsou v tabulce 2.9.



Obr. 2.22 Časování komunikace podle protokolu Gen-2 (obrázek převzat z [2]).

Tab. 2.9 Význam a hodnoty parametrů časování (převzato z [2]).

Parametr	**Minimum**	**Nominální hodnota**	**Maximum**	**Popis**
T_1	MAX(RTcal, $10T_{pri}$) × (1-|FT|)-2µs	MAX(RTcal, $10T_{pri}$)	MAX(RTcal, $10T_{pri}$) × (1-|FT|)+2µs	Zpoždění odpovědi tagu
T_2	$3{,}0\times T_{pri}$		$20{,}0\times T_{pri}$	Zpoždění mezi odpovědí tagu a následujícím příkazem čtečky
T_3	$0{,}0\times T_{pri}$			Po T_1 čtečka čeká na odpověď tagu. Tato doba T_3 je definována, ale nemá specifikovanou hodnotu
T_4	$2{,}0\times$RTcal			Požadovaná prodleva mezi dvěma příkazy čtečky

Parametr T_{pri} je symbolová perioda při komunikaci od tagu ke čtečce a platí, že T_{pri}=1/BLF.

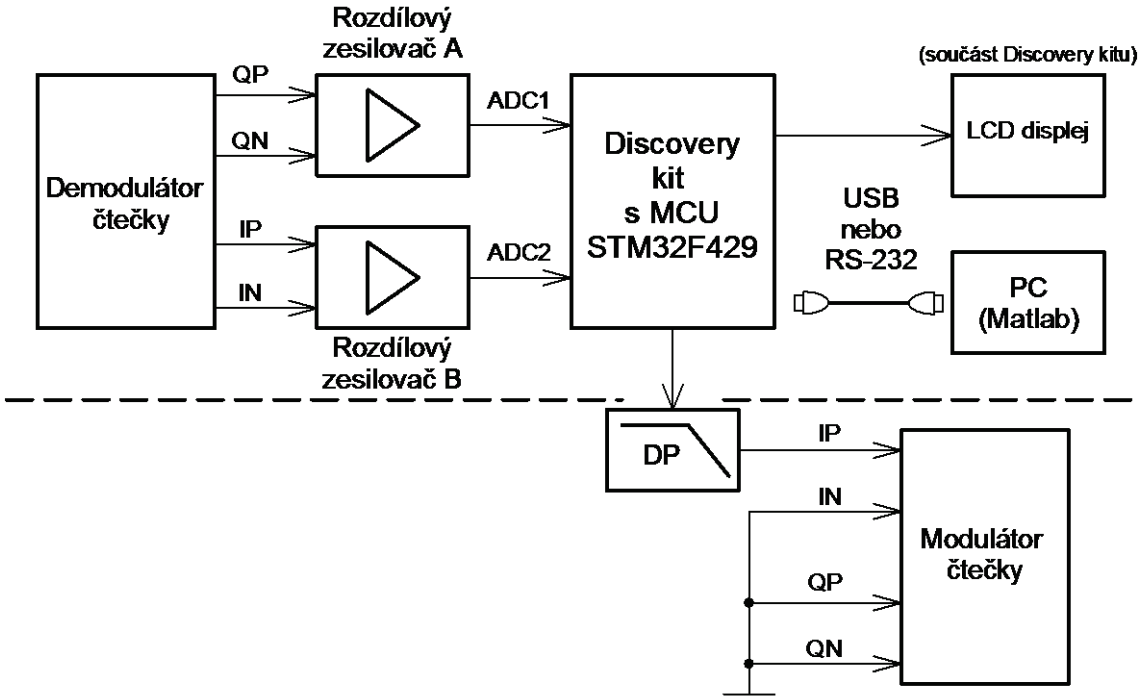
3 PRAKTICKÁ REALIZACE ŘÍZENÍ ČTEČKY A ZPRACOVÁNÍ SIGNÁLU

Tato kapitola se zabývá návrhem programu pro mikrokontrolér, který slouží pro nastavení vysokofrekvenčních integrovaných obvodů ve front endu čtečky, generování základních příkazů standardu EPC Global Class-1 Gen-2 a především pro navzorkování a dekódování signálů odezvy tagu na tyto příkazy. To je stěžejním úkolem této diplomové práce a cílem je získat identifikační čísla tagů nacházejících se v poli antény čtečky. Zmíněné funkce jsou implementovány na vývojové desce (tzv. Discovery kit) s mikrokontrolérem ST Microelectronics STM32F429ZIT6, který je v této kapitole také stručně popsán. Je nutné poznamenat, že z velkého počtu kombinací komunikačních rychlostí (BLF) a způsobů kódování, které standard Gen-2 umožňuje, se navržené řešení omezuje na komunikační rychlost 40kHz (tedy tu nejnižší) a na nejjednodušší kódování FM0 (viz kapitola 2.2.2). Pohled na výsledné zařízení je v příloze A.1.

3.1 Blokové schéma

Blokové schéma propojení front endu čtečky EXIN-1 s vývojovou deskou Discovery kit je na obrázku 3.1. Horní polovina schématu nad přerušovanou čarou je přijímací část, kde bylo nutné umístit dva rozdílové zesilovače, které převádí diferenční výstupy demodulátoru čtečky na výstupy vztažené proti zemi, aby mohl být použit vnitřní A/D převodník mikrokontroléru. Tyto výstupy jsou následně navzorkovány a obdržená data jsou zpracována tak, aby se získaly potřebné informace. Pro ověření funkčnosti dekódovacích algoritmů se v první fázi jevilo vhodné přenést navzorkovaná data do PC, konkrétněji programu MATLAB, který je vhodný pro zpracování signálu. Tento přenos je realizován asynchronní sériovou komunikační periferií mikrokontroléru (UART), kterou lze použitím externího převodníku převést na standard sériové linky RS-232 nebo na standard USB.

Vysílací část zahrnuje implementaci příkazů čtečky a jejich odeslání na vstup modulátoru čtečky. Ideálně by měla ještě zahrnovat implementaci FIR filtru typu *Raised-cosine*, aby byly splněny spektrální požadavky na signály vysílané čtečkou, a dále by měla zahrnovat digitálně – analogový převod, avšak pro ověření funkce výsledného zařízení v laboratorních podmínkách není realizace těchto částí nutná. Signál pro modulátor je tedy generován pomocí jednobitové konverze, kdy je jeden digitální výstup (pin) mikrokontroléru připojen na neinvertující soufázový vstup (IP) modulátoru a ostatní vstupy jsou připojeny na zem. Zmíněný signál je ještě upraven filtrem typu dolní propust 1. řádu, ale to nic nemění na faktu, že signál neplní spektrální požadavky.



Obr. 3.1 Blokové schéma propojení front endu čtečky s mikrokontrolérem.

3.2 Popis použitého mikrokontroléru a vývojové desky

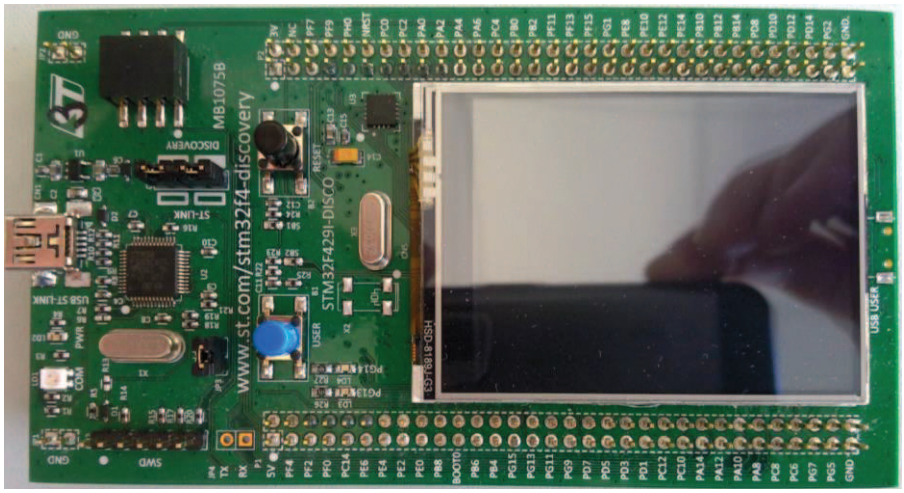
Jak již bylo řečeno, potřebné funkce jsou implementovány na vývojové desce zvané Discovery kit s mikrokontrolérem STM32F429ZIT6. Vzhled vývojové desky je na obrázku 3.2. Kromě samotného mikrokontroléru, o kterém bude řeč později, tato deska obsahuje mnoho užitečných přídavných prvků, jako jsou [6]:

- Programovací a ladicí rozhraní ST-LINK/V2, které lze použít buď pro práci s mikrokontrolérem osazeným na desce, nebo pro práci s externím mikrokontrolérem, který se připojuje přes rozhraní pojmenované SWD.
- Napájecí napětí 3V a 5V získané z USB sběrnice, které lze využít pro napájení proudově nenáročného zařízení připojeného k desce. Druhou možností je napájet desku z externího zdroje o stejném napětí.
- TFT (*Thin – Film - Transistor*) LCD dotykový displej s uhlopříčkou 2,4 palce, 246 tisíci barvami RGB, 240×320 obrazovými body.
- Paměť SDRAM 64 megabitů (2^{20} (1 mega) × 16 bitů × 4 banky) zahrnující AUTO REFRESH MODE a režim snížené spotřeby (*power-save mode*).
- 6 LED diod:
  - LD1 (červená/zelená) pro USB komunikaci.
  - LD2 (červená) indikující napájení 3,3V.
  - dvě uživatelské LED: LD3 (zelená) a LD4 (červená).
  - dvě LED pro USB OTG: LD5 (zelená) pro indikaci napětí V_{BUS} a

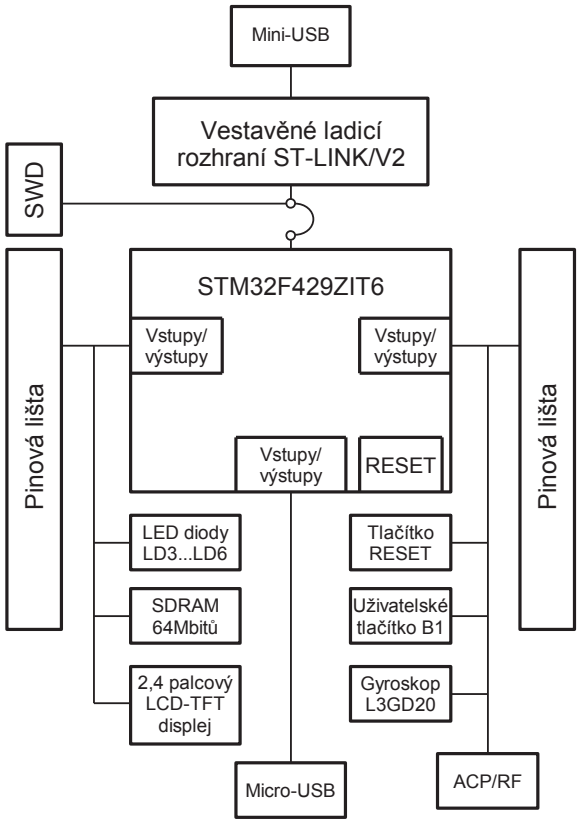
LD6 (červená) pro indikaci nadproudu (OC, *over-current*).

- Dvě tlačítka (uživatelské a resetovací).
- Periferii USB OTG (*On-the-go*) s micro-AB konektorem.

Propojení mezi mikrokontrolérem a ostatními periferiemi na Discovery kitu ilustruje blokové schéma na obrázku 3.3.



Obr. 3.2 Vývojová deska STM32F429 Discovery kit (obrázek převzat z [6]).



Obr. 3.3 Bloková struktura vývojové desky STM32F429 Discovery kit (obrázek převzat z [6]).

3.3 Mikrokontrolér STM32F429ZIT6

Celá vývojová deska je postavena kolem mikrokontroléru firmy ST Microelectronics STM32F429ZIT6, což je 32-bitový mikrokontrolér s jádrem ARM® Cortex®-M4 a také jednotkou pro operace v plovoucí řádové čárce (FPU, *Floating-point Unit*). Dalšími součástmi a přednostmi mikrokontroléru, které jsou podstatné pro realizaci výsledného zařízení, jsou [7]:

- Kontrolér pro LCD-TFT dotykový displej s rozlišením XGA a vyhrazenou jednotkou Chrom-ART Accelerator™ pro zlepšení grafické úrovně displeje (DMA2D).
- Hodinové, resetovací a napájecí bloky:
  - Napájení aplikace a vstupních/výstupních pinů 1,7V až 3,6V.
  - Krystalový oscilátor 4 až 26MHz.
- 3×12-bitový A/D převodník.
- 2×12-bitový D/A převodník.
- DMA kontrolér pro všeobecné použití: 16 datových proudů, FIFO paměť.
- 17 čítačů/časovačů: 2×32-bitový časovač, ostatní časovače 16-bitové. Pracovní kmitočet až 180MHz.
- V závislosti na pouzdru až 168 vstupních/výstupních pinů s podporou přerušení:
- USB 2.0 full-speed/high-speed kontrolér, režimy device/host/OTG.
- Jednotka pro výpočet *CRC* (*Cyclic Redundancy Check*).

Mikrokontrolér dokáže vykonávat až 225 milionů instrukcí za sekundu. Ve verzi osazené na vývojové desce je zapouzdřen v pouzdru LQFP144 se 144 vývody a rozměry 20×20mm. Veškeré další informace o všech součástech a parametrech mikrokontroléru lze nalézt v literatuře [7] a [8].

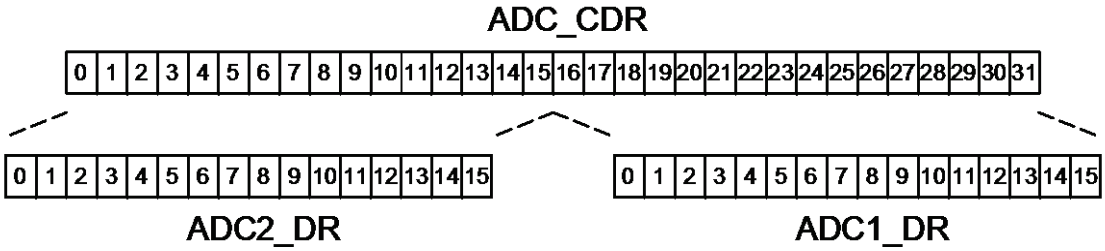
3.4 Periferie mikrokontroléru použité při realizaci zařízení

Pro řízení čtečky a zpracování signálu je důležitých především několik konkrétních periferií mikrokontroléru. Jednak je to kontrolér DMA (*Direct Access Memory*), který slouží pro přímý přenos dat z některé vnější periferie (například UART, A/D převodník, kontrolér pro čtení/zápis do vnější paměti FMC apod.) do paměti bez asistence jádra. Jádro tedy může během DMA přenosu vykonávat jinou činnost, navíc přenos je rychlejší a ničím nepřerušovaný.

Za druhé je to analogově – digitální převodník. Po zvážení všech výhod a nevýhod jednotlivých řešení A/D převodu popsaných v kapitole 1.8 bylo vybráno řešení s vnitřními převodníky mikrokontroléru. Toto řešení bylo vybráno proto, že ačkoliv mikrokontrolér nabízí mnoho vstupně výstupních pinů organizovaných převážně do 16-bitových portů, na použité vývojové desce je většina z nich dedikována pro různé periferie na desce, takže výsledkem je, že reálně lze použít asi jen 23 volných pinů,

z nichž se však nikdy více než pět nenalézá v jednom portu. Pro oba vnější převodníky je třeba vždy na mikrokontrolér přivést alespoň 16 linek (14 datových, signalizace překročení rozsahu, hodinový signál pro synchronizaci) a skládání výstupních dat z různých pinů na různých portech by bylo zbytečnou komplikací, nehledě na to, že by zůstalo volných pouze sedm ze zmíněných 23 volných pinů. Z dostupných tří vnitřních A/D převodníků jsou tedy použity dva, a to ADC1 a ADC2, každý pro vzorkování jednoho kanálu. Vzorkovací kmitočet každého převodníku je při nastaveném hodinovém taktu jádra mikrokontroléru 144MHz maximálně 2,4 milionu vzorku za sekundu (2,4MSample/s). Přenos digitalizovaných dat do paměti je samozřejmě řešen pomocí DMA pro dosažení co největší rychlosti. Původně byly převodníky nastaveny do tzv. *Multimode Interleaved módu* (prokládaný režim). Ten spočívá v tom, že výstupem každého převodníku jsou 16-bitová slova, ta jsou pak složena do 32-bitového slova, které je posláno přes DMA do paměti. 16-bitová slova jsou do 32-bitového slova organizována tak, že horních 16 bitů obsahuje data z převodníku ADC2, spodních 16 bitů obsahuje data z převodníku ADC1. Tuto situaci znázorňuje obrázek 3.4.

ADCx_DR jsou výstupní registry jednotlivých převodníků, ADC_CDR je výstupní registr pro Multimode režim A/D převodníků.



Obr. 3.4 Seřazení výstupních registrů A/D převodníků v prokládaném režimu.

Jak se ale záhy ukázalo, řešení s prokládaným režimem není příliš vhodné, protože pro dekódování signálu je třeba mít oddělený signál soufázové složky přijatého signálu a signál kvadraturní složky signálu, takže bylo nutné prokládaná data opět rozdělit, což způsobovalo zbytečný nárůst času nutný pro zpracování a dekódování signálu. Proto nakonec každý převodník posílá data do své vlastní proměnné prostřednictvím vlastního DMA kanálu.

