1 Předmluva:

Mým hlavním úkolem je zhotovit komunikační modul, který umožní ověřování a vývoj složitých regulačních algoritmů pomocí počítače PC v prostředí MATLABu. Vzhledem k existenci vhodného RT systému Real Time Toolbox distribuovaného firmou Humusoft se jedná o vytvoření driveru pro tento systém.

Vzhledem ke stavu modelu v době zahájení práce a těsné vazbě mého úkolu na HW část jsem po konzultaci s vedoucím diplomové práce přesunul těžiště svého úsilí i na řešení této problematiky. Zde spolupracuji s Pavlem Krskem, řešitelem obvodové části celého zařízení.

Tato práce společně s prací Pavla Krska navazuje na diplomovou práci ing. Pavla Beneše. Ten nemohl pro velký objem prací, vysokou složitost dílčích částí a dlouhé výrobní časy mechanických dílů celý model úspěšně dokončit.

V průběhu řešení projektu bylo potřeba řešit větší množství různorodých problémů. Jednotlivé problémy vznikaly postupně při snaze uvést celý model do provozuschopného stavu. Proto se práce skládá z většího počtu popisů dílčích úkolů, navrhnutých a realizovaných řešení uvedených problémů. Diskuse o jednotlivých částech projektu je obsažena v první kapitole.

V zadní části textu je obsaženo několik příloh s výpisy programů, jež byly napsány v rámci projektu. V textu jsou obsaženy odkazy na tyto programy. Čtenářům této práce, kteří nejsou z laboratoře automatického řízení, doporučuji nejprve začít četbu přílohou *Realizovaná zařízení*. V ní jsou obsaženy fotografie vyrobených částí, jež umožní snazší získání přehledu o celém projektu.

Hlavní řešené úkoly:

a) Dokončení konstrukce trenažéru.
b) Řízení modelu RC soupravou z počítače.
c) Snímání polohy helikoptéry (stavu systému).
d) Přenos informací o poloze k počítači.
e) Konstrukce napájecího zdroje s velkým výkonem.
f) Softwarový interface modelu. V tomto případě se jedná o interface pro matematický program MATLAB.
g) Testovací program pro ověření funkčnosti celé sestavy.

Rád bych na tomto místě poděkoval panu ing. Janu Houškovi za podporu při tvorbě komunikačního driveru, panu ing. Zoltánu Tankó a ing. Štefanu Jedlikovi za cenné informace o obvodech Xilinx a panu ing. Petru Horáčkovi CSc. za finanční a organizační zajištění celého projektu. Dále bych chtěl poděkovat panu ing. Hábovi za zhotovení mechanických dílů konstrukce trenažéru.

Tento text není původní, ale vznikl sloučením s diplomovou prací Pavla Krska. Tímto obsahuje ucelený popis problematiky vývoje modelu vrtulníku v celé její šíři.

2 Obsah

Contents

3 Úvod do problematiky

V této kapitole jsou popsány hlavní části projektu, které bylo potřeba dokončit pro uvedení modelu do provozuschopného stavu. Celá práce je členěna do kapitol, z nichž každá se zabývá jedním funkčním celkem.

3.1 Popis modelu

Základem se stal RC továrně vyráběný model helikoptéry EP CONCEPT s průměrem rotoru 890mm, jehož pohonná jednotka je tvořena SS elektromotorem LE MANS AP36 (maximální odběr 20A při 10V). Model byl vyroben firmou KYOSHO. Původně byl model řízen čtyřkanálovou RC soupravou firmy Robbe pracující v modelářském pásmu 40Mhz. Jedná se o vrtulník klasické koncepce s jedním nosným rotorem o průměru 892mm s Bell-Hillerovým ovládáním a vyrovnávacím rotorem o průměru 175mm umístěným na konci ocasu. Celková vzletová hmotnost modelu je 1200 až 1300g.

Model s elektromotorem byl zvolen pro tišší a čistější provoz. Musí však být vyřešen přívod elektrické energie do motoru. Modely vybavené spalovacím motorem mají větší rozměry a jsou uvnitř laboratoře nepoužitelné. Jejich hlavní výhodou je větší užitečná hmotnost.

Helikoptéra představuje z hlediska regulace nelineární mnoharozměrový (MIMO) systém vyššího řádu. Takový reálný systém se hodí pro praktické ověřování moderních způsobů regulace složitých dynamických systémů, protože i přes rozvoj výpočetní techniky a simulačních metod nemůže žádná simulace nahradit reálný systém. Teoreticky by bylo možno odvodit diferenciální rovnice celého modelu a provádět regulaci na simulovaném modelu. Tato úloha je však velmi komplikovaná a představovala by pro jednoho člověka práci na několik let. Vzhledem k extrémní složitosti systému by rovnice musely být odvozeny odborníkem v oboru, což by si vyžádalo vysoké náklady na jeho práci. Při odvozování rovnic vždy dojde k některým zjednodušením a vliv některých faktorů není možno vůbec předem odhadnout (např. Turbulence vzduchu v místnosti).

Naším úkolem je vytvořit dostatečně bezpečný reálný model helikoptéry, který by co nejvěrněji napodoboval chování skutečného vrtulníku a který by bylo možné řídit počítačem. K řízení je nutné vybavit model systémy pro snímání polohy a pro řízení helikoptéry.

Helikoptéra je těžko řiditelný systém a přitom lze model lehce poškodit. Proto je bezpečnost modelu zajištěna jeho umístěním na trenažéru. Trenažér se skládá ze dvou Cardanových kloubů spojených výsuvnou tyčí. Kloub I (pod vrtulníkem) zajišťuje kromě náklonu ve dvou osách i možnost otáčení helikoptéry kolem svislé osy. Kloub II (pod heliportem) zajistí opět náklon ve dvou osách a navíc vysouvání spojovací tyče. Dohromady je tak zajištěn prakticky volný pohyb helikoptéry v kuželu nad heliportem. Kloub II s heliportem je umístěn na konstrukci tak, aby se model nemohl dostat do blízkosti překážek. Bezpečnost modelu je vyk-

oupena omezením pracovního prostoru modelu a bohužel i menší změnou dynamických vlastností modelu.

3.1.1 Hlavní části helikoptéry

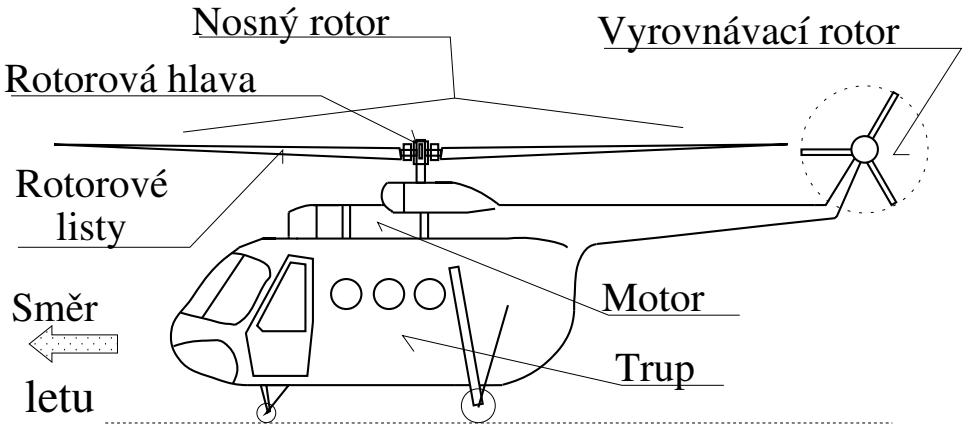


Figure 1: Hlavní části helikoptéry

Byl vybrán typ vrtulníku s jedním nosným rotorem. Proto je další popis vrtulníku zaměřen právě na tento typ. Na listech otáčejícího se *nosného rotoru* vzniká vztlaková síla, která představuje nosnou sílu celé helikoptéry. Působením odporu prostředí na listy je vytvářen ještě reakční moment. Jeho působením by došlo k rotaci *trupu helikoptéry*, a proto je nutno provést kompenzaci reakčního momentu. Kompenzace je prováděna vyrovnávacím momentem, který vzniká otáčením *vyrovnávacího rotoru*. Vyrovnávací rotor je obvykle umístěn na zádi vrtulníku, přibližně v ose souměrnosti stroje a v rovině nosného rotoru tak, aby se eliminovala nevyvážená síla vzniklá jeho tahem a velký sklon trupu. Pohon obou rotorů je zajištěn *motorem* přes *rozvodnou soustavu*. Jednotlivá ústrojí jsou v trupu rozmístěna s ohledem na těžiště, které by mělo být v optimálním případě umístěno pod středem nosného rotoru. Rozmístění základních částí vrtulníku je ukázáno na 3.1.1.

3.1.2 Nosný rotor

Dvoulistý nosný rotr má průměr 892mm. Rotor je opatřen tuhými závěsy rotorových listů a pomocným řídícím rotorem aerodinamického Hillerova stabilizátoru, který tvoří dva krátké listy na koncích opatřené řízenými ploškami. Nastavování listů nosného rotoru, jak cyklické, tak kolektivní zajišťuje systém s řídící deskou ovládaný třemi servy řídícího systému

přes pákové mechanismy.

3.1.3 Trup vrtulníku a pohonná jednotka

Trup vrtulníku má délku asi 200mm a šířku 60mm. Je tvořen základnou na níž stojí konstrukce, která spojuje základnu trupu s ocasní částí a slouží k ukotvení hřídele hlavního rotoru. Základna tvoří plochou krabičku v níž je umístěn převodový systém pro pohon rotorů. V zadní třetině trupu je ukotven hřídel hlavního rotoru spojený s převodovou soustavou přes jednosměrnou spojku (tzv. cvrčka). Spojka zaručuje otáčení hlavního rotoru i v případě, že dojde k výpadku hnací jednotky a umožňuje tak nouzové přistání autorotací. A naopak, běží-li motor, pohání přes spojku nosný rotor bez omezení.

V ose trupu před hlavním hřídelem je vertikálně umístěn elektromotor s osou směřující směrem dolů, kde je v základové desce spojena s převodovým ústrojím. Pohon vyrovnávacího rotoru je řešen hřídelem v zadní části trupu spojeným přímo s převodovkou. Tento hřídel pohání přes řemenový převod vyrovnávací rotor. Taková konfigurace náhonu rotorů je nevýhodná tím, že v případě letu autorotací není vyrovnávací rotor poháněn, ale je dokonce brzděn zablokovanou pohonnou soustavou. Tím se značně zhorší řiditelnost celého vrtulníku.

3.1.4 Ocas a vyrovnávací rotor

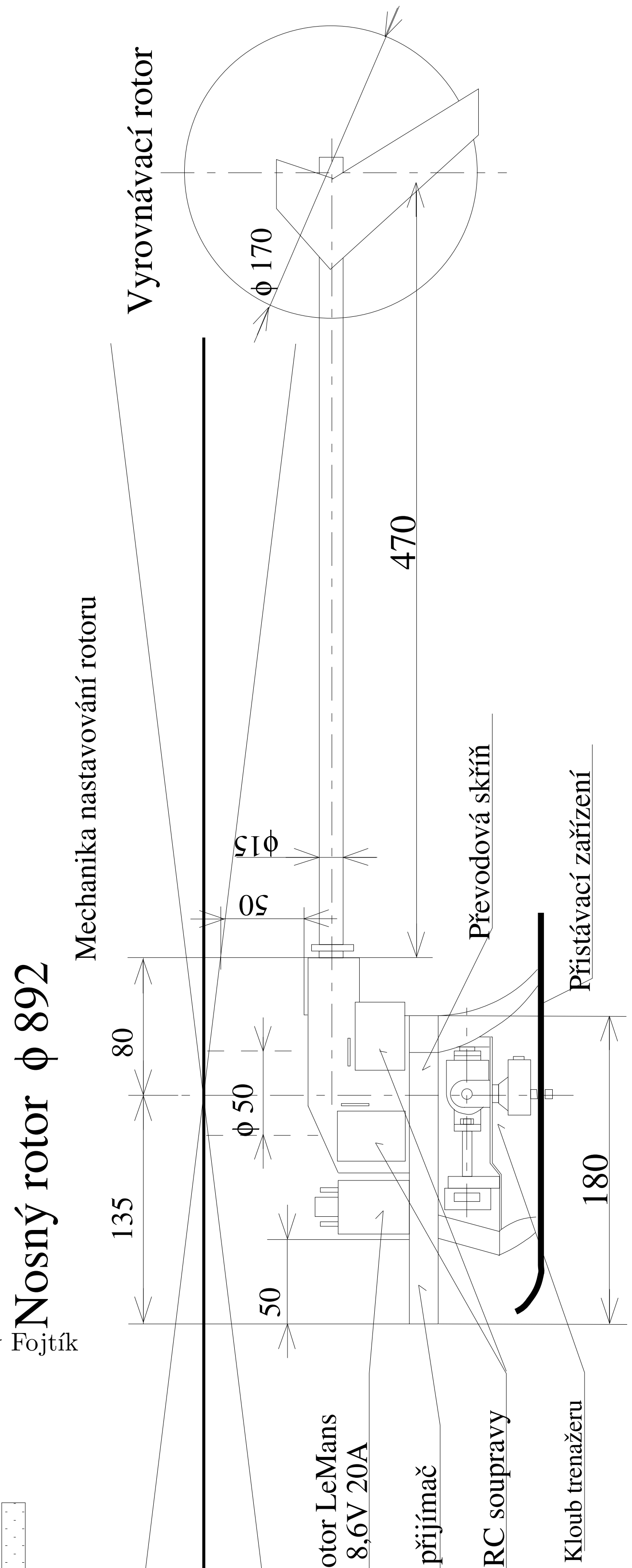
Ocas je tvořen duralovou trubkou o průměru 15mm a délce asi 470mm. Jejím středem prochází gumový pás, který pohání vyrovnávací rotor. Ten spolu s mechanikou pro ovládání náběhu jeho listů je upevněn na konci duralové trubky. Servo je spojeno s mechanismem zadního rotoru lankem. V zadní části ocasu jsou vertikální a horizontální stabilizační plochy.

3.1.5 Řídící a přistávací systémy

Akčními členy řídící soupravy jsou čtyři modelářská serva přimontovaná po obou stranách ke konstrukci trupu, vždy dvě a dvě na jedné straně. Řízení modelu zajišťuje čtyřkanálová RC soprava ATTACK-4 (FP T4NBL) firmy Robbe, která pracuje s kmitočtem 40,695MHz. Přijímač RC soupravy je umístěn v přední části trupu před hnacím motorem, kde je pro jeho umístění prostor o ploše 50x60mm. Přistávací zařízení (podvozek) tvoří dvě ližiny vytvarované z duralových trubiček o průměru 4mm.

3.2 Způsob řízení helikoptéry

Klasický vrtulník je ovládán změnou směru a velikosti vektoru tahu nosného rotoru a změnou velikosti tahu vyrovnávacího rotoru.


Vyrovnávací rotor
ϕ 170
470
Mechanika nastavování rotoru
ϕ15
50
Převodová skříň
Přistávací zařízení
Nosný rotor ϕ 892
80
ϕ 50
135
50
180
8,6V 20A
přijímač
RC soupravy
Kloub trenažeru

Změna velikosti tahu vyrovnávacího rotoru způsobí momentovou nerovnováhu a tím otáčení stroje kolem svislé osy, tedy změnu směru letu. Velikost momentu ovládá pilot pedály nožního řízení, které mění úhel náběhu listů vyrovnávacího rotoru, a tak mění i jeho tah.

Změnou velikosti tahu nosného rotoru se dociluje stoupání či klesání helikoptéry. Protože tah se řídí nastavením náběhu všech listů rotoru společně, nazývá se tento způsob kolektivní řízení či kolektiv. Páka kolektivu je podobná páce automobilové ruční brzdy, ale umisťuje se vlevo od sedadla pilota.

Obvykle na páce kolektivu bývá otočná rukověť, která ovládá přípusť motoru. Její pomocí lze nastavit otáčky motoru a tím i nosného rotoru. Pro chod motoru se však obvykle požaduje udržení otáček v oblasti, kdy motor dodává maximální výkon či má největší účinnost. Přípusť se tak většinou nepoužívá k přímému ovlivňování letu.

Změnou náběhu listů v průběhu jedné otáčky se docílí vzniku kroutícího momentu, který naklání celý vrtulník. Spolu s nakloněním roviny nosného rotoru se nakloní i vektor tahu a vytvoří se tak vodorovná síla, která způsobuje posuvný pohyb vrtulníku. Náklon je možno provést jak v podélné tak v příčné ose podle toho, kde se v otáčce zvětší či zmenší úhel náběh listů rotoru. Protože ke změně úhlu náběhu dochází cyklicky v průběhu otáčení rotoru, nazývá se toto řízení cyklické či zkráceně cyklika. Cyklika se ovládá pohybem řídící páky , která je umístěna před sedadlem pilota.

Helikoptéra je mnohem méně stabilní než klasické letadlo. Nestabilita při řízení vyžaduje zvýšenou pozornost pilota a zvětšuje nebezpečí havárie. Pro zvýšení stability jsou helikoptéry vybaveny stabilizátory, které mají za úkol tlumit naklánění vektoru tahu nosného rotoru a jsou vřazeny mezi pilota a cyklické řízení. Nejznámější jsou stabilizační systémy Bellův a Hillerův. Podstatou Bellova systému je mechanický setrvačník, zatímco Hiller doplnil rotor o pomocné řídící plošky. Dnes se tyto systémy nahrazují hydraulickými posilovači, nebo přímo autopilotem.

Vrtulníky jsou také citlivější na vítr a turbulenci, než klasická letadla. Situace je lepší při rychlém dopředném pohybu, kdy se chování helikoptéry blíží chování letounu s pevnými křídly. Obecně se dá říci, že řízení vrtulníku je složitý a náročný úkol.

3.2.1 Letové režimy

V této části uvedené režimy představují možná zadání budoucích experimentů s řízením modelu. Použitý model se snaží co nejvěrněji se přiblížit chování skutečného vrtulníku, a proto většina režimů přímo vychází z provozních režimů běžných vrtulníků.

Naklánění a otáčení trupu:

Režim je umožněn mechanickou konstrukcí trenažéru a spočívá v pevné fixaci pohyblivé tyče ve spodním Cardanově kloubu. Hlavní výhodou je značné zjednodušení celého systému vhodné v přípravných fázích návrhu regulátoru. Při fixaci tyče však vznikne pevný bod, který způsobí změny v dynamice modelu.

Start:
V každé z následujících regulačních úloh musí být vyřešen start vrtulníku. Při startu se projevuje tzv. přízemní jev, který mění chování vrtulníku a tím celý manévr komplikuje. Při použití malé přistávací plošiny, umístěné ve větší vzdálenosti od podlahy a stěn než je průměr rotoru, by se přízemní jev neměl podstatně projevovat.
Přistání:
Operace přistání je obdobná startu. Při klesání se mění způsob obtékání rotoru a spolu s přízemním jevem je značně komplikováno přiblížení k přistávací plošině.
Visení:
Vrtulník zůstává na jednom místě a udržuje si konstantní výšku. Dopředný let zlepšuje stabilitu vrtulníku. Při jeho absenci se zhoršuje stabilita celého modelu. Tento způsob letu představuje obvyklý režim činnosti vrtulníků. Předpokládám, že udržení modelu v režimu visení bude představovat nejčastější regulační úlohu.
Přímý let:
Při naklonění trupu vrtulníku v režimu visení provedeného pomocí změny cykliky se vrtulník začne pohybovat ve směru náklonu. Míra naklonění určuje rychlost letu. Při použití trenažéru je dosah vrtulníku omezen délkou vodící tyče. Proto lze provést přímý let jen po velmi krátké dráze.
Plynulá zatáčka:
Zatáčka je prováděna z režimu přímého letu plynulou změnou cykliky při součinnosti zadní vrtulky. Dochází k natáčení vrtulníku do strany. Vzhledem ke zvolené konstrukci závěsu bude pro zvládnutí diskutovaného režimu vhodný pohyb po kruhové trajektorii.
Přistání autorotací:
Je obdobou nouzového přistání letadel a provádí se většinou při vysazení motoru. Provedení autorotace vyžaduje určitou výšku. Volnoběžka odpojí hřídel rotoru od motoru a pilot nastaví negativní náklon listů. Klesáním vrtulníku dojde k prudkému nárůstu otáček. V blízkosti země pilot nastaví zpět kolektivní řízení a setrvačná energie rotoru postačí ke zpomalení klesání. Tento režim nemůže být zkoušen pomocí trenažéru.

3.3 Souřadný systém

V letectví se pro popis pohybu letadla používá různých soustav souřadnic. Pro navigaci se užívá souřadná soustava odvozená od kartografické soustavy poledníků a rovnoběžek. Jedná se vlastně o sférickou souřadnou soustavu, kde je poloha popsána zemskou délkou, šířkou a výškou nad povrchem země. Při studiu dynamiky letu je většinou zbytečné uvažovat zaoblení země a komplikovat si tak výpočty, když rozdíly jsou prakticky zanedbatelné. To platí obzvláště pro náš model helikoptéry pohybující se pomalu a na velmi malém prostoru. Pro popis dynamiky letu se proto užívá pravoúhlé soustavy souřadnic.

Systém pro popis letu letadla se skládá z několika sořadných systémů,

které jsou spolu provázány. Vztahy mezi těmito systémy pak popisují polohu letadla v prostoru a polohu vůči proudu vzduchu, která je pro let velmi důležitá.

Základní soustavou souřadnic je **zemský souřadný systém** $(\mathbf{O_g;x_g,y_g,z_g})$, jehož rovina $\mathbf{x_g,z_g}$ je vodorovná a představuje povrch země. Polohová souřadnice $\mathbf{y_g}$ pak odpovídá výšce nad zemí. Pro zjednodušení se tento souřadný systém považuje za inerciální.

Těžiště letadla tvoří počátek **letadlové soustavy souřadnic** $(\mathbf{O;x,y,z})$. Osa **x** je orientována od zádi k přídi trupu, tedy v předpokládaném směru letu, osa **y** směřuje kolmo vzhůru a tvoří tak rovinu symetrie letounu. Osa **z** je kolmá k rovině **x,y** a směřuje do strany vpravo od osy **x**.

Z hlediska letu je důležitá **aerometrická sostava souřadnic** $(\mathbf{O^a;x_a,y_a,z_a})$, jejíž osa $\mathbf{x_a}$ je rovnoběžná s vektorem rychlosti vzduchu obtékajícího letadlo a je orientována opačně, nebo-li odpovídá vektoru vzdušné rychlosti. Osy $\mathbf{y_a,z_a}$ jsou pak orientovány obdobně jako u letadlového souřadného systému.

