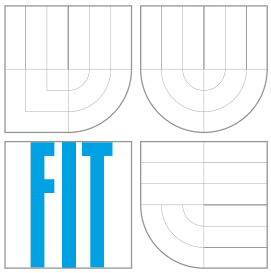
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

FITKIT JAKO PŘIJÍMAČ/VYSÍLAČ DÁLKOVÉHO OVLÁDÁNÍ

FITKIT AS A REMOTE CONTROLLER TRANSCEIVER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MICHAL RŮŽEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. ZDENĚK VAŠÍČEK
SUPERVISOR

BRNO 2010

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2009/2010

Zadání bakalářské práce

Řešitel: **Růžek Michal**
Obor: Informační technologie
Téma: **FITkit jako přijímač/vysílač dálkového ovládání**
FITkit as a Remote Controller Transceiver
Kategorie: Vestavěné systémy

Pokyny:

1. Seznamte se s platformou FITkit a komunikačními protokoly, které využívají běžně dostupné dálkové ovladače komunikující v infračerveném pásmu.
2. Navrhněte rozšiřující modul, který umožní příjem i vysílání modulovaného IR signálu.
3. Realizujte modul a navrhněte sadu knihovních funkcí, pomocí kterých bude možné ovládat jiné zařízení a dekódovat příchozí signál.
4. Činnost navrženého řešení demonstrujte na zvolené aplikaci.

Literatura:

- Gook, M.: *Hardwarová rozhraní: průvodce programátora*. Computer Press, 2006
- Nagy, C.: *Embedded systems design :using the TI MSP430 series*. Boston, 2003
- Dále dle pokynů vedoucího

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů 1 až 2 zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Vašíček Zdeněk, Ing.**, UPSY FIT VUT
Datum zadání: 1. listopadu 2009
Datum odevzdání: 19. května 2010

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

L.S.

doc. Ing. Zdeněk Kotásek, CSc.
vedoucí ústavu

Abstrakt

Tato bakalářská práce se zabývá komunikačními protokoly použitými v infračervených dálkových ovladačích spotřební elektroniky. Popisuje v současnosti nejpoužívanější protokoly RC5, NEC a SIRC, jejich vlastnosti a způsoby dekódování. Pomocí jednoduchého hardwarového přípravku a komponent implementovaných na čipu FPGA platformy FITkit jsou realizovány transceivery těchto protokolů. Řešení obsahuje také ukázkovou aplikaci v jazyce C.

Abstract

The bachelor's thesis is about communication protocols used in infrared remote controls of consumer electronics. It describes widely used protocols such as RC5, NEC and SIRC, their properities and methods of decoding. Transceivers are realized via components inside FITkit's FPGA chip, using simple HW module. Demo application written in C language is also included.

Klíčová slova

infračervené záření, paprsek, protokol, dálkové ovládání, RC5, NEC, SIRC, FPGA, vysílač, dekodér, transceiver, FITkit

Keywords

infrared radiation, beam, protocol, remote control, RC5, NEC, SIRC, FPGA, transmitter, decoder, transceiver, FITkit

Citace

Michal Růžek: FITkit jako přijímač/vysílač dálkového ovládání, bakalářská práce, Brno, FIT VUT v Brně, 2010

FITkit jako přijímač/vysílač dálkového ovládání

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Zdeňka Vašíčka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Michal Růžek
11. května 2010

Poděkování

Vedoucímu práce Ing. Zdeňku Vašíčkovi za cenné rady, které mi poskytl při tvorbě této bakalářské práce.

Obsah

Kapitola 1

Úvod

Infračervené dálkové ovládání patří v současnosti k nejrozšířenějšímu způsobu ovládání spotřební elektroniky. V počátcích dálkových ovladačů se k přenosu dat užívalo zvukového signálu. Tento nespolehlivý způsob vysílání byl rozvojem optoelektronických součástek nahrazen infračervenými prvky jako je infračervená LED a fotodioda. Část kapitoly 2 proto věnuji obecně infračervenému záření a jeho použití v ovladačích spotřební elektroniky. Během vývoje dálkových ovladačů neexistovaly žádné standardy, kterými by se měl přenos dat řídit. Každá větší společnost vymyslela vlastní formát vysílaných dat, takzvaný protokol. V kapitole popisuji nejrozšířenější protokoly, k nimž bezpochyby patří RC5 od firmy Philips, dále pak jeho rozšířenou variantu RC5X. Také nesmíme vynechat protokoly společností NEC a SONY. U každého z nich popisuji způsob kódování bitové informace i podobu a časování stavů v datovém rámci. Z uvedených informací budu vycházet při návrhu a vlastní tvorbě přijímače/vysílače těchto protokolů.

Začátek třetí kapitoly pojednává o příjmu infračerveného paprsku, popisuje obvyklé části, ze kterých se přijímač IR záření skládá. Dále píši o součástce řady TSOP17xx, integrovaném přijímači, který jsem se rozhodl použít při stavbě rozšiřujícího modulu. Díky tomuto přípravku bude možné přes platformu FITkit přijímat a vysílat infračervené informace. Podstatnou část kapitoly tvoří rozbor možností dekódování protokolů, jejich výhod či nevýhod a čím mne varianta, kterou jsem si vybral pro implementaci, zaujala. V poslední části vysvětluji, proč jsem pro implementaci transceiverů na platformě FITkit zvolil čip FPGA namísto mikrokontroléru.

Navazující kapitola líčí návrh přípravku a volbu vhodného místa připojení k FITkitu. Po výběru konkrétního typu integrovaného přijímače uvádím schéma zapojení přípravku, návrh rozložení součástek na desce plošných spojů a její realizaci.

Pátou kapitolu jsem věnoval popisu tvorby transceiverů jako komponent v jazyce VHDL. Z každého transceiveru zvlášť popisuji implementaci jeho vysílací a přijímací části. Nechybí doplňující obrázky ilustrující formát dat na vstupu/výstupu komponenty transceiveru. Z kapitoly se také můžete dozvědět, jakým způsobem jsou uvnitř FPGA připojeny na jeden adresový SPI dekodér všechny transceivery. Konec kapitoly popisuje implementaci knihovních funkcí v programovacím jazyce C, které umožní komunikaci mezi procesorem a transceivery, a zajímavé programové části ukázkové aplikace.

Kapitola 6 dokumentuje prototypy vybraných knihovních funkcí. Popisuji zde význam jednotlivých parametrů funkcí, dále pak hodnoty, jakých mohou parametry nabývat. Do této kapitoly jsem také zahrnul stručný popis ovládání ukázkové aplikace.

V závěru práce zhodnocuji dosažené výsledky, zmiňuji nedokonalosti návrhu a nástin možných řešení. Zabývám se otázkou dalšího vývoje projektu a návazností na jiné projekty.

Kapitola 2

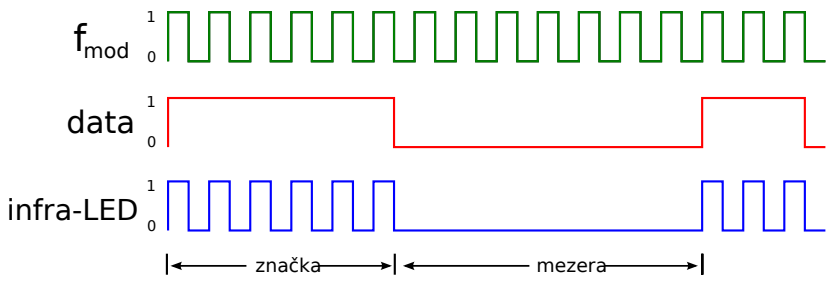
Přenos informace infračerveným zářením

Pojem „infračervené záření“ vyjadřuje jeho umístění v barevném spektru: z latinského *infra* = „pod“, „pod-červenou“. Infračervené (zkr. IR) záření je elektromagnetické záření s vlnovou délkou delší než u viditelného světla. Rovněž se projevuje tepelnými účinky na povrchu tělesa, jež tyto paprsky pohlcuje. Mezinárodní komise pro osvětlení CIE (čerpáno z [9]) rozděluje infračervené záření do tří skupin podle vlnové délky:

- IR-A: 780–1400 nm, do této kategorie patří klasické infra-LED, či například noční vidění.
- IR-B: 1400–3000 nm, pro telekomunikační účely.
- IR-C: 3–1000 μm, využito například pro teplotní navádění střel.

Jako světelné zdroje se v dálkových ovladačích používají luminiscenční diody infra-LED, emitující paprsek záření o vlnové délce kolem 950 nm. Jelikož při posílání informace způsobem log. 1 = „infra-LED svítí“, log. 0 = „nesvítí“ by bylo obtížné zajistit bezchybný přenos datového rámce z důvodu vysokého rušení (zdrojem takového infračerveného záření je i slunce či obyčejná žárovka), používá se modulace (obr. 2.1).

Při modulaci logickou úrovní 1 se infra-LED budí obdélníkovým nosným signálem o dané střídě a frekvenci, typicky mezi 30–56 kHz. Tento stav se nazývá *značka* (ang. *mark*). Po dobu logické úrovně 0 není dioda aktivní, nastala *mezera* (ang. *space*). Dobu trvání značky a mezery, stejně tak jako jejich význam specifikuje použitý protokol. V klidu, tj. když neprobíhá přenos rámce, setrvává vysílač ve stavu log. 0. Přijímač signálu je nastaven na



Obrázek 2.1: Modulace vysílaných dat

použitý nosný kmitočet. Tím, že ostatní kmitočty odfiltruje, účinně zamezuje možnému rušení okolním světlem.

Protože takto ovládaná infra-LED pracuje pouze v krátkých časových intervalech, je možné výrazně zvýšit její proud v propustném směru a tím zvětšit vyzářený světelný tok, aniž by došlo k přehřátí a poškození diody. Zavedením pulzně šířkové modulace se střídou kolem 30 % (záleží na použitém protokolu) se zkrátí aktivní doba infra-LED, díky čemuž se sníží spotřeba a prodlouží životnost baterie jako napájecího zdroje ovladače.

2.1 Protokoly firmy Philips

2.1.1 RC5

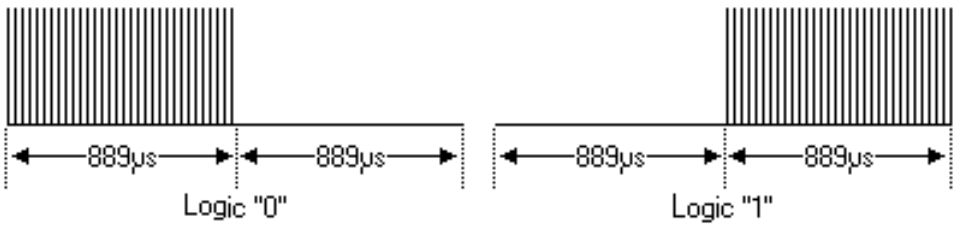
Mezi nejznámější protokoly patří RC5, původně vyvinutý firmou Philips. Do jisté míry jej kopírují i další společnosti, ne vždy jsou pak dodrženy všechny parametry protokolu (rozdílný nosný kmitočet, počet start-bitů, apod.). Datový rámec protokolu se podle [1] skládá z celkem 14 bitů, jež jsou vyslány v tomto pořadí:

- 2 start-bity (zahájení komunikace)
- toggle bit (zabezpečovací mechanismus)
- 5 bitů pro adresaci zařízení (televizní přijímač, videorekordér, satelit ...)
- 6 bitů vyhrazených pro příkaz (změna hlasitosti, přepnutí kanálu, atp.)

Oba start-bity jsou vždy logické úrovně 1 a kromě indikace začátku datového rámce podle nich přijímač automaticky dolaďuje zisk a citlivost na příchozí infračervený signál (ang. AGC = Automatic Gain Control).

Hodnota toggle (čes. překlápěcího) bitu se při každém novém stisku klávesy mění na opačnou. Díky němu lze na přijímací straně rozlišit mezi trvale stisknutým tlačítkem a jednotlivými stisky po sobě. Pokud je například trvale držena číslice 1 na dálkovém ovladači televizního přijímače, nedojde ani po vložení a následném odstranění překážky v cestě IR paprsku k přepnutí na jedenáctý kanál.

Následují bity reprezentující adresu zařízení a kód příkazu, přičemž v obou případech se nejvýznamnější bit (MSB) vysílá jako první.



Obrázek 2.2: Manchester kódování, převzato z [1]

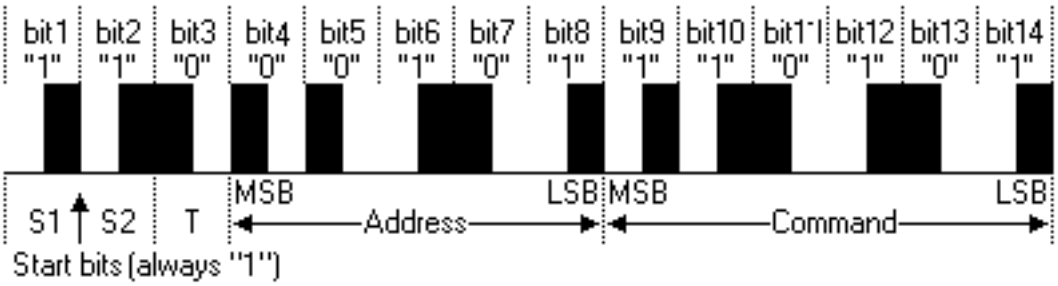
Protokol RC5 pracuje na nosné frekvenci 36 kHz a pro odesílání dat se používá kódování Manchester. Toto kódování, někdy též zvané jako bifázová modulace, spojuje datový signál se synchronizačním. Bitový interval je rozdělen na dvě stejně velké části a doprostřed intervalu je vložena změna signálu. Hodnota bitu je pak dána směrem hrany signálové změny.

Pro RC5 je bit o log. 0 definován změnou ze značky na mezeru, opačně pak pro logickou 1, viz obrázek 2.2. Bitový interval má konstantní délku rovnou 64 periodám nosného kmitočtu, tedy:

$$\frac{64}{36\ kHz} \doteq 1778\ \mu s.$$

Odtud platí, že odeslání všech 14-ti bitových intervalů potrvá $14 \cdot 1778\ \mu s \doteq 25$ ms. Doba značky či mezery je složena z 32 period nosné, tj. 889 μs. Výrobce doporučuje pro snížení spotřeby nastavit střídu modulovaného signálu na 25 % nebo 33 %.

Na obrázku 2.3 si můžete prohlédnout strukturu jednoho datového rámce protokolu RC5. Rámec bude opakovaně vysílán, dokud bude drženo tlačítko na dálkovém ovladači. Perioda opakování, počítáme-li jí jako čas mezi začátky obou rámců, odpovídá době odeslání 64 bitů, čili 113,7 ms. Pro připomenutí, hodnota toggle bitu se v průběhu opakovaného vysílaní téhož rámce nemění.



Obrázek 2.3: Datový rámec RC5, převzato z [1]

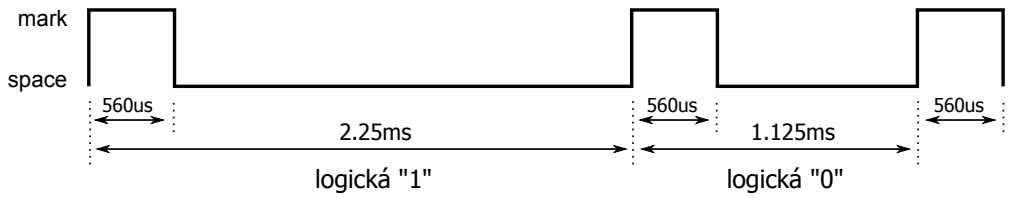
2.1.2 RC5X

Šestibitová hodnota příkazu v protokolu RC5 se časem ukázala jako nedostačující, což vedlo k přidání dalšího bitu a zvýšení počtu možných příkazů na 128. Přidaný sedmý bit nahrazuje druhý start-bit, navíc je jeho hodnota invertována. Inverze zaručuje zpětnou kompatibilitu s původním protokolem RC5: pro příkazy 0–63 bude negovaný sedmý bit roven logické 1 a zařízení vybavené starším standardem RC5 jej proto bude chápat jako obyčejný start-bit. Rozšířenému protokolu se dostalo názvu RC5X, z anglického Extended RC5.