Třetí podstatnou periferií je některá periferie schopná přenést data do PC. Původně byla použita periferie USB_OTG_FS/HS, která byla nastavena do režimu *device* (zařízení) s rychlostí *full-speed*, protože s interní fyzickou vrstvou není rychlost *high-speed* podporována, a po připojení k PC se chovala jako virtuální sériový port. Bohužel ve spojení s programem MATLAB tato periferie nefungovala spolehlivě (podrobnosti v kapitole 3.7), takže nakonec byla použita asynchronní sériová komunikace (periferie UART), kterou lze převést na standard RS-232 nebo na standard USB pomocí externích převodníků.

Dalšími použitými periferiemi byl např. časovač pro přesné nastavení časových parametrů příkazů standardu Gen-2 a SPI periferie pro řízení LCD displeje.

3.5 Rozdílový zesilovač

Protože výstupy soufázové i kvadraturní složky demodulátoru čtečky jsou diferenční, je nutné převést je na výstupy, které budou vztažené k zemi systému. To lze zajistit rozdílovými zesilovači dle schématu v příloze A.2, které bylo vytvořeno v programu Eagle. Například pro vrchní zesilovač (pro kvadraturní složku) se výstupní napětí řídí rovnicí:

$$U_{\mathrm{Q\_VÝST}} = U_{\mathrm{QP}} \cdot \left(\frac{R_3 + R_4}{R_3} \cdot \frac{R_2}{R_1 + R_2} \right) - U_{\mathrm{QN}} \cdot \frac{R_2}{R_1} \qquad [\mathrm{V};\ \mathrm{V},\ \Omega] \quad (3.1)$$

kde U_{QP} je napětí na neinvertujícím vstupu rozdílového zesilovače, U_{QN} je napětí na invertujícím vstupu rozdílového zesilovače a $U_{\mathrm{Q\_VÝST}}$ je výstupní napětí zesilovače. Tuto rovnici lze výrazně zjednodušit zavedením rovností $R_3 = R_1$, $R_4 = R_2$. Pak platí:

$$U_{\mathrm{Q\_VÝST}} = \frac{R_2}{R_1} \cdot \left(U_{\mathrm{QP}} - U_{\mathrm{QN}} \right) \qquad [\mathrm{V};\ \Omega,\ \mathrm{V}] \quad (3.2)$$

Dosazením do této rovnice se pro hodnoty rezistorů ze schématu ukáže, že každý z rozdílových zesilovačů zesiluje rozdílové napětí desetkrát.

Protože vzhledem ke vstupnímu rozsahu A/D převodníku mikrokontroléru je třeba, aby se výstupní napětí rozdílových zesilovačů pohybovalo v rozmezí 0V až 3V, je nutné posunout stejnosměrnou složku výstupních napětí zesilovačů na polovinu tohoto rozsahu, tj. 1,65V. To se děje prostým děličem napětí, který je posílen napěťovým sledovačem s operačním zesilovačem IC1C.

Poslední částí schématu je stabilizátor napětí s výstupním napětím 3,3V pro napájení operačních zesilovačů. Na vstup stabilizátoru je přivedeno hlavní 12V napájení celé čtečky. Operační zesilovače mohou být napájeny také přímo z vnějšího napětí 3,3V, je-li dostupné. Přepínání se pak provádí pomocí přepojování propojky (jumperu) JP1.

V dolní části schématu jsou pak filtry typu dolní propust pro alespoň malou úpravu obdélníkového signálu posílaného na vstup modulátoru čtečky.

Podoba desky plošných spojů navržené podle tohoto schématu stejně jako osazovací výkresy jsou v příloze A.3.

Pro propojení vývojové desky Discovery kit s front endem čtečky byla navržena ještě jednoduchá jednostranná deska plošných spojů, do které se Discovery kit vsadí a potřebné signály jsou vyvedeny na konektory se zámkem. Několik spojů, které bylo třeba vést na horní straně desky, je provedeno drátovými propojkami. Propojení s front endem čtečky je pak provedeno pomocí kabelů zakončených protikusem ke konektorům se zámkem. Schéma této desky je stejně jako schéma desky předchozí součástí přílohy A2. Předloha této desky plošných spojů spolu s osazovacím výkresem je pak součástí přílohy A.3.

3.6 Algoritmus pro dekódování signálu odezvy tagu

Pro dekódování signálu tagu bylo třeba vytvořit algoritmus, jehož vstupem by mělo být pole vzorků z A/D převodníku a výstupem pole hodnot „0“ a „1“, které odpovídají buď náhodnému šestnáctibitovému číslu *RN16*, nebo identifikačnímu číslu tagu (*EPC*). Tento algoritmus byl vytvořen nejprve v programu MATLAB, kde je snadné zobrazit si výsledky jednotlivých mezikroků zpracování.

Celkem byly vytvořeny čtyři funkce pro program MATLAB nazvané `filtrace_v1.m` až `filtrace_v4.m`, které se liší v tom, s jakým signálem pracují. Vždy je však vstupem soubor programu MATLAB s příponou `.mat`, ve kterém jsou uloženy navzorkované signály soufázové a kvadraturní složky odezvy tagu (průběhy signálů jsou na obrázku 3.5). Tyto soubory jsou nazvány `RN16_a.mat` až `RN16_o.mat`, protože algoritmus byl ověřován nejprve na navzorkovaném signálu náhodného šestnáctibitového čísla *RN16*. Jednotlivé funkce se pak liší takto: `filtrace_v1.m` vypočte z jednotlivých složek absolutní hodnotu a fázi celkového komplexního signálu a následně z absolutní hodnoty zjišťuje, jakou sekvenci symbolů „0“ a „1“ signál obsahuje na základě korelace s kopiemi těchto symbolů, `filtrace_v2.m` také vypočte z jednotlivých složek absolutní hodnotu a fázi celkového komplexního signálu a další výpočet také provádí se signálem absolutní hodnoty, ale používá mnohem jednodušší algoritmus založený na detekci hran (popsáno dále), `filtrace_v3.m` používá stejný algoritmus, ale pracuje s fází signálu a konečně `filtrace_v4.m` používá stejný algoritmus, ale pracuje pouze se signálem soufázové nebo kvadraturní složky podle toho, který má vetší rozdíl mezi maximální a minimální hodnotou. Aplikací každé ze čtyř funkcí na všech 15 realizací navzorkovaného signálu *RN16* bylo zjištěno, že nejlepší výsledky poskytuje metoda poslední, tedy funkce `filtrace_v4.m`, která fungovala ve všech případech a která pak byla vybrána pro implementaci v programu mikrokontroléru. Ještě je třeba poznamenat, že každá realizace signálu *RN16* platí pro jinou vzdálenost tagu od antény čtečky, pro jiné natočení tagu, avšak pro stejný tag, protože pro stejnou polohu tagu by odběr více realizací signálu neměl smysl.

Samotný algoritmus pro dekódování funguje následovně. Nejprve je třeba znát vzorkovací kmitočet A/D převodníků a komunikační rychlost BLF. Z těchto údajů lze vypočítat počet vzorků na symbol. Např. pro vzorkovací kmitočet f_{VZ}=2,4MHz (2,4MSample/s) a BLF=40kHz vychází:

$$N = \frac{f_{VZ}}{BLF} = \frac{2{,}4 \cdot 10^6\,\mathrm{Hz}}{40 \cdot 10^3\,\mathrm{Hz}} = 60\,\mathrm{vzorků} \qquad [-;\ \mathrm{Hz},\ \mathrm{Hz}] \quad (3.3)$$

Reálně je počet vzorků na každý symbol o několik vzorků větší nebo menší. Tento fakt je zohledněn parametrem `NS_jitter`. V případě funkce `filtrace_v4.m` se projdou pole vzorků soufázové a kvadraturní složky, aby pro každou složku bylo nalezeno maximum a minimum. Na základě rozdílu maxima a minima je vybrána složka se silnějším signálem. Na ni je pak aplikována exponenciální kumulace s koeficientem $q = 1\text{-}2^{-8}$, jejíž výsledek funguje jako plovoucí průměr (původní signál i signál po provedení kumulace je vidět na obrázku 3.6, signály jsou posunuty tak, aby měly nulový stejnosměrný posun.). Zdrojový kód části funkce provádějící

exponenciální kumulaci je uveden níže:

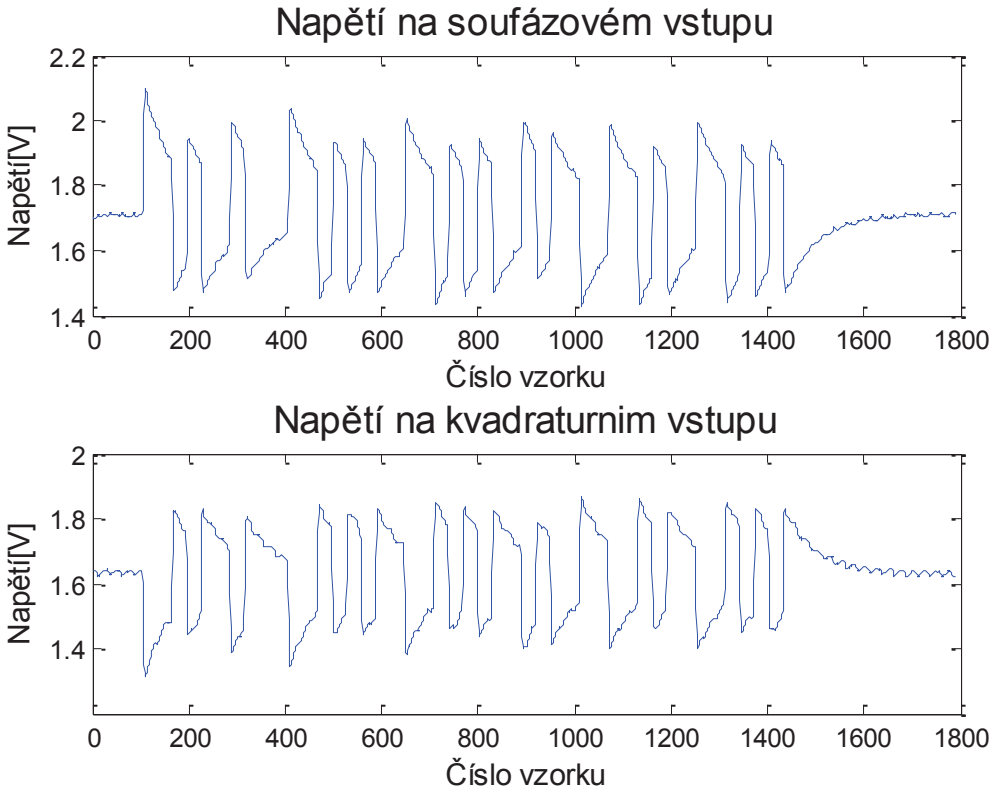
```
%vypocet filtrovaneho signalu - dynamickeho prahu
filtered = zeros(1,length(better_signal));
filtered(1)=better_signal(1)/q;
for l=2:length(better_signal)
    filtered(l)=(filtered(l-1)-filtered(l-1)*q + better_signal(l));
end;
filtered = filtered*(q);
```

Na základě porovnávání velikosti původního signálu s prahem vytvořeným exponenciální kumulací je vytvořeno pole obsahující pouze „-1“ (je-li vzorek signálu menší než práh) a „1“ (je-li vzorek signálu větší než práh). Výsledek je na obrázku 3.7, přičemž hodnoty jsou upraveny do rozsahu 0 až 1. Toto pole je pak procházeno a vždy je zaznamenána pozice prvku, pro který platí, že je jiný než prvek předchozí. Tím jsou vlastně detekovány hrany.

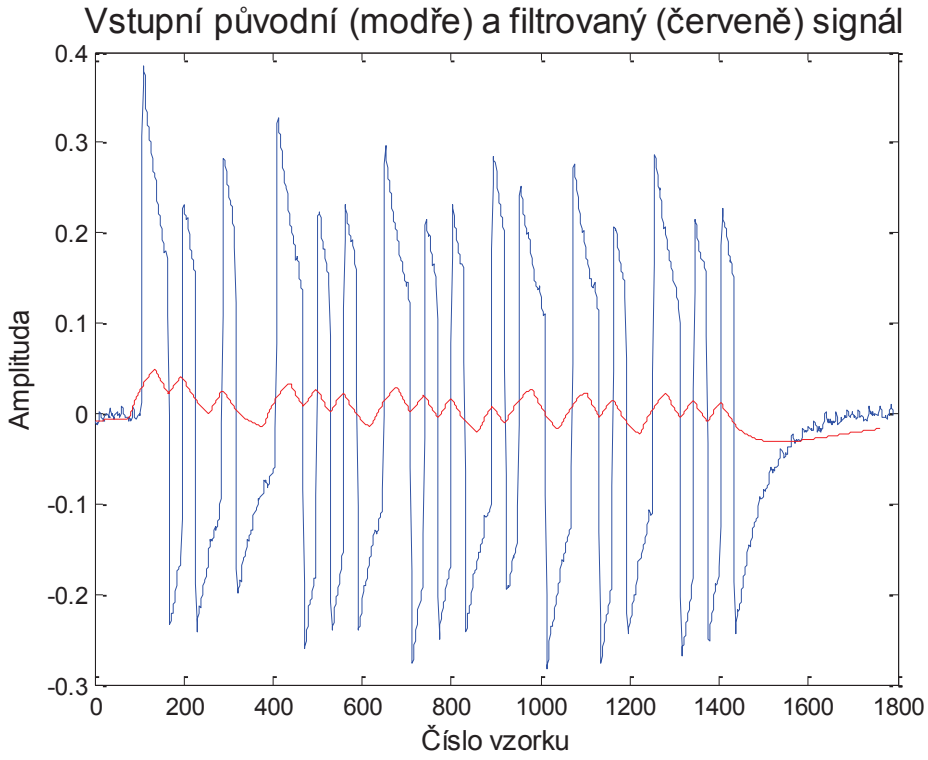
Po detekci hran jsou následně vypočteny rozdíly pozic hran, a pokud je tento rozdíl přibližně roven vypočtenému počtu vzorků na symbol, je rozhodnuto, že se jedná o symbol „1“. Pokud je roven přibližně polovině počtu vzorků na symbol, je zkontrolováno, jestli také následující rozdíl je roven polovině počtu vzorků na symbol, a pokud ano, je rozhodnuto, že se jedná o symbol „0“. Pokud je však rozdíl roven přibližně polovině počtu vzorků na symbol, ale následující rozdíl je roven jeden a půl násobku počtu vzorků na symbol, jedná se o posloupnost symbolů „0“ a „v“ s tím, že místo „v“ se do pole s dekódovanými symboly uloží „2“. Tento symbol je důležitý, protože je předposledním symbolem preambule a je od něj odvozeno, na které pozici začínají užitečné symboly čísla *RN16*. Ty jsou vykopírovány do dalšího pole, kde se tedy v každém prvku pole obdrží jeden symbol (tedy bit) čísla *RN16*. Toto pole je výstupem funkce. V poli se symboly se ještě může vyskytnout symbol „3“. Říká, že rozdíl pozic hran je více než jedena půl násobek počtu vzorků na symbol. Tento symbol se vyskytuje na konci pole se symboly, kde se signál vrací do ustáleného stavu na určitou stejnosměrnou úroveň. Pro lepší pochopení způsobu, jakým funguje vyhodnocování symbolů, je zdrojový kód části funkce, která toto provádí, uveden níže:

```
    %identifikace symbolu "0", "1", "v" a "dummy-1"
    k=1;
    zero_flag = 0;
    for m=1:length(delta_new)
        if ((delta_new(m) > (NS - NS*jitter_koef))...
            &&(delta_new(m) < (NS + NS*jitter_koef)))
            symbols(k)=1;
            k=k+1;
            zero_flag = 0;
        elseif ((delta_new(m) > (NS/2 - NS*jitter_koef))...
                &&(delta_new(m) < (NS/2 + NS*jitter_koef)))
            if (m+1)<length(delta_new)
                if ((delta_new(m+1) > (NS/2 - NS*jitter_koef))...
                    &&(delta_new(m+1) < (NS/2 + NS*jitter_koef))...
                    && (zero_flag == 0))
                    symbols(k) = 0;
                    k=k+1;
                    zero_flag = 1;
                elseif ((delta_new(m+1) > (NS/2 - NS*jitter_koef))...
                        &&(delta_new(m+1) < (NS/2 + NS*jitter_koef))...
```

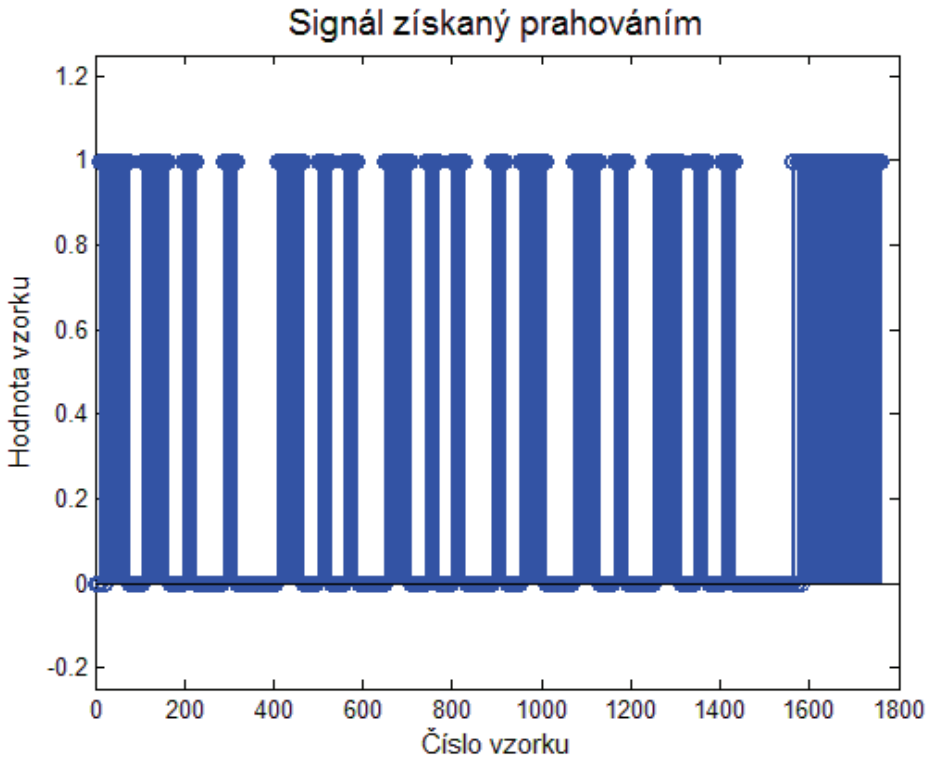
```
                    && (zero_flag == 1))
                zero_flag = 0;
            elseif ((delta_new(m+1) >...
                    (NS/2+NS - NS*jitter_koef))...
                    &&(delta_new(m+1) <...
                    (NS/2+NS + NS*jitter_koef)))
                symbols(k) = 0;
                k=k+1;
                symbols(k) = 2;
                k=k+1;
                zero_flag = 0;
            end;
        end;
    elseif (delta_new(m) > (NS/2+NS - NS*jitter_koef))
        if (m+1) > length(delta_new)
            symbols(k) = 3;
            k=k+1;
            zero_flag = 0;
        elseif ((delta_new(m+1) > (NS/2 - NS*jitter_koef))...
                &&(delta_new(m+1) < (NS/2 + NS*jitter_koef)))
            symbols(k) = 3;
            k=k+1;
        end;

    end;
end;
```



Obr. 3.5 Soufázová a kvadraturní složka signálu odezvy RN16 přijatého od tagu.