Vztah mezi zemským souřadným systémem a souřadným systémem

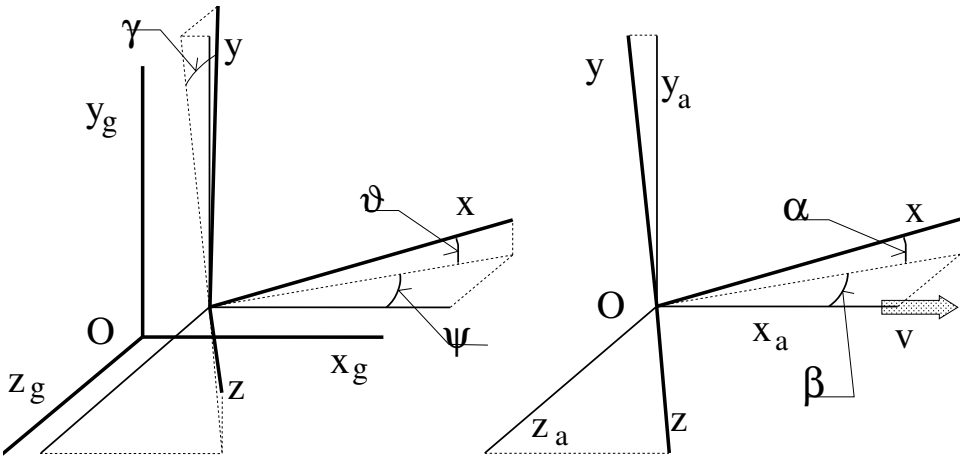


Figure 3: Souřadné systémy pro popis pohybu letadla

letadla udává obecnou polohu letadla v prostoru. Polohu letadla v prostoru popisujeme šesti parametry:

x, y, z ... Poloha těžiště letadla

ψ ..... Kurz - otočení kolem osy y souřadného systému letadla

ϑ ..... Podélný náklon - otočení kolem osy z

γ ..... Příčný náklon - potočení kolem osy x letadla.

Následující rovnice popisují vztah letadlové a zemské souřadné soustavy:

$$\mathbf{x_g} = [x_g, y_g, y_g, 1]^T \qquad \mathbf{x} = [x, y, z, 1]^T \tag{1}$$

$$\mathbf{x_g} = \begin{bmatrix} 0 & 0 & 0 & x \\ 0 & 0 & 0 & y \\ 0 & 0 & 0 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\psi & 0 & -\sin\psi & 0 \\ 0 & 0 & 0 & 0 \\ \sin\psi & 0 & \cos\psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\vartheta & -\sin\vartheta & 0 & 0 \\ \sin\vartheta & \cos\vartheta & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \cos\gamma & \sin\gamma & 0 \\ 0 & -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x} \tag{2}$$

Vztah mezi souřadnou soustavou aerometrickou a letadlovou popisuje aerodynamické podmínky vzhledem k letadlu. Aerodynamické podmínky se popisují omezeným počtem parametrů a není tak určena úplná transformace souřadných soustav, jak je tomu při udávání polohy letadla, protože nemá smysl udávat náklon letadla příčně k proudu vzduchu, ani jeho absolutní polohu vůči tomuto proudu. Aerodynamická situace je tak popsána třemi parametry:

α ..... Úhel náběhu - úhel svíraný vektorem vzdušné rychlosti a osou letadla x v rovině symetrie letadla x,y

β ..... Úhel vybočení - úhel svíraný vektorem vzdušné rychlosti a osou letadla x v rovině křídel letadla x,z.

v ..... Vzdušná rychlost - rychlost letadla vůči okolnímu vzduchu. Jedná se o vektorovou veličinu, ale zde je myšlena pouze absolutní hodnota, protože její směr je dán parametry α,β.

Následující rovnice popisují vztah letadlové a aerometrické souřadné soustavy:

$$\mathbf{x_a} = [x_a, y_a, y_a, 1]^T \qquad \mathbf{x} = [x, y, z, 1]^T \tag{3}$$

$$\mathbf{x_a} = \begin{bmatrix} \cos\beta & 0 & -\sin\beta & 0 \\ 0 & 0 & 0 & 0 \\ \sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x} \tag{4}$$

Lepší představu o popisu polohy letadla a vzdušné situace si můžeme udělat z 3.3 a výše uvedených rovnic.

Poznámka: Popsaná orientace os souřadných systémů odpovídá normě GOST. Existuje však ještě popis podle normy ISO, kde jsou zaměněny osy y, z. Je změněna i orientace osy y, takže v obou normách jde o souřadné systémy kladně orientované. Grafické porovnání norem je na 3.3.

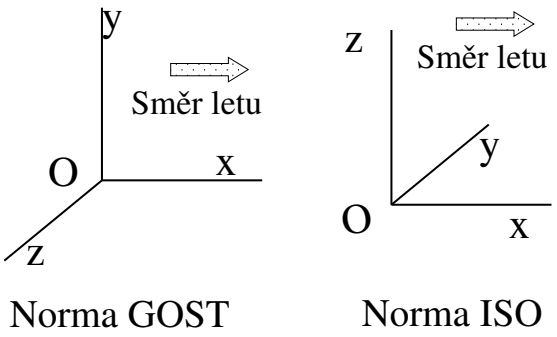


Figure 4: Rozdíl orientace os normy GOST a ISO

3.4 Funkční bloky

3.4.1 Řízení modelu RC soupravou z počítače

Řízení zajišťuje čtyřkanálová RC souprava ROBBE. Jeden kanál ovládá nastavení listů pomocného rotoru, druhý pak nastavení listů hlavního rotoru tzv. kolektiv (tah). Zbylé dva kanály zajišťují nastavování listů hlavního rotoru v průběhu jedné otáčky tzv. cyklika (náklon helikoptéry). RC souprava byla doplněna pátým kanálem pro pulsně šířkové řízení výkonu motoru a tím i otáček hlavního rotoru. Aby bylo možné řízení i z počítače, byl vytvořen interface na základě dvou časovacích obvodů 8253, které tvoří pulsněšířkovou modulaci pro serva. Tento signál je zaveden do původní RC soupravy před AM modulátor. Vysílací část a zbytek řetězce je tak využit z původní soupravy, včetně možnosti přepnout na ruční řízení [viz dipl. práce Pavel Beneš 1993].

3.4.2 Snímání polohy helikoptéry (stavu systému)

Bylo prověřováno několik možností snímání polohy (ultrazvuk, kamery, UHF...), ale nakonec se ukázalo jako nejschůdnější řešení doplnit trenažér IRC snímači polohy. Zpracování informace z těchto snímačů je poměrně náročná logická operace.

3.4.3 Přenos informace o poloze k počítači

Místa vyhodnocení snímačů a místa zpracování (interface, PC) jsou od sebe poměrně vzdálená. Jedna z desek je dokonce umístěna na volně se pohybující helikoptéře. Proto je pro předávání dat potřeba zvolit kód s vyšší bezpečností a rychlým zotavením z chyb tak, aby bylo spojení zajištěno i za zhoršených podmínek.

Přenos sériových dat z helikoptéry je zajištěn bezdrátovým spojem. Signál je amplitudově modulován na nosnou vlnu s kmitočtem 18Mhz a vysílán s velmi malým výkonem (10mW). Přijímač je umístěn pod heliportem, odkud je již signál veden vodiči spolu se signálem z desky II k počítači podle 6.5.4. Anténní systém tvoří dvě cívky, které jsou vázány nejen vzduchem, ale i přes konstrukci trenažéru.

Zájemce o podrobnější informace o této problematice odkazuji na [dipl. práce Pavel Krsek 1995].

3.4.4 Softwarový interface modelu

V laboratoři automatického řízení na ČVUT FEL se pro řešení regulačních úloh používá matematický program MATLAB. Pro komunikaci s okolními modely je využíván softwarový balík pro regulaci v reálném čase Real time toolbox. Pro zajištění přístupu k řízení modelu helikoptéry je potřeba napsat ovladač. Ovladač by měl přepočítat souřadný systém modelu, který je sférický, na systém kartézský, jež je pro regulaci vhodnější. Dodatečně lze požadovat na ovladači, aby znemožnil nesmyslné akční zásahy a zajistil vypnutí motoru v době, kdy se model vymkne kontrole.

Hardware může umožnit i zjištění kvality přenosu dat. S těmito informacemi by mohl ovladač počítat a případně varovat uživatele při podstatném zhoršení kvality přenosu pod únosnou mez. Pro regulaci je třeba znát i derivace resp. diference stavu systému. Je lépe počítá-li derivaci sám ovladač, který má k dispozici více informací, než je obsaženo v samotných stavových signálech.

Popisu rozhraní je věnována kapitola *Programové rozhraní s RT Toolboxem* a vlastní ovladač modelu je popsán v kapitole *Programová část projektu*.

4 Konstrukce trenažéru

Pro provoz modelu v laboratoři musí být bezpodmínečně zajištěno upoutání modelu s cílem zabránit případné havárii. Ta může nastat z mnoha příčin. Ideální by byla mechanická konstrukce, která na jedné straně zajistí bezpečnost modelu a zároveň umožní volný pohyb modelu. Je zřejmé, že uvedený ideál není zcela dosažitelný. Proto je třeba zvolit vhodný kompromis mezi volným pohybem a bezpečností modelu.

Je nutno přihlédnout k vysoké ceně modelu a ještě vyšší ceně náhradních dílů. Dále by mělo být počítáno s tím, že na modelu bude probíhat výuka řízení a navržené regulátory mohou občas postrádat stabilitu. Z uvedených důvodů je potřeba preferovat zejména bezpečnost modelu.

4.1 Uchycení modelu k trenažéru

Pro nácvik pilotů se někdy používá konstrukce podle obrázku 4.1. Zobrazený způsob však nechrání model před kontaktem roztočených listů rotoru s konstrukcí popřípadě zemí. Bylo ověřeno, že tento kontakt způsobí zničení listů a pokroucení konstrukce. Možné havárii by se dalo zabránit jen za cenu drastického omezení pohyblivosti modelu nastavením dorazů kloubů konstrukce.

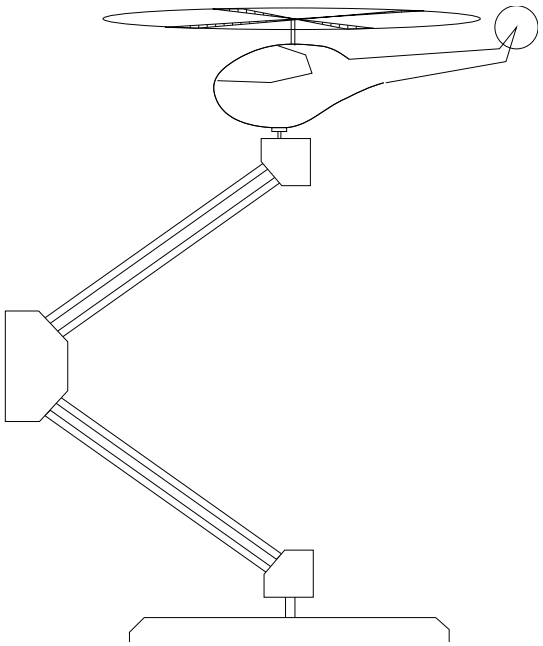


Figure 5: Cvičná konstrukce

Většina z uvedených problémů je řešena návrhem konstrukce používající dvou Cardanových kloubů. Dva možné návrhy jsou zobrazeny na obrázcích 4.1 a 4.1.

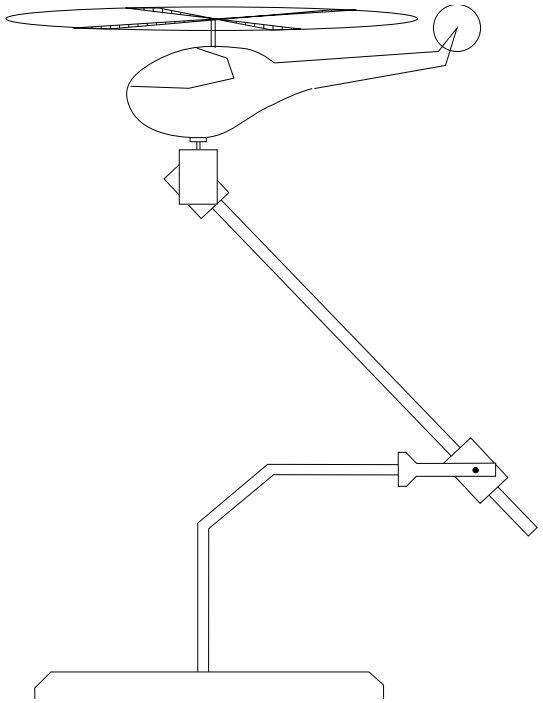


Figure 6: Vlečená tyč

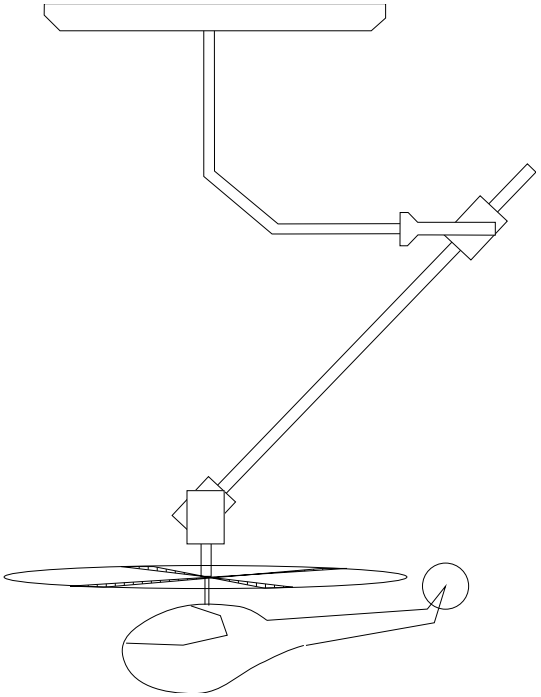


Figure 7: Konstrukce se závěsem

V prvním případě připomíná mechanické řešení kyvadlo. Za hlavní výhodu zobrazeného řešení považuji spolehlivou ochranu vrtulníku před pádem. Ten bude nejspíše představovat velmi častý typ havárie. Při pádu dojde k plynulému prokývnutí rovnovážnou polohou. Spojovací tyč, která je kritickým článkem konstrukce, nebude mechanicky namáhána na ohyb. Avšak je potřeba zachytit vrtulník v těžišti seshora. Při pokusu obejít hlavní rotor nelze při dodržení malé hmotnosti vytvořit tak velkou závěsnou konstrukci s dostatečnou tuhostí. Další možností je vedení závěsu středem hlavního rotoru. To by však vyžadovalo velký zásah do celé mechanické konstrukce.

Z těchto důvodů byl použit druhý způsob ve tvaru obráceného kyvadla, který eliminuje nedostatky předchozího řešení. Při volném pádu je hlavní tyč namáhána na ohyb. Bylo vyzkoušeno, že při vhodném tvaru tyče a malé hmotnosti vrtulníku k jejímu poškození nedojde. Na přistávací plošinu bude potřeba nalepit vrstvu pěnové gumy pro ztlumení nárazu při pádu. Dále je vhodné omezit pohyb kloubů, aby v libovolné dosažitelné poloze nedošlo ke styku vrtule s nosnou konstrukcí.

4.2 Návrh konstrukce trenažéru

Hlavním požadavkem kladeným na nosnou konstrukci je její dostatečná tuhost. Při zapnutí modelu by nemělo docházet k její rezonanci s otáčkami motoru. S tím souvisí i vhodné uchycení konstrukce k podložce zobrazené na 4.2 popř. připevnění ke zdi podle 4.2.

Další velmi důležitý požadavek představuje zamezení styku roztočeného rotoru s jakoukoli částí konstrukce. Ten by měl v lepším případě za následek zničení listů rotoru. Vodící tyč při pohybu vrtulníku vyplňuje prostor kužele. Tento prostor by také měl zůstat volný. Styk vodící tyče s nosnou konstrukcí nemá sice kritické následky pro vlastní model, avšak zavádí do systému nelinearitu, která velmi komplikuje až znemožňuje proces identifikace a regulace.

V úvahu přicházeli dvě varianty. První variantou je provést připevnění ve třech bodech ke zdi podle 4.2. Připevnění ke zdi by si vyžádalo

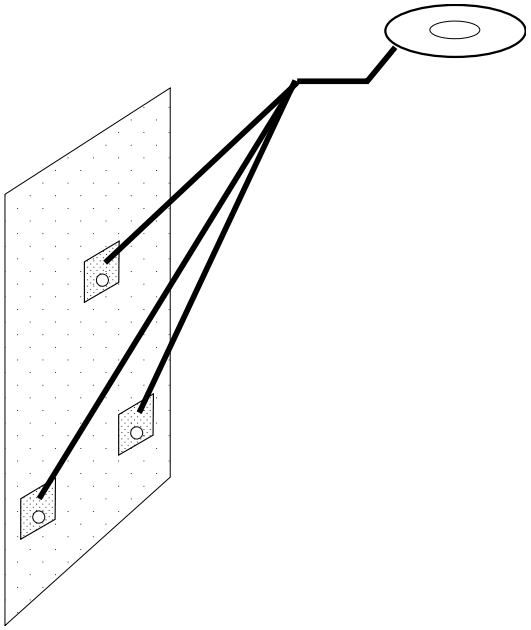


Figure 8: Tříbodové uchycení ke zdi

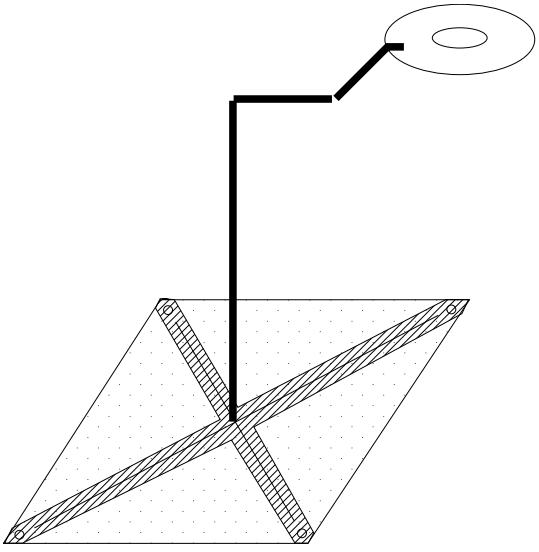
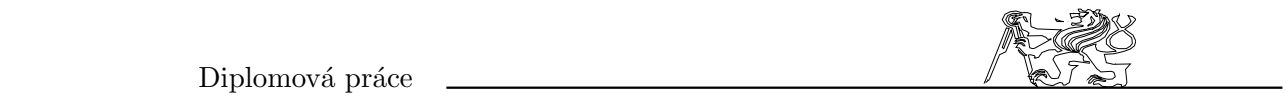


Figure 9: Uchycení k podložce



mechanickou realizaci nosníků a vrtání nosných děr. Tím by byl model odkázán pouze na jedno místo a opětovná instalace v případě stěhování laboratoře by se komplikovala. V blízkosti zdi, podobně jako země, vznikají turbulence vzduchu, které podstatně zhoršují letové vlastnosti modelu. Proto by přistávací plošina musela být umístěna co nejdále od zdi, čímž by vznikla velká neskladná konstrukce.

Druhá varianta podle 4.2 je založena na nosné tyči připevněné k

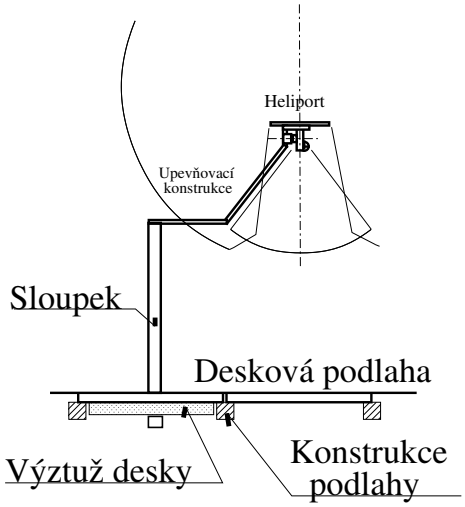


Figure 10: Ukotvení k podlaze laboratoře

zemi (podlaze). Mechanické připojení k zemi musí být dostatečně robustní, aby nedošlo k rozkmitání tyče. Protože je druhá varianta o něco výhodnější byla nakonec zvolena a úspěšně realizována. Podlaha v laboratoři se skládá z dřevěných dlaždic. K jedné z nich byl zespod přišroubován svařený ocelový kříž a do jeho středu byla vsazena nosná tyč. Zvolená a realizovaná sestava je zobrazena na 4.2.

4.5.2 obsahuje celkový přehledový pohled na vyrobený model.

4.3 Snímání polohy modelu

4.3.1 Možnosti pro určení polohy modelu

Původně se uvažovalo o možnosti bezkontaktního měření polohy vrtulníku například pomocí zpracování obrazu z několika kamer, či na základě měření doby letu radiového nebo zvukového signálu z modelu k přijímačům. Ukázalo se, že by takový způsob byl sice možný, ale vzhledem k prostředí laboratoře a určení modelu zbytečně náročný.

Model nebude vzhledem ke své velikosti nikdy schopen nést spolu s baterií i řídící systém a stát se tak autonomním létajícím prostředkem. Helikoptéra se má stát pouze laboratorním modelem systému, na němž se bude ověřovat teorie řízení, která by se dala použít pro jiný model s parametry dostatečnými pro autonomní pohyb. Protože se nepředpokládá, že by někdy vrtulník opustil trenažér, je nejschůdnější určit jeho polohu z postavení mechanických částí trenažéru.

4.3.2 Umístění snímačů polohy na trenažeru

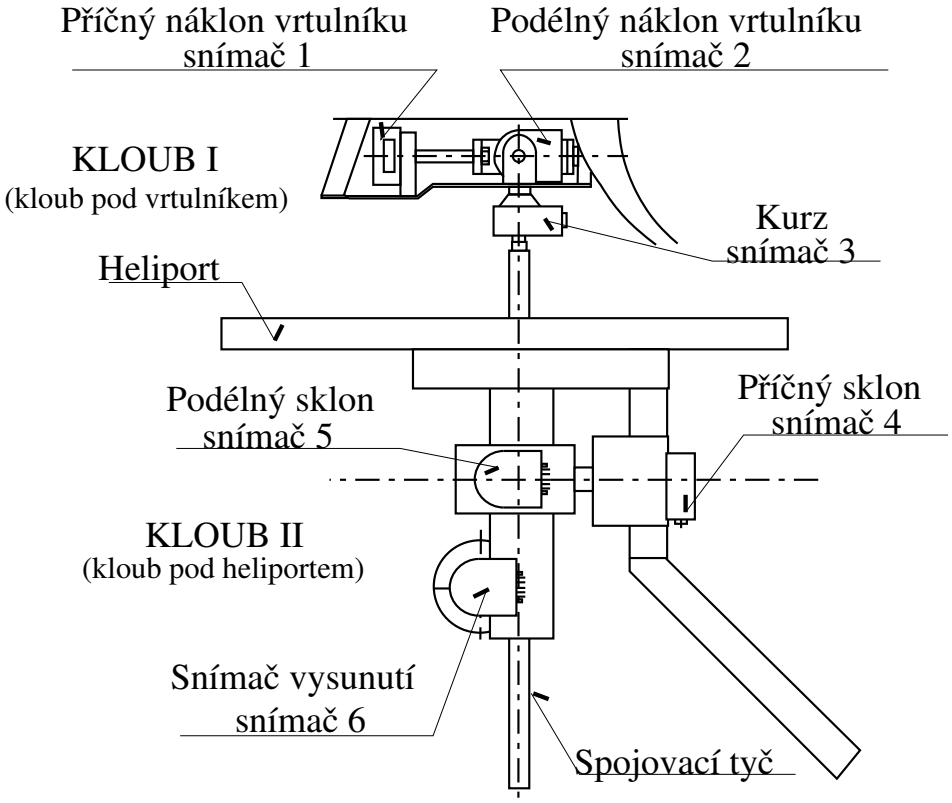


Figure 11: Rozmístění a značení inkrementálních snímačů

Trenažér byl za účelem snímání polohy opatřen rotačními inkrementálními snímači úhlu natočení.