2.2 NEC

Tento protokol, vytvořený společností NEC Electronics, je používán ve spotřební elektronice většiny japonských výrobců, například Pioneer, Onkyo, Hitachi, NEC, Teac, či Yamaha. Mezi jeho přednosti patří možnost kontroly dat proti chybám a také adresování až cca 65 000 zařízení v rozšířeném módu.

Následující informace jsou volně převzaty z [3] a [7]. Protokol operuje na nosné frekvenci 38 kHz a pro kódování bitů se používá pulzně poziční modulace (obr. 2.4). Pulzy typu značka jsou při této modulaci vždy stejně dlouhé, a to 560 μs. Hodnota bitu je potom definována dobou mezi začátky těchto dvou pulzů. Pro logickou hodnotu 1 činí tato doba 2,25 ms, pro logickou 0 přesně polovinu, tedy 1,125 ms. Výrobce uvádí doporučenou střídu nosného kmitočtu rovnou 33 %.

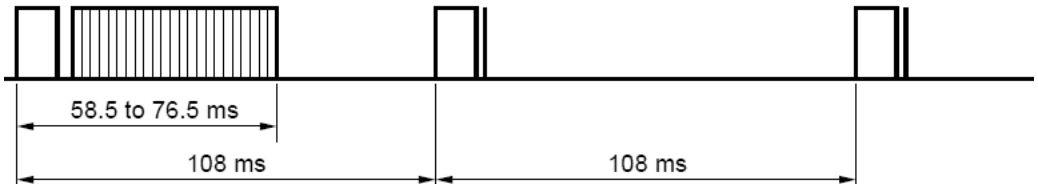


Obrázek 2.4: NEC - pulzně poziční modulace

Datový rámec protokolu NEC lze rozdělit na 3 hlavní části:

- hlavička, ang. leader code,
- část pro adresu zařízení nebo kód výrobce, zvaná též custom code,
- datová část obsahující ovládací příkaz.

Hlavička protokolu vždy začíná 9 ms značkou (v přijímači užita při AGC) následovanou mezerou. Délka mezery rozlišuje typ rámce. Novější specifikace protokolu totiž umožňuje vyslání jednoho celého rámce a poté, v případě, že je tlačítko stále stisklé, se vysílá pouze krátká informace o opakování předchozího rámce. Tato vlastnost, v anglické dokumentaci označovaná jako one-shot transmission mode, snižuje spotřebu energie až o jednu třetinu. Taktéž se s její pomocí dá rozpoznat trvalý stisk tlačítka a opakovaný stisk. Bude-li tedy doba výše zmíněné mezery rovna 2,25 ms, jedná se o opakovací kód a kromě 560 μs značky ukončující tuto mezeru se další data nevyšlou. Doba 4,5 ms naopak signalizuje, že hlavička bude následována zbývajícími 32 bity rámce. Doba k opětovnému vyslání rámce, příp. opakovacího kódu činí 108 ms (obr. 2.5).



Obrázek 2.5: Jeden datový rámec a opakovací sekvence, převzato z [7]

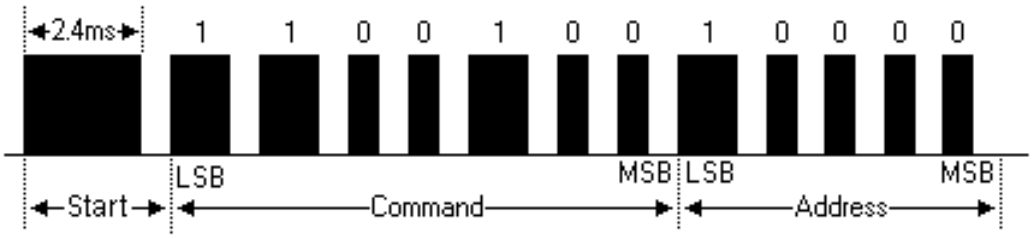
Blok adresy se skládá z 16 bitů, kódovaných podle uvedené modulace. Jako první se odesílá nejméně platný bit (LSB). V původní verzi protokolu obsahoval nižší byte adresu cílového zařízení, ve vyšším byla uložena její invertovaná podoba využitá při kontrole při příjmu. Novější specifikace dovoluje adresu rozšířit na celých 16 bitů. Připravíme se tak sice o možnost kontroly správnosti, výrazně ale vzroste počet možných adres.

Obrázek 2.6: Datový rámec protokolu NEC, převzato z [3]

Podobně jako předchozí část, i část pro příkaz obsahuje 16 bitů a jako první se vysílá nejméně platný bit. Zde ovšem zůstala zachována redundance v podobě negované informace v horním bytu. Počet příkazů tedy dosahuje maximálně 256 možných variant. Horní negovaný byte bude využit po přijetí rámce k ověření validity dat. Dopředu víme dobu vysílání příkazového bloku, ta bude mít díky přítomnosti negovaného příkazu konstantních $8 \cdot (2,25 + 1,125) = 27\ ms$. Obrázek 2.6 ukazuje podobu datového rámce protokolu NEC, zde s rozšířenou 16-ti bitovou adresou.

2.3 Sony SIRC

Mezi poslední představitele rozšířenějších komunikačních protokolů patří SIRC (Serial Infra-Red Control) od japonské firmy Sony. Přenášená data jsou dle [2] kódována pulzně šířkovou modulací a následně vysílána na nosné frekvenci 40 kHz se střídou 33 % (25 %). Hodnota bitů se při této modulaci rozlišuje délkou trvání značky. Značka o délce 1,2 ms představuje logickou jedničku. Logická nula trvá poloviční dobu, 600 μs. Značky od sebe odděluje mezera s konstantní dobou 600 μs.



Obrázek 2.7: 12-ti bitový rámec protokolu SIRC, citováno z [2]

Začátek rámce tvoří úvodní 2,4 ms trvající značka (opět slouží k automatickému doladění zisku přijímače), po níž následuje 600 μs mezera a pak již vlastní data, přičemž jednotlivé části jsou vysílány od nejméně významného bitu. Postupným vývojem vznikly 3 varianty tohoto protokolu, lišící se však pouze počtem bitů. Podle [4] existují tyto varianty:

- 12-ti bitová verze (obr. 2.7) odesílá 7-mi bitový příkaz, poté 5-ti bitovou adresu zařízení,
- 15-ti bitová varianta pouze rozšiřuje adresu na 8 bitů,
- 20-ti bitový protokol vychází z 12-ti bitového, za adresovou část navíc připojuje osmibitová rozšiřující (ang. extended) data.

Rámec je periodicky vysílán, dokud je drženo tlačítko na ovladači. Doba, za kterou dojde k opakovanému vyslání, se rovná 45 ms od začátku rámce.

Kapitola 3

Analýza problému a návrh řešení

3.1 Příjem IR záření

V počátcích dálkových ovladačů byly přijímače infračerveného záření sestaveny výhradně z diskrétních součástek. Vlastní zapojení přijímače se skládá z několika bloků:

- fotodetektor,
- zesilovač,
- pásmovou propust,
- demodulátor.

V drtivé většině plní úlohu fotodetektoru optoelektronický prvek zvaný fotodioda, zapojený v odporovém režimu. V tomto módu mění svůj elektrický odpor v závislosti na intenzitě osvětlení. V pouzdru fotodiody bývá přítomen i filtr, který propouští pouze určitou část světelného spektra, konkrétně vlnové délky kolem 950 nm. Zde bych rád upozornil na důležitost (alespoň přibližné) shody vlnových délek vysílací infra-LED a fotodetektoru. Při větším rozdílu se totiž snižuje citlivost a pro stejný dosah vysílače by bylo nutné zvýšit výkon vysílací diody.

Signál z fotodetektoru je připojen na vstup zesilovače. Část zesilovače tvoří obvod, který udržuje na jeho výstupu konstantní napěťovou úroveň přijatého pulzu, nezávisle na tom, jak moc je vysílač vzdálen od přijímače. Obvod toho dociluje automatickou změnou velikosti zesílení, v anglické literatuře známou pod termínem Automatic Gain Control. Využívá přitom počáteční AGC značku, vysílanou na začátku datového rámce protokolu. Zesilovač je kvůli vysokému zesílení citlivý na okolní rušení, proto byly starší přijímače odstíněné plechovým krytem.

Signál dále vstupuje do pásmové propusti nastavené tak, aby propouštěla pouze úzké pásmo kmitočtů se středem v použité nosné frekvenci.

Poslední část přijímače tvoří demodulátor. V případě, že do demodulátoru přichází nosný kmitočet se známou frekvencí, změní logickou hodnotu na výstupu přijímače.

3.1.1 TSOP17xx

Do své bakalářské práce jsem se rozhodl použít kompaktní přijímač řady TSOP17xx (viz [8]) od německé firmy Vishay. Poslední dvojčíslí udává nosnou frekvenci, na které přijímač

operuje, například TSOP1738 pro 38 kHz. Jedná se o rozměrově malou součástku se třemi vývody, která v sobě zahrnuje všechny výše zmíněné funkční bloky. K výhodám patří nízká spotřeba energie, jednoduché použití a také vysoká odolnost vůči rušení okolním světlem.

Výstup přijímače je aktivní v logické nule, pokud tedy na přijímač nedopadá paprsek z vysílače, bude výstup setrvávat v logické 1. Pro správnou demodulaci vyžaduje, aby pulz trval nejméně 10 period nosné. Takovýto pulz by měl být následován podobně dlouho mezerou, aby se daly bezpečně odlišit dvě po sobě jdoucí značky. Pro zamezení rušení se doporučuje vložit do napájecí cesty filtr v podobě rezistoru a elektrolytického kondenzátoru, kterým se oddělí napájení přijímače od zbytku zapojení.

3.2 Způsoby dekódování IR protokolů

3.2.1 Philips RC5X

Díky použitému kódování u tohoto protokolu je možné ze signálu odvodit synchronizační kmitočet a s ním potvrzovat platnost dat. Tento princip dekódování jsem se rozhodl nepoužít pro jeho komplikovanost, neboť pro rekonstrukci hodinového kmitočtu je potřeba navíc obvod fázového závěsu. Kromě toho lze hodinovou frekvenci při dodržení specifik protokolu určit již předem.

Obrázek 3.1: Průběh signálu RC5X na výstupu IR přijímače

Bitové intervaly jsou vždy stejně dlouhé, nabízí se tedy možnost jednoduše vzorkovat signál vždy uprostřed první poloviny bitového intervalu. Výstup z IR přijímače je invertován, viz obrázek 3.1, který zobrazuje průběh signálu zachycený programem DigiTrace [5]. Proto odpovídá levá polovina bitového intervalu vlastní hodnotě bitu. Vzorkování by se potom spustilo po detekci první sestupné hrany signálu, tedy při příjmu prvního start-bitu. Po načtení zbývajících 13 bitů se vzorkování ukončí a k dispozici máme načtený datový rámec. Problémy však mohou nastat s časováním. Na rozdíl od dekodéru, který bude řízen krystalovým oscilátorem, se v dálkových ovladačích používají levnější a méně přesné keramické rezonátory. Rovněž pokles napájecího napětí ovladače způsobený vybíjející se baterií může způsobit rozladění oscilátoru. Při vzorkování takového průběhu není zaručeno, že se vzorkuje v první polovině intervalu, protože se vzorkovací signál vůči přijatému rozchází, zvláště pak u posledních bitů. Tento způsob dekódování rovněž může považovat jakýkoli protokol za RC5X, z důvodu absence kontroly časových délek.

Jako další možnost se nabízí měřit dobu mezi dvěma sousedními sestupnými hranami. V tomto případě se všech 14 bitových intervalů rozdělí na 28 poloviční „půlbitů“ a výsledná data se potom sestaví z těchto půlbitů. Délku půlbitu 889 μs označme jako T. Doba mezi zmíněnými hranami potom bude nabývat hodnot 2T, 3T, případně 4T. Hodnoty menší či větší jsou považovány za chybu. Hodnota 2T značí, že první půlbit měl hodnotu 1. Poté přišla sestupná hrana a následující půlbit musí být nutně logické hodnoty 0. Délka 4T nastane pokud po sobě následují půlbity s hodnotami 0110. Pro dobu 3T je situace komplikovanější a je třeba rozlišit, zda se jedná o první nebo druhý načítaný půlbit v bitovém intervalu. V prvním případě se do pole půlbitů zapíše sekvence 010, v opačném hodnoty

110. Tím, že k novému čítání dochází vždy nejdéle za dobu 4T, se omezuje možnost chyby při větším rozptylu oscilátoru u dálkového ovladače. Podle hodnot posledních bitů, například při kombinaci log. 0 s log. 1, už nemusí přijít očekávaná sestupná hrana a jinak kompletní datový rámec bude zahozen (časovač překročí dobu 4T). Kvůli této komplikaci a také z důvodu nutnosti počítání T-dob mi tento způsob rovněž nevyhovoval.

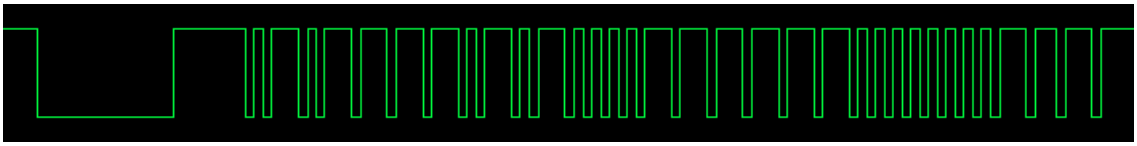
Poslední metoda je založena na skutečnosti, že uprostřed bitového intervalu vždy dojde ke změně signálu. Směr této změny udává hodnotu tohoto intervalu. Sestupná hrana signalizuje logickou hodnotu 1, zatímco vzestupná hrana značí bitový interval s hodnotou 0. Protože se v přijatém signálu na některých místech vyskytují hrany i na začátcích a koncích bitových intervalů, je nutné detekci prostřední hrany spustit ve správný okamžik. Pro tento účel jsem navrhl časové okno, které se jakoby posouvá po přijímaném signálu. Detekuje se pouze hrana, která je uvnitř tohoto okna, ostatní jsou ignorovány. Po jejím přijetí se okno posune na další bitový interval a čeká se znovu na prostřední hranu. Jestliže do určitého času daného šířkou okna hrana signálu nepřijde, vyhodnotí se přijímaný průběh jako chybný. K bezpečnému odlišení rámce RC5X od jiných protokolů se využívá faktu, že po vyslaném rámci vždy následuje pauza dlouhá

$$t_a - t_b = 113,7\ ms - 25\ ms \doteq 89\ ms,$$

kde t_a představuje periodu vysílání a hodnota t_b dobu přenosu jednoho rámce. Po vypočtenou dobu je postačující testovat, zda nepřišel další pulz, a pokud ne, prohlásíme přijatý rámec za platný.