Obr. 3.6 Zpracovávaný signál odezvy RN16 před exponenciální kumulací (modře) a po exponenciální kumulaci (červeně).



Obr. 3.7 Průběh získaný porovnáním zpracovávaného signálu odezvy RN16 s plovoucím prahem.

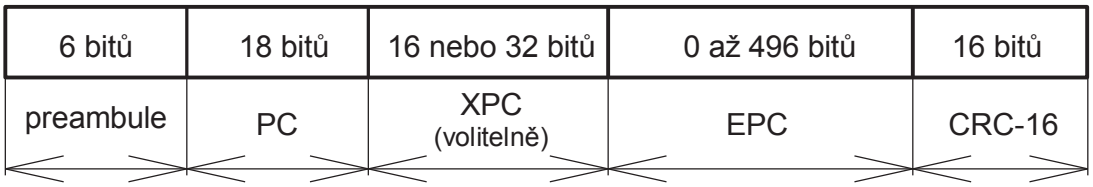
Pro variantu signálu, které odpovídají obrázky, se získá pole všech symbolů (včetně preambule) takovéto:

```
symbols = 1 1 0 1 0 2 1 0 0 1 1 0 0 1 0 1 1 1 0 1 1 0 0 3
```

Výsledné šestnáctibitové náhodné číslo *RN16* pak je:

```
RN16 = 0 0 1 1   0 0 1 0   1 1 1 0   1 1 0 0
```

Algoritmus pro dekódování identifikačního *EPC* čísla tagu funguje podobně, na závěr je však nutné rozdělit pole všech získaných symbolů na tři části. První část je tzv. *PC* pole, které udává především délku *EPC* čísla v šestnáctibitových slovech, druhá část je samotné číslo *EPC* a nakonec je to zabezpečení *CRC-16*. Bloková struktura podoby signálu obsahujícího *EPC* tagu je na obrázku 3.8, část označovaná *XPC* se u použitých tagů nevyužívá. Průběhy signálů soufázové a kvadraturní složky odezvy tagu jsou na obrázku 3.9. Obrázek 3.10 pak zobrazuje detail soufázové složky signálu obsahujícího *EPC* tagu. Je třeba poznamenat, že i zde se práce omezuje pouze na případ, že délka *EPC* je šest šestnáctibitových slov. Pro účel dekódování identifikačního čísla tagu byla vytvořena funkce `filtrace_EPC.m`. Tato funkce také implementuje výpočet *CRC-16* z *PC* a *EPC* a lze tak porovnat, jestli přečtené *CRC-16* a vypočtené *CRC-16* jsou shodné, a usoudit, jestli přečtená data jsou správná. Všechny výše zmíněné funkce spolu s několika soubory s navzorkovanými daty (jak signálu *RN16*, tak signálu *EPC*) jsou uloženy na CD přiloženém k práci.



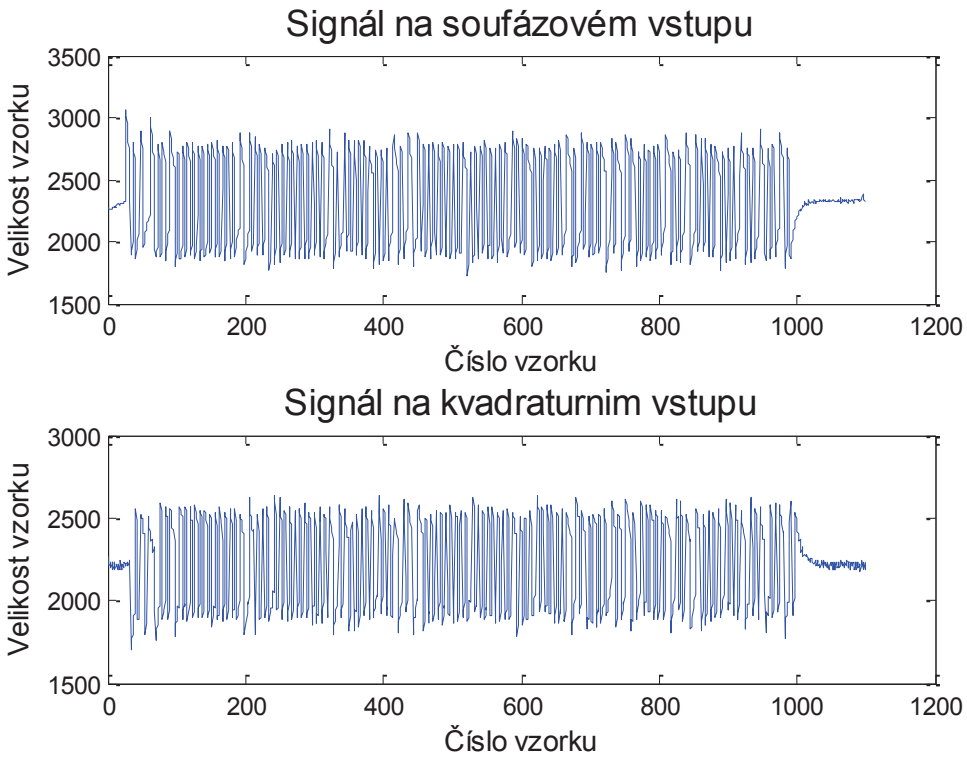
Obr. 3.8 Struktura signálu obsahujícího identifikační číslo (EPC) tagu.

Výsledné *EPC* pro tag, jemuž odpovídají průběhy signálů na obrázcích 3.9 a 3.10 je následující (v hexadecimálním vyjádření):

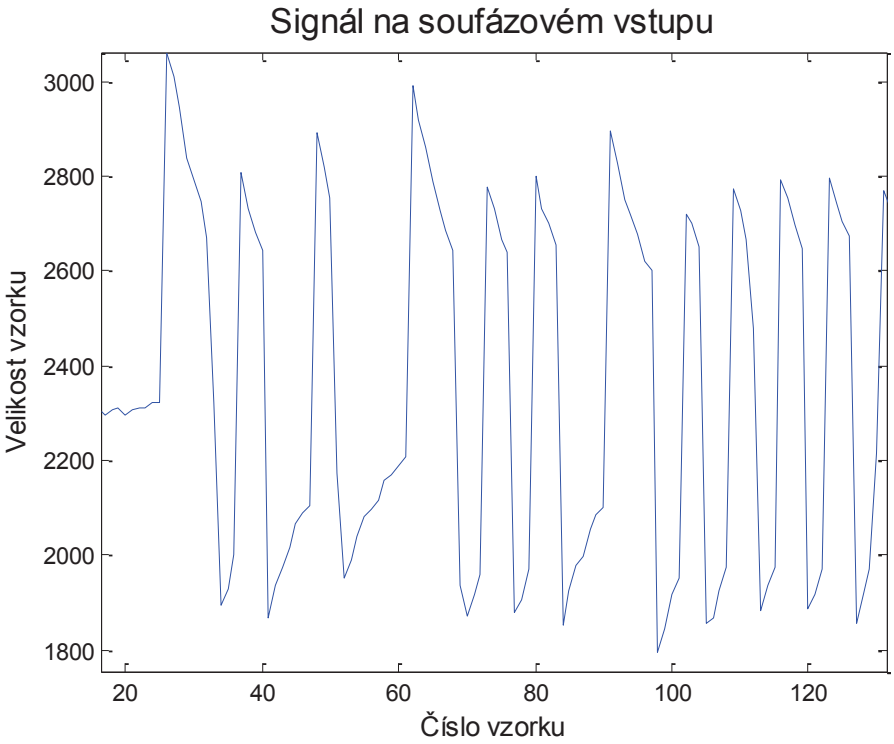
```
EPC = E200  3411  B802  0110  3436  8338
```

Zabezpečení *CRC-16* pak je (opět hexadecimálně):

```
CRC16 = 4666
```



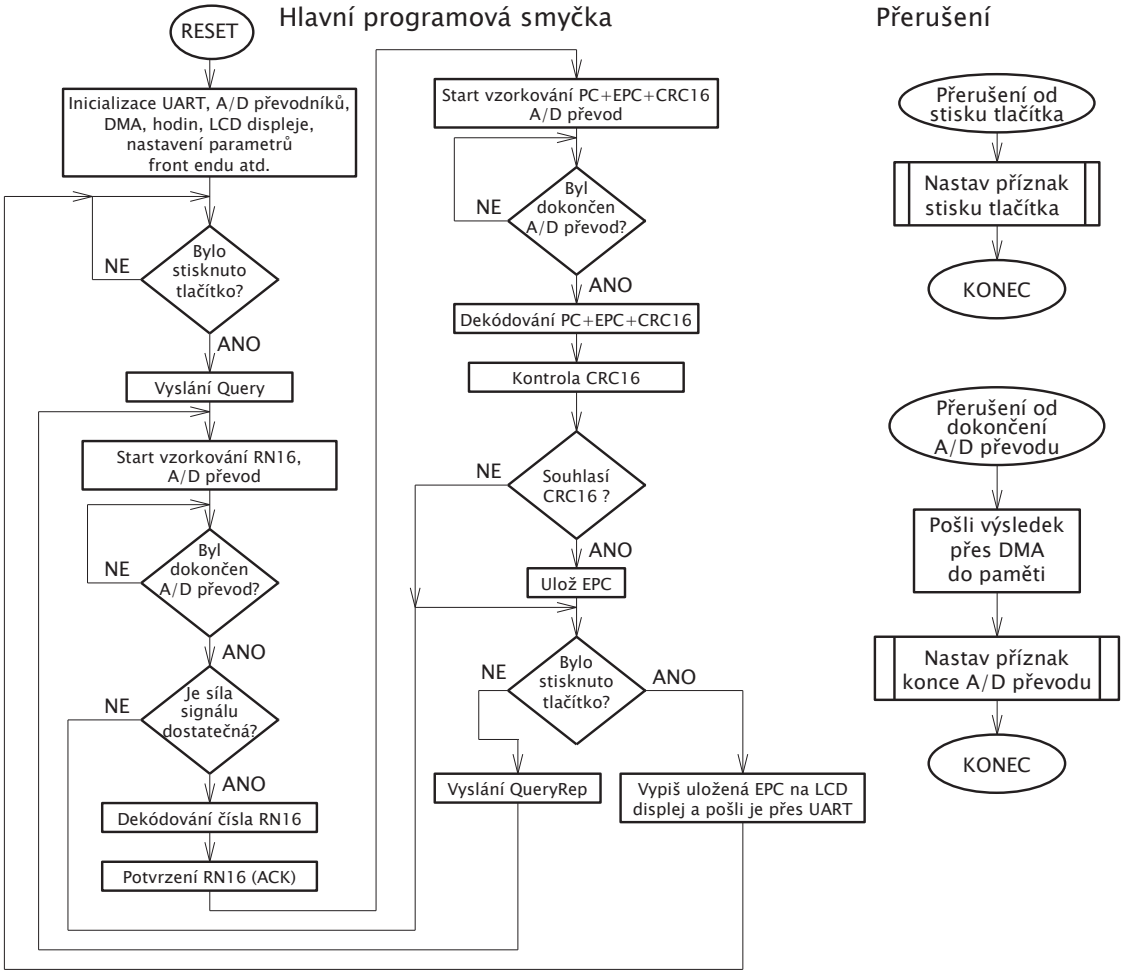
Obr. 3.9 Soufázová a kvadraturní složka signálu odezvy EPC přijatého od tagu.



Obr. 3.10 Detail začátku signálu soufázové složky odezvy EPC přijaté od tagu.

3.7 Popis programu pro mikrokontrolér

Zjednodušený vývojový diagram programu pro zvolený mikrokontrolér je na obrázku 3.11. Program je psán v jazyce C ve vývojovém prostředí Em::Blocks 2.30, ale protože nastavování všech periferií, systémů atd. mikrokontrolérů s jádrem ARM je již poměrně komplikované, je velice výhodné použít program STM32CubeMX^{TM}, který umožňuje jednoduše zvolit a nastavit požadované periferie pouhým výběrem v grafickém rozhraní a následně umožňuje vygenerovat kostru kódu pro zvolené vývojové prostředí na základě knihovny HAL (*HAL library*), která obsahuje většinu potřebných funkcí. Program STM32CubeMX je volně ke stažení na webových stránkách firmy STMicroelectronics [9]. Zdrojový kód celého dále popsaného programu je uložen na CD přiloženém k diplomové práci.



Obr. 3.11 Vývojový diagram programu pro mikrokontrolér.

Program mikrokontroléru pak funguje takto: po resetu je nejprve inicializována knihovna HAL, hodiny a všechny potřebné periferie prostřednictvím funkcí `MX_GPIO_Init()`, `MX_DMA_Init()`, `MX_ADC1_Init()`, `MX_ADC2_Init()`, `MX_UART5_Init()`, `MX_SPI5_Init()`, `MX_TIM3_Init()` a `SystemClock_Config()`, následně se provede nastavení integrovaných obvodů ve

front endu čtečky (konkrétně obvodu ADF9010), čímž se provede nastavení vysokofrekvenčních parametrů komunikace (nosný kmitočet, výkonové úrovně, zisk zesilovače v přijímací větvi, šířka pásma přijímacího filtru apod.). Toto nastavení zprostředkovávají funkce modulu `adf9010.c`. Tento modul je převzatý z původního programu pro mikrokontrolér Atmel ATmega8, který sloužil pro otestování front endu čtečky. Funkce modulu jsou např. `adf9010_init()` a `adf9010_init2()` pro inicializaci obvodu ADF9010, `adf9010_set_freq()` pro nastavení nosného kmitočtu čtečky, `adf9010_tx()` a `adf9010_rx()` pro nastavení vysílací resp. přijímací části front endu a nebo `adf9010_commit()`, který vlastně obvod ADF9010 aktivuje a spustí odeslání komunikačního příkazu front endem čtečky. Tento modul byl upraven pouze tak, že byly změněny názvy portů a pinů tak, aby bylo možné modul použít pro zvolený mikrokontrolér. Následuje ještě funkce pro inicializaci LCD displeje `TM_ILI9341_Init()` a funkce `TM_ILI9341_Rotate()`, která pootočí orientaci displeje o 90°.

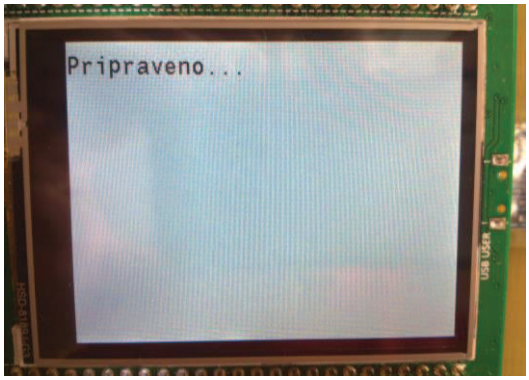
Po dokončení inicializací program v nekonečné smyčce testuje, zda došlo k externímu přerušení od stisku tlačítka. Pokud ano, vyšle se příkaz *Query*, prostřednictvím funkce `gen2_query_0()` z modulu `gen2.c`, která nastaví komunikační rychlost, kódování, počet slotů v inventarizační obrátce, populaci tagů, které se inventarizace účastní atd. a následně se funkcí `HAL_ADC_Start_DMA()` spustí oba A/D převodníky a čeká, až jsou nastaveny příznaky `EndConv_1` a `EndConv_2` indikující dokončení převodu každého z převodníků. Pokud jsou nastaveny, převodníky se vypnou funkcí `HAL_ADC_Stop_DMA()` a získaný blok vzorků je zpracován funkcí `RN16_det()` z modulu `detection.c`, která vrací jednak číselnou hodnotu přijatého čísla *RN16* a za druhé příznak, zda vstupní blok dat opravdu obsahuje odezvu *RN16*, případně ve které ze složek signálu (soufázové nebo kvadraturní) je signál silnější.