Po dvou snímačích jsou umístěny v každém kloubu, kde snímají náklon v jednotlivých pohybových rovinách kloubů. Jeden snímač, umístěný těsně pod kloubem helikoptéry, zaznamenává otáčení kolem spojovací výsuvné tyče.

Číslo snímače	Typ	Přesnost [per/ot]	Snímaná veličina (Umístění)
1	IRC 450.1/512/BZ	512+Index	Příčný náklon vrtulníku (Kloub I)
2	IRC 450.1/512/BZ	512+Index	Podélný náklon vrtulníku (Kloub I)
3	IRC 450/540/B	540	Otočení kolem osy spojovací tyče (Kloub I)
4	IRC 450.1/512/BZ	512+Index	Příčný sklon spojovací tyče (Kloub II)
5	IRC 450.1/512/BZ	512+Index	Podélný skon spojovací tyče (Kloub II)
6	IRC 450/540/B	540	Vysunutí spojovací tyče (Kloub II)

Table 1: Umístění inkrementálních snímačů

Vysouvání tyče je pogumovanou kladkou převedeno na otáčení, které je registrováno posledním šestým inkrementálním snímačem polohy. Snímač pro snímání vysunutí spojovací tyče je součástí kloubu pod heliportem. Umístění snímačů, jejich typ a snímanou veličinu přehledně ukazuje 4.3.2 a 4.3.2.

4.3.3 Použité inkrementální snímače

Vzhledem k tomu, že model bude po celou dobu své činnosti připoután, lze jeho polohu snímat podle náklonů kloubů konstrukce. Snímací členy by měly dosáhnout dostatečného rozlišení při malé hmotnosti. Nejhorší situace nastane v případě měření náklonů tyče. Tam se nepřesnost měření úhlu násobí délkou tyče, což způsobí značnou nepřesnost určení pozice vrtulníku pro větší délku tyče.

Konstrukce je osazena nejmenšími dostupnými snímači IRC 450.1

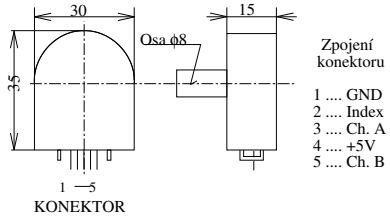


Figure 12: IRC snímač

v provedeních B a BZ, jejichž vnější vzhled ukazuje 4.3.3. Mají dvoufázový výstup v TTL logice. Průběhy výstupních signálů jsou zachyceny na 4.3.3 Přechodové hrany výstupních signálů jsou dostatečně strmé, a proto jejich další úprava není nutná. Oba typy snímačů vyžadují napájecí napětí 5V. Provedení BZ poskytuje navíc ještě nulovací pulz, který se generuje jednou za celou otáčku snímacího kotouče. Jeho využití v praxi však brání skutečnost, že při každé inicializaci musí snímač projít nulovou polohou pro

vynulování vnitřních čítačů. Do této polohy by však musel být model naveden ručně, což by značně komplikovalo obsluhu. Proto bude nulování vždy prováděno po zapnutí celého zařízení.

Rozlišení snímače typu BZ je 540 dílků/ot a typu B 512 dílků/ot. Vhodným zapojením vyhodnocovače snímačů je celkové rozlišení zčtyřnásobeno. Vysunutí tyče a otáčení trupu modelu je snímáno snímači typu B. Ostatní pozice jsou osazeny snímači v provedení BZ.

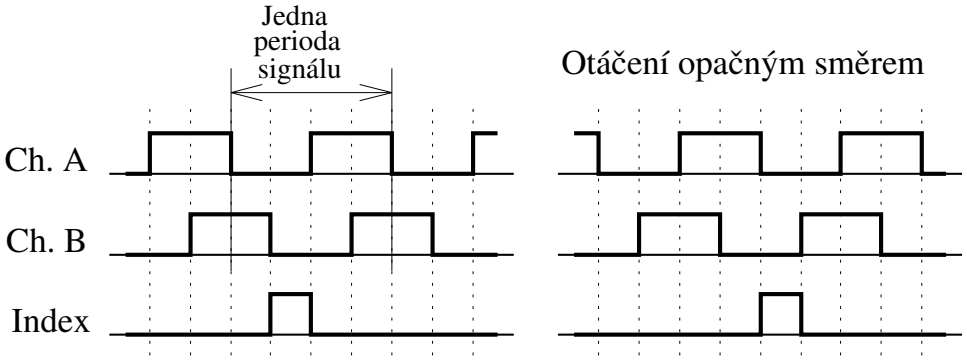


Figure 13: Průběh signálů z inkrementálního snímače

4.4 Snímač otáček nosného rotoru

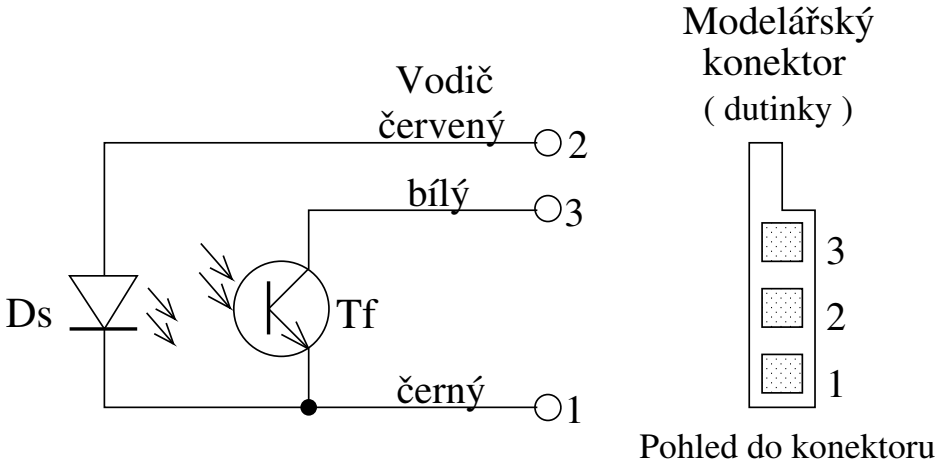


Figure 14: Zapojení snímače otáček a jeho konektoru

Snímač je realizován jednoduchou světelnou závorou jejíž clonka je umístěna na hřídeli, která pohání vyrovnávací rotor. Snímač nesnímá tedy skutečné otáčky nosného rotoru, ale otáčky hnacího motoru, s nímž je vyrovnávací rotor přes převod přímo spojen. Nosný rotor má otáčky úměrné otáčkám motoru, pokud je motor v záběru a neuplatňuje se tak volnoběžná spojka umístěná na hlavním hřídeli. V době volnoběhu nedává tedy čidlo správnou informaci o otáčkách nosného rotoru. To je pouze zanedbatelná nevýhoda, když si uvědomíme, že za letu musí motor být většinou v záběru.

Samotný snímač je umístěn uvnitř konstrukce vrtulníku a ven je vyveden kabelem zakončeným příslušným konektorem v zadní části trupu vrtulníku, pod místem připevnění ocasu. Schema a zapojení konektoru snímače otáček je na 4.4.

Snímač osadil již můj předchůdce Pavel BENEŠ a zbylo o něm jen velmi málo informací. Na 4.4 je schema zkušebního obvodu a naměřený průběh signálu. Bohužel se ukázalo, že jednotlivé pulsy jsou rozdílné co do amplitudy i délky a vzájemného odstupu. Amplituda není konstantní ani v průběhu jednoho pulsu. Neurčitost amplitudy je důležitá z hlediska zpracování a proto je na grafu označena šrafovaným polem. Pro další zpracování se bude muset signál upravovat do úrovní TTL pomocí tranzistorového zesilovače.

Na jednu otáčku nosného rotoru lze napočítat 9 až 10 pulsů.

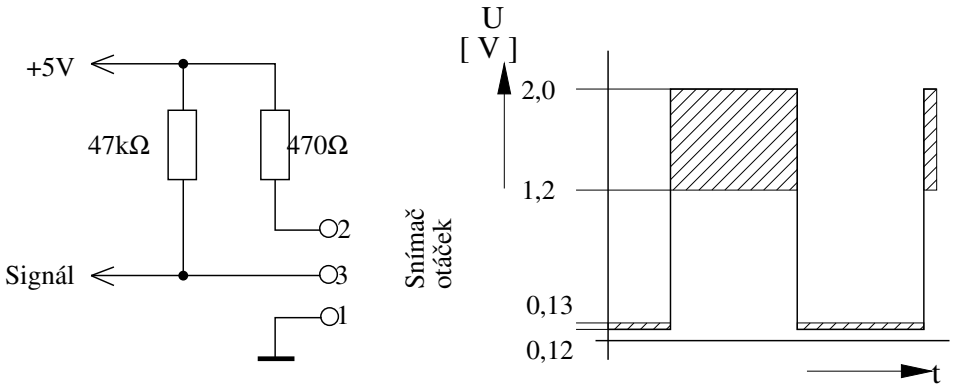


Figure 15: Měřící zapojení a průběh signálu ze snímače otáček

Nepodařilo se převod určit přesněji, ale lze se domnívat, že na jednu otáčku připadá 9,5 pulsů s přesností dvě procenta. Pro jednoduchost lze ovšem udávat: **10 pulsů na jednu otáčku hlavního rotoru**.

Tímto zjednodušením se dopouštíme chyby 5%, která je při měření otáček zanedbatelná, obvzláště když nám nepůjde o absolutní hodnotu, ale o rozdíl od hodnoty požadované. Vzhledem k různé délce jednotlivých pulsů je nutno využít pro vyhodnocení otáček metody, jejíž součástí bude integrace,

která zmírní vliv různé délky a odstupu impulsů. Například můžeme počítat počet pulsů za jistý časový okamžik. Pokud tento okamžik bude dostatečně dlouhý bude se různá délka jednotlivých impulsů projevovat pouze nepatrným šumem na výsledném údaji.

Do snímaného signálu otáček se nežádoucím způsobem přidával vf signál z vysílačky. To způsobovalo, že naměřená hodnota byla úplně nesmyslná. Zkoušeli jsme přidat kondenzátory pro odstranění rušení. Nakonec se jako nejlepší ukázala varianta s majoritním digitálním filtrem před vyhodnocovací logikou.

4.5 Napájení modelu

Vrtrulníky mají vysokou energetickou náročnost a ani vyvíjený model není výjimkou. Při plném záběru motoru je odebírán ze zdroje proud přesahující 20A. Vzniká problém odkud a jakým způsobem dodat do motoru potřebnou energii.

4.5.1 Napájecí baterie

Modeláři při létání používají speciální akumulátory připevněné ke spodní části vrtulníku. Ty jsou schopny krátkodobě pokrýt energetickou spotřebu modelu. V praxi se jedná o dobu kolem 5 min. Po vybití je nutno akumulátor znovu nabít. Doba nabíjení není nejkratší a ani počet nabíjecích cyklů není velký. Protože se jedná o speciální typ akumulátorů, jsou poměrně drahé. Jejich hmotnost také nelze zanedbat. Při umístění modelu k trenažéru je již velká část užitné hmotnosti využita pro zvedání vodící tyče a vrchního Cardanova kloubu se snímači polohy. S připevněným akumulátorem by se již model nemusel vůbec odlepit od země.

Původně byl model napájen z NiCd akumulátoru firmy SANYO se jmenovitým napětím 8,4V a kapacitou 1850mAh, který však může poskytnout energii pouze na několik minut letu. Vzhledem k hmotnosti a náročnosti na údržbu bylo nutno nahradit akumulátor jiným zdrojem a systémem přívodů umožňující omezený pohyb vrtulníku.

4.5.2 Alternativní přívod energie

Protože již byla varianta se spalovacím motorem zamítnuta, zbývá přivést elektrickou energii pomocí kabelů ze země. Kabely je vhodné přivést takovým způsobem, aby byla co nejméně ovlivněna dynamika modelu. Nejvýhodnější nejspíše bude využití duté vodící tyče pro uložení kabelů. Celá tyč by mohla posloužit jako část jednoho vodiče, protože je vyrobena z vodivého materiálu. Tímto způsobem by se podařilo mírně snížit celkovou vzletovou hmotnost.

Protažení kabelů vodící tyčí si však vynutí mechanickou úpravu horního Cardanova kloubu. Při propojení modelu se zdrojem pomocí drátových vodičů vzniká možnost jejich překroucení opakovaným otáčením modelu. Bude nutno omezit úhel natočení modelu. Při omezení povolených

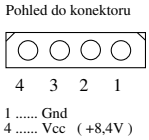


Figure 16: Zapojení baterie

kurzů nepůjde například hladce provést krouživý pohyb modelu a dojde k omezení ostatních letových režimů zmenšením stavového prostoru.

Proto jsem zvolil přivedení napětí přes pohyblivé kartáče. Ani tato volba není absolutně nejlepší, ale představuje vhodný kompromis. Od kartáčů se může šířit elektromagnetické rušení. Kartáče budou pravděpodobně silně namáhány, což mírně sníží jejich životnost (jedná se o desítky až stovky provozních hodin). Protože budou snadno vyměnitelné, nepředstavuje tato skutečnost v případě experimentálního modelu vážnou překážku.

Poslední provedené pokusy s kartáčemi ukázaly **neschůdnost této cesty** při použití běžně dostupných materiálů. Bude potřeba vytvořit jiný systém přívodu elektrické energie. Běžné kartáče jsou určeny pouze pro otáčející se stroje a při zastavení přestane být obrušován jejich povrch. To způsobí značný nárůst odporu a tím i ztrátového výkonu. V praxi se jedná až o 3V na 10A.

4.5.3 Způsob propojení napájecích kabelů

Deska interface XRI je napájena přes pojistky a vypínač ze zdroje počítače 5V a 12V. Napájení je vedeno datovými vodiči z karty PCL812. Přes desku XRI je napájena i deska VT, která je také součástí interface. Přes interface je napájen i vysílač RCTx. Deska VT2 a datový přijímač Rx umístěné pod heliportem jsou spojeny s deskou XRI datovými vodiči. Spolu s datovými vodiči z VT2 a Rx jde z interface i napájení těchto obvodů. Napájení obvodů pod heliportem je 12V. Na desce VT 2 je pak realizován zdroj 5V/1A zajišťující napájení obvodu XILINX a inkrementálních snímačů na tomto kloubu.

Vedení výkonového okruhu jde ze zdroje přímo do vrtulníku na řídící spínač hnacího motoru. Ze stejného zdroje je napájena i deska XV1 se snímači a RCRx spolu se servy. Na desce XV1 je opět realizován zdroj 5V/1A pro napájení obvodu XILINX a snímačů na vrtulníku. RCRx vyžaduje také napájení 5V, ale serva v činnosti odebírají až 1,2A. Ve vrtulníku je proto umístěn zdroj ZD1 5V/2A, který napájí RCRx a přes něj i serva. Jednotlivé komponenty jsou značeny podle 6.1.

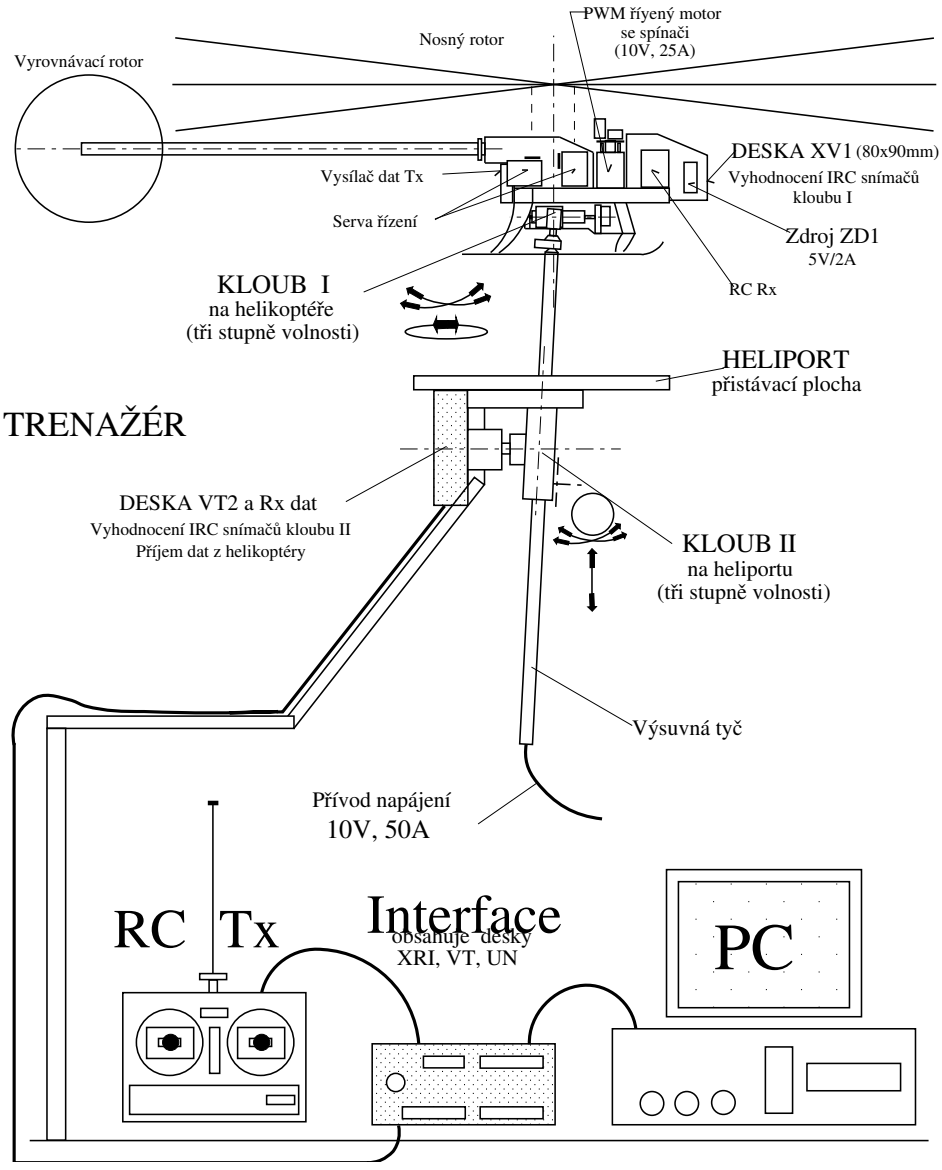


Figure 17: Celkový pohled na sestavu trenažéru

5 Bezdrátové řídící systémy

5.1 Komunikace v průmyslových systémech

S rostoucí automatizací a computerizací celého průmyslového procesu rostou nároky na komunikaci mezi jednotlivými články řízení a úrovněmi rozhodování.

Na 5.1 je znázorněna hierarchie komunikačního řetězce v

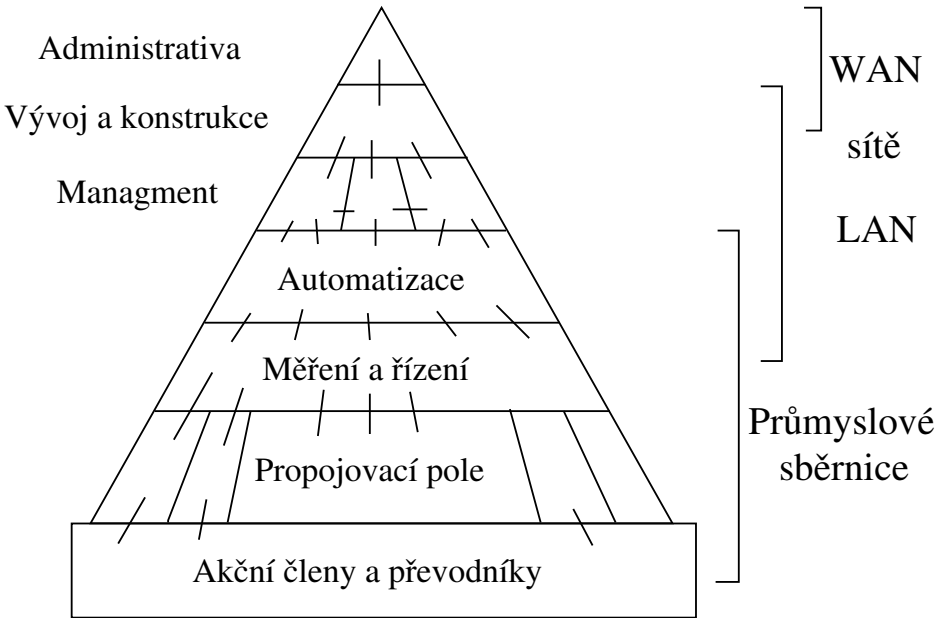


Figure 18: Hierarchie komunikačního řetězce s naznačeným druhem komunikačního media

průmyslovém procesu. Tvar pyramidy má za úkol znázornit snižující se počet účastníků a informačních spojů společně se stoupající kapacitou jednotlivých spojů. Nejnižší vrstvu tvoří senzory a akční členy propojené propojovacím polem s vrstvou měření a řízení, kde se požaduje velká rychlost reakce a bezpečnost provozu. Naproti tomu je v administrativě a vývoji požadován přenos značných objemů dat mezi méně účastníky bez nároku na reakci v reálném čase.

S požadavky na různých úrovních řízení se mění i komunikační médium tak, aby požadavkům odpovídalo. V oblasti administrativy a vývoje na vrcholu pyramidy se používají sítě LAN a WAN bez možnosti deterministického přístupu, ale s velkou přenosovou kapacitou na velké vzdálenosti. Tam kde musíme vyhovět požadavkům řízení v reálném čase,

je nutno použít průmyslové sběrnice určené pro sběr dat a řízení v reálném čase.

Na všech úrovních řízení je možno použít bezdrátový přenos dat. Nevýhodou bezdrátového přenosu dat přes okolní prostředí je jeho zatížení rušením z tohoto prostředí. Přenosové spektrum je obvykle omezené a pro přenos je nutno signál upravovat poměrně drahým zařízením. Toto zařízení předurčuje dnes bezdrátové spoje do oblastí, kde se přenáší značný objem dat přes malý počet spojů a cena zařízení se tak vyrovná ceně galvanického spoje. Takovým místem jsou například sítě LAN a WAN, kde nám družicové spoje umožňují přenos dat v rámci celého světa bez nutnosti vynakládat velké prostředky na zřízení a údržbu galvanických spojů. Výhodné může být však i použití pojítka na několik metrů, pokud je obtížné kabel položit (např. přes silnici).