3.2.2 NEC

Rámec protokolu NEC (obr. 3.2) lze poměrně spolehlivě rozeznat od ostatních protokolů pomocí úvodní hlavičky. Po detekci první sestupné hrany počítáme dobu do příchodu hrany vzestupné, a bude-li tato doba dlouhá alespoň 8,7 ms, jedná se o začátek NEC rámce. Tento čas byl vybrán k odlišení od podobných protokolů (například JVC), jejichž délka úvodní značky se pohybuje do 8,5 ms. Po rozpoznání úvodní značky se opět počítá doba do příští sestupné hrany. Nyní bude sloužit k rozlišení normálního datového rámce od opakovací sekvence. Z kapitoly 2 víme, že pro normální rámec se tato doba rovná hodnotě 4,5 ms, zatímco pro opakovací kód 2,25 ms. Pro jednodušší implementaci postačí kontrolovat uplynulou dobu proti 3,4 ms, což je přibližná střední hodnota mezi zmíněnými časy. Pokud bude naměřená doba větší jak 3,4 ms, jedná se o normální rámec, následovaný vlastními bity.



Obrázek 3.2: Průběh signálu NEC na výstupu IR přijímače

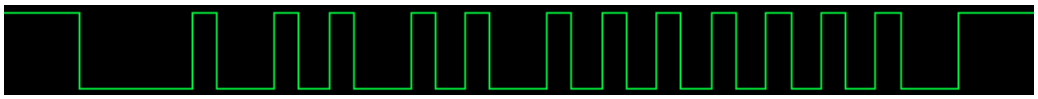
Při detekci hodnot bitů části adresové a příkazové si můžeme situaci opět zjednodušit měřením pouze doby mezi dvěma sestupnými hranami signálu. Doba pod 1,7 ms (přibližný střed z intervalu 1,125 ms a 2,25 ms) označuje logickou nulu, větší hodnota logickou jedničku. Nutností je přítomnost časovače, který hlídá maximální čas, po který může přijít sestupná hrana. V tomto případě by bylo vhodné nastavit časovač na dobu logické jedničky, včetně drobné rezervy, tedy například na čas 2,4 ms. Výrobci většinou uvádějí toleranci do

5 % od hodnot specifikovaných protokolem. Po vypršení takového časovače dojde k přerušení příjmu rámce, neboť obsahoval chyby. Požadujeme-li vysokou přesnost při detekci rámce protokolu NEC, můžeme navíc kontrolovat dodržování délky pulzu v pulzně pozičním kódování, jehož velikost by měla být rovna 560 μs.

Po přijetí všech 32 bitů zbývá provést kontrolu správnosti dat v příkazové části rámce podle posledních 8 bitů, které obsahují negovanou podobu příkazu. Vlastní kontrolu lze uskutečnit mnoha způsoby, já zde uvádím variantu s operací logického součinu mezi příkazem a jeho invertovanou verzí. Jestliže v datech není chyba, bude byte výsledku této operace roven nule a přijatý rámec můžeme prohlásit za validní.

3.2.3 Sony SIRC

Úvodní pulz SIRC lze detekovat stejným způsobem jako u protokolu NEC, tedy měřit interval mezi první sestupnou a nástupnou hranou a v případě hodnoty kolem 2,4 ms jej uznat jako platný začátek SIRC rámce. V následné datové části využijeme vlastnosti pulzně šířkové modulace a budeme zjišťovat dobu mezi sestupnou hranou a příští vzestupnou hranou. Tento interval v přijaté negované podobě rámce (obr. 3.3) odpovídá délce trvání pulzu, jež určuje hodnotu bitu. Střední hodnota mezi dobou trvání „jedničkového“ a „nulového“ pulzu činí 900 μs. Kratší doba značí bit o logické hodnotě 0 a obráceně. I zde se uplatní časovač hlídající maximální dobu čekání na vzestupnou hranu, po jehož vypršení se přijatý rámec zneplatní.



Obrázek 3.3: Průběh 12-ti bitového SIRC na výstupu IR přijímače

Na rozdíl od protokolu NEC, kde můžeme kontrolu doby pulzu vypustit, protože je konstantní a prakticky nedůležitá, zde musíme sledovat i konstantní délku trvání mezery. Ta by měla nabývat hodnoty 600 μs. Překročení této doby účelně využijeme pro detekci konce rámce. Dopředu totiž není jasné, kolika-bitový rámec protokolu SIRC se vlastně přijímá. Pokud potom dojde k překročení časového intervalu mezery, pravděpodobně to znamená, že se už přijaly všechny bity. Počítadlo bitů přitom musí obsahovat hodnotu 12, 15 nebo 20. Nerovná-li se údaj této hodnotě, jedná se o chybně přijatý rámec, případně o rámec jiného protokolu.

Jiná varianta dekódování rámce detekuje pouze sestupné hrany. Po zachycení první hrany uvedený algoritmus čeká 2,7 ms (doba úvodního pulzu + polovina mezery) a pokud po tuto dobu nepřijde další sestupná hrana, uzná přijatý signál za počátek rámce. V následující části rámce se po přijetí sestupné hrany odpočítá doba 900 μs a následně vzorkuje výstup z přijímače. Navzorkovaná hodnota se rovná invertované hodnotě odvysílaného bitu. Během uvedené čekací doby nesmí přijít hrana signálu, jinak je takový rámec považován za chybný. Načítání je ukončeno podobně jako u prvního způsobu dekódování. Zde se však neměří délka mezery, ale překročení doby čekání na další sestupnou hranu. Tento způsob dekódování se mi nezdá tak spolehlivý jako dříve zmíněný kvůli vzorkování, které neřeší přesný tvar signálu. První variantě nahrává i fakt, že ji lze implementovat modifikací dekodéru NEC.

3.3 Dekodér na platformě FITkit

Při návrhu dekodérů jsem se zabýval otázkou implementace na vývojové platformě FITkit, neboť jejich realizaci lze provést buď a) programově v mikrokontroléru (MCU) rodiny MSP430, nebo b) hardwarovou cestou v programovatelném hradlovém poli (FPGA) řady Spartan 3.

Uvedený MCU obsahuje podle [6] dva čítače/časovače A a B, s registry použitelnými pro komparační režim či mód záchytu hrany. Druhý čítač je již využíván knihovní funkcí `delay()`, proto u něj nelze přepnout na jiný zdroj hodin, případně přenastavit předděličku. Režim záchytu hrany můžeme účelně využít k měření šířky pulzu v daném protokolu. Dekodér by bylo vhodné implementovat pod přerušením, například sestupnou hranou, aby se hlavní tělo programu mohlo věnovat jiné činnosti. Výhodou a současně i nevýhodou realizace v MCU je spojení dekodérů do jednoho. V tomto případě by algoritmus již po přijetí úvodního pulzu věděl, jaký rámec kterého protokolu přijímá, a poté dokončil načítání v odpovídající větvi programu. Takové řešení ale ztrácí na univerzálnosti, neboť budeme-li chtít v aplikaci použít pouze dekodéru určitého protokolu, zůstanou ostatní nevyužity a zbytečně zabírají místo v paměti MCU. V případě modulového řešení, kde by byl každý dekodér implementován samostatně, vzniká problém při aplikaci více dekodérů naráz, protože moduly nelze jednoduše spojit. Implementace dekodérů v MCU by pravděpodobně využila oba časovače, což snižuje aplikovatelnost, protože v hlavním programu by je nebylo možné používat.

Oproti tomu implementace v hradlovém poli přináší možnost realizovat každý dekodér jako samostatnou a nezávislou komponentu se shodným rozhraním. I díky tomu nevzniká problém při jejich propojení mezi sebou, stejně tak bude snadné jednotlivé dekodéry přidávat a odebírat. Komponenty by příchozí signál zpracovávaly paralelně a ta z nich, která signál dekóduje správně, informuje o svém stavu řadič přerušení. Ten vyšle do MCU žádost o přerušení, signalizující úspěšné přijetí dat. Následně si mikrokontrolér tato data z FPGA vyčte. Kromě obsluhy externího přerušení se MCU nijak nepodílí na dekódování a může se naplno věnovat užitečné aplikaci. Implementace v FPGA však přináší jisté úskalí. Programovací jazyk pro popis hardwaru VHDL je citlivý na dodržení syntaktických struktur. Při nesplnění tohoto požadavku nemusí proces syntézy zcela pochopit námi zamýšlený návrh a vytvoří odlišný, někdy i zcela nefunkční obvod.

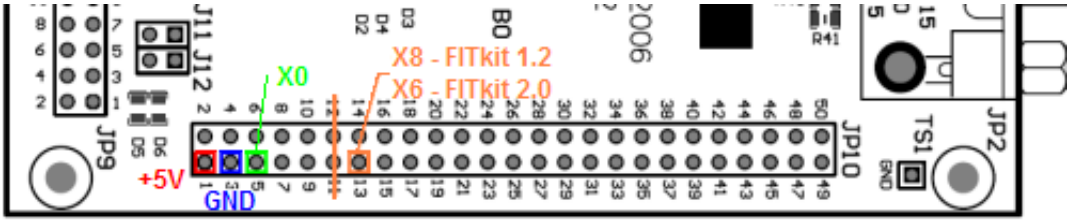
Výhodněji se z důvodů modulární podoby dekodérů a minimální zátěže mikrokontroléru jeví druhá varianta, proto jsem si ji vybral pro implementaci.

Kapitola 4

Realizace HW přípravku

4.1 Připojení modulu k FITkitu

Protože jsem si pro implementaci vybral čip FPGA, je nutné HW přípravek obsahující přijímací a vysílací část připojit ke konektoru JP10. Na FITkitu se nachází úplně dole. Uvedený konektor tvoří 25 (u FITkitu verze 2.0 je to 26) kolíků ve dvou řadách, přičemž napájecí napětí je přivedeno výhodně na krajní piny konektoru. Situaci poněkud komplikuje fakt, že prvních 6 vývodů X0–X5 je použito k interním účelům, například X0 funguje jako výstup z řadiče přerušení. Prvním pinem, který lze využít pro vlastní účely, je z toho důvodu bit z FPGA dostupný pod označením `X(6)`. Kolíky číslo 11 a 12 nejsou u FITkitu verze 2.0 zapojeny, díky tomu se pin X6 posouvá až na kolík 13, na kterém je ve verzi 1.2 připojen bod X8. Při použití jiné verze FITkitu je proto důležité správně upravit názvy pinů v kódu VHDL.



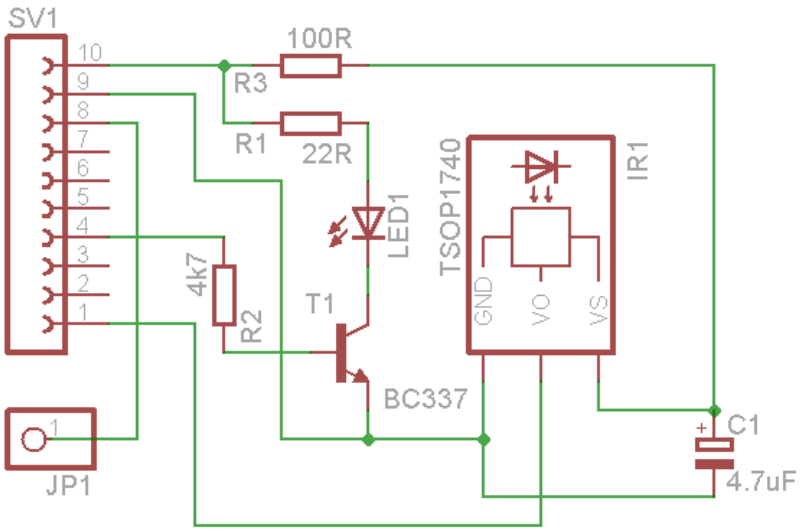
Obrázek 4.1: Konektor JP10 a popis důležitých pinů

Z obrázku 4.1 je patrné, že napětí +5V včetně nulového bodu GND, potřebné pro napájení přijímače TSOP a vysílací infra-LED se nachází ve spodní řadě za sebou. Pro připojení přípravku tedy vystačíme s jednořadou dutinkovou lištou, začínající na levé straně spodní poloviny konektoru. Využijeme lištu o délce deseti dutinek, která se běžně vyrábí.

4.2 Schéma zapojení

Původně jsem zamýšlel použít pro každou nosnou frekvenci jeden odpovídající přijímač TSOP17xx. Zjistil jsem však, že přijímače mají toleranci kolem 5 % a tedy, že by stačilo použít 38 kHz verzi pro příjem 36 kHz i 40 kHz, protože by se do této tolerance přibližně vešly. Problémy však nastaly při příjmu 40 kHz zmíněným TSOP1738. Signál vůbec neodpovídal předpokládanému průběhu. Vinu na tom nese skutečnost, že pro spolehlivé

přijetí pulzu je nutné, aby trval alespoň 10 period nosné. Tento požadavek nebyl při vysílání na 40 kHz dodržen. Experimentováním jsem tedy došel k závěru, že k příjmu signálu všech tří frekvencí postačí přijímač s nosným kmitočtem 40 kHz, tedy TSOP1740. Pomalejší kmitočty totiž splňují zmíněné pravidlo o minimálním počtu period a jsou bezchybně přijímačem dekódovány.



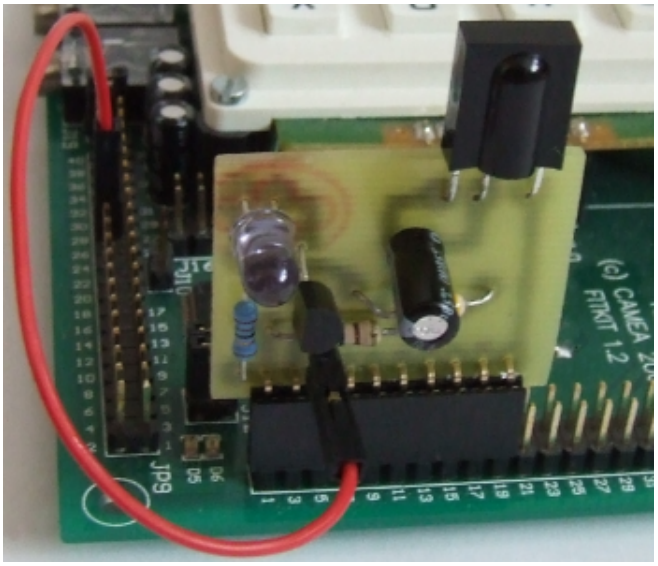
Obrázek 4.2: Schéma zapojení přípravku

Zapojení přijímače vychází z katalogového listu výrobce [8]. Kondenzátor C1 a rezistor R3 společně tvoří filtr, který potlačuje rušení vzniklé v rozvodu napájecího napětí a které by jinak nepříznivě ovlivnilo spolehlivou činnost přijímače. Vysílací část tvoří tranzistor T1 v zapojení se společným emitorem, v jehož kolektorovém obvodu je připojená infračervená LED1 s vlnovou délkou 950 nm. Rezistor R1 omezuje proud tekoucí infra-LED a pomocí něho lze měnit vyzářený světelný výkon diody. V tomto zapojení jsme limitováni použitým tranzistorem, jehož kolektorový proud I_c nesmí trvale překročit hranici 0.5 A. Hodnota odporu R2 není kritická, vyhoví v rozsahu od 2 kΩ do 10 kΩ. Pro správnou funkci přípravku je nutné u starší verze FITkitu propojit výstup z přerušení FPGA s mikrokontrolérem. Přerušení je vyvedeno na pin X0 a protože použitá dutinková lišta SV1 tento pin zakrývá, je nutné jej vyvést na desku HW přípravku (JP1) a následně tento nový kolík propojit drátem s vývodem 26 na konektoru JP9. Při použití nového FITkitu verze 2.0 slouží k tomuto spojení propojka J5, zmíněný drát není nutné připojovat.