Funkce `RN16_det()` prakticky realizuje dekódovací algoritmus popsaný v kapitole 3.6, tedy nejprve na základě poměru rozdílu maxima a minima signálu zjistí, která složka obsahuje silnější signál, přičemž pokud tento rozdíl je menší než pevně stanovená mez, je vyhodnoceno, že data obsahují pouze šum, tedy že žádný z tagů neodpověděl na příkaz *Query*, je nastaven výstupní příznak na odpovídající hodnotu a zbytek obsahu funkce je přeskočen. Pokud jsou data vyhodnocena jako platná, je na ně aplikována exponenciální kumulace, jejíž výsledek funguje jako plovoucí práh pro porovnání, zda je signál ve vysoké nebo nízké úrovni. Ve výsledku tohoto porovnání jsou pak nalezeny hrany, tedy přechody mezi vysokou a nízkou úrovní. Podle rozdílů pozic těchto hran je pak vyhodnoceno o jaké symboly (bity) se jedná. Ze všech symbolů jsou pak vybrány ty, které představují číslo *RN16* a toto číslo tvoří výstup funkce.

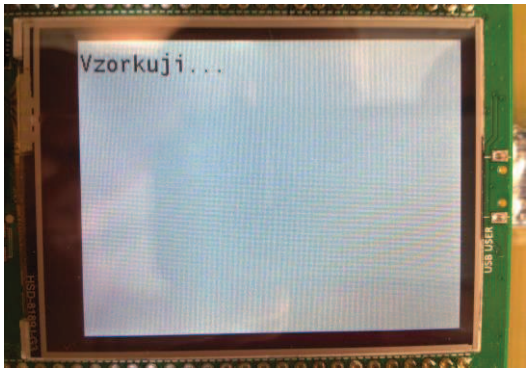
Číslo získané funkcí `RN16_det()` je potvrzeno tagu pomocí příkazu *ACK*, který implementuje funkce `gen2_ack()` opět z modulu `gen2.c`. Modul `gen2.c` byl také částečně převzat z původního programu pro mikrokontrolér AVR, byl však doplněn o realizaci příkazů *ACK*, *QueryRep* a s nimi související funkce `gen2_frame_sync()`, která vytváří na začátku příkazů synchronizační rámec (obrázek 2.7 v kapitole 2.1.3) a samozřejmě byl modul upraven tak, aby byl funkční na zvoleném mikrokontroléru. Po vyslání příkazu *ACK* se opět spustí A/D převodníky a očekává se přijetí signálu obsahujícího číslo *PC* (nejdůležitější jeho částí je část udávajícího délku *EPC*), číslo *EPC* (identifikační číslo tagu) a cyklickou šestnáctibitovou kontrolu *CRC-16*.

Vzorky získané A/D převodem jsou zpracovány funkcí `EPC_det()`, která vrací délku *EPC* v šestnáctibitových slovech, samotné *EPC* jako pole 32-bitových hodnot a číslo *CRC-16*. Tato funkce funguje podobně jako funkce `RN16_det()`, avšak už není třeba zjišťovat, která ze složek obsahuje silnější signál (to je známo z funkce `RN16_det()`). Dále ještě funkce vrátí jako pole 32-bitových hodnot čísla *PC* a *EPC* dohromady, tento výstup je použit ke kontrolnímu výpočtu *CRC-16* funkcí `CRC_Control()`. Pokud jsou si vypočtené *CRC-16* a *CRC-16* dekódované ze signálu tagu rovny, je dekódované *EPC* uloženo. Následně je vyslán příkaz *QueryRep* (funkce `gen2_query_rep()`), kterým čtečka sníží všem tagům čítače slotů a očekává přijetí *RN16* od dalšího tagu a celý cyklus se opakuje. Před tím je ale ještě rozsvícena zelená LED dioda. Červená LED dioda je rozsvícena na začátku vysílání každého příkazu čtečky a naopak zhasnuta do dokončení vysílání každého příkazu.

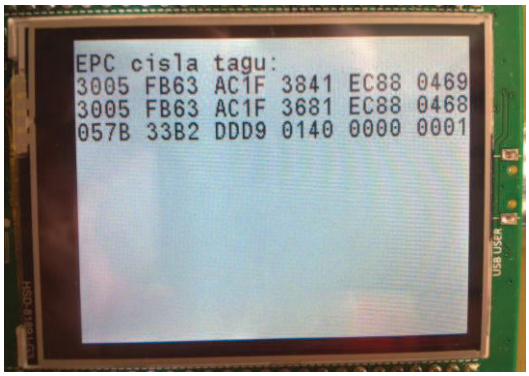
Pokud dojde k tomu, že data navzorkovaná A/D převodníky nejsou vyhodnocena jako *RN16*, ale jako šum, přeskočí se celý zbytek zpracování (tedy získání *EPC*, kontrola *CRC-16* atd.) a pokračuje se rovnou vysláním příkazu *QueryRep*. Zastavení vysílání příkazů se provádí opětovným stiskem tlačítka. Po tomto druhém stisku tlačítka se zhasne zelená LED dioda, uložená *EPC* čísla tagů jsou upravena pro přehlednější zobrazení, zformátována na textové řetězce standardní funkcí `sprintf()` a zobrazena na LCD displeji pomocí funkce `TM_ILI9341_Puts()`. Obrázky 3.12 až 3.14 ukazují podobu výpisu na displeji po resetu zařízení, po spuštění vzorkování a po zastavení vzorkování a vypsání *EPC* čísel tagů.



Obr. 3.12 Podoba výpisu na displeji po resetu (před spuštěním vzorkování).



Obr. 3.13 Podoba výpisu na displeji během spuštěného vzorkování.



Obr. 3.14 Displej se zobrazením EPC čísel zachycených tagů.

Zároveň s výpisem na displej jsou *EPC* čísla tagů také odeslána do počítače prostřednictvím periferie UART (funkce pro odeslání je `HAL_UART_Transmit()`), kde mohou být načtena v programu MATLAB. Za tímto účelem se nejprve posílá počet načtených *EPC*, aby bylo v programu MATLAB známo, kolik dat se má následně přečíst z daného sériového portu. Způsob, jakým jsou data v programu MATLAB čtena, je popsán v kapitole 3.10.

Co se týče zobrazení získaných identifikačních čísel tagů na LCD displeji, je třeba poznamenat, že knihovna pro ovládání LCD byla převzata z webových stránek (viz literatura [13]). I zde byla však nutná úprava, protože původní autor využíval jinou knihovnu než *HAL library*. Proto bylo nutné změnit funkce pro ovládání SPI sběrnice na standardní funkce knihovny *HAL library* (především se jedná o použití funkce `HAL_SPI_TransmitReceive()` v těle funkce `TM_SPI_Send()`).

Ve snaze o co nejkvalitnější digitalizaci měřeného signálu byla původní vzorkovací frekvence A/D převodníků nastavena na 2,4 milionu vzorku za sekundu (2,4MSample/s). Při následném vývoji a ladění programu se však ukázalo, že vzorků je takové množství, že jejich zpracování trvalo několikanásobně déle, než aby bylo možné dodržet požadavky na časování dle standardu (viz obrázek 2.22 a tabulka 2.9 v kapitole 2.3.5). Vzorkovací frekvence tedy musela být snížena osmkrát pro vzorkování *RN16* a dokonce dvanáctkrát pro vzorkování paketu obsahujícího *PC*, *EPC* a *CRC-16*. Tímto krokem byla snížena citlivost čtečky a tedy i její dosah.

Druhým výrazným problémem byla snaha využít vestavěnou USB periferii mikrokontroléru zvanou USB_OTG_HS, kterou lze nastavit na funkci virtuálního sériového portu, pro přenos navzorkovaných dat do programu MATLAB, kde měly být navrženy a ověřeny dekódovací algoritmy. Ve spojení s programem MATLAB však tato periferie vykazovala zvláštní chování, kdy po odpojení napájení vývojové desky a jeho opětovném připojení již program MATLAB nebyl schopen daný virtuální sériový port najít, přestože jiné jednodušší programy jej byly schopné najít a dokonce se k němu připojit. Opětovného připojení se dalo dosáhnout pouze restartováním programu MATLAB a přeprogramováním mikrokontroléru. Toto bylo nakonec vyřešeno přenosem dat přes UART s vnějším převodníkem na RS-232 nebo USB.

Přerušení od stisku tlačítka, dokončení A/D převodu, přerušení od časovače atd. jsou obsluhovány pomocí speciálních tzv. *callback* funkcí, protože knihovna HAL je postavena tak, že každé přerušení volá svou vlastní *callback* funkci, kterou si uživatel může editovat dle svých potřeb. Zde se konkrétně jedná o funkce

`HAL_GPIO_EXTI_Callback()` pro obsluhu přerušení od stisku tlačítka, `HAL_TIM_PeriodElapsedCallback()` pro obsluhu přerušení od vypršení periody časovače a `HAL_ADC_ConvCpltCallback()` pro obsluhu přerušení od dokončení A/D převodu.

3.8 Výsledky měření vytvořené komunikace

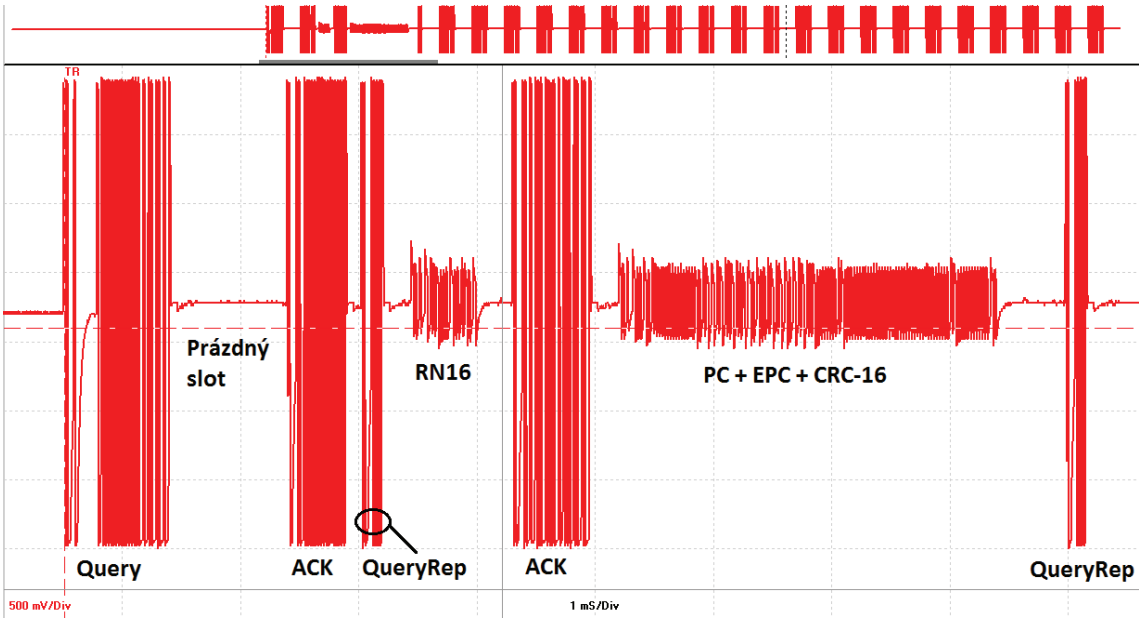
Tato kapitola dokumentuje výsledky měření komunikace mezi čtečkou a tagy. Při měření byly použity tři tagy zobrazené na obrázcích 3.15 až 3.17. Nosný kmitočet čtečky byl nastaven na 867,5MHz, použitá komunikační rychlost (BLF) byla 40kHz, kódování FM0 (viz kapitola 2.2.2). Číslo Q, které určuje počet slotů v inventarizační obrátce a ze kterého tagy vypočítávají, ve kterém slotu budou odpovídat, bylo nastaveno na hodnotu Q=3. V praxi se může stát, že některé tagy vůbec neodpoví. V tomto případě to znamená, že odpoví buď všechny tři tagy, nebo jen dva nebo pouze jeden a je také možné, že neodpoví žádný z tagů.

Obr. 3.15 Tag UPM RAFLATAC.

Obr. 3.16 Tag UPM Short Dipole na podkladu z průhledné fólie.

Obr. 3.17 Tag UPM Short Dipole na papírovém podkladu.

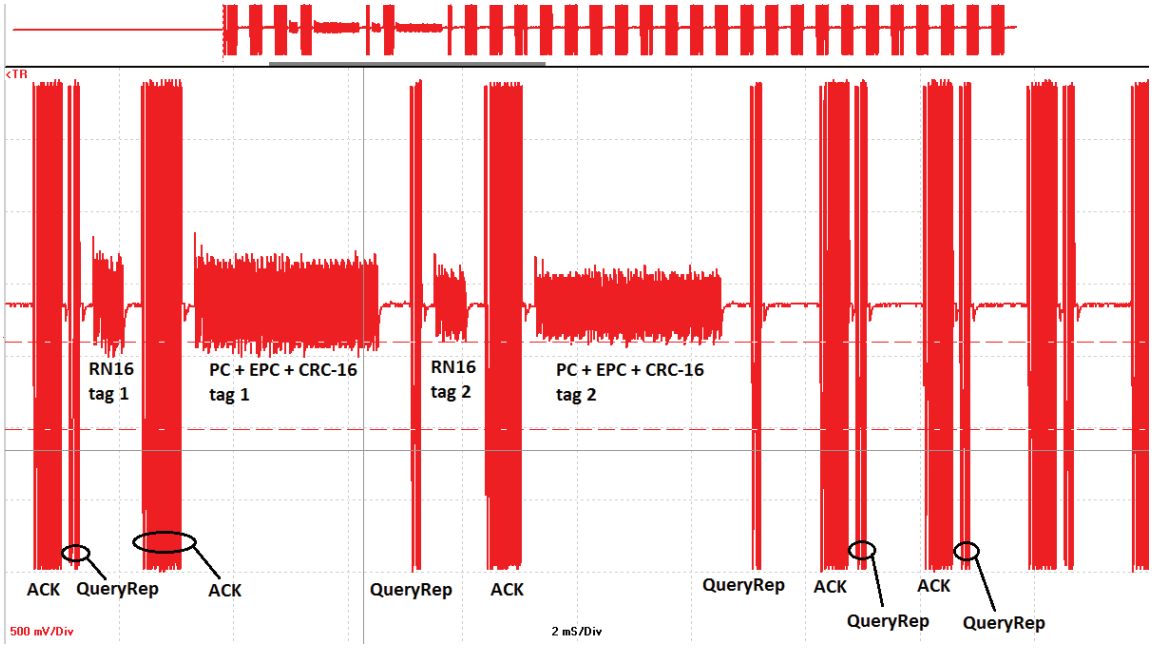
Na obrázku 3.18 je vidět průběh komunikace, kdy odpověděl právě jeden tag, a to ve druhém slotu. Je patrná odezva *RN16* stejně jako paket obsahující údaje *PC*, *EPC* a *CRC-16*. Je také patrné, že v ostatních slotech, kde žádný tag nevysílá, čtečka nečeká na paket obsahující číslo *EPC* a rovnou vysílá příkaz *QueryRep*.



Obr. 3.18 Průběh komunikace s odpovědí právě jednoho tagu (napěťová osa – 500mV/dílek, časová osa – 1ms/dílek).

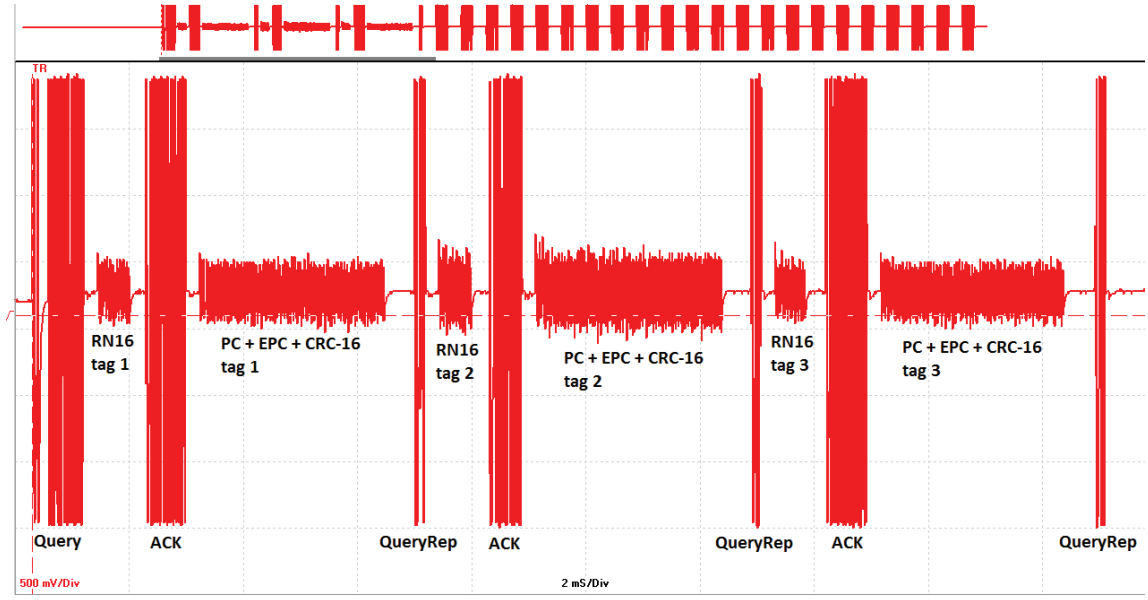
Na obrázku 3.18 je také patrné, že čtečka potvrzuje příkazem *ACK* také prázdný slot (slot, kde není žádné *RN16*). To je dáno tím, že práh rozhodující, zda je ve slotu přítomen užitečný signál nebo se jedná pouze o šum, je nastaven na poměrně nízkou hodnotu. Algoritmus pak šum o vyšší napěťové úrovni než je tento práh vyhodnotí jako platný signál *RN16*, pokusí se jej dekódovat a potvrdí jej. Řešením by bylo nastavit rozhodovací práh na vyšší hodnotu. To by však na druhou stranu mohlo mít za následek, že pokud by signál odezvy tagu byl slabý, mohl by být považován za šum a čtečka by nepotvrzovala ani platný signál *RN16*. Tag by tím pádem nebyl zaznamenán, což je nežádoucí.