Na úrovni snímání a řízení, kde je tok dat pomalejší, nároky na spolehlivost větší a spojů více, je snaha využívat bezdrátový přenos jen tehdy, když je to naprosto nezbytné. Pokud se bezdrátového přenosu užije, bývá to obvykle v rámci na objednávku vytvářeného řídícího systému, kdy se bezdrátový spoj maximálně musí přizpůsobit podmínkám a možnostem aplikace.

V rámci této práce budu považovat za bezdrátový spoj, jen takový spoj, který nevyžaduje nejen galvanické propojení vysílače a přijímače zpráv, ale který nevyžaduje žádné mechanické propojení. Jedná se tedy o spoj umožňující do jisté míry volný pohyb přijímače a vysílače, kterého je užito v praktické části diplomové práce.

5.2 Přenosová cesta a přenosový kanál

Pojmem **přenosová cesta** rozumíme fyzikální prostředí, v němž se uskutečňuje přenos signálu pomocí měronosné veličiny. **Měronosná veličina** je materiálním nosičem přenášené zprávy v prostoru a čase, tedy libovolná fyzikální veličina jejíž informační parametr se v čase mění souhlasně s přenášenou zprávou. Pro lepší využití můžeme celou časovou a frekvenční oblast přenosové cesty rozdělit do několika částí. V každém takovém časovém či frekvenčním segmentu lze pak přenášet signály a zprávy současně a nezávisle na signálech v ostatních segmentech. Části přenosové cesty nazýváme **přenosové kanály**.

5.3 Přenosové cesty

Přenosové cesty můžeme rozdělit podle použitého média, frekvenčního pásma a charakteru prostředí.

5.3.1 Akustická přenosová cesta

K přenosu informace se užívá mechanického vlnění okolního prostředí. Podstatou tohoto vlnění jsou elastické kmity, které se šíří rychlostí

závislou na mechanických vlastostech média. Ve vzduchu je rychlaost šíření asi 330 m/s.

Podle frekvence vlnění dělíme akustickou přenosovou cestu na:

infrazvukové pásmo	pod 16Hz
slyšitelné pásmo	16Hz až 20kHz
ultrazvukové pásmo	nad 20kHz

Pro přenos dat v automatizační technice se používá pouze ultrazvukového pásma. Infrazvuk není pravděpodobně příliš zdravý pro člověka a může poškozovat i mechanické konstrukce. Slyšitelné pásmo je vyhrazeno pro komunikaci mezi lidmi a proto se pro přenos dat neužívá.

5.3.2 Elektromagnetická přenosová cesta

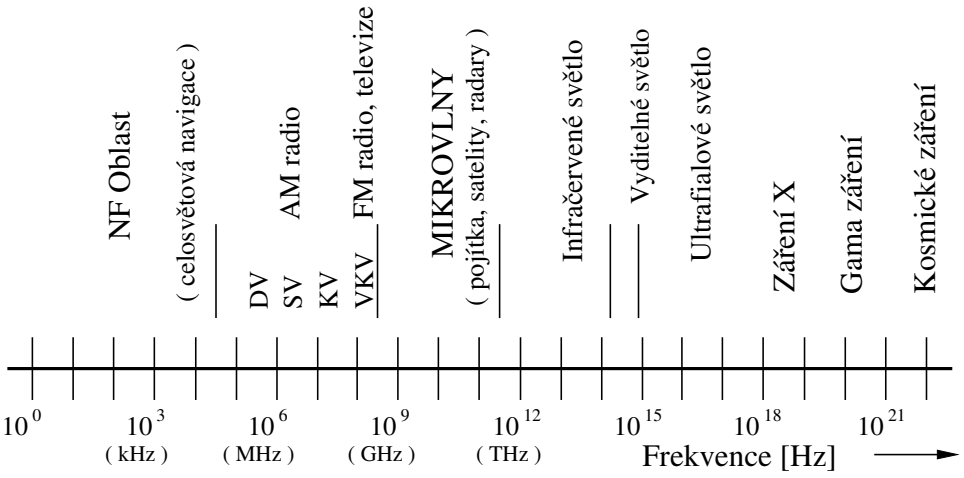


Figure 19: Elekromagnetické frekvenční spektrum

Pod tímto označením je myšlena přenosová cesta využívající k přenosu elektromagnetické vlny. Vyzářená energie se šíří prostorem v podobě elektromagnetického pole od vysílací antény. Šíření elektromagnetických vln je značně závislé na parametrech prostředí, které se v závislosti na čase, vzdálenosti a délce vlny značně mění. Elektromagnetické cesty pro bezdrátový přenos se dělí podle použitého frkvenčního spektra jak je znázorněno na 5.3.2.

V automatizačních systémech se užívá telemetrických systémů pro sběr dat obvykle v pásmu VKV od 30 do 300MHz. S rozvojem datové komunikace a elektroniky vůbec se začíná pro přenos dat užívat i vyšších pásem až do desítek GHz.

5.3.3 Optická přenosová cesta

Jedná se vlastně o elektromagnetický přenos s kmitočtem vlny odpovídajícím světlu. V elektromagnetickém spektru světlu odpovídají kmitočty přibližně od 10^{12}Hz do 10^{16}Hz. Viditelné světlo je přitom jen úzké pásmo s kmitočtem asi 500THz. Nižší kmitočet než viditelné světlo má světlo infračervené. Druhou stranu světelného spektra tvoří ultrafialové světlo.

Pro přenos dat lze využít modulovaného zdroje světla. Jako přijímač pak může sloužit jakýkoliv fotocitlivý prvek. Pouze je nutno dbát, aby se kmitočet vysílače shodoval s frekvenčním pásmem citlivosti přijímače. Takový přenos lze použít nejen ve viditelném spektru, ale i v infračervené či ultrafialové oblasti. Nevýhodou je nutná přímá viditelnost a vysoká hladina přírodního rušení . Naproti tomu výhodou je možnost použití velké šířky přenosového pásma a zhoršená možnost odposlechu směrového spoje.

Ukázalo se také, že světelné záření lze s úspěchem přenášet pomocí optických vláken a to i na velké vzdálenosti. Tato oblast prodělala v minulých letech rychlý vývoj. Optické kabely se dnes považují za moderní prostředek pro spojení v telefonii a datových sítích. Rychlost a nízká cena umožňuje nahrazovat dnes galvanické spoje světlovody. Protože světlovod představuje mechanické spojení mezi vysílačem a přijímačem, které brání v jejich volném pohybu, nebudu se touto moderní technikou spojů dále zabývat.

5.3.4 Metalická přenosová cesta

Jedná se o přímé propojení vysílače a přijímače vodičem. Vodivého propojení se používá nejčastěji pro nejrůznější vzdálenosti a kmitočty přenášeného signálu. Zde je metalická přenosová cesta uvedena pouze pro úplnost výčtu, protože se nejedná o bezdrátovou přenosovou cestu.

5.4 Přenos v základním a přeloženém pásmu

Většina bezdrátových přenosových cest nedovoluje přenášet signál v **základním pásmu**, tedy v pásmu v němž je generován zdrojem signálu. Důvodem je požadavek na využití více přenosových kanálů a případné technické obtíže s přenosem v některých frekvenčních oblastech. Pokud není možno využít k přenosu základního pásma, musíme si pomoci **modulací**. Modulovaný signál je výsledkem procesu modulace, který probíhá v **modulátoru**. Do modulátoru vstupuje periodický **nosný signál** jehož některé parametry (frekvence, amplituda ...) jsou v modulátoru řízeny **modulačním signálem**, který nese informaci. Výsledkem modulace je transformace frekvenčního pásma signálu a po potlačení základního frekvenčního pásma vznikne signál v **přeloženém pásmu**.

K přenosu se může použít i několikanásobné modulace, kdy je signál zpracováván několika modulátory za sebou při použití různých nosných signálů. Modulovaný signál se tak vždy stává modulačním pro následující

modulátor. Takového přenosu se užívá obvzláště na velmi vysokých kmitočtech.

5.5 Modulace pro přenos analogových signálů

Analogovým signálem rozumíme signál spojitý v čase i amplitudě. Tímto signálem je modulován harmonický nosný signál, nebo impulsní nosný signál. Podle toho se rozlišuje analogová a impulsní modulace. Samostatnou kapitolu tvoří modulace zajišťující číslicové kódování analogového signálu a jeho digitální přenos.

5.5.1 Amplitudová modulace

Při této modulaci se provádí v modulátoru prostý součin nosného a modulačního signálu. Protože tato modulace tedy pouze transformuje spektrum signálu do okolí nosného kmitočtu říká se jí lineární modulace. Představu o frekvenčním spektru amplitudově modulovaného signálu si můžeme udělat z 5.5.1. Princip amplitudové modulace lze vyjádřit vztahem:

$$a_A(t) \;=\; [A + \Delta A\frac{x(t)}{X_m}]\,\cos(\omega_0 t) \;=\; [1 + m_A\frac{x(t)}{X_m}]\,A\,\cos(\omega_0 t) \qquad (5)$$

kde

$a_A(t)$	je	amplitudově modulovaný signál
a(t)		nosný signál, pro AM platí $a(t) = A\cos(\omega_0 t+\varphi_0)$
ω_0		kmitočet nosného signálu
A		amplituda nosného signálu
x(t)		modulační signál (signál, který se má přenášet)
X_m		maximální hodnota modulačního signálu
$x(t)/X_m$		normovaný modulační signál
ΔA		maximální změna amplitudy nosného signálu
m_A		činitel amplitudové modulace, $m_A = \Delta A/A$, $m_A>1$ se nepoužívá, protože pak při demodulaci dochází ke zkreslení.

Informaci nesou pouze postranní pásma a to obě stejnou. Pro zlepšení výkonnové bilance se používá modulace DSB, při níž jsou vysílána plně postranní pásma a případně značně potlačená nosná. Za cenu složitější demodulace lze potlačit i druhé postranní pásmo, čímž vznikne modulace SSB. Je jedno které postranní pásmo potlačíme, ale při demodulaci je nutno na to brát zřetel. Vysíláno je pak jen jedno postranní pásmo čímž se opět zlepší výkonová bilance.

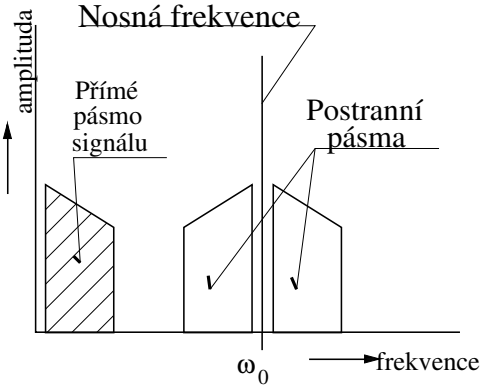


Figure 20: Frekvenční pásmo AM

5.5.2 Frekvenční modulace

Frekvenční modulace patří mezi úhlové nelineární modulace. Frekvence modulovaného signálu odpovídá okamžité hodnotě amplitudy modulačního signálu. Při této modulaci dochází k nelineární transformaci frekvenčního spektra a vzniku teoreticky neomezeného spektra. Z praktického hlediska lze zanedbat složky spektra, které mají v součtu menší výkon jak 1% z výkonu nemodulované nosné. FM signál a jeho okamžitá frekvence jsou dány následujícími vztahy:

$$\omega(t) \;=\; \omega_0 + \Delta\omega_m \, g(t) \tag{6}$$

$$a_F\,(t) \;=\; A\,\cos[\omega_0\, t + \Delta\omega_m \int g(t)\, dt + \varphi_0\,] \tag{7}$$

kde ω_0 je úhlová frekvence nosného signálu
A amplituda nosného signálu
$\Delta\omega_m$ frekvenční zdvih modulace, změna frekvence odpovídající maximální hodnotě signálu g(t)
g(t) normovaný modulační signál, g(t)=x(t)/X_m
a_F(t) frekvenčně modulovaný signál

Pokud budeme modulovat harmonickým signálem g(t) = cos (Ωt + Φ), dostaneme rovnici pro okamžitou hodnotu FM signálu:

$$a_F\,(t) \;=\; A\,\cos\,[\omega_0\, t \;+\; m_F\;\sin(\Omega t) \;+\; \varphi_0 \;+\; \Phi] \tag{8}$$

kde m_F je činitel frekvenční modulace, $m_F = \Delta\omega_m/\Omega$.

Podle velikosti činitele m_F rozlišujeme dva druhy frekvenční modulace. Pokud je m_F malé (m_F<1) hovoříme o **úzkopásmové frekvenční modulaci**. Její frekvenční spektrum se svou šířkou blíží frekvenčnímu spektru amplitudové modulace. Pokud je naopak činitel m_F velký (m_F>1) jedná

se o **širokopásmovou frekvenční modulaci**. S rostoucím činitelem m_F roste rychle i šířka potřebného frekvenčního pásma. Z toho vyplývá také to, že pro vysoké kmitočty přenášeného signálu se tvoří širší spektrum a tím se zvětšuje náchylnost k rušení. FM radiové stanice na VKV využívají modulaci s činitelem m_F=5 (při Ω=15kHz je tak $\Delta\omega_m$=75kHz).

5.5.3 Fázová modulace

Fázová modulace lze charakterizovat tím, že okamžitá odchylka fázového úhlu modulovaného signálu je lineárně závislá na okamžité hodnotě modulačního signálu. Fázová modulace je tedy dána vztahem:

$$\varphi(t) \;=\; \omega_0 t + \Delta\varphi_m\, g(t) + \varphi_0 \;=\; \omega_0 t + \; m_p g(t) + \varphi_0 \tag{9}$$

$$a_P\,(t) \;=\; A\,\cos[\omega_0\, t + \; m_p\, g(t) + \varphi_0\,] \tag{10}$$

kde ω_0 je úhlová frekvence nosného signálu
φ_0 fáze nosného signálu
A amplituda nosného signálu
$\Delta\varphi_m$ maximální změna frekvence odpovídající maximální hodnotě signálu g(t)
m_P činitel fázové modulace, $m_P = \Delta\varphi_m$
g(t) normovaný modulační signál, g(t)=x(t)/X_m
a_P(t) frekvenčně modulovaný signál

Fázová modulace patří k úhlovým nelineárním modulacím podobně jako modulace frekvenční. Od frekvenční modulace se liší především činitelem modulace nezávislým na kmitočtu modulačního signálu. Maximální fázový úhel modulace je omezen z důvodů jednoznačnosti na $\Delta\varphi_m$ = +-180°. Toto omezení způsobí, že fregvenční pásmo je užší než v případě frekvenční modulace. Pro $m_P << 1$ se spektrum fázové modulace blíží spektru modulace amplitudové, podobně jako tomu je pro frekvenční modulaci.

Fázová a frekvenční modulace jsu si velice podobné, ale přesto jsou to dvě různé modulace. Rovnice 5.5.3 popisuje jak lze pomocí fázové modulace získat frekvenčně modulovaný signál. Opačný postup je popsán rovnicí 5.5.3.

$$a_F\,(t) \;=\; A\,\cos[\omega_0\, t + \; m_p \int g(t)\, dt] \tag{11}$$

$$a_P\,(t) \;=\; A\,\cos[\omega_0\, t \;+\; m_F\, g'(t)\,] \tag{12}$$

5.5.4 Impulsové analogové modulace

Tato modulace patří k moderním metodám. Její podstatou je převod analogového signálu na signál diskrétní představovaný sledem impulsů. Parametry posloupnosti impulsů (amplituda, frekvence, fáze, šířka

) se mění v závislosti na modulačním signálu. Cílem impulsové modulace je snížit vliv rušivých signálů na přenos informace. Přenosový kanál obvykle nejde prakticky využít k přenosu stejnosměrné složky a signálů s nízkým kmitočtem z důvodu rušení způsobeného nestabylitou zařízení. Impulsová modulace zajistí přeložení frekvenčního spektra směrem k vyšším kmitočtům. Nevýhodou je pouze rozšíření kmitočtového pásma.

Protože musíme dodržet vzorkovací teorém, je nutno aby kmitočet impulsního nosného signálu byl nejméně dvojnásobkem mezního přenášeného kmitočtu f_m. Šířka pásm lze orientačně odhadnout z Furierova rozvoje impulsního signálu. Pro přenos 90% výkonu je nutná šířka pásma $\Delta F=1/\tau$ a pro přenos 95% je zapotřebí mít kanál široký $\Delta F=2/\tau$, kde τ je šířka impulsu. Tento hrubý odhad platí pro všechny typy impulsních modulací.

Impulsové modulace se rozdělují podle ovlivňovaného parametru impulsů :

Amplitudová impulsová modulace - amplituda impulsů je závislá na okamžité hodnotě modulačního signálu. Je možné užít způsobu, kdy modulovaný signál sleduje po dobu impulsu hodnotu modulačního signálu, takže modulovaný impuls nemá konstantní velikost. Druhou možností je modulace při níž mají impulsy po celou dobu trvání konstantní velikost odpovídající hodnotě modulačního signálu v okamžiku jejich počátků. Oba dva způsoby patří k amplitudové modulaci a liší se jen nepatrně.

Fázová impulsová modulace - při této modulaci odpovídá fáze impulsů, vyjádřená vzdáleností impulsu oproti synchronizační značce, okamžité hodnotě přenášeného signálu.

Frekvenční impulsová modulace - podle modulačního signálu se mění frekvence impulsů, představovaná vzdáleností mezi nimi. Délka impulsu přitom může být konstantní, nebo se měnit nepřímoúměrně s frekvencí a zachovávat tak poměr vzdálenosti pulsů a jejich délky (střídu).

Pulsně šířková modulace (PWM) - při PWM se zachovává kmitočet impulsů a mění se pouze jejich šířka a tím i střída (poměr impulsu k periodě). Pokud se šířka impulsu mění pouze jedním směrem oproti určujícímu bodu mluvíme o takzvané jedostranné PWM. Při oboustrané šířkové modulaci se mění puls symetricky oproti určujícímu bodu.

Impulsové modulace se pro její vlastnosti často využívá ve spojení s některou základní modulací. Takové modulaci, kdy je modulovaný signál použit pro další modulaci se říká výcenásobná modulace.

5.5.5 Pulsně kódová modulace (PCM)

Tato modulace patří mezi impulsní modulace, ale není vlastně modulací analogovou. Analogový signál je vzorkován, kvantován a převáděn na digitální údaj. Údaj v číslicové formě je pak vyslán digitálním přenosovým kanálem. V místě příjmu lze z dodaných vzorků opět rekonstruovat původní analogový signál s omezeními danými vzorkováním a kvantováním. Digitální údaj může vyjadřovat nejen absolutní hodnotu přenášeného signálu, ale i jen

jeho změnu, čímž docílíme zmenšení objemu přenášených dat za cenu zůžení přenášeného frekvenčního pásma původního analogového signálu.

5.6 Přenos digitálních signálů

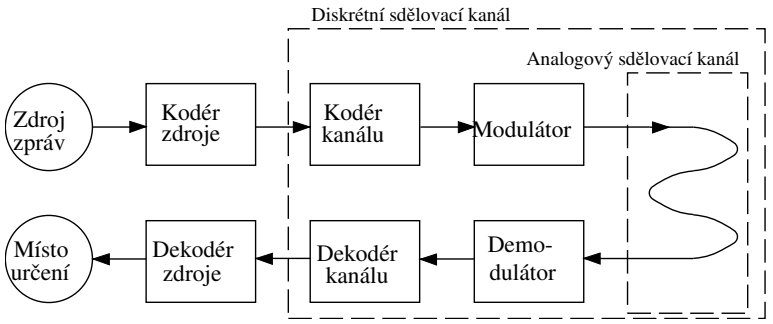


Figure 21: Blokové schéma digitálního komunikačního systému

Číslicové signály považujeme za signály diskrétní v amplitudě i čase. Jsou charakterizovány posloupností prvků z konečné množiny prvků. Digitální signály jsou obvykle dvoustavové (binární), třístavové a vyjímečně vícestavové. Na 5.6. je znázorněn digitální komunikační systém. Zprávy jsou kódovány do digitální formy, poté jsou kodérem vysílače upraveny pro přenos. Signál je poté modulován a přenášen přenosovým kanálem. Na přijímací straně musí být provedena demodulace a dekódování.

5.7 Kódování

Kódováním lze obecně nazvat přiřazení stavu prvků jedné množiny (zprávy) stav prvků druhé množiny. Z hlediska přenosu zpráv nám kódování definuje jak elektrickou reprezentaci, tak přizpůsobení signálu přenosovému kanálu, zlepšení synchronizačních vlastností, ale na vyšší úrovni i zabezpečení proti chybám a zneužití dat. Protože různé kódy mají různý účel, je obvyklé použití vícenásobného kódování, kdy je zpráva naříklad nejdříve zabezpečena proti chybám a poté upravena pro přenos. Na straně přijímače je pak opačným postupem dekódována. Pro běžné použití je nutno trvat na jednoznačnosti kódu a na možnosti jeho snadného a jednoznačného dekódování.

Kódování je rozsáhlý obor a zde se zmíním jen o základních pojmech v oblasti reprezentace signálu a jeho přípravy pro přenos. Právě na této úrovni se musí signál přizpůsobit bezdrátovému přenosovému kanálu.

5.7.1 Kódová reprezentace zpráv

Pokud definujeme L jako počet prvků zdroje zpráv pak můžeme říci, že pro zachování jednoznačnosti musí mít kód nejméně L prvků. Prvek kódu je představován slovem o délce n prvků. Každý prvek slova je vybírán z množiny možných hodnot, tedy abecedy. Počet různých slov takového kódu lze vypočítat vztahem:

$$L = m^n \tag{13}$$

kde m je počet prvků v příslušné abecedě. Dnes se obvykle používá binární kód kde m=2 a potřebného počtu různých slov se dosahuje délkou kódového slova.

Nejjednodušší je případ kdy slovo má pouze jeden prvek. Takové kódy se označuje jako **jednoprvkové kódy** a používají se v řízení a snímání s málo stavy . V případě, že má kódové slovo více prvků mluvíme o **víceprvkovém kódu**. S rozvojem počítačů se přešlo na víceprvkové binární kódy, které odpovídají přímo reprezentaci dat v počítači.

Obvykle z důvodů komprese se užívá **nerovnoměrného kódu** jehož jednotlivá slova mají různou délku. Jejich nevýhodou je složitější dekódování oproti **rovnoměrnému kódu**, který má stejnou délku všech kódových slov. Lze říci, že na úrovni přenosového kanálu se vždy užívá rovnoměrného kódu a komprese se nechává na vyšší úrovni kódování zpráv.

5.7.2 Elektrická reprezentace

Pro přenos je důležitá elektrická reprezentace signálu a to především signálů binárních. Podle typu elektrického signálu, kterým kódujeme binární signál mluvíme o:

Unipolárním signálu - signál pouze jedné polarity, označované také jako signály RZ (Return to Zero). Ke kódování užívá nulové úrovně a úrovní jedné polarity. Nevýhodou takového signálu je velká stejnosměrná složka.

Polární signál - ke kódování se užívá signálu s opačnou polaritou takže se výrazně potlačí stejnosměrná složka. Nevýhodou je nutnost vložení synchronizační informace.