4.3 Deska plošných spojů

Desku plošných spojů jsem navrhl, stejně jako schéma zapojení, v počítačovém programu Eagle 5.7.0. Jedná se o jednostrannou desku, s šířkou mírně přesahující namontovanou dutinkovou lištu. Tu jsem použil v 90° verzi, ve které jsou dutinky vodorovně s deskou. Po připojení k platformě FITkit bude tedy deska umístěna nastojato. Tomu jsem přizpůsobil i rozmístění ostatních součástek na desce. Vysílací infra-LED i senzor TSOP1740 se nachází v horní části plošného spoje, přičemž přijímač svým tělem tuto desku převyšuje. Předpokládá se přitom ohnutí a zkrácení spodní části jeho vývodů o 90° směrem dozadu, do plošného spoje. V dolní části desky u dutinkové lišty je připájen pin, na který je vyve-

deno přerušení z FPGA. Tento vývod je potřeba spojit s mikrokontrolérem tak, jak již bylo uvedeno výše. Návrh desky plošných spojů je k dispozici v příloze B.



Obrázek 4.3: Hotový přípravek připojený k FITkitu

Prototyp plošného spoje jsem pro jeho jednoduchost vytvořil ručně kresbou lihovým perem a následně vyvolal v roztoku chloridu železitého. Na obrázku 4.3 si můžete prohlédnout hotový rozšiřující modul připojený k FITkitu.

Kapitola 5

Implementace transceiverů

5.1 IR transceiver jako komponenta VHDL

Řadiče protokolů jsem navrhl jako komponenty tak, aby měly shodná rozhraní a poskytovaly plně duplexní provoz. Uvnitř jedné komponenty (řadiče) proto najdeme přijímač (dekodér) i vysílač (enkodér), schopný nezávislé činnosti. Například u všech komponent lze nalézt 24 bitový port `DATA_IN` a `DATA_OUT`, kterým je možno zapsat či naopak přečíst data z transceiveru. Pokud použitý protokol neobsahuje 24 datových bitů, jsou zbývající bity doplněny nulami (v případě výstupu) nebo ignorovány (v případě vstupu).

K řídícím signálům patří `RST`, inicializující komponentu do výchozího stavu, dále pak signál pro aktivaci vysílání, nazvaný `WRITE_EN`. Logická jednička přivedená na tento vstup způsobí uložení dat z portu `DATA_IN` do vnitřního registru a následně dojde k jejich postupnému odvysílání. Po celou dobu vysílání je stavový výstup `BUSY` v logické úrovni 1 a vysílač bude ignorovat další požadavky na zápis dat. Mezi jednotlivými rámci protokolu se vždy vyskytuje mezera definované délky. Po odvysílání rámce proto příznak `BUSY` setrvá v logické jedničce, dokud neuplyne doba mezery. Takto lze vždy jen počkat na ukončení příznaku `BUSY` a následně ze strany mikropočítače odeslat další data.

Výstupní signál určený k modulování vysílací infra-LED jsem pojmenoval `IR_OUT`. Naopak k příjmu dat, pro napojení na výstup TSOP1740, je určen signál `IR_IN`. Po úspěšném dekódování protokolu se na signálu `DATA_VLD` objeví po dobu jednoho hodinového taktu logická hodnota 1. Ta signalizuje, že se na portu `DATA_OUT` nachází platná data. Data na výstupu setrvávají až do úspěšného dekódování dalšího rámce.

Poslední část rozhraní tvoří vstupní signál `CLK`, na který se předpokládá připojení hodinového kmitočtu `SMCLK` s frekvencí 7,3728 MHz. Z tohoto kmitočtu se uvnitř komponenty dělením získají další periodické průběhy, důležité pro činnost vysílací i dekódovací části.

Na tomto místě bych ještě rád popsal skupinu obvodů, které jsou v dekódovacích částech komponent prakticky totožné. Zejména se jedná o obvod detekce hran v signálu z TSOP1740. Detektor je tvořen dvěma signály `ir_reg_a` a `ir_reg_b` ve funkci klopných obvodů typu D. Ty jsou zapojeny za sebou, přičemž na vstup prvního z nich je přiveden sledovaný signál. S každou nástupnou hranou hodin se informace posune na další výstup. Tím se přiváděný signál zpozdí, v praxi to ale ničemu nevadí. Sestupnou hranu potom představují hodnoty `ir_reg_a`=0 a `ir_reg_b`=1, pro vzestupnou hranu platí opačné hodnoty.

Další část se shodnou implementací tvoří registr `dout`, do kterého se po vystavení `DATA_VLD` do log. 1 uloží dekódovaná data. Výstup registru je dle mapování připojen na port `DATA_OUT`. Způsob mapovaní datových bitů na výstupní sběrnici se liší dle protokolu, podrobněji se mu proto budu věnovat až později.

5.1.1 RC5X

Přijímač

Srdcem této přijímací a dekódovací části je konečný Mooreův automat. Po inicializaci setrvává v klidovém stavu `S_IDLE`. Jakmile se však v signálu objeví sestupná hrana, označující de facto střed prvního start-bitu, přejde do stavu `S_WAIT` a zároveň spustí 19-ti bitový časovač hodinového signálu `cnt`. Úkolem tohoto stavu je počkat do doby, která představuje levý okraj okna zmíněného v kapitole 3. Při dodržení standardu by se měla příští hrana uprostřed bitového intervalu objevit za dva půlbity, každý trvající 889 μs. Pro spolehlivé zachycení ovladačů různých výrobců jsem zavedl čtvrtinovou toleranci kolem očekávané hrany. Čtvrtina se odvíjí pouze od jednoho půlbitu. Tomu potom odpovídá čas

$$889\ \mu s + \frac{3}{4} \cdot 889\ \mu s \doteq 1555\ \mu s.$$

Jakmile čítač dosáhne hodnoty odpovídající této době, přesune se stavový automat do stavu `S_RECV`. V něm očekává příchod hrany, signalizovaný pulzem na signálu `sh_wr`. Čas, do kterého musí hrana přijít, je včetně tolerance definován jako

$$2 \cdot 889\ \mu s + \frac{1}{4} \cdot 889\ \mu s \doteq 2000\ \mu s.$$

Po jeho překročení se nastaví signál `timeout`, čímž se celý přijímač resetuje a uvede opět do klidového stavu. Přijde-li hrana do stanoveného limitu, automaticky se zapíše patřičná hodnota do posuvného registru `data_shreg`, přičemž registr se posouvá vlevo. Hodnota, která se má zapsat, se objeví v signálu `sh_val`. Pro sestupnou hranu je to jednička, pro vzestupnou logická nula. Signál tvořící posuvný registr se při inicializaci vynuluje a do krajního pravého bitu se vloží log. 1. Tato jednička slouží k detekci ukončení rámce. Jakmile se objeví v registru úplně vlevo, došlo k přečtení všech bitů rámce a automat může přejít do stavu `S_SPACE1`. V opačném případě přejde automat po příchodu hrany do stavu `S_CNTRSET`, který pouze nuluje čítač, a následně se vrátí zpět do stavu `S_WAIT`. Stav nazvaný `S_SPACE1` má za úkol také jen resetovat čítač. V následujícím stavu `S_SPACE2` se hlídá, zda po dobu trvání mezery mezi rámci náhodou nepřišel další pulz. Pokud ano, jedná se o chybný rámec a algoritmus se vrátí do výchozí pozice. Doba hlídání je daná maximální hodnotou 19-ti bitového čítače:

$$2^{19} \cdot \frac{1}{7,3728\ MHz} \doteq 71\ ms.$$

Časový úsek neodpovídá celé mezeře, tj. 89 ms, v praxi však bohatě postačí. Po uplynutí uvedené doby dojde v následujícím stavu `S_COMPLETE` k nastavení příznaku platnosti dat `DATA_VALID` a současně ke zkopírování přijaté informace z posuvného registru do klopného obvodu na výstupu. Poté se automat vrátí do výchozího stavu. Protože protokol RC5X nese pouze 13 užitečných bitů a sběrnice má šířku 24 bitů, použil jsem následujícího mapování:

Obrázek 5.1: Mapování dat RC5X na sběrnici

Vysílač

V této sekci hraje důležitou roli 13-ti bitový čítač `pcount`. Tento čítač počítá impulzy generované děličkou hodinového kmitočtu. Dělička je nastavena tak, aby pulzy přicházely s dvojnásobnou nosnou frekvencí 72 kHz. Signál k buzení infračervené diody se potom odebírá z prvního bitu `pcount(0)`, neboť na něm bude uvedená frekvence dělena dvěma a dostaneme 36 kHz. Střída tohoto signálu se rovná 50 %, nejedná se tedy o doporučenou hodnotu. Menší střída slouží pouze ke snížení spotřeby, jiná hodnota proto příjem signálu nijak neovlivní.

Zde trochu odbočím a popíši činnost implementovaného konečného automatu. Po aktivaci signálu `WRITE_EN` přechází automat z klidové polohy do stavu `S_LOAD`, ve kterém dojde k zapsání dat do posuvného registru `sh_reg`. Zároveň s tím se nastaví příznak `BUSY` do logické 1. Posuvný registr načte data ze signálu `din`, který dle obrázku 5.1 mapuje bity ze vstupní sběrnice. Kromě toho také přidá na začátek datového rámce logickou jedničku ve funkci start-bitu. Poté přejde do stavu `S_TRANS`, v němž zůstane do doby, než se odvysílá celý rámec.

Vyslání dat pomocí kódování Manchester lze implementovat s využitím operace XOR mezi hodnotou bitu a datovým hodinovým kmitočtem. Bude-li výsledek operace XOR roven jedné, připojí se budící 36 kHz signál na výstup `IR_OUT`. V opačném případě se výstup nastaví do logické nuly. Pro naše účely potřebujeme generovat datový hodinový signál s periodou rovnou délce bitového intervalu, což odpovídá počtu 64 period nosného signálu. Vzestupná hrana takového signálu přijde vždy na začátku bitového intervalu a po uplynutí 32 period nosné se signál obrátí. Zmíněnou hranu jsem použil pro posun dat v posuvném registru. Datový kmitočet je odebírán ze sedmého bitu `pcount(6)`, tento bit se mění každých 64 půlperiod, čili 32 period nosné.

Detekci odeslání všech bitů plní komparátor `data_end`. Logicky by mělo platit, že odeslání 14ti bitů o délce 64 period nosné odpovídá celkovému počtu

$$2 \cdot (14 \cdot 64) = 1792$$

pulzů signálu `gen_cmp`. To by platilo v případě, že při vysílání prvního bitu by na počátku byl datový hodinový signál v logické úrovni 1. Kvůli použitému zdroji tohoto signálu se však signál nastaví do jedničky až po uběhnutí 32 period nosné. Do této doby se v posuvném registru v posledním levém bitu, který obsahuje hodnotu určenou k odvysílání, nachází jedničkový start-bit. Operace XOR nad touto hodnotou a datovými hodinami (které jsou zatím v log. 0) povolí na výstupu buzení infra-LED. V konečném důsledku tak dojde k přenosu celého prvního bitu, přestože došlo pouze k odvysílání jeho druhé poloviny. Celkový počet period musí být proto o chybějící polovinu snížen na:

$$2 \cdot (13.5 \cdot 64) = 1728.$$

Jakmile čítač dosáhne této hodnoty, vrátí komparátor `data_end` logickou úroveň 1. Na vzniklou situaci zareaguje konečný automat přechodem do stavu `S_WAIT`, ve kterém se deaktivuje výstup `IR_OUT`. Jmenovaný stav slouží pro odpočet časové mezery mezi datovými rámci. Z kapitoly 2 víme, že doba mezi začátky dvou rámců se rovná době odeslání 64 bitů. Protože čítač půlperiod běží od počátku vyslání prvního bitu, stačí detekovat dosažení hodnoty

$$2 \cdot (64 \cdot 64) = 8192.$$

Za tímto účelem jsem navrhl komparátor `frame_end`, který svou aktivní hodnotou informuje stavový automat o dokončení přenosu rámce i mezery. Ten následně přejde do výchozího stavu `S_IDLE` a zruší příznak `BUSY`.

5.1.2 NEC

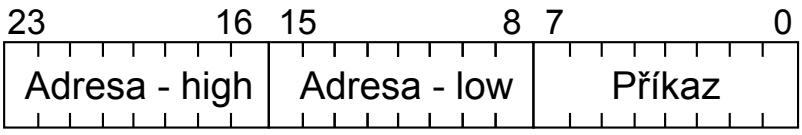
Přijímač

Podobně jako u dekodéru RC5X i zde hlavní část tvoří konečný automat. Jeho činnost postupně popíši tak, jak zpracovává přijímaný signál. Po příchodu první sestupné hrany vstoupí automat do stavu `S_LEAD1`, ve kterém se spustí čítač `cnt` pro měření délky pulzu. Pulz by měl mít 9 ms, při toleranci 5 % považuji za validní i délku 8,55 ms. Automat proto setrvá ve zmíněném stavu, dokud čítač nedosáhne hodnoty odpovídající 8,55 ms. Pokud během této doby přijde vzestupná hrana označující konec pulzu, nejedná se o korektní začátek NEC rámce. Po překročení uvedené doby přechází automat do stavu `S_LEAD2`, ve kterém se naopak očekává, že vzestupná hrana přijde. Pakliže se hrana v signálu do 9,45 ms (9 ms + 5% tolerance) neobjeví, je načítání rámce rovněž přerušeno.

Po rozpoznání počátečního AGC pulzu je nutné rozlišit, zda se přijímá celý rámec či pouze opakovací sekvence. K tomu jsou určeny stavy `S_REP` a `S_NORM`. Do prvního automat přejde ihned po korektním zachycení AGC pulzu a setrvá do doby, než čítač napočítá hodnotu odpovídající době 3.4 ms. Tento čas tvoří hranici mezi dobou 2,25 ms indikující opakovací sekvenci a 4,5 ms představující normální rámec. Poté automat vkročí do stavu `S_NORM`. V obou zmíněných stavech se mimo jiné sleduje příchozí signál, a po detekci sestupné hrany se dle aktuální stavu určí další akce. Časový limit do příchodu hrany je včetně 5% tolerance stanoven na 4,725 ms, po této době se příjem rámce přeruší a konečný automat se vrátí do výchozího stavu. Pro jednodušší popis budeme nyní předpokládat příjem kompletního rámce.

Proto po zjištění sestupné hrany přejde automat z `S_NORM` do stavu `S_BIT0`, který společně s `S_BIT1` definuje hodnotu bitu dle uplynulého času do následující sestupné hrany. Rozhodujícím okamžikem je v tomto případě čas 1,7 ms, neboli střední hodnota mezi dobou log. 0 a log. 1. Čas do příchodu sestupné hrany jsem omezil na 2,363 ms, poté je přijímaný rámec zneplatněn. Dekódovaný bit se následně vloží do registru, jeho obsah se přitom posouvá vpravo.