Obrázek 3.19 pak ukazuje případ, kdy odpověděly dva tagy. V tomto případě odpovídají ve dvou bezprostředně sousedících slotech, což je spíše neobvyklé. Ve většině případů bude mezi odpověďmi několik slotů prázdných.

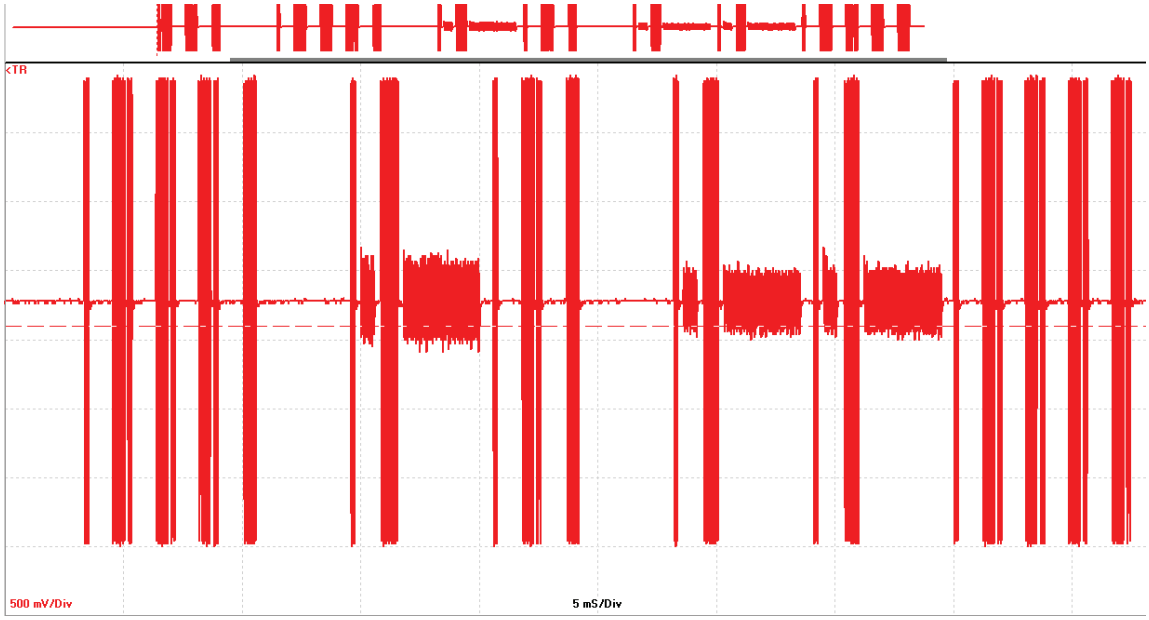


Obr. 3.19 Průběh komunikace s odpovědí dvou tagů – tagy odpovídají v bezprostředně sousedících slotech (napěťová osa – 500mV/dílek, časová osa – 2ms/dílek).

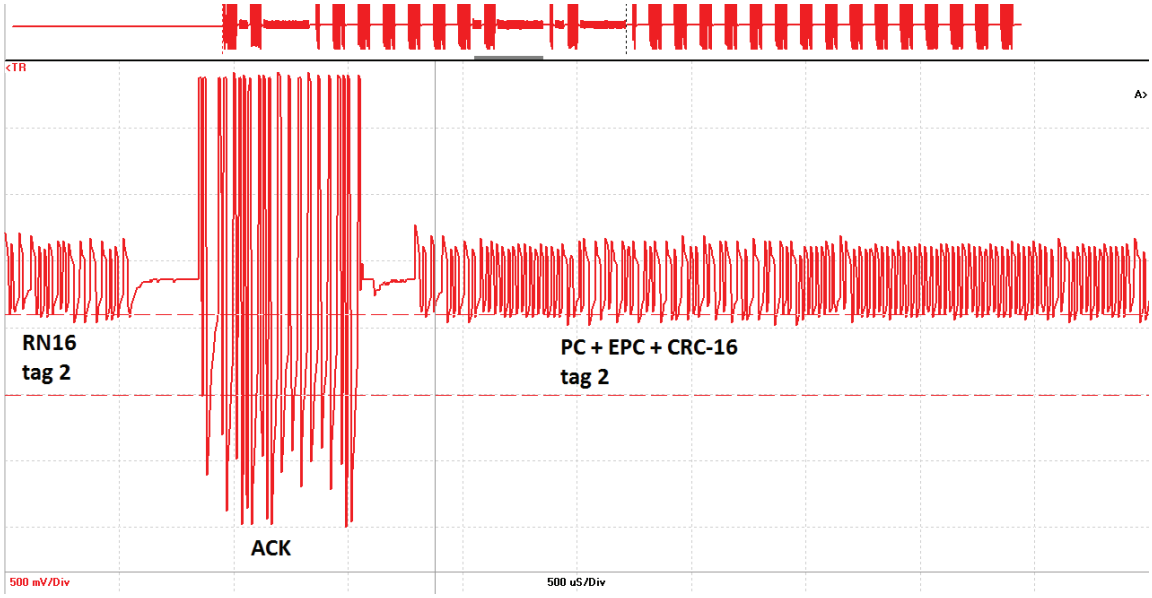
Obrázky 3.20 a 3.21 ukazují dva zachycené případy, kdy došlo k odpovědi všech tří tagů. Podle amplitudy signálu je vidět, že se opravdu jedná o tři různé tagy, a také je dobře patrné, že výběr slotu, ve kterém bude tag odpovídat, je náhodný (resp. pseudonáhodný). Obrázek 3.22 pak zobrazuje detail signálu odpovědi druhého tagu z obrázku 3.20.



Obr. 3.20 Průběh komunikace s odpovědí tří tagů – verze 1 (napěťová osa – 500mV/dílek, časová osa – 2ms/dílek).

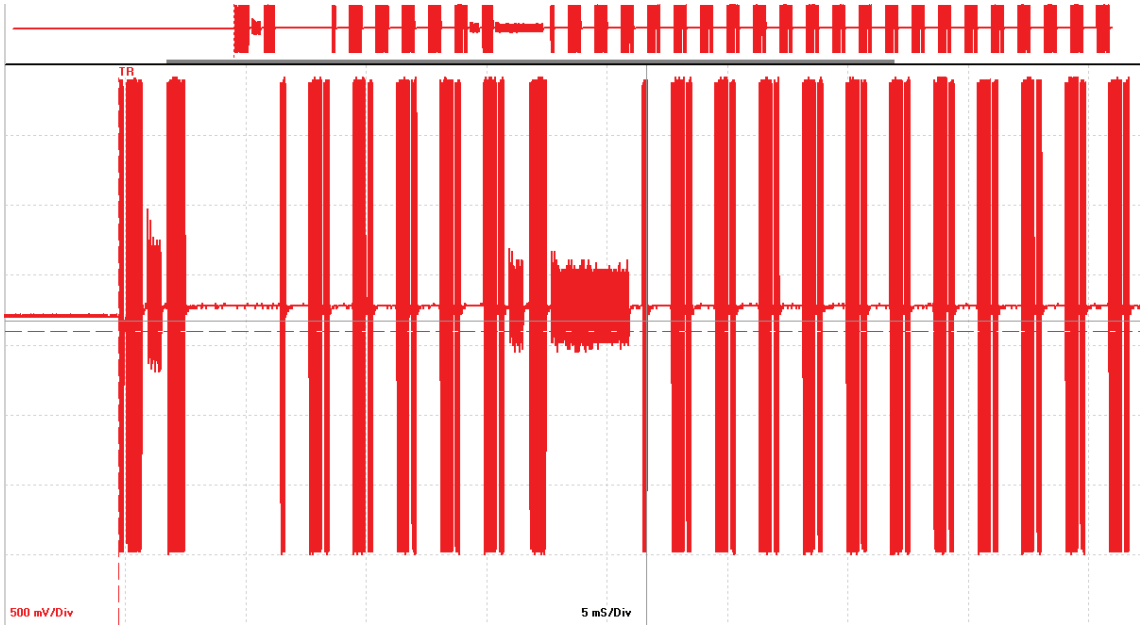


Obr. 3.21 Průběh komunikace s odpovědí tří tagů – verze 2 (napěťová osa – 500mV/dílek, časová osa – 5ms/dílek).

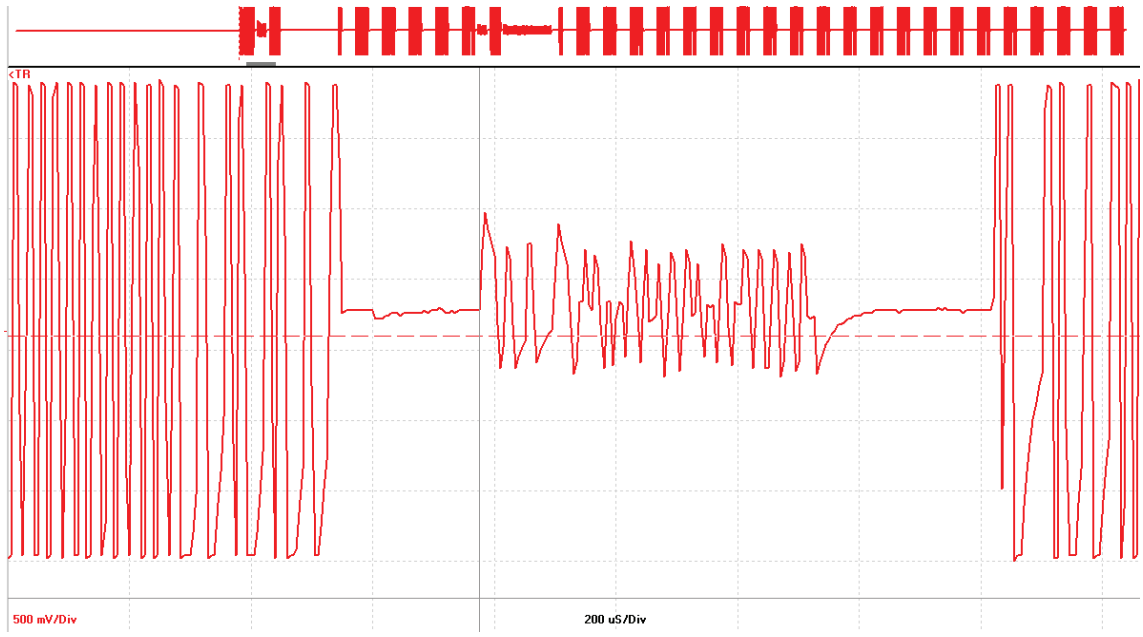


Obr. 3.22 Detail příkazu ACK a paketu obsahujícího EPC druhého tagu v inventarizační obrátce s odpověďmi tří tagů (napěťová osa – 500mV/dílek, časová osa – 500µs/dílek).

Velmi zajímavý je průběh na obrázku 3.23, kde kromě kompletní odpovědi jednoho tagu je vidět také *RN16* v prvním slotu, po kterém však nenásleduje blok obsahující *EPC*. Detail na obrázku 3.24 prozrazuje, že se jedná o případ, kdy dva tagy začaly odpovídat současně. Je vidět, že preambule je nepoškozená (protože je vždy a u všech tagů stejná), zatímco zbytek signálu je znehodnocen kolizí mezi odpověďmi tagů.



Obr. 3.23 Průběh komunikace s jednou kompletní odpovědí tagu a dvěma překrývajícími se RN16 – odpověď dvou tagů zároveň (napěťová osa – 500mV/dílek, časová osa – 5ms/dílek).



Obr. 3.24 Detail kolidujících RN16 v případě současné odpovědi dvou tagů (napěťová osa – 500mV/dílek, časová osa – 200µs/dílek).

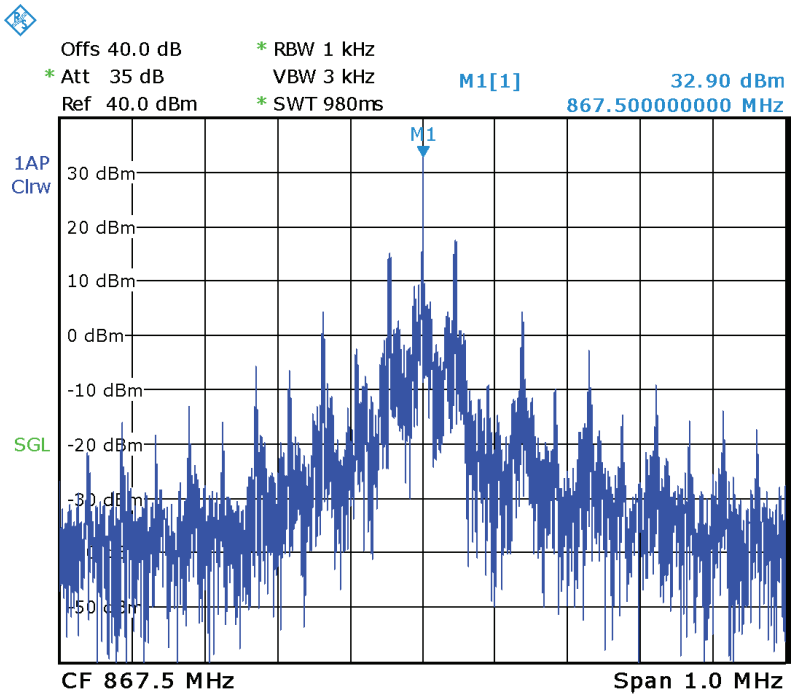
3.9 Parametry vyvinuté čtečky

Níže uvedený výčet stručně shrnuje vybrané parametry vyvinuté čtečky. Jedná se jednak o parametry elektrické, za druhé o parametry časové, tedy především časové prodlevy mezi příkazy čtečky, které zároveň vypovídají o tom, jak dlouho trvá

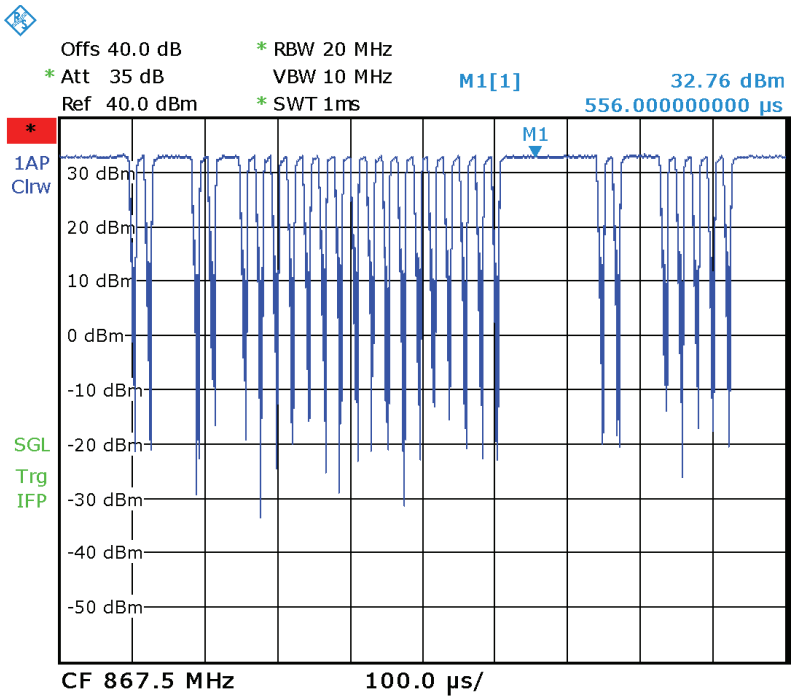
zpracování signálů přijatých od tagů, a za třetí se jedná o obecná nastavení čtečky. Parametry čtečky jsou tedy takovéto:

- Napájecí napětí: 12V
- Proudový odběr: 2,05A
- Průměrná časová prodleva mezi *RN16* a příkazem *ACK* (T_{2A}): 305µs
- Průměrná časová prodleva mezi *PC* + *EPC* + *CRC-16* a příkazem *QueryRep* (T_{2B}): 573µs
- Kódování: FM0
- Komunikační rychlost tag → čtečka (BLF): 40kHz
- Nastavená nosná frekvence: 867,5MHz
- Vysílací výkon čtečky: 2W (33dBm)
- Dosah čtečky pro spolehlivé přečtení tagu: přibližně 1m až 1,5m pro jediný tag v poli antény čtečky (závisí na použitém tagu), pro více tagů je vzdálenost pro přečtení alespoň jednoho tagu zhruba třetinová.