Bipolární signál - kromě signálu opačné polarity je součástí kódu i nulová úroveň, proto se tento kód označuje jako pseudo ternární. Větší nároky na přenosový kanál se kompenzují možností využít třetí úroveň signálu pro přenos synchronizační informace.

Z hlediska bezdrátového přenosového kanálu, který obvykle má problém s přenosem stejnosměrné složky, je unipolární a polární signál prakticky identický, protože stejnosměrná složka bude při přenosu potlačena.

5.7.3 Kódy pro přenos v lokálních sítích

Lokální sítě obvykle používají jako pracovní médium metalický spoj s jedním respektive dvěma vodiči. Tak je k dispozici pouze jeden přenosový

kanál, který musí nést nejen data, ale i synchronizační informaci. Taková konfigurace se podobá přenosu bezdrátovým spojem, a proto mohou být využívány stejné kódy, pro které mluví i dostupnost součástkové základny.

Na 5.7.3 jsou druhy kódů užívaných v lokálních sítích. Tyto kódy

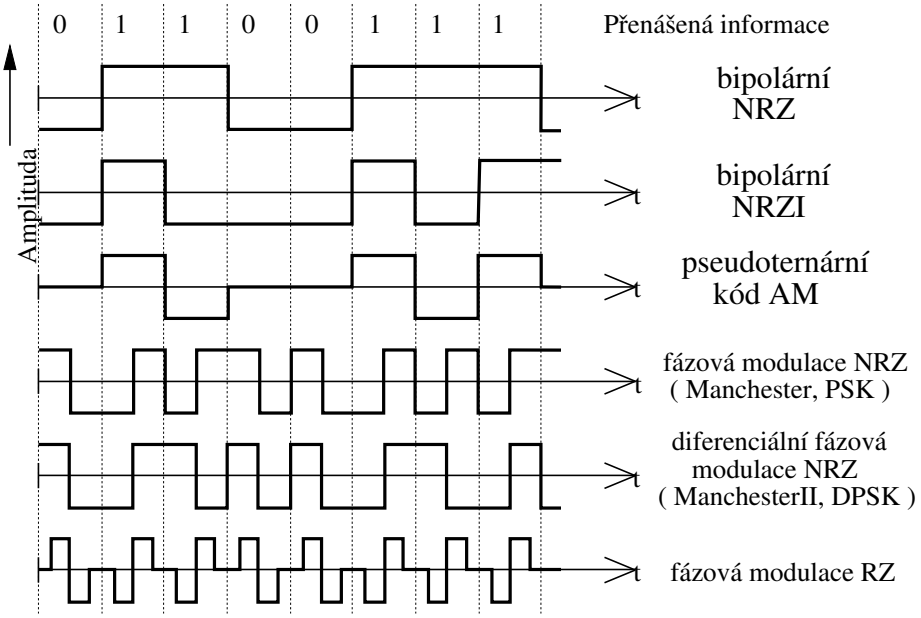


Figure 22: Kódy (modulace) pro přenos v lokálních sítích

lze použít nejen pro přenos v základním, ale i přeloženém pásmu. K tomu se používá většinou kmitočtová nebo fázová modulace.

5.7.4 Vyšší úroveň kódování

Na mysli mám samotné kódování zprávy a kódy pro ochranu dat proti poruchám i zneužití. Do této kategorie se řadí i protokol sítě na úrovni zpráv a jejich implementace. Podle síťového protokolu OSI se tedy jedná linkovou a vyšší vrstvy. Na této úrovni je jedno jaké přenosové médium použijeme. Pro bezdrátový přenos tak není nutno užívat zvláštních standardů odlišných od síťových. Jsou vyjímky, kdy je nutno z důvodů časového zpoždění zachycovat celé zprávy a předávat je dál na úrovni směrovače či brány tak, aby síť nemusela počítat se spožděním daným přenosovým kanálem.

Pro kódování zpráv se obvykle užívá binárního kódu, či Grayův kód, které se liší pouze přiřazením znaků k odpovídajícím číslům (pokud budeme

kódovat číslo). Jedná-li se o přenos jedním přenosovým kanálem musíme vysílat jednotlivé prvky kódu za sebou, tedy sériově. Nepsanou úmluvou je, že jako první se vysílá vždy nejnižší významový prvek. Na úrovni protokolu je nutno označit začátky slov kódu a jejich skupin (zpráv). Tato část kódování musí být vždy přesně definována protokolem přenosu a dnes se prosazuje i jistá standartizace.

5.8 Modulace pro přenos datových signálů

Máme-li již zakódovanou zprávu, musíme se postarat o její přenos kanálem. Při použití bezdrátového spoje není většinou možné přenášet zprávu v přímém pásmu a je nutno použít modulace. Modulace pro přenos číslicového signálu jsou odvozeny od modulací analogových. To bylo důvodem proč jsem se jimi v kapitole 5.5 zabýval. Dnes se pro bezdrátový přenos většinou používá digitální modulace, která je odolnější proti rušení a dnes umožněna rozvojem elektroniky.

5.8.1 Amplitudová modulace ASK (Amplitude-shift keying)

Harmoniský nosný signál je amplitudově modulován datovým signálem, který nabývá diskrétních hodnothodnot (pro binární signál obvykle 0 a 1). V nejjednodušším případě pak logické jedničce odpovídá například plný výkon vysílaného signálu a logické nule naopak odpovídá nepracující vysílač. ASK s hloubkou modulace 100% je sice jednoduchá metoda modulace, ale náchylná k poruchám. Zlepšení odolnosti se dociluje použitím menší hloubky modulace, takže nedochází k přerušení vysílání. I přes malou odolnost proti rušení je ASK pro svojí jednoduchost často užívána.

Frekvenční pásmo ASK odpovídá frekvenčnímu pásmu impulsu symetricky rozloženému okolo kmitočtu nosného signálu. Pro přenos je tak potřeba přibližně $\Delta F=2/\tau$, kde τ je délka nejkratšího přenášeného impulsu (modulační rychlost).

5.8.2 Frekvenční modulace FSK (Frequency-shift keying)

Podstatou modulace FSK je přiřazení jednotlivých frekvencí diskrétním hodnotám modulačního signálu. V případě nejčastěji používaného binárního signálu se přiřazují pouze dvě frekvence, jedna logické nule a druhá logické jedničce. Frekvenční spektrum se podobá ASK, ale rozloženému kolem obou použitých kmitočtů. Velikost frekvenčního pásma nutného pro přenos vychází opět z delky nejkratšího přenášeného pulsu τ (modulační rychlosti) a je dána vztahem $\Delta F=4/\tau$.

Modulace FSK je odolnější proti rušení než modulace ASK a je také dnes rozšířenější. FSK používají na příklad modemy pro přenos dat po telefonních linkách. Dnes je nezanedbatelnou výhodou této modulace dostupnost technických prostředků pro její realizaci.

5.8.3 Fázová modulace PSK (Phase-shift keying)

Jedná se o moderní modulaci při níž se jednotlivým stavům diskrétního modulačního signálu přiřazuje různá fáze signálu modulovaného. Kromě binární modulace BPSK, kde jsou dvěma stavům přiřazeny dvě různé fáze, se pro zvětšení přenosové rychlosti, při zachování modulační rychlosti, užívá kódování pomocí více stavů. Běžně je tedy užívána QPSK (Quadrature...) se čtyřmi odlišnými stavy fáze a 8Φ-PSK, která využívá pro přenos 8 rozlišitelných stavů fáze. Nelze ovšem zvyšovat počet fázových stavů do nekonečna, protože se tím zvyšují nároky na detektor a snižuje odolnost proti rušení.

5.8.4 Kvadraturní amplitudová modulace QAM (Quadrature amplitude modulation)

QAM modulace je vlastně spojením amplitudové a fázové modulace. K přenosu digitálního signálu se užívá více stavů (například 16 stavů), které jsou dány nejen fází signálu ale i jeho amplitudou. Tato modulace představuje opět možnost, jak zvýšit přenosovou rychlost při zachování rychlosti modulační. Z hlediska odolnosti proti rušení není obvykle možné použít poměr amplitud menší než 1:2.

Jak modulace PSK, tak modulace QAM jsou dnes používány pro přenos počítačových dat po telefonních linkách. S rozvojem datových sítí se tento obor začal velmi bouřlivě rozvíjet, a dnes se dosahuje přenosových rychlostí okolo 20kb/s při nezměněné šířce pásma (2.4kHz). Těchto modulací se dá také samozřejmně použít pro bezdrátový přenos, protože umožňují dobře využít audio kanál pro přenos datového signálu. Další jejich výhodou je již zmíněné zvýšení přenosové rychlosti vzhledem k šířce přenosového pásma, což je i přes nutnost složitějšího zařízení předurčuje k bezdrátovému přenosu dat v počítačových sítích.

5.9 Způsoby přenosu dat

Tato kapitola má poskytnout přehled terminologie používané v telekomunikační technice pro označení způsobů přenosu dat.

5.9.1 Simplexní, duplexní přenos

Simplexní provoz - data a řídící znaky jsou přenášeny pouze jedním směrem. Datový spoj pro tento provoz je jednoduchý, ale neumožňuje potvrzení přenosu, takže se tohoto druhu provozu užívá především pro sběr dat.

Poloduplexní provoz - data jsou předávána jedním směrem, ale zpět je přenášen údaj potvrzující přenos. Poloduplexní provoz však vyžaduje již obousměrný spoj, i když zpětný spoj má menší nároky na šířku pásma a je možné použít jediný obousměrný kanál.

Duplexní provoz - přenos probíhá oběma směry současně. Pro

současný příjem a vysílání je nutno mít k dispozici dva nezávislé přenosové kanály, a proto je duplexní provoz méně vhodný pro bezdrátový přenos.

5.9.2 Paralelní, sériový přenos

Sériový přenos - jednotlivé bity jsou přenášeny za sebou v časové posloupnosti. Sériový přenos je nejpoužívanějším druhem přenosu pro bezdrátovou komunikaci, protože právě při takovém přenosu je k dispozici obvykle jen jeden přenosový kanál.

Paralelní přenos - lze charakterizovat tím, že všechny bity jednoho kódového slova jsou přenášeny současně.

Sérioparalelní přenos - kombinace obou předchozích způsobů. Je ho nutno užít tam kde je kódové slovo příliš dlouhé a není k dispozici potřebný počet přenosových kanálů.

5.9.3 Synchronní, asynchronní přenos

Pro zpracování přenášeného signálu na straně přijímače je nutné, aby byl určen začátek zprávy a jednotlivé bity musí být detekovány v příslušných časových intervalech. Z hlediska synchronizace přijmu jednotlivých bitů (určení časových intervalů pro jejich příjem) můžeme rozlišit :

Synchronní přenos - předpokládá synchronní práci přijímače a vysílače. Přitom synchronizace musí být zajišťována zvláštním přenosovým kanálem, který nese tuto informaci, nebo jiným společným zdrojem časového rastru, který má k dispozici jak přijímač tak vysílač. Synchronní přenos umožňuje dobře využít komunikační kanál a proto se užívá při vyšších rychlostech přenosu.

Asynchronní přenos - je definován vysíláním časově nevázaných dat, takže příjem a vysílání nemusí být nijak časově spojeno. Takový přenos neumožní detekci seriových posloupností a proto je také asynchronní přenos užíván pouze vyjímečně.

Aritmický přenos - je kompromisem mezi synchronním a asynchronním přenosem. Jednotlivé bloky jsou vysílány v časově asynchronních okamžicích, ale v průběhu přenosu se předpokládá synchronní spolupráce přijímače s vysílačem. Časová základna je spouštěna vždy na začátku přenášeného bloku, který tak musí nést synchronizační informaci, v průběhu přenosu bloku se pak považuje přijímač za synchronní s vysílačem. Jejich časování musí být natolik časově stejné, aby během relace nevznikl časový rozdíl větší než je polovina doby trvání jednoho bitu.

5.10 Shrnutí

Bezdrátové řídící systémy se v průmyslu používají jen zřídka. Důvodem je velká náročnost zařízení a vysoká hladina rušení právě v průmyslových provozech. Rušení způsobuje snížení spolehlivosti a zvyšuje náklady na zařízení. Pokud se bezdrátový systém realizuje, vychází jeho

návrh z dříve popsaných způsobů přenosu a dostupné součástkové základny.

Dnes se využívá bezdrátových mikrovlnných tras pro spojování počítačových sítí. Jde o realizaci oboustranného spoje s přímou viditelností přijímače a vysílače a to jak na kratší vzdálenosti, tak na ty nejdelší - družice. V mnohých případech vychází mikrovlnná trasa levněji než jiné řešení.

S rozvojem počítačové techniky se rozvíjí i modemy pro přenos dat po telefonní síti. Modemy využívají podobné modulace jako bezdrátové spoje. Protože telefonní síti se dnes spojuje stále více počítačů, vynutilo si to i určitou standardizaci v této oblasti přenosu dat.

Snad nejvíce je bezdrátové řízení rozšířeno ve spotřební elektronice, pokud vezmeme v úvahu všechna dálková ovládání různých zařízení. Za pozornost stojí i bezdrátové řízení modelů, pro které se používá vysílačů již mnoho let. Za tu dobu došlo v této oblasti k určité standardizaci. Všichni výrobci používají stejný systém modulace a to PWM-AM, i když ani zde nejsou obvykle jednotlivá zařízení úplně záměnná.

6 HW část modelu

Jako nejjednodušší řešení se nabízí využít nějakého jednočipového mikroprocesoru ze široké nabídky. Sériová data by měla být vysílána a přijímána velmi rychle (asi 100kb/s) se zvýšeným stupněm odolnosti proti rušení realizovaným ochranným kódem. Žádný z dostupných procesorů nemá možnost předefinovat sériově vysílaná slova. Navíc by jen operace vysílání a vyhodnocování IRC snímačů vyžadovala dostatečně rychlý (desítky MHz) procesor. Protože signály ze snímačů přicházejí asynchronně, není možno 100% programově zaručit jejich okamžité zpracování. To by mohlo vést ke ztrátě impulzu z IRC snímače.

HW realizace diskrétními obvody řeší všechny nedostatky předchozí varianty. Všechny signály lze zpracovávat paralelně a návrh se může plně přizpůsobit požadavkům kladeným na zařízení. V helikoptéře je však velmi omezena užitečná hmotnost zařízení. Ta by zvláště při využití obvodů nižší integrace byla značná. Dalším nedostatkem je nemožnost změny pevně zadrátované funkce při změně požadavků nebo požadavku na vylepšení.

Všechny uvedené problémy jsou řešeny programovatelným logickým polem. Pole GAL nebo PAL mají příliš malý počet ekvivalentních hradel na obvod. Z uvedeného důvodu jsme se rozhodli pro realizaci HW programovatalnými poli XILINX. Vzhledem ke složitosti práce s těmito obvody je způsob jejich programování diskutován ve zvláštní kapitole.

6.1 Celková koncepce

K povelovému řízení bylo využito původní RC soupravy s tím, že modulační signál je tvořen časovači (**Deska VT**) řízenými z počítače. Vytvořený modulační signál je zaveden do původního vysílače **RCTx**. V helikptéře je signál zpracován původním přijímačem **RCRx** a převeden na polohu pomocí modelářských serv. Serva a RCRx vyžadují napájení 4,8 až 7V, které zajišťuje zdroj **ZD1** +5V/2A umístěný ve vrtulníku.

Snímání polohy je prováděno pomocí inkrementálních snímačů polohy. Přímo na místě snímání je převeden inkrementální signál na absolutní údaj o poloze. Vzhledem k odlehlosti a možnosti bezdrátového přenosu, je polohová informace převedena na sériový kód a přenášena vedením k počítači, kde jsou data uložena do registrů z nichž si může přečíst informaci počítač. Protože celý úkol je značně obvodově náročný byly pro realizaci vybrána velkokapacitní programovatelná logická pole XILINX. Přímo ve vrtulníku zpracovává údaje ze snímačů polohy na kloubu pod vrtulníkem (**kloub I**) obvod XILINX **XV1**. Signál z vrtulníku je přenášen AM vysílačem **Tx** na heliport, kde je umístěn přijímač **Rx**. Údaje ze snímačů na kloubu pod heliportem (**kloub II**) jsou zpracovány a převedeny na sériový kód obvodem XILINX **XV2**. Od heliportu jsou data z obou zdrojů, již v sériové formě, přenášena do interfaceu a tam rozdělena do jednotlivých registrů. Dekódování se provádí opět obvodem XILINX na desce **XRI**, která tvoří základ interfacesu. Kromě obvodu XILINX je na desce XRI i rozhraní

Zkratka	Část zařízení	Umístění
RCTx	Původní vysílač RC soupravy sloužící k řízení modelu.	
RCRx	Přijímač RC soupravy umístěný na modelu. K přijímači jsou připojena serva.	Vrtulník
ZD1	Zdroj zajišťující napájení 5V/2A pro RCRx a serva.	Vrtulník
XV1	Obvod pro zpracování signálu ze snímačů a sériové vysílání dat. (obvod XILINX)	Vrtulník
XV2	Obvod pro zpracování signálu ze snímačů a sériové vyslání dat. (obvod XILINX)	Heliport
XRI	Obvody zajišťující příjem sériových dat, jejich dekódování a komunikaci s počítačem. (obvod XILINX)	Interface
VT	Vysílací časovací obvod, který na základě dat z počítače generuje modulační signál poro RCTx.	Interface
UN	Univerzální deska s konektorem pro RCTx v interface.	Interface
Tx	AM vysílač pro přenos sériových dat z vrtulníku.	Vrtulník
Rx	AM přijímač pro přenos sériových dat z vrtulníku.	Heliport

Table 2: Značení částí systému

pro počítač, přes který se připojuje nejen XILINX, ale i vysílací časovače VT. Interface se skládá z desky XRI, VT a universální desky **UN** na, které je konektor pro připojení RCRx. Používané značení je shrnuto v 6.1.

6.2 Způsob připojení k PC

Celé zařízení by bylo možno navrhnout jako zásuvnou kartu do motherboardu počítače. Z hlediska ovládacího programu je možno pohodlně číst hodnoty z jednotlivých snímačů polohy buď z paměti mapované do adresního prostoru mikroprocesoru nebo ze série po sobě následujících portů. Protože jsou používány i jiné rozšiřující karty, musel by být mapovaný prostor popř. pozice portů snadno přesunutelný do jiné oblasti adresního prostoru. Tímto by se však podstatně zhoršila přenositelnost na jiné typy počítačů. I výměna mezi dvěma počítači stejného typu by se poněkud zkomplikovala vzhledem k nutnosti rozebírat vnější kryt.

Dále se přímo nabízí použít pro komunikaci sériový port. Na rychlost přenosu jsou však kladeny vysoké nároky. Nároky jsou ještě vyšší než u systému přijímače, který může zpracovávat data po celou dobu své činnosti. Centrální počítač se musí navíc starat ještě o relativně komplikovaný regulační algoritmus a vlastní komunikace s vnějším zařízením by měla spotřebovat jen zanedbatelný zlomek celkové doby. Linka by musela být velmi rychlá (zde už není tak kritické omezení na velikost jako v kabině modelu). Pro tyto linky jsou vyráběny speciální obvody. Ale na straně PC by v diskutovaném případě musel být umístěn přijímač. Přijímače uvedeného typu jsou často připojovány k zásuvným kartám se všemi výše popsanými nevýhodami dříve rozebíraného řešení.

Vzhledem k tomu, že většina A/D karet poskytuje i digitální vstupy/výstupy, jsme se rozhodli pro připojení přes paralelní rozhraní realizované přes kartu PCL-812. Pro komunikaci je využito tří 8 bitových portů (dva výstupní a jeden vstupní). Dva porty - jeden vstupní a jeden výstupní - jsou použity pro externí datovou sběrnici. Třetí výstupní port generuje řídící signály pro sběrnici. Popsaný interface vlastně vytváří obousměrnou sběrnici odpovídající systému 8080. Průběhy napětí na jednotlivých vodičích při čtecím respektive zápisovém cyklu jsou nakresleny na 6.2 a 6.2. V zásadě je možno celé zařízení připojit k libovolnému I/O modulu s uvedeným počtem digitálních vstupů/výstupů. Logické výstupy musí být opatřeny pamětí, aby data byla k dispozici stále a ne pouze po dobu cyklu ISA sběrnice počítače PC.

Signál z PC	Význam na zařízení	Popis
D/O0	D0	Datová sběrnice
D/O7	D7	D0-D7 pro zápis
D/O8	A0	Jednosměrná
D/O12	A4	adressová sběrnice
D/O13	-	nepoužito
D/O14	˜RD	Čtení
D/O15	˜WR	Zápis
D/I0	D0	Datová sběrnice
D/I7	D7	D0-D7 pro čtení

Nabízí se varianta použít pouze jednu obousměrnou datovou sběrnici. Tím by se snížil počet připojovacích vodičů. Uvedené řešení bylo vynuceno neexistencí obousměrného digitálního portu na použitém typu I/O karty.

Doba potřebná pro čtení dat T=500nS v případě, že má PC

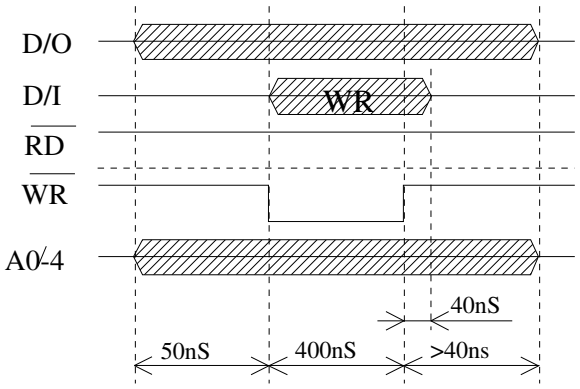


Figure 23: Průběh zápisového cyklu

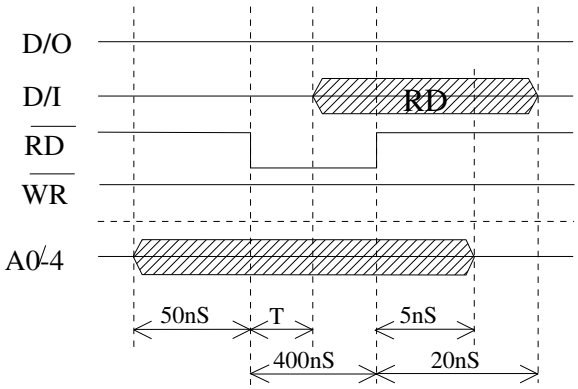


Figure 24: Průběh čtecího cyklu

prioritu přístupu **1** k interní sběrnici. Zařízení s prioritou **3** může čekat v nejhorším případě 4μS. Při přiřazení priority **1** hostitelskému počítači by teoreticky mohlo dojít k zahlcení interní sběrnice opakovanými požadavky v intervalu kratším než 1μS (dva interní sběrnicové cykly). Vzhledem k rychlosti I/O karty diskutované v kapitole *Propojení z hlediska PC* CPU takovýmto způsobem interní sběrnici zahltit nedokáže. Nejkratší časový interval mezi dvěma požadavky je 2.8μS a vychází z časování sběrnice ISA (jedná se o dva zápisy na výstupní port). Při běžném provozu se tato doba ještě výrazně prodlouží vzhledem k nutnosti zpracovat přečtená data.