Posuvný registr jsem stejně jako u RC5X vytvořil o bit delší než je počet všech bitů protokolu, tj. $32 + 1$ bitů. Při inicializaci dekodéru je do nejlevějšího bitu registru vložena hodnota 1. Jakmile se tato hodnota objeví v nejpravějším bitu, víme, že se načetl celý rámec. Tato kontrola probíhá v části `S_CHECK`. Jestliže kontrola potvrdí přítomnost jedničky, přejde automat do stavu `S_COMPLETE`, přičemž nastaví signál `recv_complet`. Tento signál ale potvrzuje jen přijetí všech bitů, nezaručuje správnost. Tu ověřuje komparátor `check_ok`, který zneguje byte příkazu a porovná jej s přijatou invertovanou podobou. Při shodě vrátí logickou hodnotu 1, čímž se nastaví i příznak `DATA_VLD` a data se zkopírují do klopného obvodu na výstupu dekodéru. Pro připojení ke sběrnici jsem navrhl toto mapování:



Obrázek 5.2: Mapování dat NEC na sběrnici

Potvrzovacím signálem `data_valid` se kromě aktivace výstupního příznaku také překlopí do log. 1 klopný obvod typu R-S, tvořený signálem `last_valid`. Jedničková hodnota popsaného klopného obvodu oznamuje, že předchozí přijatá data byla platná. Zároveň se signálem `rep_tmr_rst` vynuluje čítač `rep_tmr`. Jak název čítače možná napoví, definuje interval, po který bude přijatá opakovací sekvence považována za platnou. V tomto případě jsem čítač nastavil na hodnotu odpovídající době 108 ms s přibližně 5% povolenou odchylkou, tj. 115 ms. Nejedná se sice o nejdokonalejší řešení (časovač se spustí až po přijetí celého rámce, ne hned při detekci AGC pulzu), na druhou stranu se jednoduše implementuje. Pokud se do uvedené doby nepřijme opakovací sekvence, čítač se automaticky vynuluje a zároveň s tím se vrátí R-S klopný obvod do logické nuly.

Nyní se vraťme do okamžiku, kdy se rozpozná opakovací sekvence. Ve stavu `S_REP` se na vstupu objeví sestupná hrana, jedná se tedy o opakovací kód. Následně se zkontroluje hodnota R-S klopného obvodu a bude-li rovna jedné, vstoupí automat do stavu `S_DATA_REP`. Zde za pomoci signálu `repeat_data` opět aktivuje příznak `DATA_VLD` a resetuje čítač `rep_tmr`. Řízení automatu se poté vrátí do stavu `S_IDLE`. Dekodér je tak připraven na přijetí dalšího opakovacího kódu, případně nového rámce. Přijde-li opakovací sekvence v době, kdy se R-S klopný obvod nachází v log. 0, vrátí se automat do výchozí pozice, protože „neví“ jaká data se na výstupu mají opakovat.

Vysílač

Pro univerzálnější použití jsem navrhl vysílač tak, že se opakovací sekvence nepošle automaticky, ale pouze při vnější žádosti. Takto lze zapsat dvě stejné hodnoty po sobě, aniž by druhá hodnota byla odvysílána jako opakovací kód. Do rozhraní komponenty jsem proto přidal vstupní signál `WR_REPEAT`. Funguje obdobně jako `WRITE_EN`, avšak s tím rozdílem, že obvod nevyšle data přečtená ze vstupní sběrnice, ale pouze opakovací sekvenci. Protože tyto žádosti na vstupu trvají pouze 1 hodinový takt, je uvnitř vysílače implementován registr `is_repeat`, který uchovává žádost o vyslání opakovacího rámce.

K hlavním součástem vysílače patří generátor nosného kmitočtu 38 kHz s výstupem `irout`, dále pak čítač půlperiod zmíněného signálu. Informaci o počtu přenesených bitů obsahuje čítač `bitcount`. Nechybí ani posuvný registr `sh_outreg` s paralelním vstupem, pomocí kterého se do něj skrz mapovací signál `din` zapíšou data ze vstupní sběrnice. Při zápisu se k bytu určujícímu příkaz spočítá jeho negovaná podoba, která se do registru také uloží. Mezeru mezi dvěma rámci zajišťuje čítač `space_cnt`. Činnost vysílače řídí konečný automat. V původním návrhu obsahoval automat Mealyho výstup, způsoboval však problémy se synchronizací a proto jsem přešel na Mooreovu verzi.

Po aktivaci signálu `WRITE_EN` přejde automat do stavu, ve kterém načte obsah datové sběrnice do posuvného registru a následně se přesune do stavu `S_HEADER` k odvysílání hlavičky - AGC pulzu. Při žádosti `WR_REPEAT` je načtení dat do registru přeskočeno, pouze se zapíše jedničková informace do `is_repeat`. Úvodní pulz trvá 9ms, tomu odpovídá počet

$$2 \cdot \frac{9\ ms}{\frac{1}{38\ kHz}} = 684$$

půlperiod napočítaných v čítači `tcnt`. Po dosažení této hodnoty se přechodem do stavu `S_TMRRST1` čítač vynuluje. Takto pojmenované stavy slouží vždy pouze k resetování čítače. Po opuštění `S_HEADER` došlo rovněž k deaktivaci signálu `out_en`, od výstupu `IR_OUT` se proto odpojí nosný kmitočet z `irout` a vrátí se na něj log. 0. Předpokládejme, že nyní požadujeme vyslat datový rámec. Proto automat vstoupí do stavu `S_NORM`, ve kterém setrvá po dobu

definující tento typ rámce, to jest 4,5 ms. Hodnota čítače `tcnt` se přitom spočte vzorcem uvedeným výše. Po vynulování čítače přestoupí automat do stavu `S_MARK`, který povolí na výstup 560 μs pulz. Po jeho ukončení se opět resetuje čítač a podle hodnoty krajního pravého bitu v posuvném registru přejde automat do stavu `S_BIT0` nebo `S_BIT1`. Dle protokolu udává doba mezi začátky dvou těchto pulzů hodnotu bitu. Pro logickou úroveň 1 je to 2,25 ms, pro nulu pak 1.125 ms. Od uvedených časů musíme odečíst šířku pulzu 560 μs, protože ten byl již odeslán. Automat proto při vysílání mezery o bitové hodnotě 1 setrvá ve stavu `S_BIT1` po dobu

$$2,25\ ms - 0,56\ ms = 1,69\ ms,$$

během vysílání logické nuly ze stavu `S_BIT0` se tato hodnota rovná

$$1,125\ ms - 0,56\ ms = 0,565\ ms.$$

Posun bitů v registru `sh_outreg` zajišťuje signál `dshift` generovaný ze stavu `S_SHIFT`. Zmíněný signál také způsobí zvýšení hodnoty počítadla odeslaných bitů. Uvedený stav se také stará o to, aby byly přeneseny všechny bity. Zvýšení hodnoty počítadla se projeví až v příštím taktu hodin, takže nový údaj bude k dispozici až v příštím vstupu do stavu `S_SHIFT`. Proto se údaj počítadla srovnává s hodnotou o jedničku menším, s 31. Dokud je počet přenesených bitů menší, vrátí se automat po vynulování čítače `tcnt` k vysílání dalšího bitu. V opačném případě vstoupí konečný automat do stavu `S_STOP`, během kterého odvysílá závěrečnou 560 μs značku v roli stop-bitu a následně setrvá po jistý časový úsek ve stavu `S_WAIT`. Délku úseku určuje čítač `space_cnt`, který bez přerušení počítá půlperiodové impulzy `gen_cmp` už od stavu `S_HEADER`. Doba od začátku rámce do příchodu dalšího je specifikována protokolem NEC a činí 108 ms. Stav čítače je proto porovnáván s hodnotou odpovídající této době a po zjištění shody se automat vrací do výchozího stavu `S_IDLE`. Zároveň se zruší příznak `BUSY`, který byl jinak po celou dobu aktivní.

Zbývá vysvětlit princip odeslání opakovacího rámce. Po aktivaci signálu `WR_REPEAT` a následném odvysílání hlavičky přejde automat do stavu `S_REPEAT`, díky jedničkové hodnotě registru `is_repeat`. Za pomoci čítače `tcnt` se vytvoří 2,25 ms mezera, která značí opakovací kód. Jakmile čítač pulzů dosáhne hodnoty odpovídající času 2,25 ms, přesune se řízení automatu do stavu `S_STOP` k vysílání stop-bitu. Další kroky činnosti se shodují s výše popsaným odesláním datového rámce.

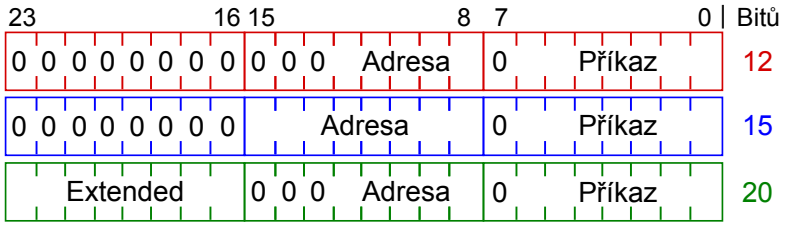
5.1.3 SIRC

Protože existují tři varianty tohoto protokolu, navrhl jsem následující transceiver tak, aby byl schopen zpracovat všechny varianty. K tomu jsem byl nucen upravit rozhraní této komponenty. Nově v něm existují signály `DATA_12_VLD`, `DATA_15_VLD` a `DATA_20_VLD`. Po dekódování 20-ti bitové verze se proto vystaví signál `DATA_20_VLD` a podobně. Pro řízení zápisu slouží signály `WRITE_12_EN`, `WRITE_15_EN` a `WRITE_20_EN`.

Přijímač

Příjem signálu SIRC začíná přechodem konečného automatu do stavu `S_LEAD1`, plnícího naprostou stejnou funkci jako v dekodéru NEC. Délka zaváděcího pulzu v protokolu SIRC měří 2,4 ms, při toleranci 5 % je to i čas

$$2,4\ ms \cdot 0,95 = 2,28\ ms.$$



Obrázek 5.3: Mapování dat SIRC na sběrnici

Poté, co čítač `cnt` dosáhne hodnoty odpovídající tomuto času, přestoupí automat do stavu `S_LEAD2`. V něm se, na rozdíl od předchozího stavu, očekává příchod vzestupné hrany, která označuje konec pulzu. Pásmo, ve kterém musí hrana přijít, je shora limitováno časem

$$2,4\ ms \cdot 1,05 = 2,52\ ms.$$

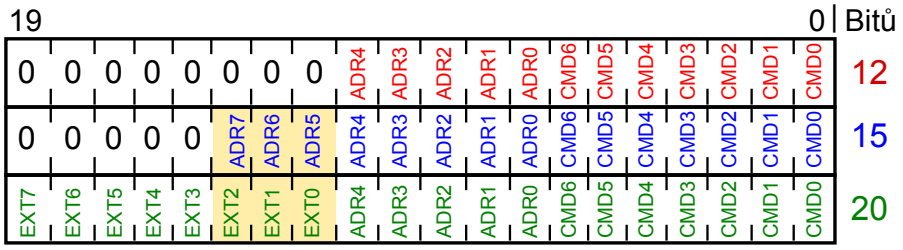
Po jeho překročení se automat vrátí do výchozího stavu `S_IDLE`. Nyní ale vycházejme ze situace, že zmíněná hrana přišla včas. Řízení automatu se proto přepne na stav `S_SPACE`, který slouží k omezení délky trvání mezery. Omezuje se přitom pouze maximální doba trvání, pokud mezera trvá kratší časový úsek než standardizovaných 600 μs, není to považováno za chybu. Limit v sobě zahrnuje i 5% toleranci, nejdelší mezera proto může být délky 630 μs. Překročení této doby může znamenat chybný protokol, nebo častější případ, že byl odvysílán celý rámec. Předpokládejme nyní včasný příchod sestupné hrany, ukončující tuto mezeru.

Automat se důsledkem toho dostane do stavu `S_BSTART`, ve kterém se odměřuje interval, po jehož překročení se daný časový průběh stává validním z hlediska definice hodnoty bitu. Má-li pulz logické 0 trvat 600 μs, dostaneme po odečtu 5 % interval 570 μs. Do té doby nesmí právě přijímaný pulz skončit, jinak dojde k vyvolání chyby a k přerušení načítání rámce. Po dosažení uvedeného času automat přejde do stavu `S_BIT0`, v němž očekává buď konec pulzu a tedy dokončení přenosu bitu o hodnotě log. 0, nebo pokračování pulzu do dalšího stavu, protože se pravděpodobně jedná o pulz definující log. 1. Přechod mezi stavy `S_BIT0` a `S_BIT1` je realizován časovou hranicí, která je dána střední hodnotou mezi dobou trvání logické nuly a jedničky, konkrétně 0.9 ms. Maximální doba trvání jedničkového pulzu je opět omezena, překročení hodnoty 1,26 ms (1,2 ms + 5 %) způsobí přerušení příjmu rámce. Při detekci konce pulzu v některém ze jmenovaných stavů, přejde konečný automat do `S_STORE0` nebo `S_STORE1`, ve kterém se zapíše hodnota přijatého bitu do posuvného registru `sh_reg` a zároveň se zvýší hodnota počítadla bitů `reccount` o jedna. Algoritmus se poté vrátí do stavu `S_SPACE` a cyklus načítání bitů se opakuje.

Určení konce rámce se provádí právě ve stavu `S_SPACE`, když je překročen časový limit maximální délky mezery. Po této události se kontroluje údaj z počítadla bitů `reccount`. Údaj musí nabývat hodnoty 12, 15, nebo 20, což odpovídá přijaté variantě SIRC. Jestliže počítadlo obsahuje odlišnou hodnotu, bude přijatý rámec označen za neplatný. Naopak, kupříkladu při shodě s číslem 15, vstoupí automat do stavu `S_15_FINISH`, který aktivuje příznak `DATA_15_VLD` a zkopíruje data z posuvného registru do výstupní registru `dout`. Mezi posuvný registr a registr `dout` jsem vložil multiplexor `format_out`, jehož úkolem je podle varianty protokolu přeuspořádat přijatou informaci tak, aby vyhovovala některému z mapování jaké ukazuje obrázek 5.3.

Vysílač

Na úvod bych začal popisem mapování sběrnice do posuvného registru, protože zde je situace poněkud komplikovanější. Posuvný registr `sh_outreg` v tomto vysílači dosahuje šířky 20 bitů, čemuž odpovídá nejdelší varianta rámce SIRC. Obsah registru je zarovnán doprava, proto se při použití nižších verzí protokolu volné místo zleva doplní nulami. Impulzem na signálu `dshift` se obsah registru posune vždy o jednu pozici doprava. Z obrázku 5.4 vyplývá, že po zarovnání do registru je část příkazu a prvních 5 bitů adresy vždy stejně umístěna.



Obrázek 5.4: Zarovnání rámců v posuvném registru

Komplikace nastává u verze s 15-ti a 20-ti bity, protože zde jsou na stejných místech umístěny 3 nejvyšší bity adresy společně s třemi nejnižšími bity pole *extended*, přičemž tato data se ze vstupní sběrnice dle obr. 5.3 odebírají z různých míst: adresa z 2. bytu, *extended* ze 3. bytu. Vzniklý problém jsem vyřešil spojením zmiňovaných tříbitových částí sběrnice logickou funkcí OR. Takové řešení ale předpokládá, že při vysílání 20-ti bitové verze budou 3 nejvyšší bity adresy nulové a naopak, při 15-ti bitové variantě musí být spodní část pole *extended* nulová. Uvedený předpoklad lze snadno splnit ze SW strany maskováním bitů.

Konečný automat vysílače se uveden v činnost aktivací signálu `write_en`, který logickým součtem spojuje všechny 3 řídící zápisové vstupy. Zároveň se do registru `comp_val` zapíše číslo udávající počet bitů, které odpovídá použité variantě protokolu. Údaj čítače bitů `bitcount` se porovnává s obsahem uvedeného registru a při shodě se vystaví příznak `all_sent` do log. 1. Příznak je později použit pro detekci odeslání všech bitů.