Obrázek 3.25 zobrazuje spektrum modulovaného vysílaného signálu. Protože modulační signál je obdélníkový, obsahuje spektrum mnoho vedlejších spektrálních složek. Přesto jsou patrná obě postranní pásma v kmitočtovém odstupu ±40kHz od hlavní nosné. Také je vidět, že výkon nosné je opravdu přibližně 33dBm. Obrázek 3.26 pak zobrazuje časový průběh příkazů *ACK* a *QueryRep* získaný ze spektra signálu při nastavené nulové šířce frekvenčního rozmítání spektrálního analyzátoru (*zero frequency span*).



Obr. 3.25 Spektrum modulovaného signálu vysílaného čtečkou.



Obr. 3.26 Časový průběh příkazů ACK a QueryRep získaný ze spektra signálu při nastavené nulové šířce frekvenčního rozmítání spektrálního analyzátoru.

3.10 Zobrazení získaných dat v programu MATLAB

Po odeslání z mikrokontroléru přes UART do počítače je třeba obdržená data zobrazit v programu MATLAB. Protože použitý převodník UART na USB se po připojení k počítači chová jako virtuální sériový port, stačí, aby se v programu MATLAB tento port otevřel, data se z něj načetla do proměnné a pak už jen stačí data vhodně zformátovat a vypsat. Pro tento účel byla vytvořena v programu MATLAB funkce `EPC_read()`. Zdrojový kód funkce je uveden níže:

```
function [] = EPC_read()

clear all; close all; clc;
s = serial('COM6');

set(s, 'InputBufferSize', 1500000);
set(s,'BaudRate', 115200);
set(s, 'Timeout', 10);
set(s, 'DataBits', 8);
try
fopen(s);
catch ME
    fclose(s);
    delete(s);
    clear s;
    error('Chyba');
end;
number_of_EPCs_bytes = fread(s,1,'uint32');
EPC_array = fread(s,number_of_EPCs_bytes/2,'uint16');
fclose(s);
```

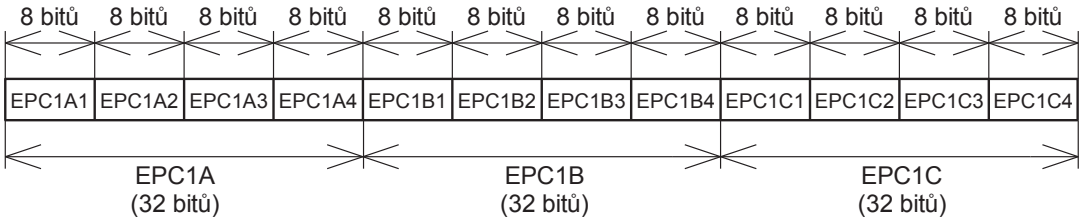
```
delete(s);
clear s;

q=0;
for q=0:(number_of_EPCs_bytes/12)-1
     sprintf('EPC%d je: %.4X %.4X %.4X %.4X %.4X %.4X', q+1,...
     EPC_array(2+q*6), EPC_array(1+q*6), EPC_array(4+q*6),...
     EPC_array(3+q*6), EPC_array(6+q*6),EPC_array(5+q*6))
end;
```

Funkce nejprve vyčistí pracovní plochu programu MATLAB, vytvoří spojení s příslušným sériovým portem (zde COM6), nastaví parametry přenosu, tj. velikost vstupního zásobníku pro přijatá data, čas, po kterém je spojení automaticky uzavřeno a počet bitů každého přijatého znaku (čísla). Poté jsou z portu do 32-bitové proměnné přečteny čtyři bajty udávající délku přijatých dat v bajtech. Následně jsou samotná data ze sériového portu načtena do 16-bitové proměnné `EPC_array()` a spojení je ukončeno a zrušeno.

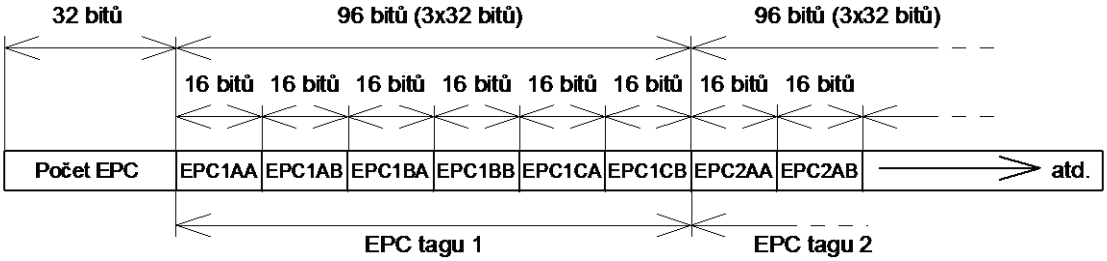
V proměnné `EPC_array()` nyní každých šest pozic představuje jedno *EPC* číslo. V cyklu `for` je pak každá šestice 16-bitových pozic pomocí funkce `sprintf()` vypsána na hlavní plochu programu MATLAB jako šest čtveřic hexadecimálních cifer, přičemž čtveřice jsou od sebe odděleny mezerou. Výpis na hlavní pracovní ploše programu MATLAB je pak pro případ zaznamenání dvou tagů například následující:

```
EPC1 je: E200 3411 B802 0110 3436 8338


EPC2 je: 3005 FB63 AC1F 3681 EC88 0468
```

Obrázek 3.27 naznačuje, že v paměti mikrokontroléru jsou *EPC* čísla uložena jako trojice 32-bitových čísel, ale do počítače jsou zasílána po osmi bitech. V programu MATLAB jsou však čtena po šestnácti bitech, takže výsledkem je, že původní trojice 32-bitových čísel je rozdělena na šestici 16-bitových čísel, jak naznačuje obrázek 3.28.



Obr. 3.27 Formát odesílání dat přes UART do počítače.



Obr. 3.28 Organizace dat při čtení EPC čísel tagů v programu MATLAB.

ZÁVĚR

Je možno říci, že zadání diplomové práce se podařilo splnit. Bylo navrženo blokové schéma propojení mikrokontroléru s jádrem ARM a front endem čtečky EXIN-1, program mikrokontroléru implementuje základní příkazy standardu EPC Global Class-1 Generation-2 a díky vytvořenému algoritmu je navržené zařízení nyní schopno přečíst identifikační čísla několika tagů nacházejících se v poli působnosti čtečky a zobrazit je na připojeném LCD displeji. Tím je demonstrována základní funkce RFID čtečky.

Jistou nevýhodou vytvořeného zařízení je absence filtrace vysílaných signálů a schopnost zařízení pracovat pouze na jediné komunikační rychlosti a s jedním typem kódování. Důvodem, proč nebyly možnosti zpracování rozšířeny, bylo časové zdržení vzniklé řešením problémů s tím, že dekódovací algoritmus byl pro velký počet vzorků příliš pomalý, takže nebylo možné dodržet časové parametry komunikace, a také problémů s chováním sběrnice USB, která měla být použita pro přenos dat do PC. Implementace jiných komunikačních rychlostí by však byla celkem snadná. Implementace jiných možností kódování by byla obtížnější, avšak nevyžadovala by významný zásah do konceptu celého programu mikrokontroléru. Nejobtížnější by z tohoto pohledu byla implementace filtrace vysílaných signálů, která by vyžadovala významný zásah do programu mikrokontroléru a pravděpodobně použití externího digitálně – analogového převodníku místo interního převodníku mikrokontroléru.

Na druhou stranu lze konstatovat, že v současném stavu je vytvořené zařízení funkční a je schopno spolehlivě přečíst identifikační číslo tagu až na vzdálenost 1,5metru (v závislosti na použitém tagu). Identifikační čísla zaznamenaných tagů jsou přehledně zobrazena na připojeném LCD displeji. Na něm je také vypisována informace o tom, zda se čtečka nachází ve stavu po resetu nebo ve stavu zaznamenávání dostupných tagů nebo ve stavu výpisu získaných identifikačních čísel. Stav čtečky je navíc indikován pomocí LED diod. Získaná identifikační čísla je také možné odeslat pomocí sériové linky do počítače, což je funkce nad rámec zadání práce. Osobní přínos práce pak spočívá v získání zkušeností se zpracováním signálu pomocí mikrokontroléru. Současný stav zařízení umožňuje další vývoj, na jehož konci by mohla vzniknout plnohodnotná RFID čtečka použitelná v praktických aplikacích.

LITERATURA

[1] POVALAČ, A., ŠEBESTA, J. *Experimental Front End for UHF RFID Reader*. [online]. Elektrorevue-Internetový časopis (http://www.elektrorevue.cz), roč. 2011, č.1, s. 55-59. Dostupné z www: http://goo.gl/z8uOg2.

[2] EPCglobal Inc. *EPC™ Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860 MHz – 960 MHz, Version 1.2.0* [online]. 2008 – [cit. 1. prosince 2014]. Dostupné z www: http://www.gs1.org/gsmp/kc/epcglobal/uhfc1g2.

[3] DOBKIN, Daniel Mark. *The RF in RFID: passive UHF RFID in practise*. Burlington: Newnes, 2008, ix, 493 s. ISBN 978-0-7506-8209-1.

[4] Analog Devices, Inc. *14-Bit, 20MSPS/ 40MSPS/ 65MSPS Dual A/D Converter AD9248*. [online]. 2010 – [cit. 1. prosince 2014]. Dostupné z www: http://www.analog.com/static/imported-files/data_sheets/AD9248.pdf.

[5] Analog Devices, Inc. *14-Bit, 20MSPS/ 40MSPS/ 65MSPS/ 80MSPS, 1.8V Dual Analog-to-Digital Converter AD9251*. [online]. 2009 – [cit. 1. prosince 2014]. Dostupné z www: http://www.analog.com/static/imported-files/data_sheets/AD9251.pdf.

[6] STMicroelectronics, Inc. *UM1670 User manual, Discovery kit for STM32F429/439 lines*. [online]. 2013 - [cit. 9. prosince 2014]. Dostupné z www: http://www.st.com/web/catalog/tools/FM116/SC959/SS1532/LN1848/PF259090.

[7] STMicroelectronics, Inc. *STM427xx, STM429xx. ARM Cortex-M4 32b MCU+FPU, 225MIPS, up to 2MB Flash/256+4KB RAM, USB OTG HS/FS, Ethernet, 17 TIMs, 3 ADCs, 20 comm. Interfaces, camera & LCD-TFT*. [online]. 2014 – [cit. 9. prosince 2014]. Dostupné z www: http://www.st.com/stonline/stappl/resourceSelector/app?page=fullResourceSelector&doctype=datasheet&SeriesID=1577.

[8] STMicroelectronics, Inc. *RM0090 Reference manual, STM32F405xx/07xx, STM32F415xx/17xx, STM32F42xxx and STM32F43xxx advanced ARM®-based 32-bit MCUs*. [online]. 2014 – [cit. 10. prosince 2014]. Dostupné z www: http://www.st.com/stonline/stappl/resourceSelector/app?page=fullResourceSelector&doctype=reference_manual&SeriesID=1577.

[9] STMicroelectronics, Inc. *STM32CubeMX, STM32Cube initialization code generator*. [online]. 2014 - [cit. 9. prosince 2014]. Dostupné z www: http://www.st.com/web/en/catalog/tools/PF259242.

[10] Analog Devices, Inc. *900MHz ISM Band Analog RF Front End ADF9010*. [online]. 2008 – [cit. 15. prosince 2014]. Dostupné z www: http://www.analog.com/static/imported-files/data_sheets/ADF9010.pdf.

[11] Analog Devices, Inc. *700MHz to2.7GHz Quadrature Demodulator ADL5382*. [online]. 2012 – [cit. 15. prosince 2014]. Dostupné z www: http://www.analog.com/static/imported-files/data_sheets/ADL5382.pdf.

[12] European Telecommunications Standarts Institute. *ETSI EN 302 208-1 V1.4.1. Electromagnetic compatibility and Radio spectrum Matters (ERM); Radio frequency Identification Equipment operating in the band 865 MHz to 868 MHz with power levels up to 2 W; Part 1: Technical requirements and methods of measurement*. [online]. 2011 – [cit. 23. března 2015]. Dostupné z www: http://www.etsi.org/deliver/etsi_en/302200_302299/30220801/01.04.01_40/.

[13] MAJERLE, T. *STM32F4 Discovery. Libraries and tutorials for STM32F4 series MCUs by Tilen Majerle – Library 08 – ILI9341 LCD for STM32F4*. [online]. 2015 – [cit. 1. května 2015]. Dostupné z www: http://stm32f4-discovery.com/2014/04/library-08-ili9341-lcd-on-stm32f429-discovery-board/.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

f	kmitočet obecně
f_0	rozdílový kmitočet
f_C	kmitočet vysokofrekvenčního nosného signálu
f_n	kmitočet vysokofrekvenčního nosného signálu n-tého rádiového kanálu
I	elektrický proud obecně
R	elektrický odpor obecně
R_x	rezistor s indexem x
$s_x(t)$	časový průběh signálu s indexem x
Tari	referenční časový interval (*Type A Reference Interval*)
T_{pri}	perioda opakování pulzu při komunikaci tag→čtečka (*Backscatter – link pulse – repetition interval)*
t	čas
U	elektrické napětí obecně
U_{CM}	souhlasné elektrické napětí (*Common Mode Voltage*)
U_{IN}	elektrické napětí na invertujícím vstupu rozdílového zesilovače pro soufázovou složku signálu
U_{IP}	elektrické napětí na neinvertujícím vstupu rozdílového zesilovače pro soufázovou složku signálu
U_{QN}	elektrické napětí na invertujícím vstupu rozdílového zesilovače pro kvadraturní složku signálu
U_{QP}	elektrické napětí na neinvertujícím vstupu rozdílového zesilovače pro kvadraturní složku signálu
$U_{Q\_VÝST}$	výstupní napětí rozdílového zesilovače pro kvadraturní složku signálu

A/D	analogově – digitální
ACK	potvrzení (*Acknowledge*)
ADC	analogově – digitální převodník (*Analog-to-Digital Converter*)
ASK	amplitudové klíčování (*Amplitude shift keying*)
BLF	komunikační rychlost ve směru tag→čtečka (*Backscatter - link frequency*)
BW	šířka pásma (*Bandwidth*)

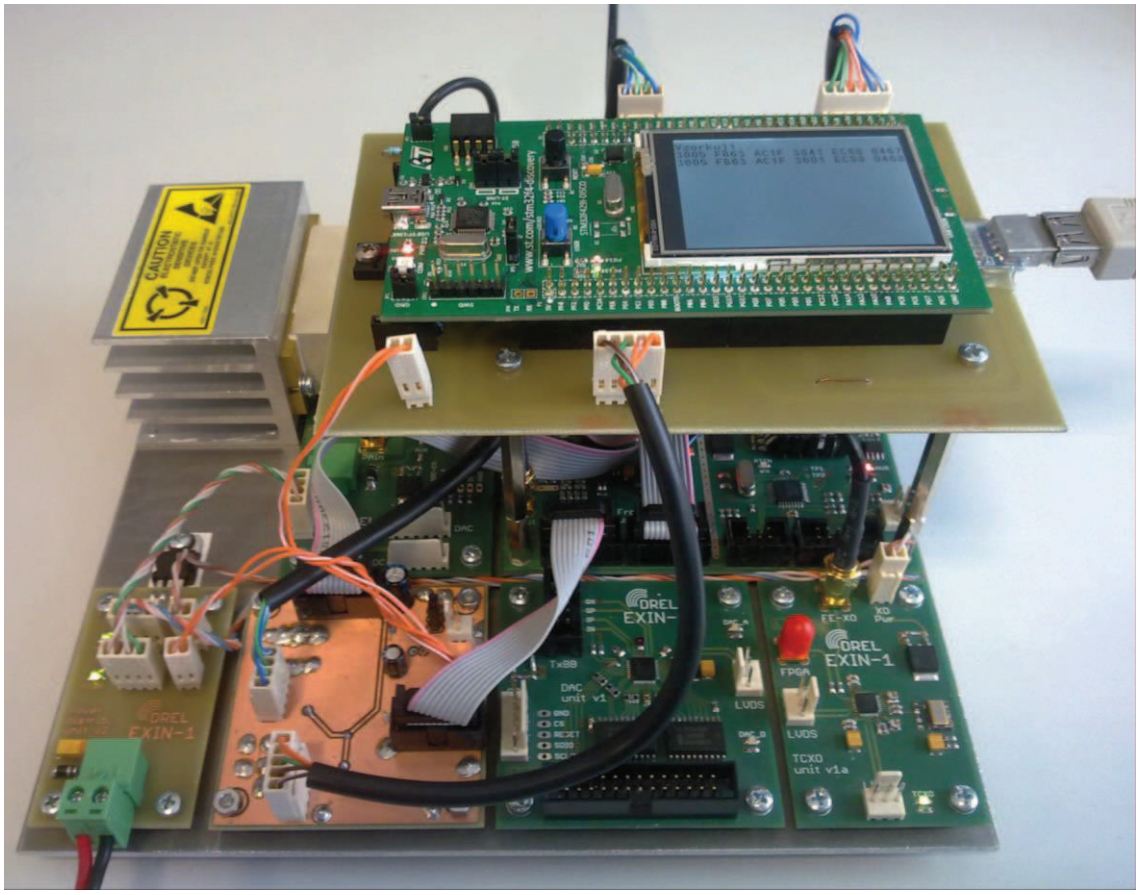
CHxx	kanál s indexem xx (*Channel xx*)
CRC	cyklický redundantní kód (*Cyclic redundancy check*)
CW	nemodulovaná nosná vlna (*Continuous wave*)
D/A	digitálně – analogový
DAC	digitálně – analogový převodník (*Digital-to-Analog Converter*)
DC	stejnosměrná veličina (*Direct Current*)
DMA	jednotka pro přímý přístup do paměti (*Direct Memory Access*)
DR	dělicí poměr (*Divide ratio*)
DSB - ASK	amplitudové klíčování se dvěma postranními pásmy (*Double sideband amplitude shift keying*)
EBV	adresování pomocí vektorů s rozšiřujícím bitem (*Extension bit vector*)
EN	evropská norma (*European Norm*)
EPC	identifikační číslo tagu (*Electronic product code*)
ERP	efektivně vyzářený výkon (*Effective Radiated Power*)
ETSI	Evropský institut pro normalizaci v telekomunikacích (*European Telecommunications Standards Institute*)
FIFO	paměť, u které jsou dat čtena ve stejném pořadí, jako jsou zapisována (*First-in-first-out*)
FIR	filtr s konečnou impulzní charakteristikou (*Finite impulse response*)
FS	plná rychlost (*Full speed*)
FT	tolerance kmitočtu (*Frequency tolerance*)
HF	vysoký kmitočet (*High Frequency*)
HS	vysoká rychlost (*High speed*)
I/Q	soufázový/kvadraturní (*In-phase/Quadrature*)
IC	integrovaný obvod (*Integrated Circuit*)
LCD	displej s tekutými krystaly (*Liquid Crystal Display*)
LED	svítící dioda (*Light-emitting diode*)
LF	nízký kmitočet (*Low Frequency*)
LSB	nejméně významný bit (*Least Significant Bit*)
MCU	mikrokontrolér (*Microcontroller unit*)
MSB	nejvíce významný bit (*Most Significant Bit*)
NAK	nepotvrzení (*Non-acknowledge*)
OTG	druh USB komunikace (*On-the-go*)