Podrobný popis činnosti a funkce řadiče vnitřní sběrnice je součástí [dip. práce P. Krsek - Řídící systém helikoptéry]. Činnost této části je rozebírána s ohledem na tvorbu programového rozhraní.

6.3 Komunikace PC-model

Systém řízení byl řešen v diplomové práci **"Řízení helikoptéry" (Pavel Beneš r.1993)**. Větší část tohoto systému již byla ve funkčním stavu a bylo nutno provést pouze drobné úpravy pro začlenění do nově vytvářeného systému snímání polohy. Základem pro řízení je původní modelářská RC souprava. Úpravy původního vysílače a přijímače nesměly přitom znemožnit původní funkci.

6.3.1 Funkce původní RC soupravy

Pro objasnění funkce programového ovladače je vhodné ve stručnosti popsat konstrukci a způsob činnosti RC soupravy. Model je ovládán dvěma křížovými ovladači, z nichž každý sdružuje 2 funkce. Páka ovladače je mechanicky spojena s jezdcem potenciometru, který převádí výchylku na napěťovou úroveň. Úrovně napětí jsou zpracovány a časově multiplexovány

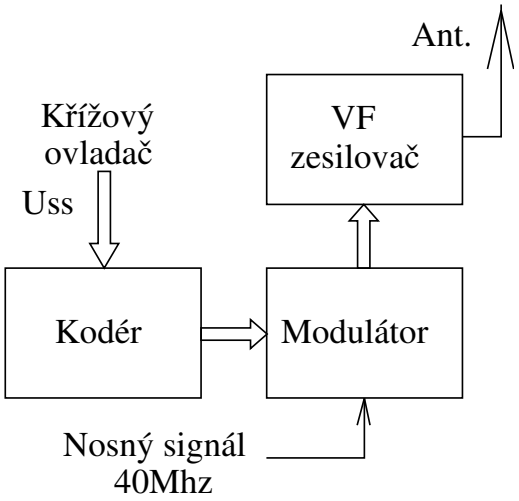


Figure 25: Blokové schéma RC vysílače

kodérem, který převádí jednotlivá napětí na délku impulsu.

Ve VF části je signál amplitudově modulován a šíří se vzduchem do přijímače. Po demodulaci v přijímači jsou získány obdélníkové průběhy, jejichž střída je úměrná výchylce ovladače. Ty jsou rozvedeny na jednotlivá serva.

RC souprava využívá pulsně šířkové modulace s časovým dělením

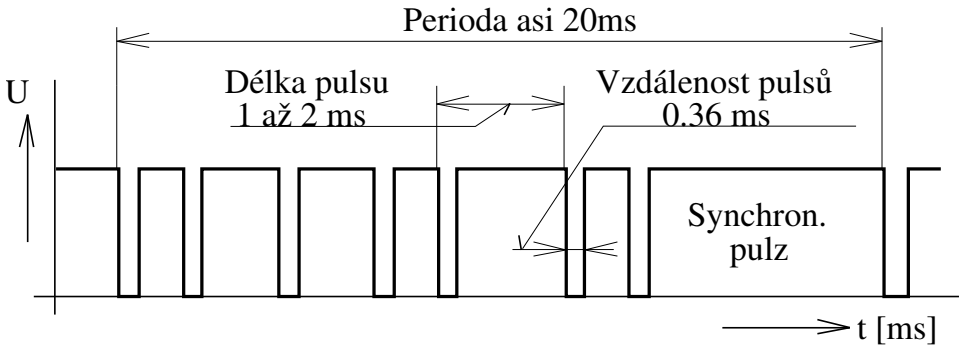


Figure 26: Signál RC vysílače (obalová křivka)

přenosového kanálu. Tento signál je amplitudově modulován modulací A1 na nosný signál 40,695MHz. Průběh signálu je zobrazen na 6.3.1. Informaci o požadované poloze serva nese vzdálenost mezi jednotlivými pulsy, která může být 1 až 2ms. Pulsy v jedné periodě, která trvá 20ms nesou informaci pro jednotlivá serva (kanály). Délka periody je závazná (do jisté míry).

Nutno je i dodržet minimální délku synchronizačního pulsu (asi 8ms), který doplňuje periodu od posledního informačního pulsu do stanovené délky. V případě, že použijeme širší informační pulsy a zkrátíme tak puls synchronizační dojde k rozpadu přenosu a s tím vznikne nekoordinovaný pohyb serv. Popsaná situace může nastat v případě chybné programové manipulace s časovači VT.

Způsob vyhodnocení signálu umožnil jednoduché rozšíření původně čtyřkanálové soupravy na šesti kanálovou. Pátý kanál byl použit pro řízení motoru. Šestý kanál je nevyužit. Duležité je nezapomenout programově nastavovat i délku tohoto pulsu (asi 1,5ms), aby nedošlo k rozpadu synchronizace.

6.3.2 Úprava pro řízení počítačem

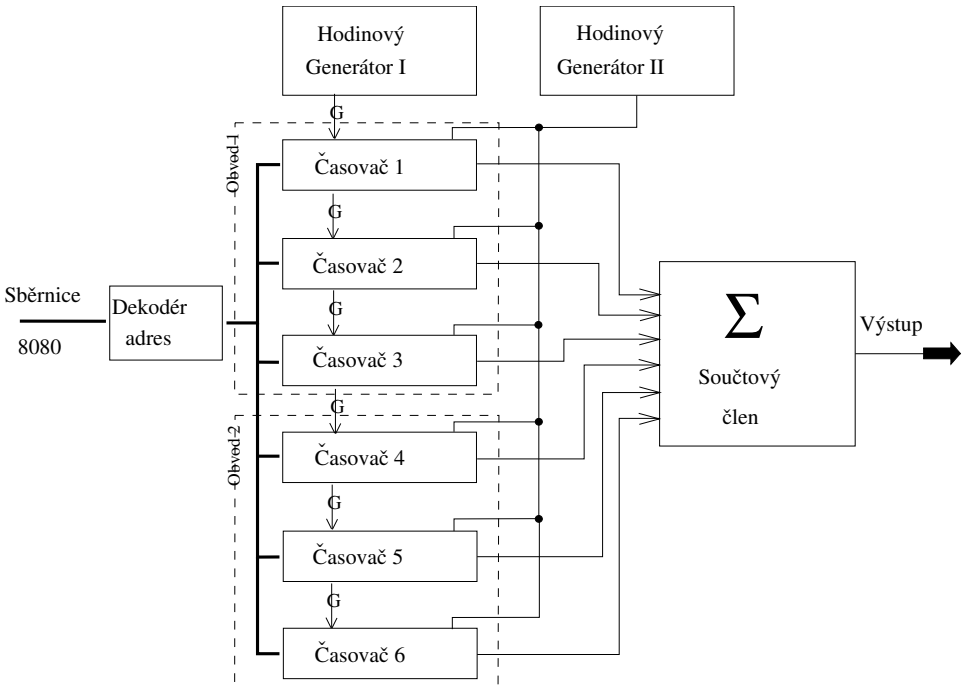


Figure 27: Blokové schéma upravené vysílací části

Pro ovládání pomocí počítače byl výše popisovaný řetězec přerušen v místě před modulátorem. Multiplexovaný signál je vytvářen kaskádou programovatelných časovačů. Celá kaskáda je spouštěna Generátorem II. Jeho perioda určuje tzv. rámec vysílání. Po spuštění dojde k postupnému spuštění 6 monostabilních pulsů. Jejich délka je určena digitálním vstupem a frekvencí generátoru I.

Celé zařízení je realizováno pomocí 2 univerzálních programo-

vatelných čítačů 8253. Ty jsou připojeny k vnější datové sběrnici, která simuluje standard sběrnice 8080. Zjednodušené blokové schéma této části je nakresleno na 6.3.2.

Programovatelné obvody však nutně vyžadují inicializaci po svém zapnutí. Bez inicializace se celé zařízení nechová korektně a nelze vyloučit ani neočekávaný rozběh motoru!

V době řešení této části se jevilo použití programovatelných časovačů jako optimální. Avšak při použití modernějších logických programovatelných polí by bylo možno vytvořit vysílací část, která by nevyžadovala inicializaci z počítače a generovala vhodný signál ihned po svém zapojení. Pro poslední variantu jsme se nerozhodli pouze z toho důvodu, že celý obvod byl již zapojen a odladěn.

Adresování vysílacích čítačů Obvody 8253 jsou připojeny k simulované sběrnici systému I8080. Každý z obvodů obsahuje tři registry předvolby čítačů a jeden registr řídící. Přehled o jednotlivých adresách je v 6.3.2.

Na začátku je nutno nahrát do řídících registrů data určující práci

Registr (pouze pro zápis)	Adresa (bin, hex)	Číslo obvodu
Předvolba čítač 1	00000B=00H	
Předvolba čítač 2	00001B=01H	Obvod 1
Předvolba čítač 3	00010B=02H	
Řídící slovo obvodu 1	00011B=03H	
Předvolba čítač 4	00100B=04H	
Předvolba čítač 5	00101B=05H	Obvod 2
Předvolba čítač 6	00110B=06H	
Řídící slovo obvodu 2	00111B=07H	

Table 3: Adresy registrů vysílacích čítačů

čítačů. Čítače musí být nastaveny do režimu MKO s 16 bity použitými pro čítání. Konfigurační data se nahrávají zvlášť pro každý čítač na adresu řídícího registru. Pro který čítač je konfigurace určena udává její obsah. Nastavit je nutno všechny čítače. Zápis šestnáctibitových dat do předvolby čítačů se provádí postupně po osmi bitech na adresu příslušného čítače. Jako první se zapisuje nižší významový byte a poté vyšší významový byte. Podrobnější informace jsou v [3].

6.3.3 Propojovací kabely

Fyzicky je propojení s počítačem provedeno dvěma plochými kabely opatřenými na obou stranách samořeznými konektory s dvaceti vývody. Jeden kabel je vstupní (spojuje signály D/I) a druhý je výstupní (spojuje signály D/O). Oba kabely přivádí do desky XRI a tím i do celého interfaceu napájení +5V a +12V. Bohužel jsou tyto dva kabely záměnné a to na obou

stranách. Na počítači také nelze tak jednoduše zjistit, který konektor je výstupní. Pro určení existuje jednoduchý test. Postačí pouze do výstupu zapsat data a adresu (pokud možno v nevyužité části adresního prostoru) a aktivovat signál WR. Tím se dostanou data na sběrnici a musí je tedy jít přečíst na vstupu (D/I0-D/I7). Pokud nejsou přečtena data správně je pravděpodobné, že došlo k záměně konektorů, nebo není zapnuto napájení. Záměna konektorů skutečně zamezí funkci, ale napájení je při ní v pořádku a nedojde k žádnému fyzickému poškození. Po správném zapojení kabelů bude zařízení opět korektně pracovat.

Poznámka: Jak deska XRI tak i laboratorní karta jsou opatřeny vždy párem stejných dvaceti vývodových konektorů pro plochý kabel. Číselné značení vývodů na konektorech je shodné pro desku XRI i laboratorní kartu. Konektor K1 je výstupní a na kartě je označen D/O. Vstupní konektor K2 odpovídá konektoru karty D/I.

6.3.4 Propojení sběrnic

Protože logické vstupy a výstupy jsou jednosměrné a sběrnice systému 8080 je obousměrná bylo nutno použít obvodů pro zřízení této sběrnice. Tyto obvody jsou na desce XRI společně s přijímacím obvodem XILINX a jeho podpůrnými obvody. Schema desky XRI je součástí přílohy C. Vstup je řešen přímým připojením vstupů karty D/I0 - D/I8 k příslušným bitům datové sběrnice. Správnost čtení (jeho časování) zajišťuje programové vybavení. Stálé připojení vstupních signálů karty k simulované sběrnici tuto sběrnici nezatěžuje. Podobně je připojena i adresová sběrnice k výstupním signálům D/O8 - D/O12. Přímé spojení je v případě adresy umožněno tím, že počítač je jejím jediným zdrojem. Naopak datové signály výstupní D/O0 - D/O7 musí být odděleny od datové sběrnice jednosměrným třístavovým budičem, který reprezentuje obvod 74HCT373 - IO6 na desce XRI. Řízení IO6 je logickým obvodem odvozeno od signálu WR, přičemž je zajištěn i přesah dat na sběrnici po skončení signálu WR. Řídící sběrnice tvořená signály RD a WR, aktivními v nule, je generována logikou (IO7, IO8, IO9) na základě signálu D/O14 odpovídající RD a D/O15, který představuje signál WR. Logika zajišťuje, aby nemohly být aktivovány oba signály (RD, WR) současně a nedošlo tak ke kolizi na sběrnici. Přehled o propojení odpovídajících signálů je v 6.4.2.

6.4 Komunikace Model-PC

6.4.1 Výchozí předpoklady a způsob řešení

Protože informace z inkrementálních snímačů natočení i údaj otáčkoměru jsou digitální signály, bylo jednoduché rozhodnout o číslicovém zpracování, a přenosu informace do počítače. Signál z inkrementálního snímače je za pomoci čítače převedena na údaj o absolutní poloze, obdobně

je pak převeden i signál z otáčkoměru. Všechny údaje nyní již reprezentované osmibitovými slovy se po vnitřní sběrnici obvodu přenáší do vysílacího shift registru, kde je toto slovo doplněno o adresu zdroje a synchronizační značky, čímž vznikne vysílané 32bitové slovo složené ze čtyř osmibitových slabik.
Data jsou vysílána sériovým kódem, který lze užít i pro AM modulaci bezdrátového spoje, do přijímače umístěného na desce XRI.

Do přijímače přicházejí data ze dvou vysílačů XV1 a XV2. V něm jsou dva shift registry, vždy jeden pro jeden přenosový kanál. Přijatá data jsou po kontrole přenášena podle jejich adresy přes vnitřní sběrnici do odpovídajících paměťových registrů. Z paměťových registrů mohou posléze být na vyžádání předány po simulované sběrnici systému 8080 do počítače. Problémem, který musel být vyřešen, je sdílení vnitřní sběrnice obvodu více zdroji dat.

Z důvodů rozumné realizovatelnosti musela být omezena délka přenášených slov na 8 bitů, i když údaj o absolutní poloze je vícebitový. Dojde tak k tomu, že poloha udávaná zařízením je vlastně relativní s větší periodou neurčitosti. Počítač pak rekonstruuje absolutní polohu. Jediným omezením je perioda vzorkování, která nesmí být tak velká, aby v této době mohl údaj překročit pásmo absolutního určení polohy. Osmibitový číslicový údaj se nesmí změnit o více jak 127 kroků, aby mohla být rekonstruována poloha a směr pohybu.

Celkovou představu o koncepci systému si čtenář může udělat z 6.5.4. Všechny tři obvody XILINX jsou řešeny jako systém čítačů a registrů okolo interní sběrnice. Protože ústředním motivem je přenos dat, o právě vykonávané činnosti rozhoduje časovač vysílání nebo příjmu. Je pravda, že v případě přijímače je to komplikovanější o vzájemnou spolupráci dvou přijmačů a počítače. O tomto problému bude ještě diskutováno při popisu konkrétních schemat v příloze.

6.4.2 Komunikace z hlediska počítače

Pro přijímmací obvod XILINX na desce XRI je na simulované sběrnici systému 8080 určen adresní prostor od adresy 10H do adresy 1FH. Na vnitřní sběrnici obvodu XILINX to představuje adresy 0 až FH.

Na adresách 0 až 7H jsou registry představující obrazy registrů a čítačů snímajících polohu a otáčky. Jsou to registry jejichž obsah byl přijat po dvou sériových kanálech z desky XV1 (adresa 0 až 3) a desky XV2 (adresa 4 až 8). Registry s adresou 9 a AH jsou obrazem dvou čtyřbitových čítačů, které počítají korektní přenosy v jednotlivých kanálech. Počet přenosů by měl být asi 4500 za sekundu v každém kanálu. Čítače přenosů se nulují čtením libovolného znaku z kontrolního řetězce. Kontrolní řetězec je představován znaky "PK&JF" uloženými v registrech s pevným obsahem na adresách B až FH. Čtení kontrolního řetězce se používá ke kontrole správného připojení zařízení. Rozpis registrů s udaným významem a jednotkami je v 6.4.2.

Registry udávající polohu jsou pouze osmibitové i když rozsah poloh je vesměs mnohem větší (odpovídalo by tomu například 10 až 14 bitů) a údaj může mnohokrát přetéci obsah osmibitového registru. Údaj v registrech není

Adresa (hex.) sběrnice I8080	Adresa (hex.) interní sběrnice	Údaj v registru	Zdroj dat, deska	Jednotky na dílek
10	0	Příčný náklon modelu	XV1	360°/(4x512) O,17578°/b
11	1	Podélný náklon modelu	XV1	360°/(4x512) O,17578°/b
12	2	Kurs modelu	XV1	360°/(4x540) 0,16667°/b
13	3	Otáčky nosného rotoru	XV1	(ot.x10)/227ms opakování 227ms
14	4	Příčný náklon spojovací tyče	XV2	360°/(4x512) O,17578°/b
15	5	Podélný náklon spojovací tyče	XV2	360°/(4x512) O,17578°/b
16	6	Vysunutí spojovací tyče	XV2	0,203mm/b
17	7	Nepoužit		
18	8	Kontrolní reg. 1. kanál	XRI	Počet přenosů
19	9	Kontrolní reg. 2. kanál	XRI	počet přenosů
1A	A	Nepoužit		
1B	B	P	XRI	
1C	C	K	XRI	
1D	D	&	XRI	
1E	E	J	XRI	
1F	F	F	XRI	

Table 4: Rozložení datových registrů

tedy absolutním údajem, ale pouze relativním s větší periodou opakování než signál z inkrementálních snímačů polohy. Pokud ovšem budeme číst počítačem tyto registry dostatečně rychle, tedy tak aby změna mezi jednotlivým čtením nebyla více jak polovina rozsahu osmibitového registru, je možná rekonstrukce absolutní polohy v počítači. Tento problém ovšem limituje minimální vzorkovací periodu počítače. Maximální vzorkovací perioda, která má ještě smysl, je dána opakováním přenosu do jednotlivých registrů. Do jednoho registru se provádí asi 1000 přenosů za sekundu.

Registry obvodu XILINX lze číst stejně jako jakékoliv zařízení na sběrnici systému I8080, pouze s tím, že doba potřebná k dodání dat (ozanačovaná T) je v případě přijímacího obvodu XILINX asi 870ns. Toto zpoždění je dáno časováním obvodu a sdílením interní sběrnice společně s dvěma přijímači. Aby se zajistila možnost přístupu ke sběrnici i pro oba přijímače nesmí počítač opakovat čtení z obvodu XILINX s kratší periodou než 1,5μs.

Signál karty PCL812	Signál sběrnice standardu 8080 (význam signálu)	Číslo vývodu konektoru	Konektor na desce XRI
D/O0 až D/O7	D0 až (Datová sběrnice pro zápis) D7	1 až 8	K1
D/O8 až D/O12	A0 až (Adresová sběrnice) A4	9 až 13	K1
D/O13	Není použito	14	K1
D/O14	RD (Řídící signál čtení)	15	K1
D/O15	WR (Řídící signál zápisu)	16	K1
	GND	17 a 18	K1
	$+5V_z$	19	K1
	$+12V_z$	20	K1
D/I0 až D/I7	D0 až (Datová sběrnice pro čtební) D7	1 až 8	K2
D/I8 až D/I15	Není použito	9 až 16	K2
	GND	17 a 18	K2
	$+5V_z$	19	K2
	$+12V_z$	20	K2

Table 5: Připojení k počítači přes kartu PCL812

6.4.3 Časování obvodů

Podstatou časové základny všech obvodů pro snímání polohy se stal krystal (krystaly) s kmitočtem 18,432MHz. Od tohoto kmitočtu je odvozen

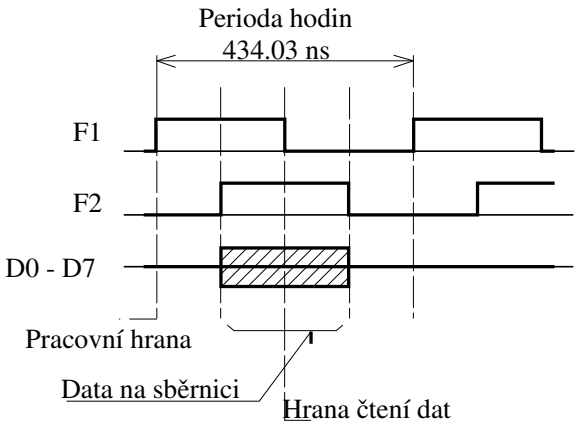


Figure 28: Hodinový signál a časování vnitřní sběrnice

dělením kmitočet dvoufázových hodin, které jsou základem pro časování všech zapojení uvnitř logického pole. Signály hodin F1 a F2 mají kmitočet 2,304MHz a jejich průběh je zachycen na 6.4.3. Tyto hodinové signály se staly základem i pro takt sběrnice.

Nástupná hrana F1 se používá pro počítání a zápis do zdrojových registrů. V době kdy je signál F2=H (log.1), jsou všechny registry již v klidu a může se zahájit přenos. Adresou vybrané zařízení (čítač, registr) v této době dodává data na interní sběrnici. Tato data jsou čtena do cílového zařízení sestupnou hranou signálu F1. Sestupná hrana signálu F2 již může opět mněnit stav registrů a čítačů. Tímto způsobem je zajištěn přenos po interní sběrnici obvodů XILINX s vyloučením hazardů. Znázornění způsobu přenosu po sběrnici je součástí 6.4.3.

V případě přijímacího obvodu XRI je vnitřní sběrnice sdílena více zdroji. Proto musel vzniknout jednoduchý systém správy sběrnic s pevně stanovenou prioritou požadavků. Nejvyšší priorita je přidělena počítači. Protože doba odbavení je jeden takt hodin (430ns) a o sběrnici se dělí tři zdroje neměl by se přístup počítače opakovat dřív jak za 1,5ms, aby měli možnost dostat se ke sběrnici všechny tři zdroje. Bližší popis obvodu pro správu sběrnic s průběhy signálů je v příloze C.

6.4.4 Sériový přenos dat

Z důvodů značné vzdálenosti jednotlivých částí zařízení a jejich vzájemného pohybu jsem byl nucen použít sériový přenos dat k počítači. Sériový jednosměrný simplexní provoz při použití vhodného kódu umožní snadnou náhradu vodiče bezdrátovým přenosovým kanálem s amplitudovou modulací.

Pro kódování byl zvolen bipolární NRZ kód, který potřebuje relativně nejužší frekvenční pásmo přenosového kanálu a po přidání synchronizačních bitů do přenášených dat je i poměrně jednoduchý pro detekci.

Přenosový kanál je tedy jednosměrný simplexní a přenos v něm probíhá aritmicky.