Před zahájením vysílání se ve stavu `S_LOAD` signálem `din` namapují data ze vstupní sběrnice do posuvného registru `sh_outreg`. Následně se automat uchýlí k odeslání úvodního pulzu rámce. Setrvá proto ve stavu `S_HEADER` tak dlouho, než počet odvysílaných půlperiod nosného signálu bude odpovídat délce hlavičky 2,4 ms. Počet půlperiod obsahuje čítač `tcnt` a pro úvodní pulz se musí rovnat hodnotě

$$2 \cdot \frac{2,4\ ms}{\frac{1}{40\ kHz}} = 192.$$

Během vysílání hlavičky je díky aktivnímu signálu `out_en` povoleno propagovaní signálu o kmitočtu 40 kHz na výstup komponenty. Po vkročení do stavu `S_TMRRST` se výstup opět zablokuje. Také se vynuluje počítadlo půlperiod. V nadcházejícím stavu `S_SPACE` setrvá automat po dobu 600 μs, což odpovídá délce mezery v protokolu SIRC. Hodnota, oproti které se porovnává údaj z čítače, se vypočte z výše uvedeného vztahu. Jakmile čítač dosáhne potřebné hodnoty (48 pulzů), přejde automat do stavu `S_DECISION`, klíčového z hlediska větvení algoritmu. Pokud byly přeneseny všechny bity, bude `all_sent` rovno jedné a automat přejde do čekacího stavu. V opačném případě se dle nejpravější hodnoty v posuvném registru rozhodne, zda bude vysílat pulz o délce log. 0 či log. 1, čemuž odpovídají stavy

S_BIT0 a S_BIT1. V nich zůstává automat po dobu, která odpovídá délce pulzu definujícího hodnotu bitu.

Poté se konečný automat odebere do stavu S_SHIFT, v němž se kromě posunu registru o jeden bit doprava také vynuluje čítač tcnt a zvýší hodnota počítadla bitů bitcount o jedna. Poté se automat vrátí k vysílání mezery přechodem do známého stavu S_SPACE a popsaný přenos dat se opakovaně uplatní pro zbývající bity v rámci.

Konec rámce nastaví signál all_sent do logické 1, načež automat zareaguje přechodem do stavu S_WAIT. V něm kontroluje hodnotu čítače space_cnt, který běží trvale od začátku vysílání. Jakmile dosáhne hodnoty odpovídající požadovaným 42 ms, ukončí se příznak BUSY a celý automat se vrátí do výchozího stavu S_IDLE.

5.2 Propojení komponent v FPGA

Propojení komponent jsem se snažil realizovat co nejuniverzálnějším způsobem, aby bylo jednoduché řadiče jednotlivých protokolů přidávat a ubírat. Proto ve finálním rozložení existuje pouze jeden SPI dekodér SPI_adc_ir pro všechny transceivery. Výběr požadovaného přijímače/vysílače se uskutečňuje prostřednictvím 3 nejnižších bitů adresy, připojených k signálu addr. Bázovou adresu SPI dekodéru jsem nastavil na 90h. Při výběru se přepínají pouze signály zápisu a sběrnice ke čtení z komponent. K výběru výstupní sběrnice slouží multiplexor mux_out, který dle hodnoty v addr připojuje na vstup dekodéru SPI_adc_ir jeden z výstupů komponent ir_RC5, ir_NEC, či ir_SIRC. Naopak vstupní sběrnice je sdílena všemi komponentami, výstup z dekodéru je signálem data_bus rozveden do vstupů transceiverů.

Další důležitou část tvoří řadič přerušení IRQctrl. Na jeho vstup jsem přes vektor interrupts připojil všechny výstupy typu DATA_VLD. Při aktivaci některého z nich vyšle řadič směrem k mikrokontroléru přerušení signalizující úspěšné přijetí nějakého kódu. Mikrokontrolér potom dle nastaveného přerušovacího bitu přečte z patřičné adresy dekódovaná data.

Řízení zápisu je poněkud komplikovanější. Problémem jsou rozdílné hodinové kmitočty. Pro transceivery je použito kmitočtu SMCLK o frekvenci 7,3728 MHz, zatímco pro SPI dekodéry se používá hodin CLK s kmitočtem 25 MHz. Zmíněný problém se projeví při požadavku zápisu generovaného dekodérem SPI_adc_ir. Výstup WRITE_EN přejde do aktivního stavu pouze po dobu jednoho taktu rychlých CLK hodin, proto si vyvolaného krátkého pulzu nemusí transceivery vůbec všimnout. Situaci jsem vyřešil již existující komponentou clk_sync, která převede přiváděný rychlý zápisový signál na pomalý, nazvaný write_en_sl. Tuto komponentu jsem navíc doplnil o registr časovaný pomalým SMCLK kmitočtem, do kterého se aktivací zápisového signálu vloží hodnota z addr. Důvodem pro toto opatření je rovněž platnost původní adresy pouze po dobu jednoho taktu rychlých hodin. Signál dále vstupuje do demultiplexoru, který jej pošle do vybraného transceiveru. Demultiplexor jsem implementoval spojením zápisového signálu write_en_sl s vektorem addr_r, který je připojen na výstup adresového registru upravené komponenty clk_sync. Vektor vzniklý uvedeným spojením se porovná na shodu s některou kombinací, která vyjadřuje adresu transceiveru dohromady se zápisovým pulzem o hodnotě log. 1. Při nalezení shody se jeden bit výstupního vektoru nastaví do log. 1, ostatní zůstávají nulové. Na jednotlivé bity výstupního vektoru jsou potom připojeny signály řídící zápis do jednotlivých transceiverů.

Pro korektní činnost vysílací části je také nutné sledovat stav signálů BUSY. Jeho aktivní hodnota vyjadřující právě probíhající vysílání se pro detekci příliš nehodí. Proto jsem navrhl obvod, který vygeneruje krátký pulz o log. 1 v případě, že se signál BUSY vrátí zpět do klidové

úrovně, tedy po ukončení vysílání. Protože naráz nemůže vysílat víc než jeden transceiver, je tento generátor společný pro všechny komponenty. Vytvořený pulz je připojen k vektoru řadiče přerušení, který následně informuje MCU o dokončeném přenosu.

Výstupy sloužící k připojení infra-LED jsou logickou funkcí OR sloučeny do jednoho a připojeny na pin `X(8)` konektoru JP10. Uvedený stav platí pro FITkit verze 1.2, ve verzi 2.0 je nutné signál připojit na pin `X(6)`. Výstup z čidla TSOP1740 byl původně přímo připojen na vstupy transceiverů, avšak díky odrazům se odeslaná informace ihned přijala. Proto jsem mezi vstupy a výstup ze senzoru vložil selektor `ir_recv`, který propouští příchozí signál pouze, pokud se nevysílá, tj. výstupy `BUSY` jsou nulové. K FITkitu jsem senzor připojil pinem `X(14)`, v novější verzi kitu je to pin `X(12)`.

5.3 Komunikační knihovna v jazyce C

Úkolem knihovny `infra_comm` je umožnit komunikaci mezi aplikací v mikrokontroléru a vlastními transceivery v FPGA. Pro plynulejší přenos dat jsem použil dvě vyrovnávací paměti typu FIFO, jednu pro příjem a druhou pro vysílání, nazvané `infra_rx_buf` a `infra_tx_buf`. Obě jsou realizované technikou rotační kruhové fronty, při které se nad indexy do patřičných polí provádí matematická operace zbytek po celočíselném dělení, kde dělitelem je velikost pole, jehož rozměr lze případně modifikovat z hlavičkového souboru. Jedna buňka pole v sobě obsahuje strukturu nesoucí informaci o typu protokolu a malé tříbajtové pole pro uložení dat. V následujícím textu budu popisovat pouze frontu pro příjem dat, neboť jediným rozdílem v implementaci přijímacího a vysílacího bufferu je předpona `rx` (u vysílacího `tx`) v názvu proměnných. Pro práci s frontou existují dva indexy, `infra_rxRindex` pro čtení z fronty, který ukazuje na zatím nepřečtený prvek, a zapisovací index `infra_rxWindex`, ukazující na první volné místo v poli. Při prázdné frontě se oba indexy rovnají. Naopak rovnost ukazatelů až při zvýšení hodnoty zápisového indexu zmiňovanou metodou znamená plnou frontu, další příchozí data proto budou zahozena. Test na prázdnost front realizují funkce `infra_rx_empty()` a `infra_tx_empty()`. Pro korektní činnost knihovny je požadováno na startu programu zavolat proceduru `infra_init()`, která nastaví všechny indexy na stejnou hodnotu, tj. 0 a aktivuje příznak `infra_tx_free`. Více se o tomto příznaku zmíním dále.

Příjem dat z FPGA zajišťuje funkce `infra_rx_interrupt_handler()` s parametrem udávajícím protokol. Její volání musí být umístěno v těle procedury pro obsluhu přerušení `fpga_interrupt_handler()`, která se nachází v hlavní aplikaci. Výše uvedená funkce vybere z pole `proto_offsets` podle typu protokolu odpovídající offset, který následně přičte k bázové adrese `INFRA_BASE_ADDR` SPI dekodéru a funkcí `FPGA_SPI_RW_A8_DN()` se vyčtou data z FPGA. Offset slouží k přepnutí multiplexoru při výběru řadiče IR protokolu. Před uložením vyčtených dat se provede kontrola plnosti fronty, pakliže je čtecí fronta plná, data se zahodí. Po úspěšném zápisu se samozřejmě posune zápisový ukazatel. Z hlavního programu potom můžeme obsah bufferu číst funkcí `infra_read()`. Bude-li fronta prázdná, vrátí funkce nulu. Jinak do parametrů předaných odkazem zkopíruje typ protokolu a jeho data, návratová hodnota přitom bude rovna jedné. Po zavolání funkce `infra_read()` dojde automaticky k uvolnění přečtené položky z bufferu. Nejedná se přitom o fyzické odstranění, pouze se navýší a následně moduluje čtecí index.

Zápis do fronty dat určených k odvysílání zajišťuje funkce `infra_send()`, která přebírá parametr určující typ protokolu a poté trojmístné pole s daty, kterými bude naplněn rámec požadovaného protokolu. U protokolu SIRC varianty 15 a 20 bitů dochází ještě k vynulování bitů, které nejsou využity, aby nezpůsobily problém při vysílání. V této části se uplatňuje

příznak `infra_tx_free`, který nastavený do jedničky označuje stav, kdy neprobíhá žádný přenos, tzn. vysílače v FPGA nejsou ve stavu `BUSY`. Pokud je zároveň prázdný i vysílací buffer, dojde k okamžitému odeslání dat do FPGA a zrušení příznaku `infra_tx_free`. V opačném případě se při volném místě uloží data do fronty k pozdějšímu odeslání.

Ke správné činnosti algoritmu je zapotřebí do obsluhy `fpga_interrupt_handler()` přidat také volání funkce `infra_tx_interrupt_handler()`, které bude vykonáno po obdržení přerušení informující mikrokontrolér o navrácení vysílačů do klidového stavu. V rámci této funkce proběhne kontrola na prázdnost odchozího bufferu a při pozitivním výsledku se nastaví už zmiňovaný příznak `infra_tx_free`. Není-li fronta prázdná, odešlou se data z první obsazené buňky do transceiveru v FPGA a následně se odpovídajícím způsobem posune čtecí index.

Stav příznaku `infra_tx_free` je zajímavý i z pohledu hlavního programu, proto lze jeho stav testovat voláním funkce `infra_no_tx_transmit()`.

5.4 Ukázková aplikace

Aplikace demonstruje možnosti transceiverů IR protokolů. K ovládání je použito terminálu, případně klávesnice a LCD displeje, proto jsou v FPGA přítomny i řadiče těchto komponent (`lcd_raw_controller`, `keyboard_controller`). Aplikace umožňuje nejen posílat IR příkazy dle zvoleného protokolu pomocí klávesnice, ale i dekódovat data od dálkového ovladače. Typ protokolu použitého při vysílání se zobrazuje na LCD displeji a stiskem klávesy `#` je možné jej měnit.

Nezbytnou části aplikace tvoří obsluha FPGA přerušení `fpga_interrupt_handler()`, která se automaticky zavolá při požadavku přerušení a nastaví bity v `bits` podle toho, od které komponenty v FPGA přerušení přišlo. V těle funkce se potom provede volání obslužné rutiny pro vyčtení dat z FPGA. Dalším nezbytným krokem je inicializace IR knihovny a také řadiče přerušení v čipu FPGA. Tyto úkoly se vykonají ve funkci `fpga_initialized()`. Funkce se zavolá po naprogramování FPGA, tedy po startu programu.

Obsluhu příkazů z terminálu zajišťuje tělo funkce `decode_user_cmd()`, která určí typ protokolu a převede textovou podobu hodnot na číselnou. Ke snímání stavu klávesnice slouží funkce `keyboard_idle()`. Zde bych rád poukázal na způsob čtení stavu klávesnice. V původním návrhu se stav klávesnice testoval v hlavní programové smyčce a pokaždé, když bylo stisknuto, případně drženo tlačítko, docházelo k odesílání infračervené informace. Toto testování je však příliš rychlé a i velice krátký stisk způsobí pokus o odeslání desítek datových rámců. Rámce se však posílají řádově pomaleji, proto docházelo k zaplňování vyrovnávací paměti i při pokusu odeslat „krátkým“ stiskem jeden rámec. Řešení jsem nalezl v použití funkce `infra_no_tx_transmit()`, která vrací kladný výsledek v případě neprobíhajícího vysílání. Dokud je návratová hodnota této funkce nulová, bude obsluha tlačítek ignorována. V opačném případě se po dekódování tlačítka odešlou pomocí globální proměnné `txdata` a funkce `infra_send()` data do FPGA. Kódy tlačítek jsou předem uloženy do konstanty typu pole struktur. Tlačítko na klávesnici FITkitu je potom indexem do tohoto pole. Sadu tlačítek (protokol) lze za běhu aplikace změnit stiskem křížku, data se potom načítají z jiného pole.

Příjem infračervených dat se uskutečňuje v hlavní programové smyčce. Na terminál se tiskne přijatý typ protokolu a následně vlastní data do doby, než se vyprázdní přijímací buffer. Poté běh programu pokračuje dále.

Kapitola 6

Dokumentace

6.1 Knihovna pro IR komunikaci

Pro správnou funkci knihovny je nutné mít správně nastavenu bázovou adresu SPI řadiče a přiřazeny offsety k použitým dekodérům. Výchozí hodnota bázové adresy `INFRA_BASE_ADDR` se rovná 90h. Pokud například požaduji aplikaci pouze s transceiverem protokolu RC5X, nastavím konstantu `IR_RC5_OFFSET` na hodnotu 0, transceiver tak bude přístupný přímo přes bázovou adresu.

K různým aplikacím se může hodit různá velikost přijímací či vysílací fronty, jejich implicitní pětimístnou velikost lze upravit změnou konstant `IR_RX_BUF_LEN` a `IR_RX_BUF_LEN`.

Sémantiku většiny funkcí jsem již popsal v implementační části, zde se proto budu věnovat formátu parametrů, které se do jednotlivých rutin předávají. Jako první zmíním proceduru `infra_rx_interrupt_handler(unsigned char proto_type)`, která přečte data z FPGA a uloží je do přijímacího bufferu. Parametr `proto_type` přitom určuje typ protokolu, a může nabývat těchto hodnot:

- `IR_RC5`,
- `IR_NEC_NORM`,
- `IR_SIRC_12B`,
- `IR_SIRC_15B`,
- `IR_SIRC_20B`.

Pomocí typu se určuje offset, který bude připočten k bázové adrese. Ke čtení položky z bufferu slouží funkce `infra_read(unsigned char *proto, unsigned char *data)`. Parametry jsou předávány odkazem, do prvního funkce uloží typ protokolu a do druhého se nahrají data z bufferu. Druhý parametr musí být typu pole o délce 3 byty. Data jsou v poli zarovnána způsobem, kde je spodní byte z FPGA uložen do prvku s indexem 2, zatímco horní byte do prvku s nejnižším indexem, tj. 0. Po provedení této rutiny se přečtená položka z bufferu automaticky odstraní. Návratová hodnota funkce je při úspěšném čtení rovna jedné, jinak nula.