PC	osobní počítač (*Personal Computer*) nebo nastavení protokolu (*Protocol control*)
PIE	pulzně - intervalové kódování (*Pulse-interval encoding*)
PLL	smyčka fázového závěsu (*Phase lock loop*)
PR - ASK	fázově reverzní amplitudové klíčování (*Phase-reversal amplitude shift keying*)
PSK	fázové klíčování (*Phase shift keying*)
RAM	paměť s náhodným přístupem (*Random Access Memory*)
RFID	radiofrekvenční identifikace (*Radio frequency identification*)
RTcal	kalibrační symbol komunikace čtečka→tag (*Interrogatorr-to-tag calibration symbol*)
SL	příznak výběru (*Selection flag*)
SPI	sériového periferní rozhraní (*Serial Peripheral Interface*)
SRAM	statická paměť s náhodným přístupem (*Static Random Access Memory*)
SSB - ASK	amplitudové klíčování s jedním postranním pásmem (*Single sideband amplitude shift keying*)
TFT	technologie s vrstvou tenkých tranzistorů (*Thin-film-transisitor*)
TID	paměťová banka sloužící k identifikaci tagu (*Tag-identification*)
TRcal	kalibrační symbol komunikace tag→čtečka (*Tag-to-interrogator calibration symbol*)
UART	univerzální asynchronní přijímač–vysílač (*Universal Asynchronous Receiver – Transmitter*)
UHF	velmi vysoký kmitočet (*Ultra High Frequency*)
USB	univerzální sériové rozhraní (*Universal serial bus*)
VGA	zesilovač s řízeným zesílením (*Variable Gain Amplifier*)

SEZNAM PŘÍLOH

A ČTEČKA EXIN-1

A.1 Pohled na celkovou podobu čtečky EXIN-1



A.2 Schémata desek plošných spojů

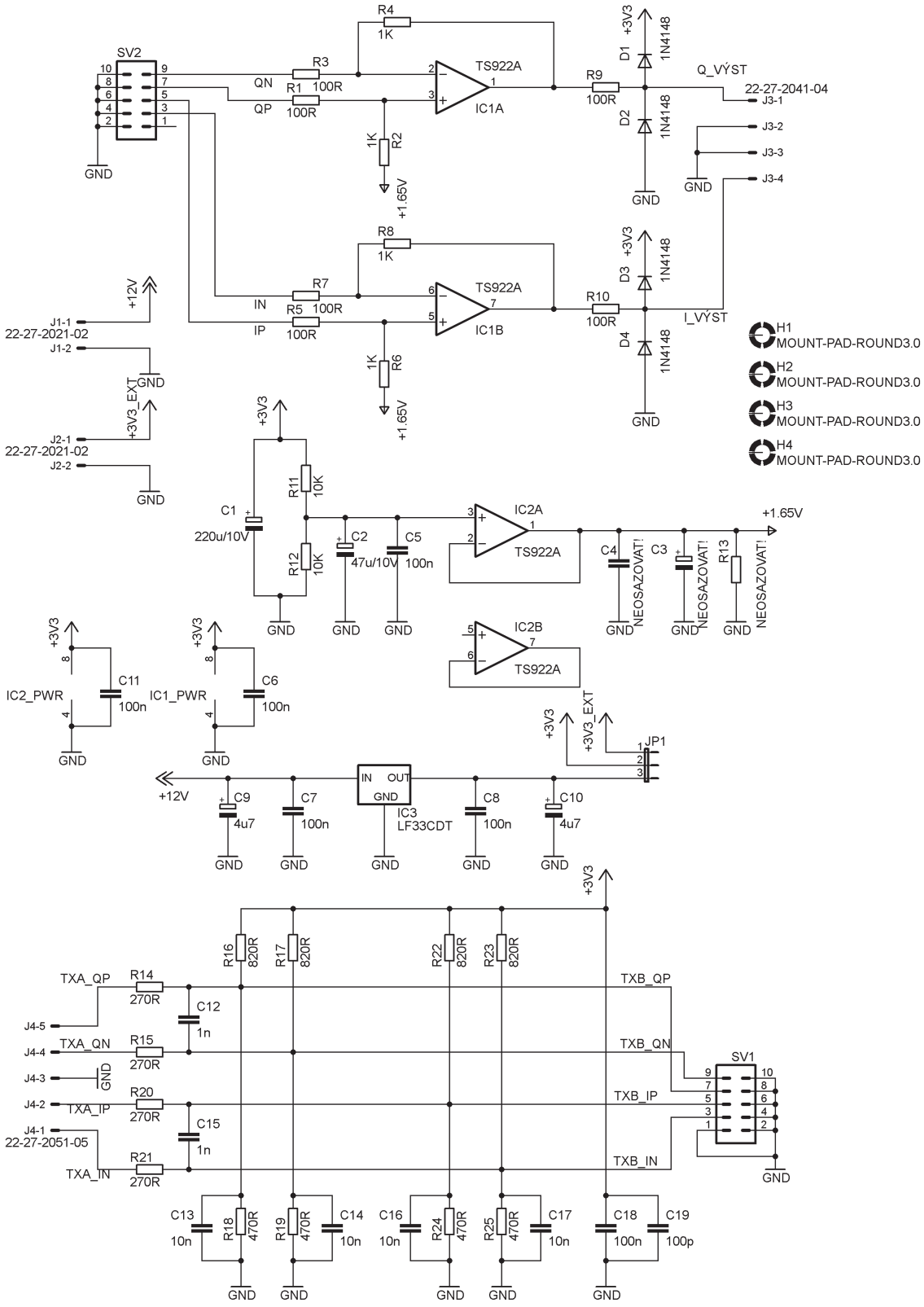


Schéma zapojení desky plošných spojů s rozdílovými zesilovači.

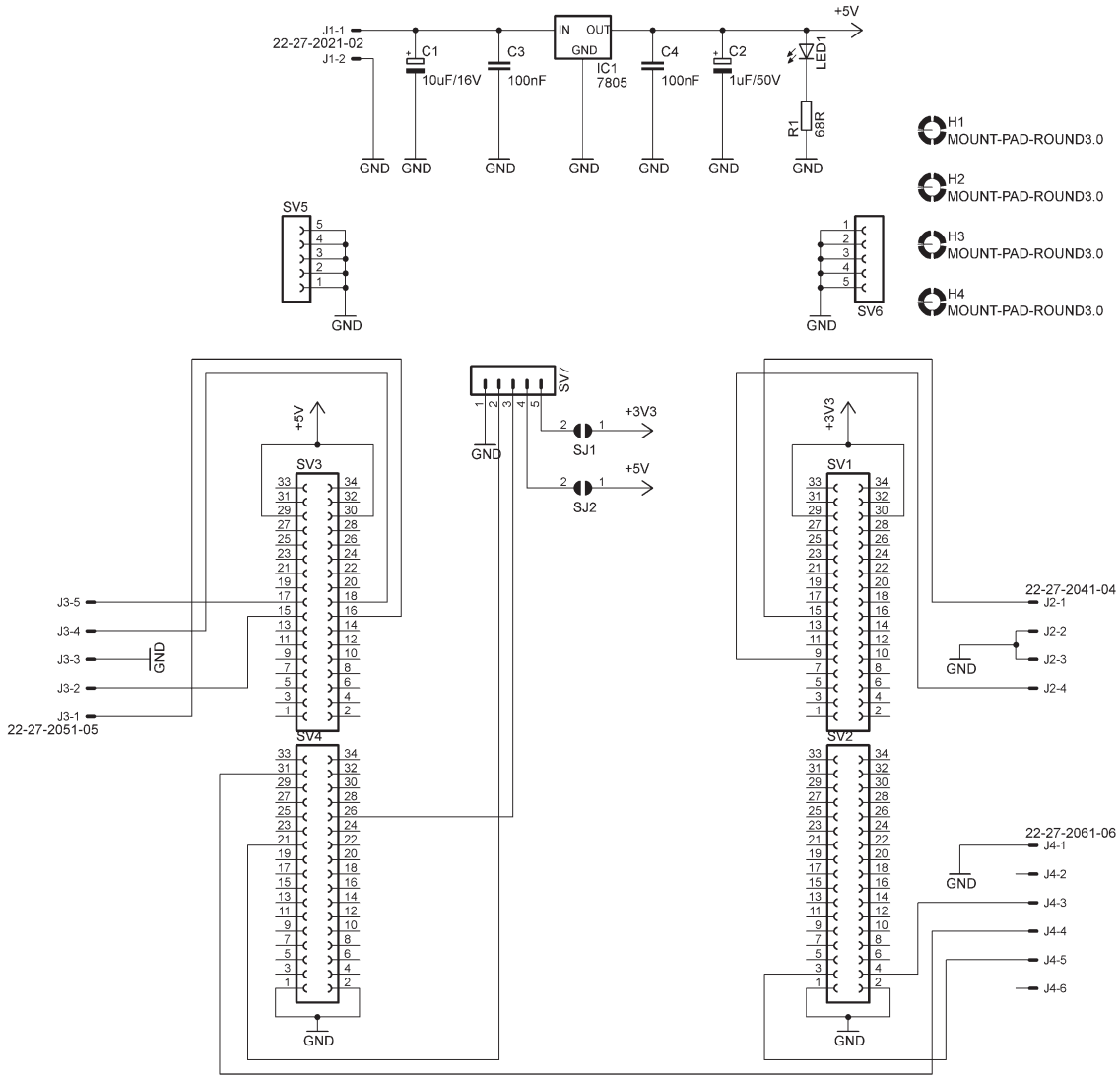
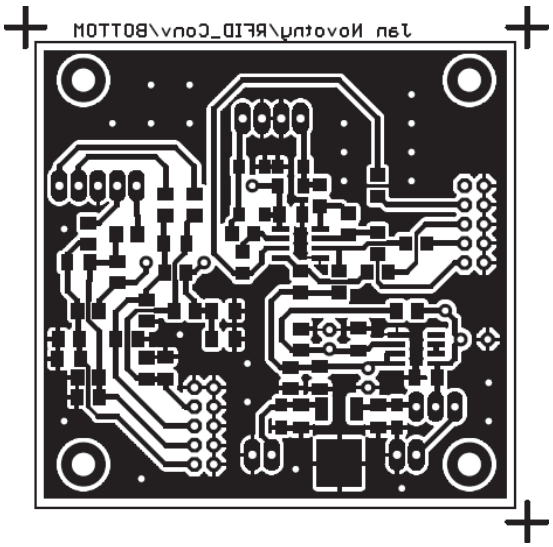
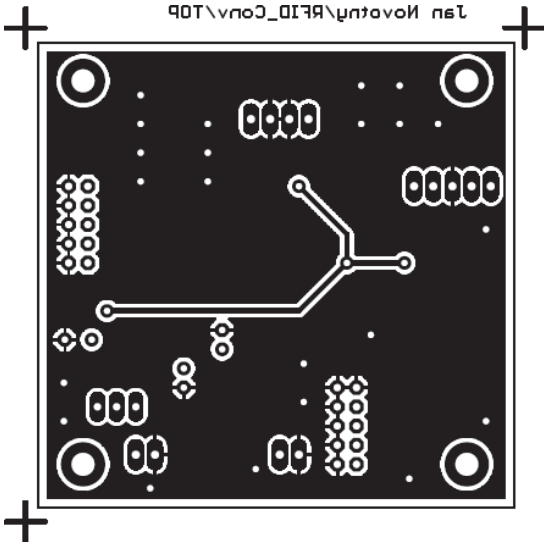


Schéma zapojení desky plošných spojů pro vsazení Discovery kitu.

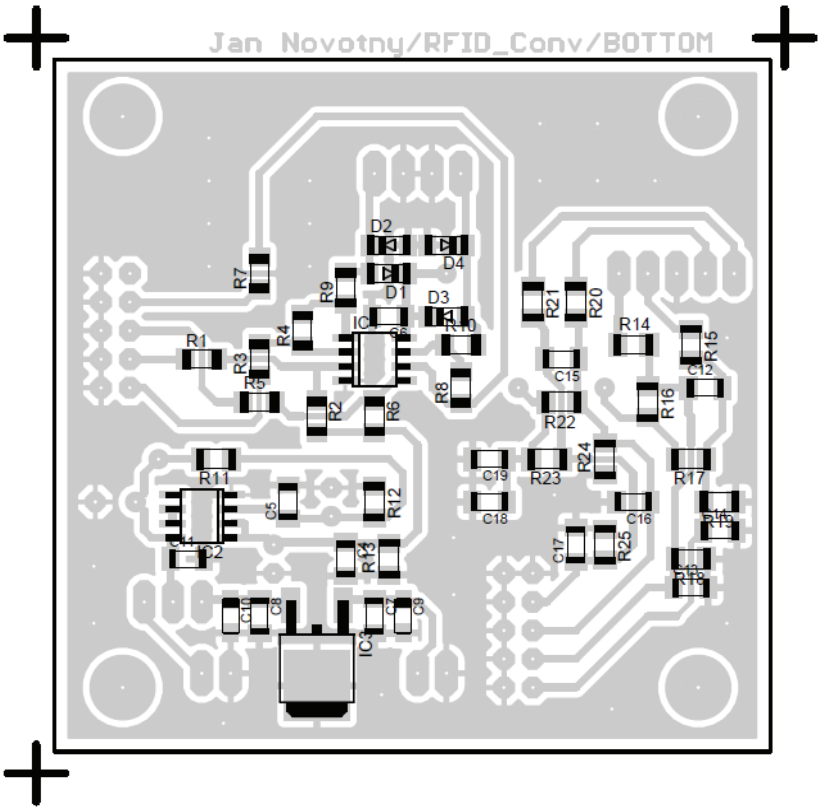
A.3 Předlohy a osazovací plánky desek plošných spojů



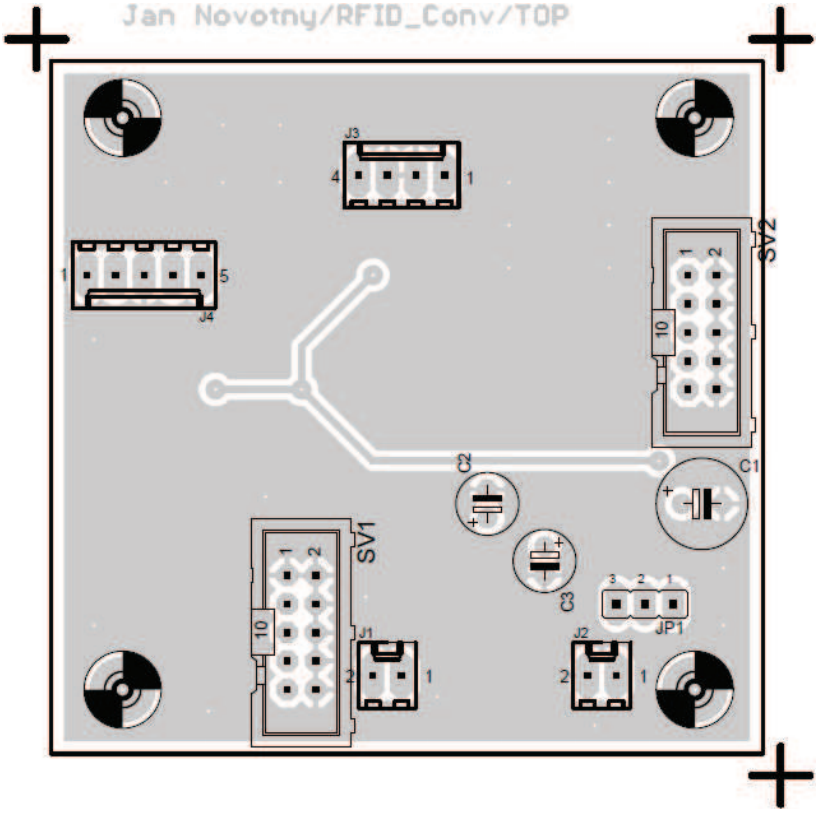
Předloha desky plošných spojů pro rozdílové zesilovače – spodní strana (měřítko 1:1, skutečná velikost desky 63,5×61,595 mm).