Data jsou vysílána v 32bitových slovech, která byla z důvodů

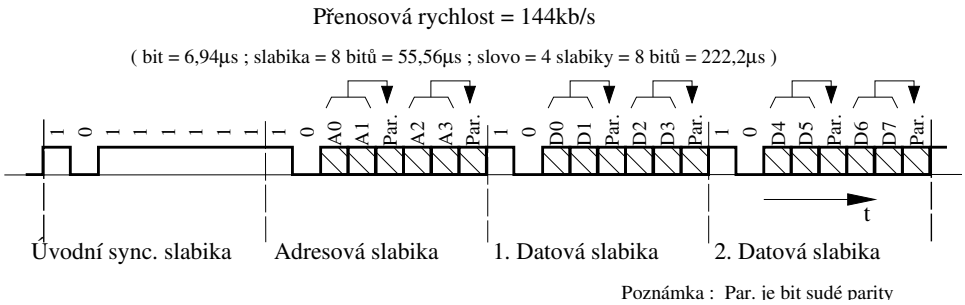


Figure 29: Tvar vysílaného sériového datového slova

jednodušší realizace z diskrétních součástek rozdělena do čtyř osmibitových slabik. Příklad jednoho slova je na 6.4.4. Každá slabika nese pouze čtyři informační bity zajištěné dvěma bity sudé parity (jeden paritní bit jistí vždy dva bity datové). Slabika začíná dvěma startovními a synchronizačními bity. První bit je tak log. 1 a druhý bit log.0. Následují dva datové bity, jištěné třetím sudým paritním bitem. Poté pokračují další dva datové bity se svým paritním bitem. Datové bity se vysílají postupně od nejméně významného bitu. První slabika slova je synchronizační a kromě dvou startovních bitů je obsah všech šesti dalších bitů log. 1, takže není dodržena parita. A právě tím se liší synchronizační slabika od datových. Také se tak ve vysílaných datach objeví jedinečná sekvence sedmi logických jedniček za sebou, která umožní synchronizaci začátku slova. Druhá slabika nese v datových bitech čtyřbitovou adresu zdroje (zároveň i cíle). Další dvě slabiky obsahují osmibitová přenášená data. Méně významná čtyřbitová půlka přenášeného bytu je obsahem třetí slabiky a významnější část datového bytu je přenášena ve slabice čtvrté. Přestože osmibitová data nejsou zcela dostačující,toto omezení bylo zvoleno z důvodů rychlosti přenosu a jeho jednodušší realizovatelnosti.

Každý z vysílacích obvodů XV1 a XV2 obsahuje čtyři zdroje osmibitových zpráv. Data z jednotlivých zdrojů jsou vysílána ve slovech postupně za sebou. Slova sebou nesou i adresu, která přenášená data identifikuje. Tento způsob přenosu dat umožňuje poměrně spolehlivou detekci chyb. Přitom při chybě dojde pouze ke ztrátě jednoho slova, protože slova jsou jednoznačně identifikována nezáleží na jejich pořadí a tím ani na ztrátě jednoho z nich.

6.4.5 Časování sériového vysílače

Vysílací hodiny jsou odvozeny z kmitočtu hodinového signálu F1 (2,304MHz) pomocí čtyřbitového čítače. Rychlost vysílání je tedy F1/16 =

144kb/s. Počet vyslaných slov je 4500 za jednu sekundu. Protože vysílána jsou postupně slova ze čtyř zdrojů, je opakovací kmitočet přenosu jednoho údaje (z jednoho zdroje) 1,125kHz. Tento kmitočet je nejvyšším možným vzorkovcím kmitočtem systému snímání polohy (předpoklad bezchybnosti přenosu).

Rychlost vzorkování je více než dostatečná pro řízení vrtulníku a tak nebude vadit ani skutečnost, že jednotlivé kanály nejsou vzorkovány současně (v jeden časový okamžik), ale postupně tak jak jsou přenášeny. Tento problém je ostatně adekvátní nesynchronnosti počítače a zařízení pro snímání polohy či systému povelového řízení.

6.5.1 Úkoly a koncepce přenosu
6.5 Bezdrátový datový spoj

Úkolem bylo relizovat bezdrátový spoj z helikoptéry, aby se mohl model volněji pohybovat. Přenos bude probíhat na vzdálenost několika cm až asi jednoho metru. Hlavním požadavkem kromě funkčnosti byla i malá hmotnost vysílače a jednoduchost realizace celého systému.

Jak bylo se dříve zmíněno, byl vybrán sériový kód pro přenos údajů z desek XV1 a XV2 nejen s ohledem na úsporu propojovacích drátů, ale i s výhledem na bezdrátový přenos. Sériový NRZ kód doplněný o synchronizační bity a bity pro ochranu dat, je vhodný i zdůvodu poměru přenosové rychlosti k frekvenční šířce přenosového kanálu pro bezdrátový přenos. Použitý kód je dostatečně odolný proti chybám a je schopen rychlého zotavení z případné chyby (do 32bitů tj. 222μs). To jsou vlastnosti nezbytné pro použití v přenosovém kanálu s velkým rušením, jakým může být bezdrátový spoj. Protože jak vysílač tak přijímač sériového signálu byly již s ohledem na bezdrátový přenos navrženy je jeho realizace jednoduchá. Pro přehled je na 6.5.1 zachycen standardní řetězec bezdrátového přenosu s drobnými poznámkami týkajícími se již konkrétního řešení.

Protože není možný přenos v základním pásmu, byla pro jednodu-

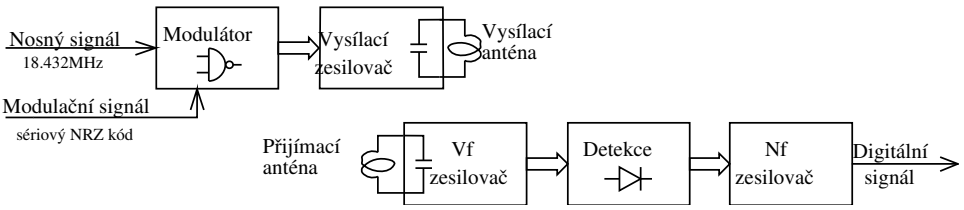


Figure 30: Přenosový řetězec bezdrátového datového spoje

chost realizace vybrána amplitudová modulace (AM). Při hloubce modulace 100% je možné realizovat modulátor jediným hradlem, které je uvnitř obvodu XILINX. Za nosný signál byl zvolen signál s kmitočtem řídícího krystalu 18,432MHz. Zapojení vysílače se tím velice zjednoduší.

Důležité je poznamenat, že vysílací výkon je pouze asi 10 až 20mW. Tak malý výkon je dostatečný pro přenos na krátkou vzdálenost a přitom z hlediska předpisů spadá mezi pomocné vysílače. Mezi ně řadí vysílač i absence antény s větším ziskem. Protože pomocné vysílače nemají žádná

frekvenční ani modulační omezení a nepodléhají registraci, je celkem pochopitelná snaha, aby vysílač patřil mezi vysílače pomocné. Malý výkon a použitá anténa samozřejmně také zajistí, že vysílač nebude rušit jiné zařízení a hlavně nebude rušit ani původní RC soupravu, kterou využíváme pro povelové řízení a jejíž antena je v těsné blízkosti tohoto vysílače.

6.5.2 Vysílač

Vysílač se skládá z modulátoru, který je tvořen hradlem NAND. Aplitudově modulovaný signál z výstupu hradla je zaveden do výstupního zesilovače. Zesilovač tvoří tranzistor PNP s resonančním obvodem v kolektoru laděným na kmitočet nosného signálu. Cívka resonančního obvodu tvoří i vysílací anténu. Napájecí napětí vysílače je 10V z hlavního napájecího zdroje. Průběh modulačního signálu je volen tak, aby v době kdy se nemá vysílat byl vysílací tranzistor určitou dobu otevřen a rychleji zatlumil anténní resonanční obvod. Dojde tak k zrychlení přechodu mezi dvěma stavy použité digitální amplitudové modulace, což zlepší situaci při zpracování sériového signálu v přijímačovém obvodu XILINX.

6.5.3 Přijímač

Přijímač tvoří vstupní resonanční obvod jehož cívka je i přijímací anténou. Signál z tohoto obvodu je zesilován jednotranzistorovým odporovým VF zesilovačem. Z jeho výstupu přichází signál přímo na detekci (usměrnění a filtrace) a pokračuje dál do Nf zesilovače, kterým je tranzistor určený k úpravě napěťových úrovní. Výstupem přijímače je přímo sériový signál v úrovních TTL, jako byl před modulátorem. Přijímač není vybaven žádným systémem AVC, protože se předpokládá dostatečná síla signálu při libovolné pozici modelu a nepředpokládám ani větší změnu síly signálu jak 20dB, při čemž amplitudové omezení signálu nevadí, nebo je dokonce výhodné.

6.5.4 Anténní systém

Obě antény tvoří cívky resonančních obvodů. Toto řešení je pravděpodobně nejjednodušší pro daný nosný kmitočet. Je nutno si totiž uvědomit, že délka vlny je asi 16,3m.

Byla provedena řada pokusů s umístěním antén. Požadavkem bylo mít dostatek signálu pro zpracování v přijímači a to i při pohybu modelu na němž je vysílač umístěn. Nevýhodou cívkových antén je velký útlum přenášeného signálu v případě jejich vzájemného kolnmého postavení. Naopak výrazného zlepšení přenosu bylo dosaženo pokud vysílací a přijímací cívka byla na společném vodivém jádře.

Z těchto pokusů vyšlo i umístění cívek okolo jádra tvořeného spojovací tyčí trenažeru. Vysílací cívka je umístěna těsně pod kloubem I (pod vrtulníkem) navinutá na umělohmotové trubce, která volně procází spojovací tyč. Cívka je upevněna na snímači pro snímání kurzu, jak ukazuje 6.5.4. Tímto umístěním je zajištěno, že vysílací cívka směřuje i za pohybu stále k heliportu a její rovina nebude moci být nikdy k rovině heliportu kolmo.

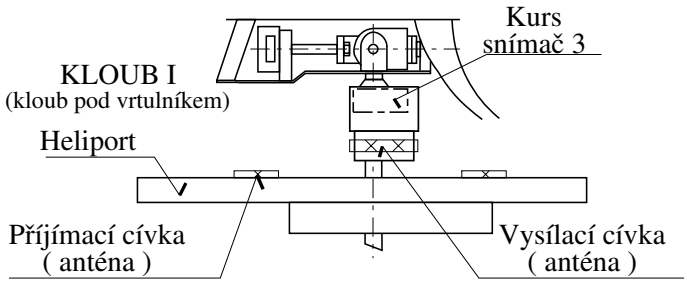


Figure 31: Umístění antén datového spoje

Přijmací cívka je navinuta na heliportu. Oba resonanční obvody musí být laděny s ohledem na umístění jejich cívek.

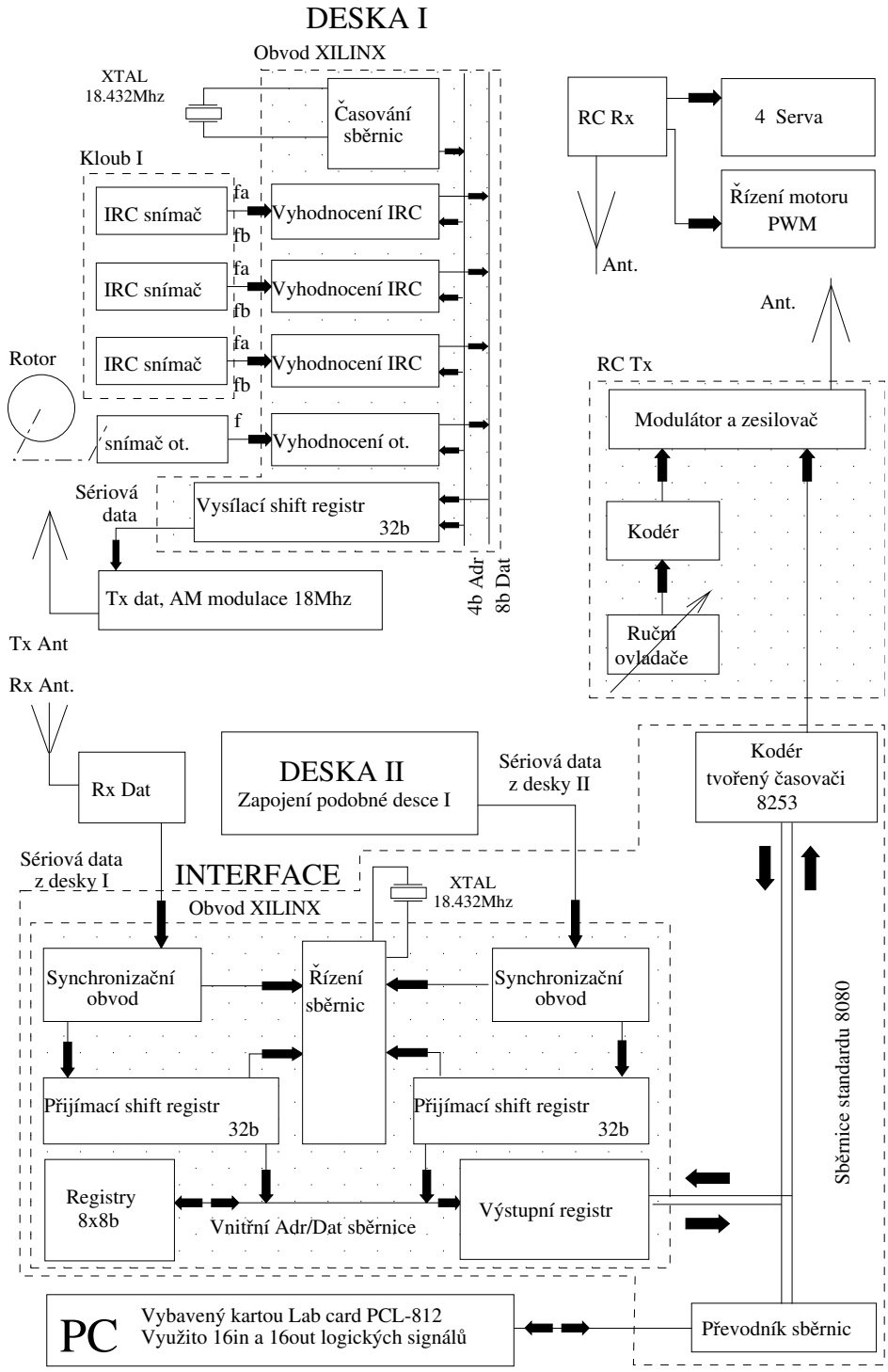


Figure 32: Blokové schéma celé sestavy

7 Programovatelná pole Xilinx

Protože vytvářené zařízení je velice obvodově náročné, rozhodl jsme se použít moderních velkokapacitních programovatelných logických polí firmy XILINX. Následující odstavce mají dát základní informace o obvodech XILINX a způsobu jejich programování. V zařízení jsou osazeny tři obvody XILINX stejného typu. Protože se jejich základní zapojení shodují s doporučením katalogu, je popsáno použité základní konfigurační zapojení pro všechny najednou.

7.1 Co je to integrovaný obvod XILINX

Každý obvod Xilinx se navenek chová jako uživatelsky programovatelný LSI obvod podobně jako hradlové pole. Obvody Xilinx jsou snadno znovu přeprogramovatelné. Základní stavební jednotkou logického pole je konfigurační logický blok (CLB = configurable logic block), jehož zjednodušené vnitřní zapojení je na 7.1.

Firma XILINX přivedla na trh programovatelná logická pole s velkou integrací. Ekvivalentem každého obvodu je několik tisíc hradel. My jsme se rozhodli pro střední třídu těchto obvodů, kterou představuje řada XC3000. Tato řada je pro daný účel vhodná jak svými možnostmi tak i svou cenou (600 až několik tisíc Kč).

7.2 Vnitřní struktura obvodu

Kombinační logika je umístěna do malých tabulek, z nichž do každé vede několik vstupů z vnější sběrnice (z hlediska jednoho bloku). Výstup může být připojen buď ke vstupu D-klopného obvodu, k další logice nebo k výstupu z obvodu. Obvod obsahuje matici totožných logických bloků, která se dělá nejčastěji čtvercová. Její rozměry jsou závislé na typu obvodu a pohybují se v rozmezí od 8x8 do 32x32.

Většina pinů může být obousměrná s výjimkou napájení a několika dalších vyhrazených pinů. Programovatelně lze nakonfigurovat elektrické parametry vstupů pro logiku TTL nebo CMOS. Vstupy mají hysterezi realizovanou Schmittovým KO.

Každý obvod má globální signál RESET, který vynuluje všechny interní KO. Signál RESET je přiveden na vyhrazený pin. V řadě XC4000 může být libovolný pin nakonfigurován jako globální RESET.

Řada XC3000 je osazována do pouzder s 64 vývody a větších. Pro zavedení externích vstupů a výstupů (z pouzdra) je možno použít téměř všechny vývody. Každý vývod je opatřen vstupně výstupním blokem (IOB = input output block). Výstup může být veden přímo nebo přes klopný obvod, je možné také použít třístavového výstupního obvodu a případně připojit Pull-Up rezistor. Vstup může pracovat jak samostatně tak i současně s výstupem (obousměrná třístavová sběrnice). Vstup je opět možno zapsat do klopného obvodu či latche. IOB neumožňuje realizovat žádnou logickou funkci s vyjímkou invertování některých signálů. IOB obklopují dokola

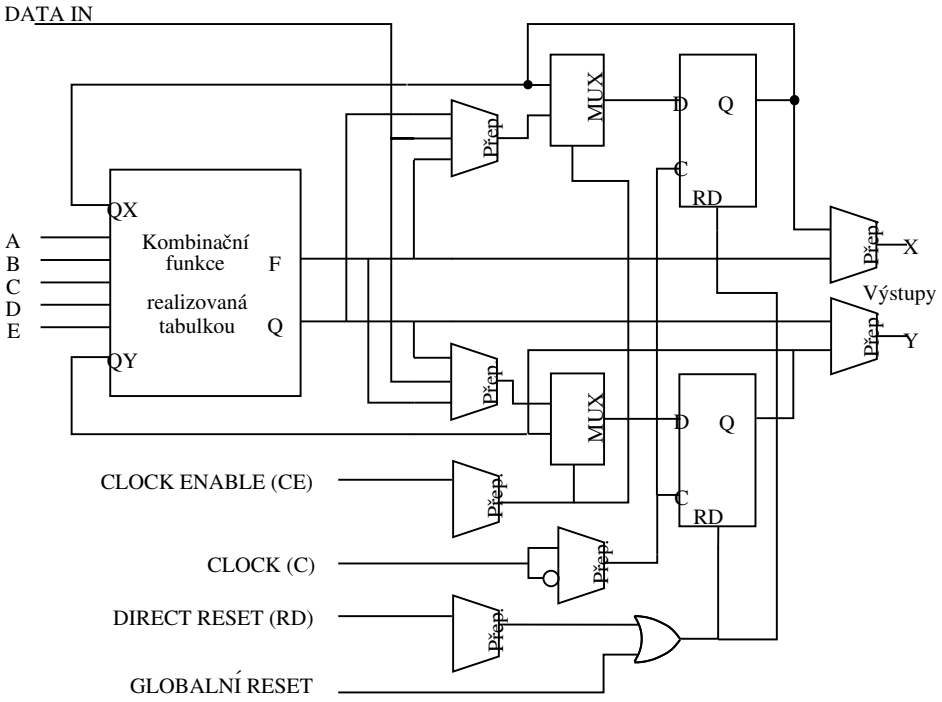


Figure 33: Zapojení konfiguračního logického bloku (CLB)

matici CLB. Vstupní a výstupní signály jsou volitelně jak v úrovních logiky TTL tak i logiky CMOS (napájené z +5V).

Propojení mezi bloky zajišťují vodiče a propojovací pole v mezerách mezi jednotlivými bloky. Za zmínku stojí takzvané dlouhé spoje (LL = long lines). Vertikální LL slouží pro rozvod hodinových signálů a signálů RESET. Vertikální LL jsou opatřeny horizontálními spojkami, a tak mohou jednoduše vytvořit rozvod po celém obvodu. Horizontální LL je vždy po každé straně CLB (zhora a zdola). Každý CLB má tak přístup ke dvěma horizontálním LL přes třístavové oddělovače. Horizontální LL umožňují v obvodech XILINX vytváření interní třístavové sběrnice. Horizontální LL mohou být opatřeny Pull-Up rezistory. V případě, že jimi opatřeny nejsou, pamatují si poslední stav.

Je využita i možnosti vytvořit v obvodu XILINX krystalový oscilátor pouhým připojením externího krystalu k příslušným vývodům.

7.3 Nahrávání programu

Pro uchování programu slouží speciální paměť CMOS-SRAM, jejíž klidový odběr je velmi malý. Spotřeba obvodu se dynamicky mění v závislosti na výkonu, množství propojených vstupů/výstupů a počtu propojených vnitřních uzlů. Obvod lze snížením napájecího napětí uvést do klidového stavu. Všechny piny jsou automaticky uvedeny do 3 stavu a obvod uchovává pouze informace v interní paměti.

Po přiložení napájecího napětí se obvod snaží nahrát do paměti konfigurační program. Jako zdroj programu může sloužit buď sériová paměť, standardní paměť typu PROM nebo sériová linka.

Na realizovaných zařízeních jsme zvolili dvě varianty současně. Sériová linka z počítače je podporována pro ladění vnitřní struktury obvodu. Definitivní podoba zapojení je nahrána do sériové paměti PROM. Nahrávání dat ze sériové paměti představuje nejjednodušší způsob konfigurace logického pole. Použití paralelního nahrávání nepřináší podstatné zrychlení počáteční inicializace. Obvod si totiž interně paralelní data překódovává na sériová.

7.4 Předdefinované části schématu

Pro usnadnění návrhu vnitřní struktury jsou na vyšší úrovni používány klasické schématické prvky jako např. logická hradla, čítače, posuvné registry atd. Několik jednoduchých prvků bylo vybráno jako primitiva. Všechny ostatní prvky jsou definovány jako makra složená z primitiv. Při překladu musí mít překládající program k dispozici definiční soubory popisující makra použitá v projektu.

Pro představu o možnostech programovatelného logického pole jsou v následující části textu podrobně rozebrána použitá primitiva.

Mód	M2	M1	M0	Popis
Slave	1	1	1	Sériový zápis s externím řízením je užíván pro nahrání z počítače či při kaskádovém zapojení více obvodů.
Master-serial	0	0	0	Sériový zápis při němž sběrnici řídí sám obvod XILINX je používán pro čtení ze sériové konfigurační paměti PROM (EEPROM).
Peripheral	1	0	1	Paralelní zápis s externím řízením pro zápis z počítače
Master-parallel H	1	1	0	Paralelní čtení konfigurace z běžných pamětí EPROM řízené obvodem XILINX. Čte od horních adres.
Master-parallel L	1	0	0	Paraelní čtení konfigurace z běžných pamětí EPROM řízené obvodem XILINX. Čte od spodních adres.