Funkce `infra_send(unsigned char proto, unsigned char *data)` slouží k zápisu dat do bufferu a k jejich odvysílání. Její parametry se prakticky shodují s výše uvedenými, pouze k typu protokolu přibyla možnost `IR_NEC_REPEAT`, při jejímž zvolení se místo klasického rámce NEC posílá rámec opakovací.

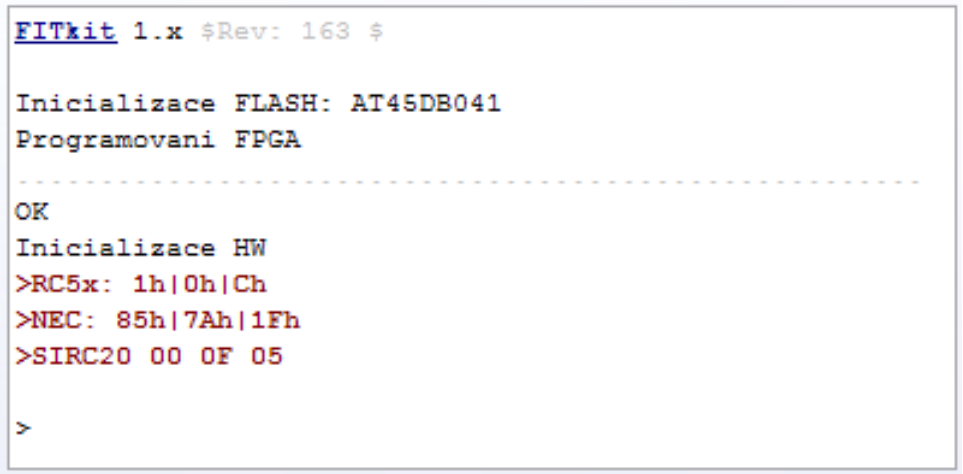
6.2 Ovládání ukázkové aplikace

Jako primární způsob ovládání aplikace jsem zvolil komunikaci přes terminálové okno. Po spojení s počítačem je možné z něj vyslat libovolný kód, případně nastavit i počet opakování. Po příjmu sekvence z dálkového ovladače a po jejím dekódování se výsledek automaticky odešle na obrazovku terminálu. Seznam příkazů včetně syntaxe lze zobrazit příkazem `HELP`. Pro ilustraci uvádím odeslání dvou totožných rámců protokolu RC5X:

```
REPEAT 2
RC5 00 0C
```

První příkaz nastaví (globálně) počet opakování rámce, zatímco druhý již nese vlastní data. Při opakovaném zadávání příkazů RC5X se toggle bit mění automaticky v aplikaci, není proto nutné jej uvádět. Hodnoty v hexadecimálním tvaru značí v uvedeném případě adresu/zařízení 00h (televizní přijímač), a hodnota 0Ch příkaz k přechodu televize do pohotovostního režimu. U protokolu NEC dochází při nastavení opakování na hodnotu větší než 1 k odeslání pouze jednoho datového rámce, zbytek vysílání bude tvořen opakovací sekvencí. Obrázek 6.1 zachycuje obsah terminálového okna po příjmu dvou rámců a po odeslání jedné datové sekvence protokolu SIRC.

Jinou možnost odeslání kódu nabízí použití klávesnice a LCD displeje. Sekvence určená k odeslání je v tomto případě uložena ve zdrojovém kódu aplikace a pro její změnu je nutné aplikaci znovu přeložit a nahrát do FITkitu. Není to příliš flexibilní způsob, použití nalezne spíše jako náhrada dálkového ovladače, poté co jsme požadované kódy odladili přes terminál. Tlačítkem # lze přepínat mezi různými protokoly, přičemž ke každému z nich může být přiřazena rozdílná sada kódů. LCD displej informuje uživatele o aktuálně zvolené sadě (protokolu). Přepínání toggle bitu protokolu RC5X, stejně jako vysílání opakovacího rámce NEC, probíhá automaticky a není vyvedeno na žádnou klávesu.



Obrázek 6.1: Snímek obrazovky terminálu

Kapitola 7

Závěr

Cílem práce bylo rozšířit FITkit o možnost dálkového ovládání spotřebičů a dekódování příchozího signálu z ovladače. Mě se podařilo navrhnout univerzální modul podporující několik nejrozšířenějších přenosových protokolů.

Při návrhu přípravku jsem experimentováním zjistil, že pro příjem dat v infračerveném pásmu na rozdílných nosných kmitočtech postačuje jeden demodulátor. Pásmová propust uvnitř přijímače se totiž vyznačuje relativně velkou tolerancí vůči nosné frekvenci.

Během implementace a ladění jsem odhalil nutnost použít pro správný chod transceiverů komponentu pro přenos informace mezi rozdílnými časovými doménami (rozdílné kmitočty hodinového signálu).

Ačkoliv jsou transceivery navrženy pro plně duplexní provoz, stávající rozšiřující modul tuto vlastnost nevyužívá. Důvodem je vysoká citlivost demodulátoru. Při současné podobě rozšiřujícího modulu se mi nepodařilo úplně odstínit vysílaný paprsek, který je tak ihned zpětně dekódován. Propojení komponent v FPGA jsem proto realizoval tak, že se příjem dat uskutečňuje pouze pokud neprobíhá vysílání. Pro obousměrnou komunikaci by bylo nutné upravit přípravek, použít vysílací LED s úzkým kuželem vyzařovaného paprsku, případně ji uzavřít do vhodného pouzdra.

Komponenty v FPGA, včetně řadičů klávesnice a LCD displeje použitých v ukázkové aplikaci, zabírají plochu cca 500 slices, to je přibližně 67 % z jejich celkového počtu. Pokud nepředpokládáme použití všech transceiverů v jediné aplikaci, nabízí se možnost implementovat zpracování i dalších protokolů. Spolu s vhodně napsanou počítačovou aplikací může projekt zastávat funkci dálkového ovládání počítače, například ovládat hlasitost, přehrávání videa, vypnutí PC, apod. Projekt by se také mohl použít, po doplnění komunikační knihovny o funkce zajišťující spolehlivé doručení dat, jako základ pro nízkorychlostní komunikaci mezi dvěma, teoreticky i více FITkity. Taková komunikace by našla uplatnění mimo jiné při hraní her pro více hráčů, například v projektu „Implementace hry Lodě na platformě FITKit“, kde by odpadla nutnost propojit zařízení kabelem, neboť dosah infračerveného paprsku je poměrně vysoký, v ideálních podmínkách větší než tři metry.

Literatura

[1] Bergmans, S.: SB-Projects: IR remote control: Philips RC-5 [online]. http://sbprojects.com/knowledge/ir/rc5.htm, 2009-09-23 [cit. 2010-04-06].

[2] Bergmans, S.: SB-Projects: IR remote control: SIRC protocol [online]. http://sbprojects.com/knowledge/ir/sirc.htm, 2009-09-23 [cit. 2010-04-08].

[3] Bergmans, S.: SB-Projects: IR remote control: NEC protocol [online]. http://sbprojects.com/knowledge/ir/nec.htm, 2009-11-19 [cit. 2010-04-08].

[4] Griffiths, P.: Sony SIRC infrared protocol [online]. http://picprojects.org.uk/projects/sirc/sonysirc.pdf, 2010 [cit. 2010-04-08].

[5] JWA Systems: A logic analyzer using the PC's parallel port [online]. http://www.xs4all.nl/~jwasys/old/diy2.html, 2009-09-10 [cit. 2010-05-09].

[6] Nagy, C.: *Embedded Systems Design Using the TI MSP430 Series*. Newnes Boston, 2003, iSBN 0-7506-7623-X.

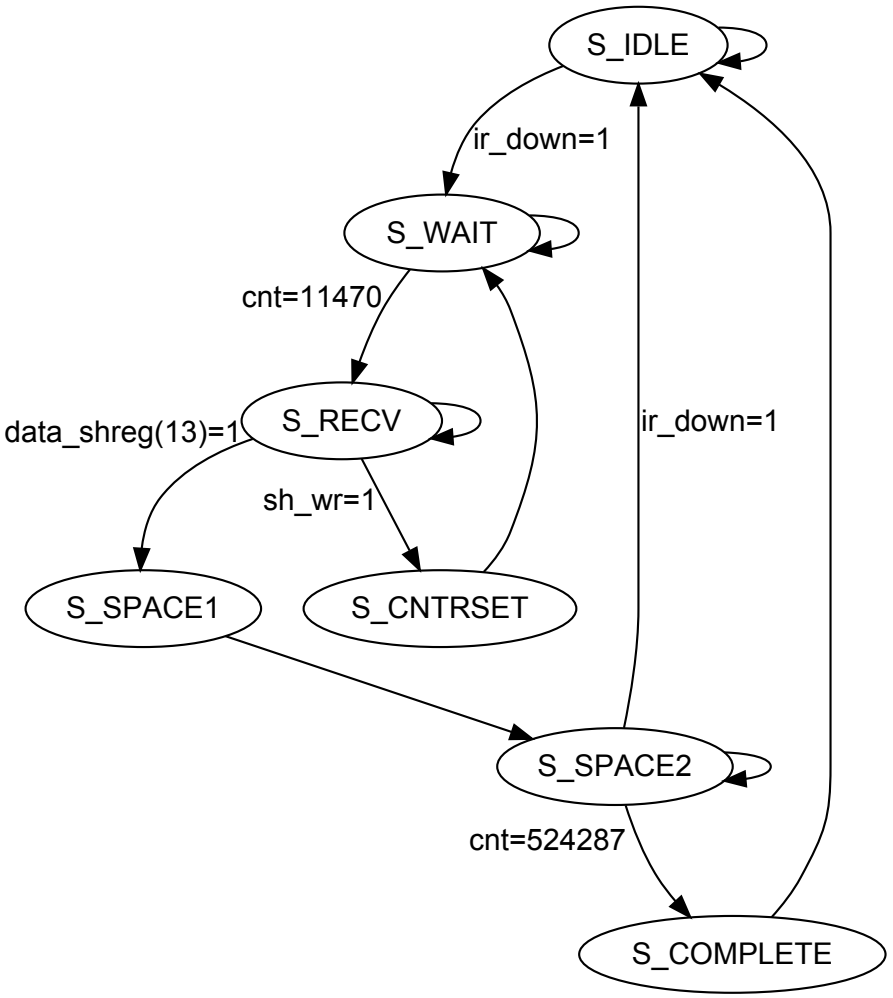
[7] NEC Corporation: MOS INTEGRATED CIRCUIT μPD6121, 6122 [online]. http://www.datasheetcatalog.org/datasheet/nec/UPD6122G-002.pdf, 1995 [cit. 2010-04-08].

[8] Vishay Semiconductor GmbH: Photo Modules for PCM Remote Control Systems [online]. http://www.datasheetcatalog.org/datasheets/208/301092_DS.pdf, 2001-04-02 [cit. 2010-04-12].

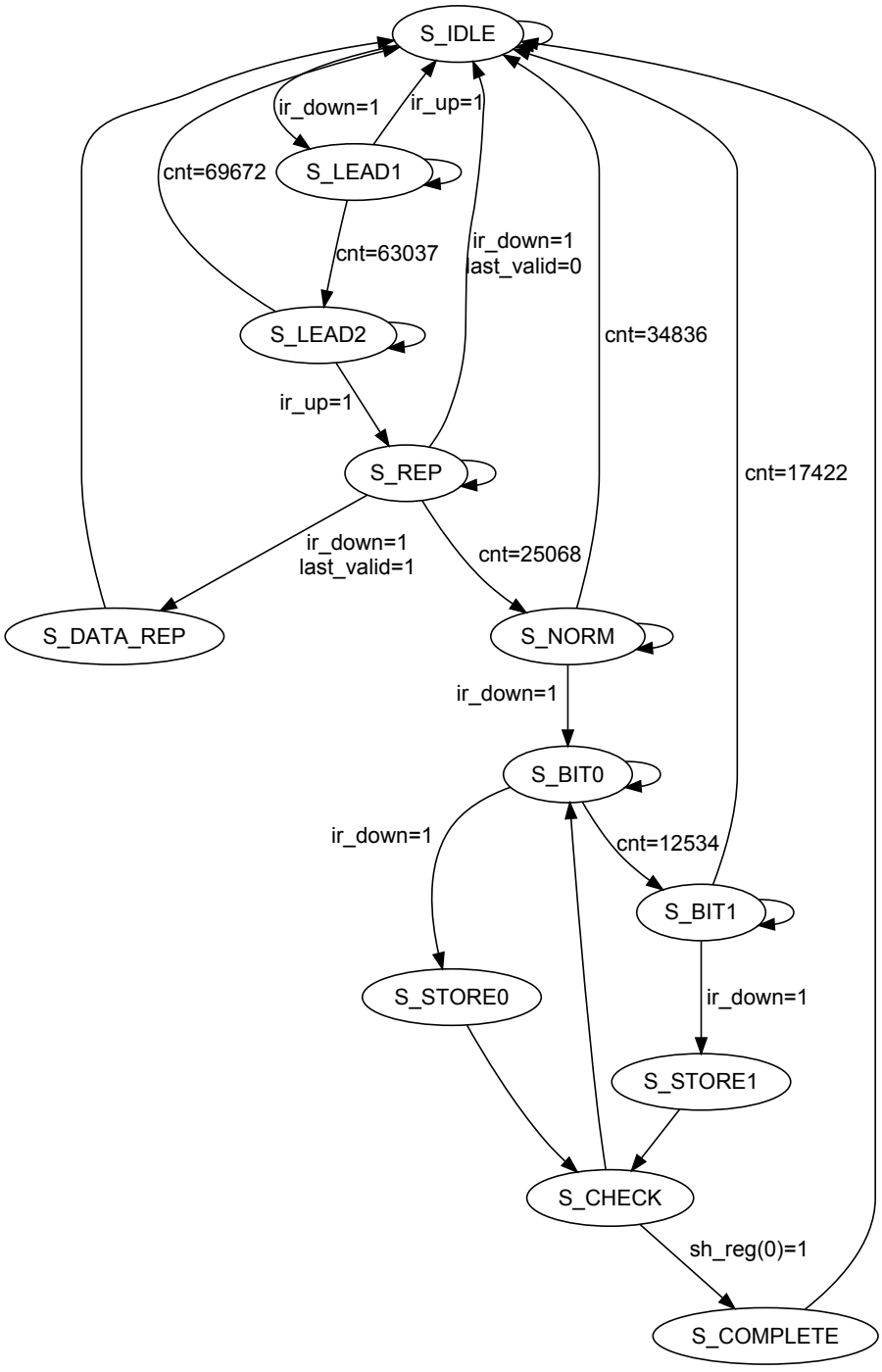
[9] Wikipedia: Infrared [online]. http://en.wikipedia.org/wiki/Infrared, 2010 [cit. 2010-04-05].