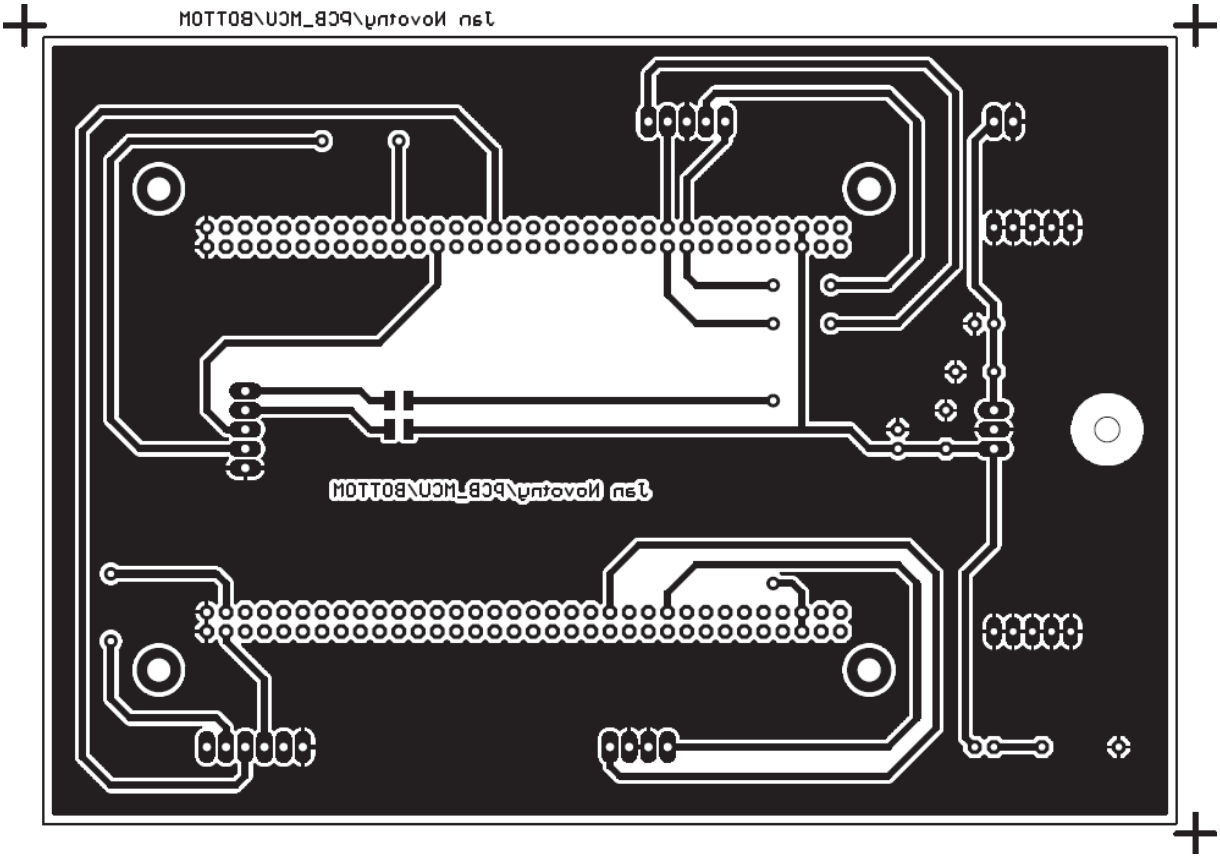
Předloha desky plošných spojů pro rozdílové zesilovače – horní strana (měřítko 1:1, skutečná velikost desky 63,5×61,595 mm).



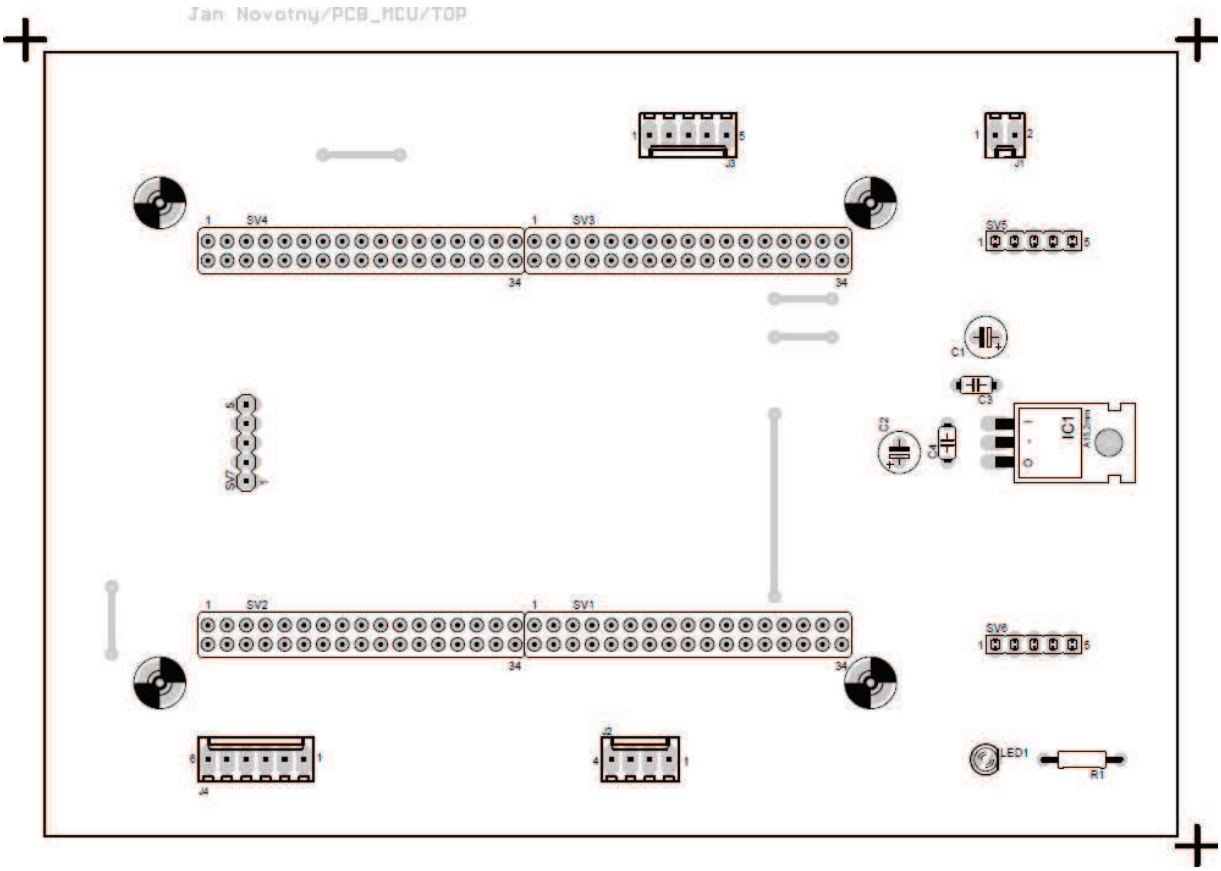
Osazovací výkres desky plošných spojů pro rozdílové zesilovače – spodní strana (1,5× zvětšeno, velikost desky 63,5×61,595 mm).



Osazovací výkres desky plošných spojů pro rozdílové zesilovače – horní strana (1,5× zvětšeno, velikost desky 63,5×61,595 mm).



Předloha deska plošných spojů pro vsazení Discovery kitu – spodní strana (měřítko 1:1, velikost desky 149,86×104,14 mm).



Osazovací výkres desky plošných spojů pro vsazení Discovery kitu – horní strana (měřítko 1:1, velikost desky 149,86×104,14 mm).

B SEZNAMY SOUČÁSTEK

Označení součástek v seznamu koresponduje s označením součástek na deskách plošných spojů v příloze A.2.

B.1 Seznam součástek pro desku diferenčních zesilovačů

Reference	Hodnota/typ	Cena/ks (orientačně)	Popis
C1	220µF/16V	2,50 Kč	Elektrolytický kondenzátor - radiální
C2	47µF/10V	1,20 Kč	Elektrolytický kondenzátor - radiální
C3, C4	NEOSAZENY	-	Elektrolytický kondenzátor - radiální
C5, C6, C7, C8, C11, C18	100nF/50V	0,80 Kč	Keramický kondenzátor SMD - pouzdro C1206
C9, C10	4,7µF/16V	3,20 Kč	Tantalový kondenzátor SMD - pouzdro SMC A
C12, C15	1nF/50V	1,20 Kč	Keramický kondenzátor SMD - pouzdro C1206
C13, C14, C16, C17	10nF/50V	1,30 Kč	Keramický kondenzátor SMD - pouzdro C1206
C19	100pF/50V	1,10 Kč	Keramický kondenzátor SMD - pouzdro C1206
D1, D2, D3, D4	1N4148	1,30 Kč	Univerzální dioda SMD 75V/150mA, pouzdro SOD80
IC1, IC2	TS922A	28,00 Kč	Dvojitý BiCMOS operační zesilovač SMD, pouzdro SO08
IC3	LF33CDT	19,00 Kč	Napěťový stabilizátor SMD 3,3V/0,5A, pouzdro DPAK
J1, J2	PSH02-02PG	1,40 Kč	Konektor se zámkem do DPS – 2 piny
J3	PSH02-04PG	2,70 Kč	Konektor se zámkem do DPS – 4 piny
J4	PSH02-05PG	3,70 Kč	Konektor se zámkem do DPS – 5 pinů
JP1	S1G20	cca 1,00 Kč	Konektorové kolíky lámací – 3 piny
R1, R3, R5, R7, R9, R10	100Ω	1,20 Kč	SMD rezistor 0,25W/1%, pouzdro R1206
R2, R4, R6, R8	1kΩ	1,20 Kč	SMD rezistor 0,25W/1%, pouzdro R1206

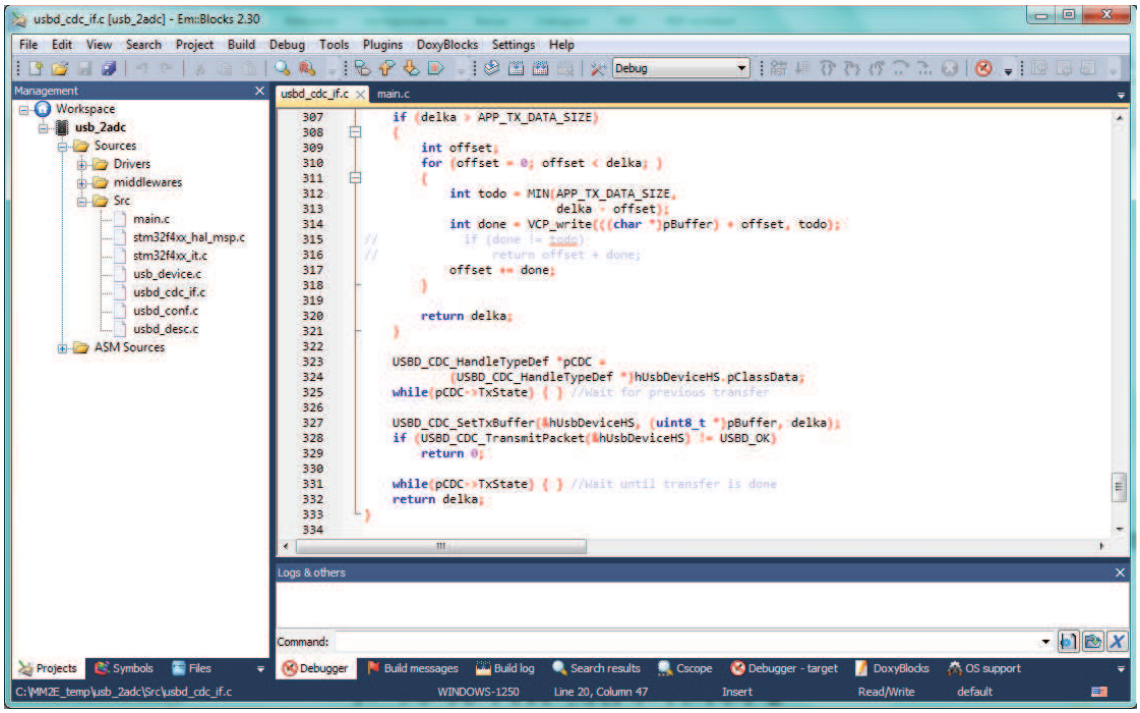
R11, R12	10kΩ	1,20 Kč	SMD rezistor 0,25W/1%, pouzdro R1206
R13	NEOSAZEN	-	SMD rezistor 0,25W/1%, pouzdro R1206
R14, R15, R20, R21	270Ω	1,20 Kč	SMD rezistor 0,25W/1%, pouzdro R1206
R16, R17, R22, R23	820Ω	1,20 Kč	SMD rezistor 0,25W/1%, pouzdro R1206
R18, R19, R24, R25	470Ω	1,20 Kč	SMD rezistor 0,25W/1%, pouzdro R1206
SV1, SV2	MLW10G	4,30Kč	Konektor pro ploché kabely do DPS přímý, 2×5 kontaktů.

B.2 Seznam součástek desky pro vsazení Discovery kitu

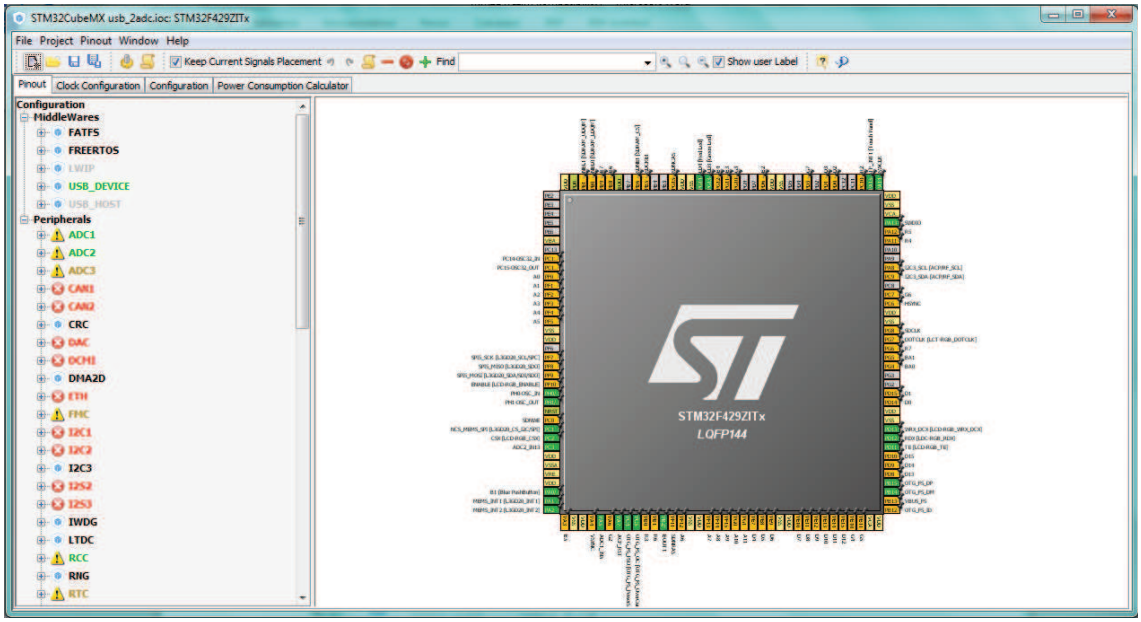
Reference	Hodnota/typ	Cena/ks (orientačně)	Popis
C1	10µF/40V	1,00 Kč	Elektrolytický kondenzátor - radiální
C2	1µF/63V	1,20 Kč	Elektrolytický kondenzátor - radiální
C3, C4	100nF/63V	1,20 Kč	Keramický kondenzátor
IC1	KEC 7805	9,70 Kč	Napěťový stabilizátor 5,0V/1A, pouzdro TO-220
IC3	LF33CDT	19,00 Kč	Napěťový stabilizátor SMD 3,3V/0,5A, pouzdro DPAK
J1	PSH02-02PG	1,40 Kč	Konektor se zámkem do DPS – 2 piny
J2	PSH02-04PG	2,70 Kč	Konektor se zámkem do DPS – 4 piny
J3	PSH02-05PG	3,70 Kč	Konektor se zámkem do DPS – 5 pinů
J4	PSH02-06PG	3,40 Kč	Konektor se zámkem do DPS – 6 pinů
LED1	LED 5MM YELLOW 120/110°	2,50 Kč	LED dioda žlutá, průměr 5mm
R1	120Ω	1,20 Kč	Metalizovaný rezistor 0,6W/1%
SV1, SV2, SV3, SV4	BL834GD	15,00 Kč	Dutinková lišta 2×17 pinů
SV5, SV6	BL805G	4,10 Kč	Dutinková lišta 5 pinů
SV7	S1G20	cca 1,00 Kč	Konektorové kolíky lámací – 5 pinů

C POUŽITÁ VÝVOJOVÁ PROSTŘEDÍ

C.1 Vývojové prostředí Em::Block



C.2 Konfigurační nástroj STM32CubeMX



Záložka pro nastavení funkce pinů mikrokontroléru.