Příloha A

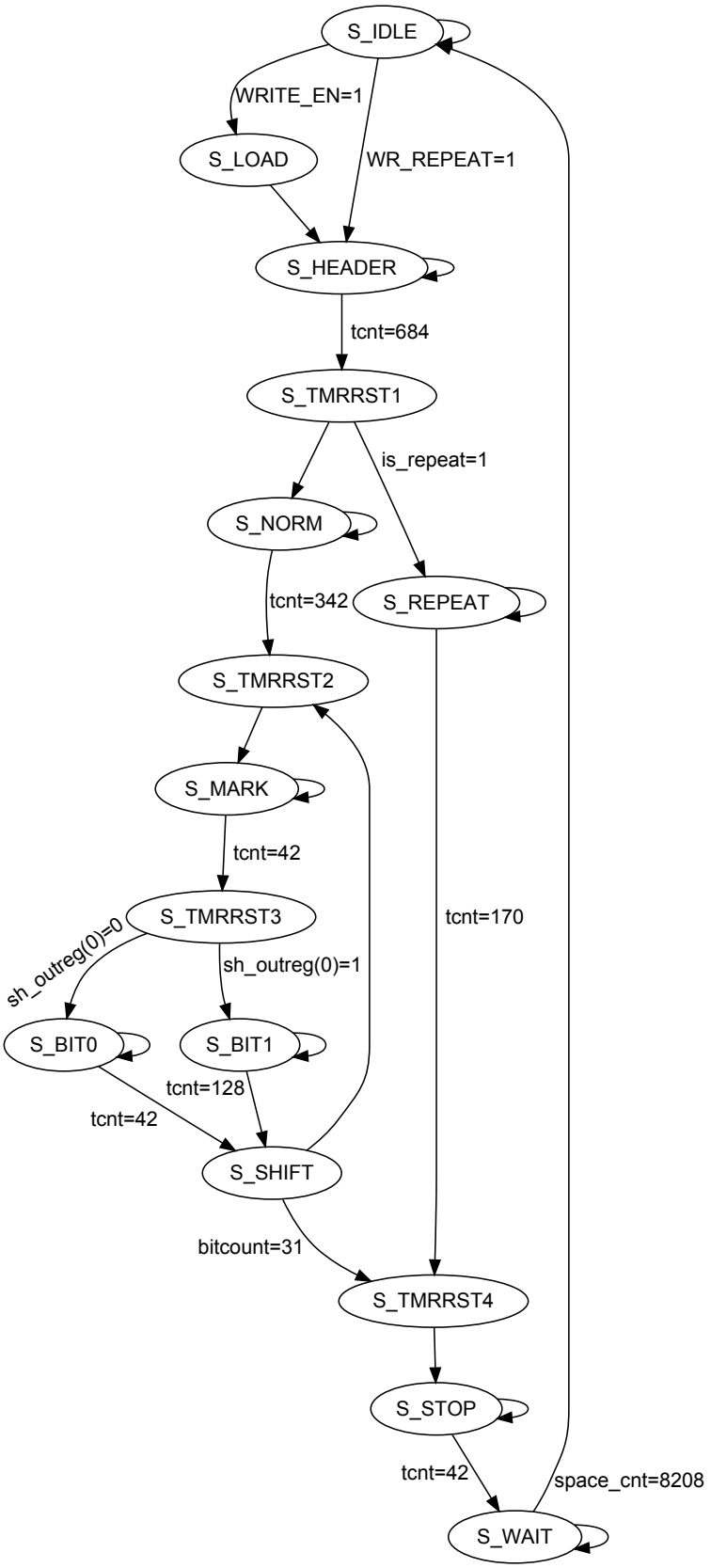
Přechodové diagramy stavových automatů



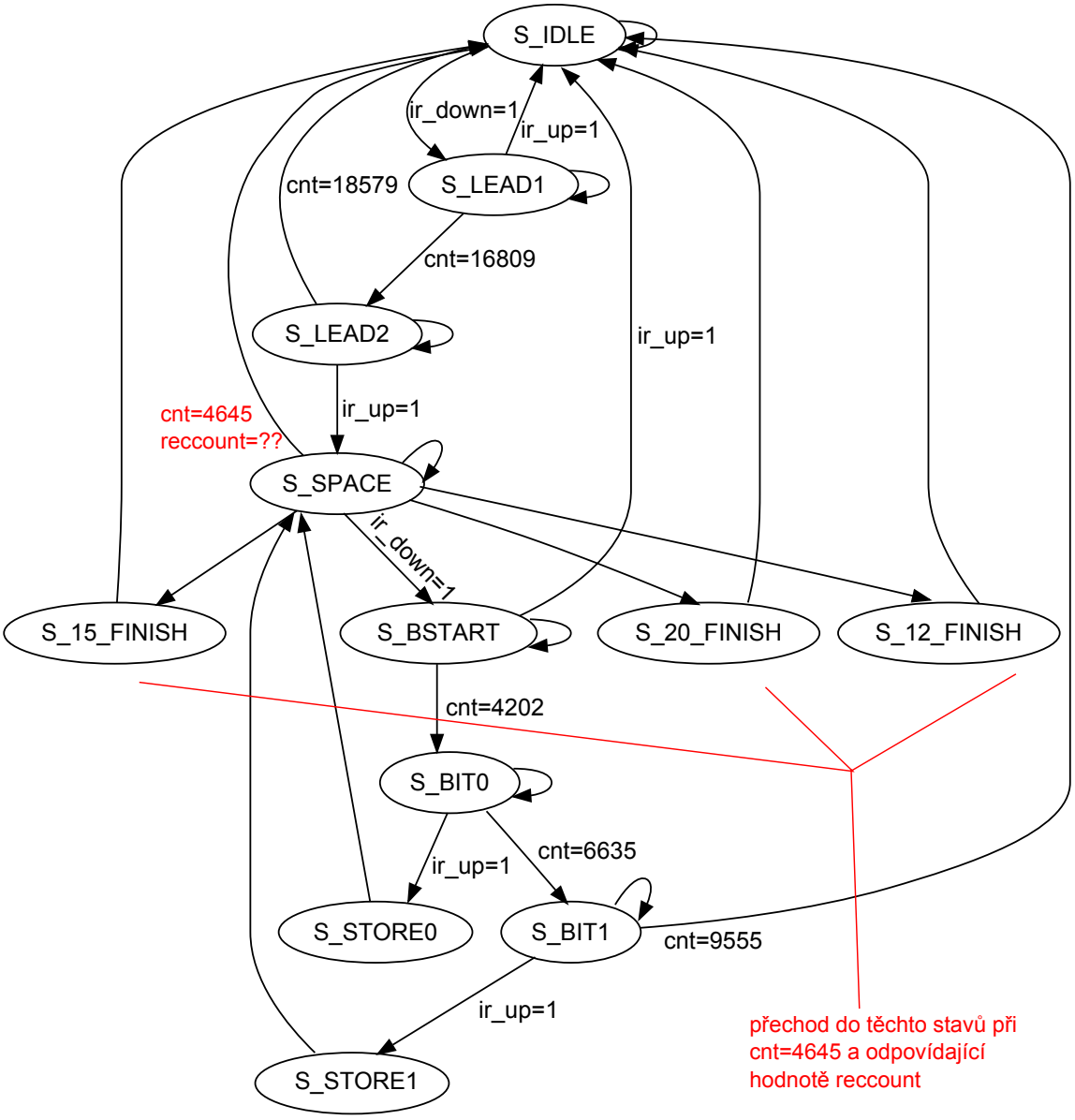
Obrázek A.1: Přechodový diagram stavového automatu přijímače RC5X



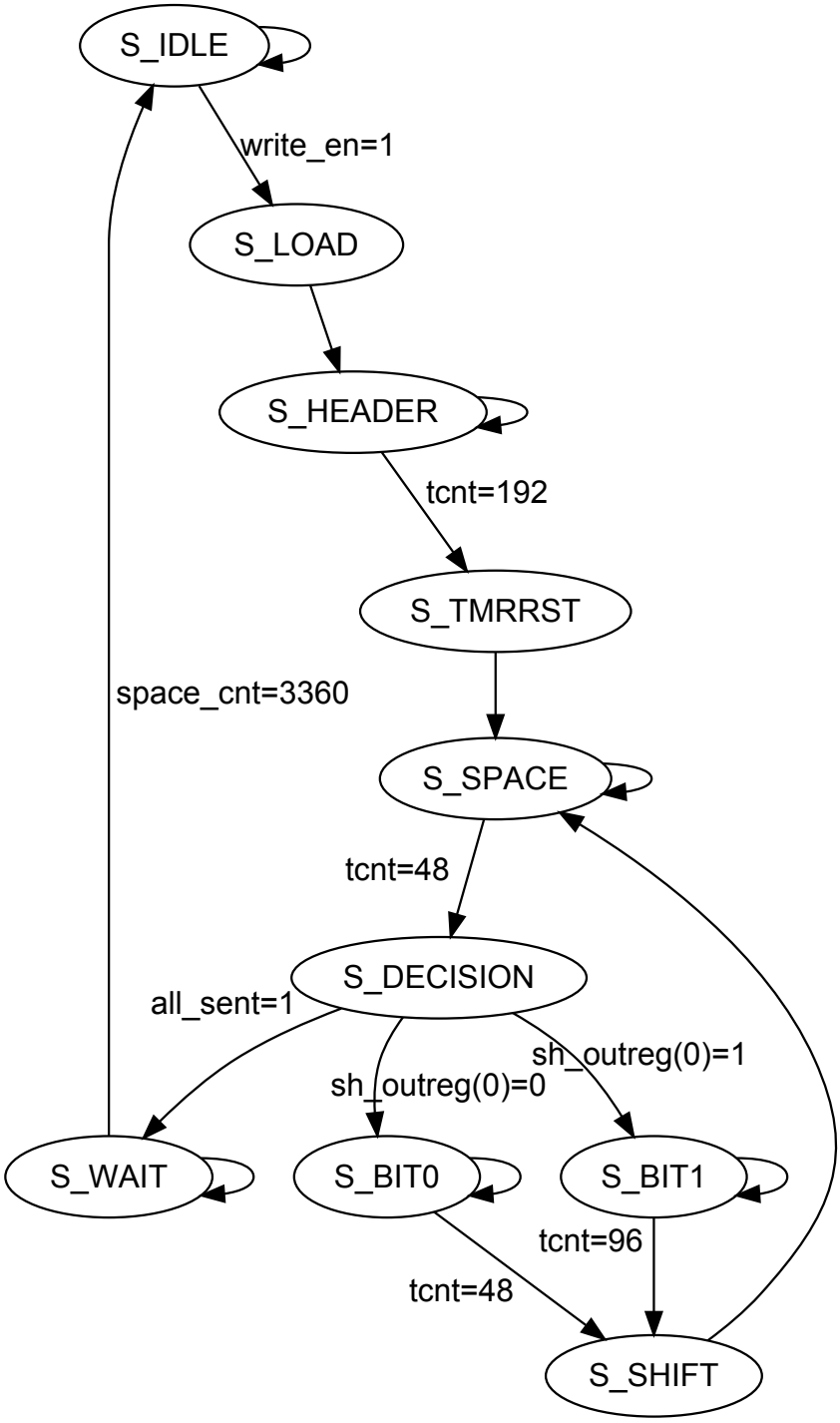
Obrázek A.2: Přechodový diagram stavového automatu přijímače NEC



Obrázek A.3: Přechodový diagram stavového automatu vysílače NEC



Obrázek A.4: Přechodový diagram stavového automatu přijímače SIRC

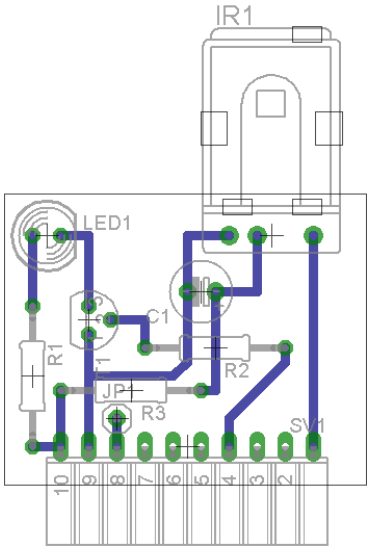


Obrázek A.5: Přechodový diagram stavového automatu vysílače SIRC

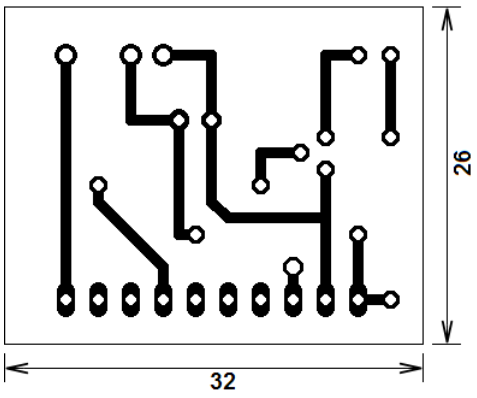
Příloha B

Rozšiřující modul

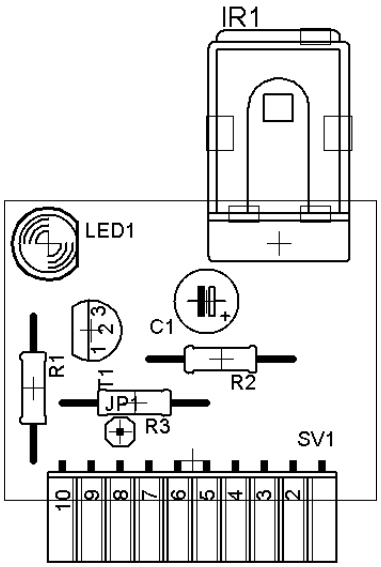
B.1 Deska plošných spojů



Obrázek B.1: Návrh desky plošných spojů



Obrázek B.2: Pohled ze strany spojů



Obrázek B.3: Osazení - pohled ze strany součástek

B.2 Seznam součástek

JP1	pin S1G01
SV1	dutinková lišta BLW810G
R1	22 R
R2	4k7
R3	100 R
C1	4,7 μF/16 V
LED1	LD274-3
T1	BC337-40
IR1	TSOP1740

Tabulka B.1: Rozpiska součástek

Příloha C

Obsah CD

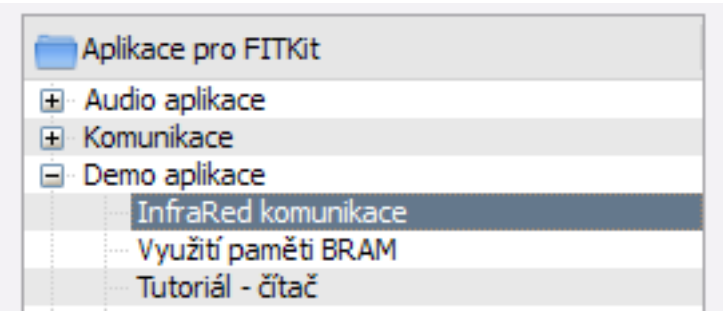
C.1 Seznam souborů

- `FITkit_SVN` – Zdrojové kódy transceiverů, komunikační knihovny a ukázkové aplikace, určené k nahrání do adresářové struktury SVN platformy FITkit.
- `Pripravek` – Soubory programu Eagle, obsahující schéma zapojení a DPS přípravku.
- `tex` – Zdrojové kódy technické zprávy bakalářské práce.
- `projekt.pdf` – Technická zpráva v elektronické podobě.

C.2 Instalace

Pro korektní instalaci předpokládám plně zprovozněný FITkit, na daném počítači musí být nainstalovaný QDevKit s GCC překladačem pro MCU a programem Xilinx ISE Webpack pro syntézu VHDL kódu. Doporučuji také stažení aktuální revize zdrojových kódu ze SVN repozitáře.

Vlastní instalace spočívá ve zkopírování obsahu adresáře `FITkit_SVN` z tohoto CD do lokální kopie SVN. Obsah podsložky `apps` se kopíruje do adresáře `apps` v SVN, stejně tak soubory z podadresářů `fpga` a `mcu` patří do odpovídajících složek v systému subversion. Ukázkový program se nachází v seznamu projektů ve skupině „Demo aplikace“ pod názvem „InfraRed komunikace“, viz obrázek C.1. Dvojitým poklepáním na označenou položku se automaticky spustí překlad projektu a následně se nahraje do připojeného FITkitu.



Obrázek C.1: QDevKit - Umístění aplikace v seznamu projektů