Table 6: Módy konfigurace obvodu XILINX

7.4.1 Primitiva kombinačních funkcí

Řada XC3000 podporuje kombinační funkce až do 5 vstupních proměnných. Jsou implementována hradla AND, NAND, OR, NOR, XOR, XNOR se 2,3,4 a 5 vstupy. Navíc může být kterýkoliv ze vstupů invertován (s výjimkou hradel typu XOR a XNOR). Dále do této kategorie patří primitiva BUF, INV a TBUF. TBUF a BUF jsou použity pro napájení vnitřních sběrnic. Několik primitiv kombinačních funkcí je nakresleno na 7.4.1.
Symboly jsou pojmenovány podle následující konvence:

hradlo[počet_vstupů[B[počet_výstupů]]]

např. hradlo AND2B1 představuje hradlo AND s jedním invertovaným vstupem.

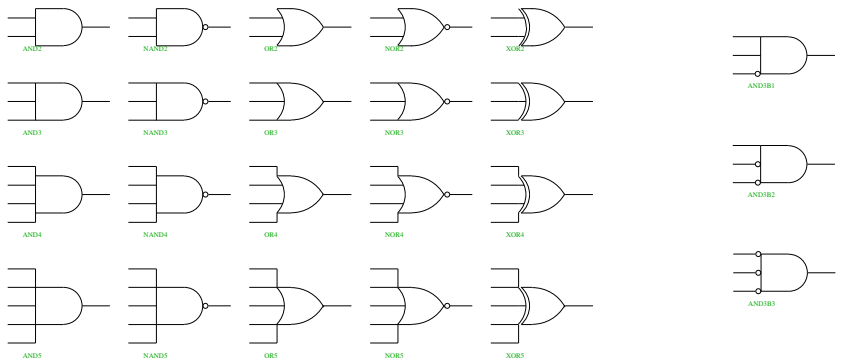


Figure 34: Ukázka některých vybraných kombinačních hradel

7.4.2 Primitiva vstupů a výstupů

Pomocí kombinace symbolu vstupního/výstupního pinu a bufferu je provedena konfigurace jednoho IO bloku. Každý signál z pinu musí být připojen přes symbol bufferu. Buffer lze použít pouze pro jediný signál. Vstupně výstupní primitiva jsou nakreslena na následujícím obrázku 7.4.2.

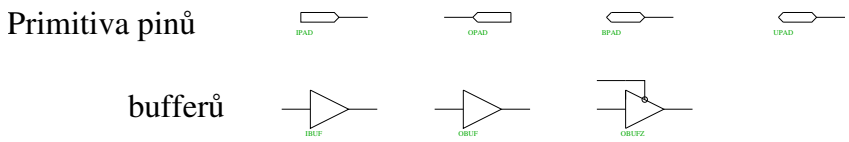


Figure 35: Primitiva vstupů a výstupů

7.4.3 Primitiva klopných obvodů

Každý CLB umožňuje umístění dvou klopných obvodů. Žádné z primitiv nemá přímo vyveden asynchronní vstup SET. Polarita hodinového signálu C je volitelná. Při použití invertoru před hodinovým vstupem program *XNFMAP* automaticky vyjme invertor a změní polaritu hodinového vstupu. Pokud zůstane asynchronní signál RESET nepřipojen, je celý klopný obvod pro jistotu ze schématu vyřazen. Dostupná primitiva KO

jsou nakreslena na 7.4.3.

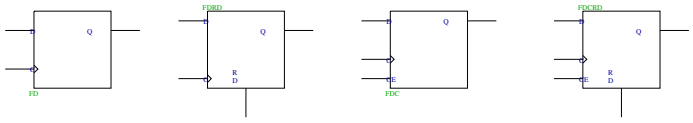


Figure 36: Základní primitiva KO

7.4.4 Primitiva oscilátoru a hodinových bufferů

K dispozici jsou dvě primitiva hodinových bufferů GCLK a ACLK. Ty odpovídají globálnímu a alternativnímu rozvodu hodin v LCA. Každé z nich může být použito v návrhu pouze jednou. Vždy je vhodné použít GCLK nebo ACLK pro nejvyšší rozváděný hodinový kmitočet.

Primitivum oscilátoru představuje vnitřní vf invertující zesilovač, který je použitelný pro realizaci krystalového oscilátoru. Pro oscilátor jsou rezervovány dva piny XTAL1 a XTAL2. K výstupu oscilátoru (je-li použit) musí být připojen vstup ACLK bufferu. V jednom LCA obvodu je dostupný pouze jediný oscilátor.

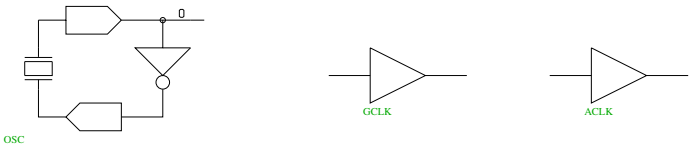


Figure 37: Primitiva oscilátoru a hodinových bufferů

7.5 Programování obvodu Xilinx

Navržené schéma je potřeba nějakým způsobem zkonvertovat do bitového pole akceptovatelného obvodem. K obvodům je dodáváno vývojové prostředí realizující automatický návrh a umísťování schématu do vnitřní logiky. Vstupním bodem sady optimalizačních programů je soubor .XNF

obsahující veškeré informace o navrhované logice včetně typu obvodu a umístění nožiček.

Konverze schémat je umožněna z následujících formátů: DASH-LCA, Schema II+, OrCAD, Daisy, Mentor, Valid, ViewLogic, PCAD, Case. Bylo by sice možné vytvářet přímo soubor fyzických uzlů a jejich spojování přes hradla, ale tato metoda postrádá grafickou reprezentaci schémat. Při rozsáhlejších projektech by mohlo dojít ke ztrátě konzistence celého schématu a případné chyby by se hledaly jen velmi obtížně.

Nejdostupnějším grafickým editorem schémat na naší škole je OrCAD, a proto jsme se rozhodli pro jeho využití při kreslení zdrojových schémat.

7.6 Způsob překladu schématu

Překlad vyžaduje provedení většího množství kroků. Nejprve je potřeba zkonvertovat schéma ve grafické podobě do struktury uzlů a spojů mezi nimi (soubor .PIN). Pak je potřeba rozložit složitější součástky (čítače, multiplexery) na primitiva podle knihovních souborů. Přibližný způsob překladu je vystižen na 7.6.

Následuje popis jednotlivých kroků při překladu. Vzhledem k obtížné dostupnosti literatury je každý použitý program podrobně rozebrán. Příkazy jsou popisovány v tom pořadí, v jakém jsou při překladu volány. U každého příkazu je uvedeno i několik dalších možných parametrů.

Uvedené informace by měly posloužit pouze pro toho, kdo bude vyžadovat drobné úpravy ve vnitřním zapojení obvodů. Pak nebude muset pracně řadit jednotlivé programy a může se věnovat pouze překladu upravených schémat. Navíc byly některé korekční programy dodatečně napsány autorem této diplomové práce.

7.6.1 Cleanup

Spuštění: *CLEANUP %1.SCH*

Program zabezpečí odstranění duplicitního umístění součástek, spojových čar a návěští ve schématu. Ty mohou být nakresleny vícekrát přes sebe, což normálně na obrazovce není vidět. V lepším případě je hlášena chyba v některé další fázi překladu. Někdy uvedená skutečnost může způsobit naprosto nečekané chování navrhnutého zařízení.

Parametry:
- /h Schémata vícekrát volaná v komplexní hierarchii se zpracovávají jen 1x.
- /r Více průchodů. Tato volba je vhodná pro velká schémata.

7.6.2 Annotate

Spuštění: *ANNOTATE %1.SCH /M/U*

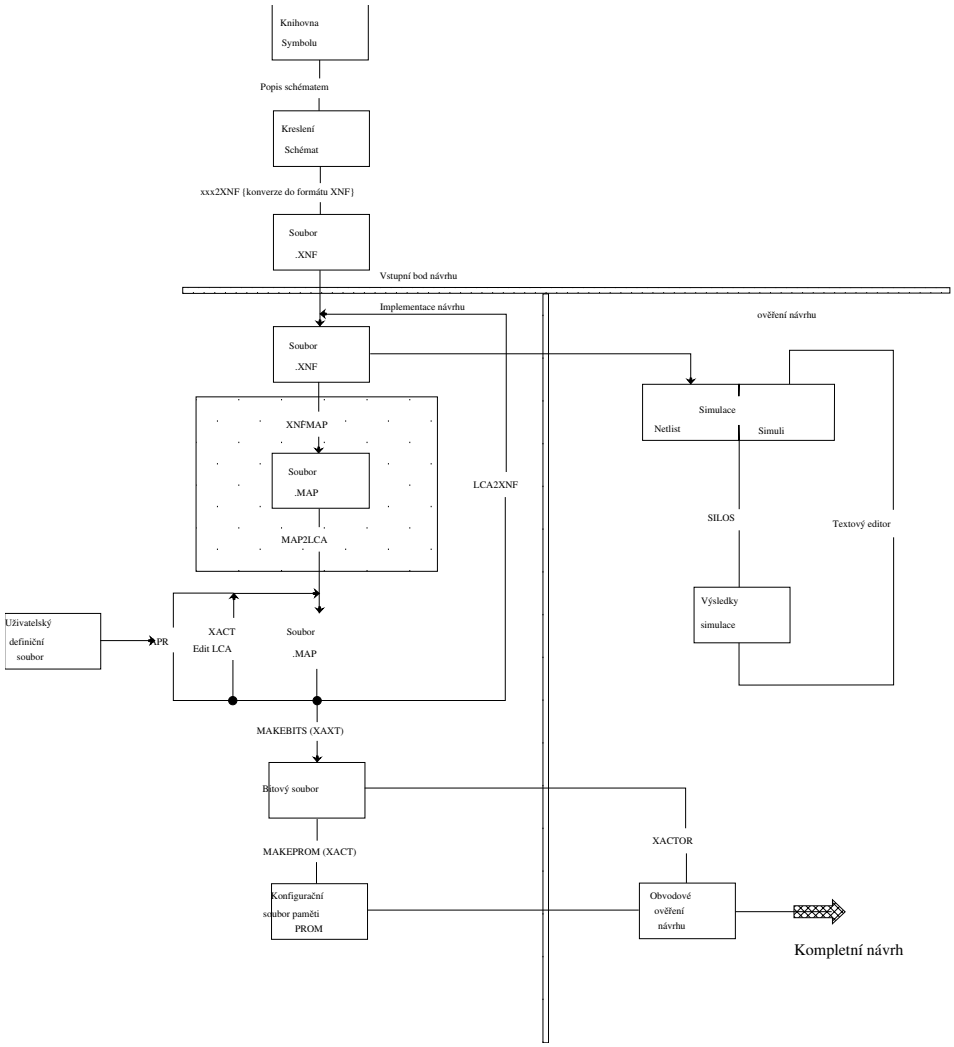


Figure 38: Struktura datových toků při překladu schématu

Prohlíží buď hierarchie schémat nebo navzájem související soubory na stejné úrovni (FLAT soubory) a přepisuje reference součástek. Reference součástek je možno aktualizovat buď přírůstkově nebo bezpodmínečně. Anotovaný soubor je možno buď použít pouze pro další programy nebo je možno s ním pracovat místo původního schématu.

Reference jsou použity v dalších krocích algoritmu pro označení součástek jako jejich jména. Při zpracování hierarchických schémat tímto programem je na aktuální disk nahráno větší množství swap souborů. Je potřeba si dát pozor na dostatečnou kapacitu (několik Mb) a na rychlost anotace. Při pokusu anotovat soubor na disku vzdáleném přes 4 síťové segmenty trval celý proces kolem půl hodiny. U ostatních programů tento problém nenastává.

Parametry: /d Vstupuje i do podschémat, které mají ve schématu vytvořenou grafickou značku.

/h Spouští v každém schématu číslování referencí od 1. Využívá se u hierarchických schémat.

/m Přesměruje výstupní soubor do vstupního schématu.

/o Zpracuje pouze zadané schéma bez hierarchického řazení.

/u Bezpodmínečná změna referencí. Změní se i ty reference, které už byly uživatelem ručně zadány.

7.6.3 Ercheck

Spuštění: *ERC %1.SCH /s*

Vykonává kontrolu zapojení a výpis chybových hlášení v případě nalezení nezapojeného vstupu, uzeměného výstupu, zkratovaného zdroje atp. Zjišťuje také chybějící návěští vodičů vstupujících do sběrnice a nezapojené Module porty.

Výstupní chybová hlášení je potřeba prověřit, protože mohou vzniknout i při správném návrhu zapojení.

Parametry: /l Vytváří seznam všech návěští.

/s Provádí v celé hiererchii schémat kontrolu shody I/O bodů (Module porty).

/u Výpis nepřipojených vývodů.

7.6.4 Netlist

Spuštění: *NETLIST %1.SCH %1.PIN FUTURENET /S*
NETLIST %1.SCH %1.PIN VSTMODEL /S

Vytvoří seznam propojení mezi jednotlivými komponentami, který je vhodný pro další zpracování. V podstatě se jedná o výstupní bod OrCADu. V našem případě je pak spojový .PIN konvertován do tvaru .XNF akceptovatelným vývojovým prostředím Xact.

Parametry: /o Zpracuje se pouze zadané schéma bez hierarchického řazení.
/u Zapíše seznam nezapojených výstupů do .NC souboru.
/q Vypne generaci výpisů při nahrávání.
/s<formát> Výstup pro program PCB.

7.6.5 Pin2Xnf

Spuštění: *PIN2XNF −O −P 3042pc84−70 %1*

Provádí konverzi souboru ve formátu .PIN do formátu .XNF. Při konverzi umožňuje rozložení hierarchické struktury podobně jako program XNFMerge. Pouze jednotlivé dílčí elementy musí být ve formátu .PIN. Tohoto způsobu je využito při práci s bloky, uloženými v knihovnách OrCadu. Při tomto způsobu práce není nutno schémata z knihoven znovu překládat.

Jedná se pravděpodobně o starší verzi programu. Výsledný vygenerovaný soubor .XNF je nekompatibilní s ostatními programy. Z tohoto důvodu musí být dodatečně upraven programem CorrXno. Protože program neumožňuje definovat adresář s popisy jednotlivých bloků, musí být všechny použité .PIN soubory nakopírovány do stejného adresáře, což značně komplikuje překlad schémat.

Parametry: -p <typ> Umožní definovat cílový obvod.

-m Provedení rozkladu hierarchicky řazených částí schématu. Nelze-li popis symbolu nalézt, pak je na obrazovku vypsáno varování.

-x Rozložitelné symboly budou rozkládány až v dalších fázích překladu. Parametry x a m se navzájem vylučují.

-o Použito pro FutureNet pinlist soubory vytvořené v OrCadu.

7.6.6 CorrXno

Spuštění: *CORRXNO %1.XNF %1.XXX %1.VST %1.PIN*

Uvedený program není součástí programového balíku OrCad ani Xilinx. Byl dodatečně napsán a byl vložen do této fáze překladu pro odstranění nekompatibilit mezi verzemi jednotlivých programů. Program je napsán v Pascalu a jeho výpis je uveden v příloze.

Protože OrCad ve verzi 3.0 není schopen do formátu .PIN vyexportovat informace o umístění nožiček na schématu, musí být tyto informace získány z přídavné tabulky. Všechny důležité informace pro tvorbu tabulky jsou obsaženy pouze ve formátu .VST. Z tohoto důvodu program umožňuje i automatické vytvoření tabulky ze souboru .VST.

Soubor *.XXX představuje vlastně opravený .XNF soubor. V tomto typu souboru však nejsou již uvedena označení jednotlivých pinů ve schématu, ale pouze čísla externích signálů. Pozice pinu je reprezentována atributem jednotky pinu (z hlediska OrCadu Part field). Signál spojující pin s vnitřní logikou má své pevné jméno, které je obsaženo v souboru .PIN. Podle tohoto jména signálu pak lze jednoznačně určit ke kterému pinu je

připojen a přiřadit mu správnou pozici. Pokud není pozice pinu překladači známa, zvolí si ji náhodně.

Další zjištěnou nekompatibilitou je špatná reprezentace invertovaných signálů. Programem je opravena i tato nekorektnost v souboru .XNF. Pokud by zůstal .XNF soubor v původním tvaru, je činnost dalších fází překladu znehodnocena. Podrobnější informace o struktuře jednotlivých souborů jsou uvedeny v kapitole: *Formáty a struktury datových souborů*.

Parametry: xxx.XNF Opravovaný vstupní soubor.

xxx.XXX Výstupní opravený soubor.

xxx.PIN Soubor s dodatečnými informacemi potřebnými pro korekci.

xxx.VST Soubor s informacemi o umístění vnějších pinů.

7.6.7 XNFMerge

Spuštění: *XNFMERGE %1 $*

Odstraňuje hierarchickou strukturu v návrhu při zachování identického chování celé struktury. Program přihrává XNF nebo MAP soubory z hierarchicky nižších úrovní do jediného souboru, který neobsahuje žádné odkazy na jiné soubory MAP nebo XNF.

Nejprve je načten a prohledán soubor popisující nejvyšší úroveň hierarchické struktury. Jako rozložitelný je považován každý symbol jehož jméno není obsaženo v množině primitivních symbolů. Po nalezení složeného symbolu, je vyhledán soubor s popisem jeho struktury. Je-li nalezen současně jak .XNF tak .MAP soubor pro popis jednoho elementu, je vybrán nejnovější z nich.

Při rušení hierarchické struktury dochází k modifikaci jmen signálů a symbolů v návrhu. Kde je to možné, tam zůstávají původní jména signálů v kombinaci s prefixem. Tímto je odstraněn konflikt jmen, ke kterému by docházelo při vícenásobném použití jednoho souboru. Pouze jména v nejvyšší hierarchické vrstvě zůstávají zachována.

Parametry: vstupní_soubor

výstupní_soubor

-d <adresář> Jméno adresáře s podřízenými soubory. Lze použít i vícenásobně.

-p <typ> Umožní definovat cílový obvod.

-q Používání nerozložených souborů. Není-li tento parametr zadán, pak je na obrazovku vypsáno varování.

-x Zpracování pouze souborů XNF. MAP soubory jsou ignorovány.

7.6.8 XNFDRC

Spuštění: *XNFDRC $*

Kontroler pravidel návrhu projektu je určen pro nalezení a identifikaci možných problémů na počátku překladu. Provádí také test konzistence

souboru .XNF a vyjímání 'mrtvé' logiky z celého projektu (např. hradlo s nezapojeným výstupem).

Při běžném překladu je obyčejně hlášeno velké množství různých varovných hlášení. Ty pouze upozorňují na možné zdroje chyb, nebo doporučují lepší techniku návrhu, ale struktura zapojení není jakýmkoliv způsobem transformována.

Parametry -p <typ> Umožní definovat cílový obvod.

-o <file> Výstupní soubor obsahující všechna hlášení. Implicitně má jméno kontrolovaného souboru s koncovkou .DRC.

7.6.9 XNFMAP

Spuštění: *XNFMAP −N $*

Program *XNFMAP* přiřadí logiku definovanou v souboru XNF do jednotlivých LCA elementů (CLB, IOB a TBUF). Soubor .XNF představuje vstupní bod programu. Na výstupu je vytvořen .MAP soubor, který popisuje rozčlenění logiky. Dále program *XNFMAP* generuje soubor .CRF, který obsahuje seznam křížových referencí, informace o vyřazené nepoužité logice, nesprávně použitých symbolech a shrnující informace o návrhu.

V každém CLB bloku jsou obsaženy dva klopné obvody (dále jen KO). Pro minimalizaci počtu vnějších spojů je nejvýhodnější provést sloučení dvou vzájemně vázaných KO (např. dělič 8). Program *XNFMAP* porovnává jména jednotlivých KO. Pro porovnání je z konce názvu odtržena poslední číselná část s výjimkou znaku '_' (např. REGA09 a REGA10 budou sloučeny). Vyšší prioritu při párování mají přímé požadavky na umístění KO do předem definovaného CLB.

Pro číslování bloků používá OrCad dvojici čísel oddělenou '_'. První číslo udává pořadí elementu ve schématu a druhé definuje číslo schématu v hierarchii. Proto budou sloučeny KO s označením REG34_20 a REG35_20.

Informace o umístění LCA elementů lze uchovat v průvodním souboru .PGF a použít pro další iteraci návrhu projektu. Při každém spuštění programu je soubor .PGF obnovován a korigován. Pro zanesení informací ze souboru .LCA je nutno nejprve použít program *LCA2XNF*. Použití průvodního souboru však musí být povoleno parametrem -k a program by měl nalézt soubor .AKA popisující křížové reference mezi původními a zkrácenými jmény signálů.

Parametry: -e Zakáže vyjímání nepoužité logiky. Vhodné pro odhad počtu vyjmutých CLB.

-f Vytváření hustšího návrhu. Může způsobit větší počet nepropojených pinů v další fázi překladu.

-i Umožní přímé řízení vstupů KO ze vstupních pinů.

-k Napodobuje rozmístění LCA elementů podle .PGF souboru z předchozí iterace.

-p <typ součástky> Umožňuje předefinovat popř. zvolit typ použitého obvodu.

7.6.10 MAP2LCA

Spuštění: *MAP2LCA $*

MAP2LCA překládá .MAP soubor do pole logických jednotek (LCA Logic Cell Array), které je využíváno vývojovým prostředím XACT. Program vykonává poslední krok konverze XNF->LCA.

Výstupem je .LCA soubor, který popisuje rozdělení schématu do vnitřních bloků. Neobsahuje však zatím informace o vzájemném propojení jednotlivých bloků. Tento soubor je možno nahrát a dále ručně zpracovávat programem XACT, nebo provést automatickou optimalizaci umístění a spojování bloků programem APR. Dalším výstupním souborem je upřesňující soubor .SCP, který popisuje omezení pro umísťování a spojování bloků (např. specifikaci umístění jednotlivých pinů).

Parametry: /a Zakáže vytváření .AKA souboru, který obsahuje hierarchická jména symbolů a signálů.

/i Přikáže programu, aby ignoroval všechna omezení nalezená v .MAP souboru.

/p <typ součástky> Umožňuje předefinovat popř. zvolit typ použitého obvodu.

7.6.11 CorrLca

Spuštění: *CORRLCA $.lca $$.lca*

Tento program podobně jako *CORRXNO* byl napsán dodatečně a není součástí standardního balíku. Je zapsán v jazyce Pascal a jeho výpis je uveden v příloze.

Při automatické konverzi schémat do prvků LCA jsou programem *MAP2LCA* do souboru LCA přidány i upřesňující informace o rozložení třístavových budičů TBUF. Umístění těchto budičů je velmi omezeno vnitřní strukturou obvodu. Z každého bloku CLB mohou vycházet pouze dva budiče, které mohou být připojeny pouze k některým Long line. Program *APR* je schopen si takovéto rozmístění sám poměrně uspokojivě navrhnout.

Pokud má však zadánu pevnou pozici budičů, nemůže být měněna jejich pozice. Při menším počtu sběrnic uvedená skutečnost způsobí pouze horší umístění, které se na vnější funkci vůbec neprojeví. Při větším počtu však docházelo k "záhadným" chybám, kdy při přidání jednoho invertoru schéma najednou nebylo možno programem APR přeložit.

Na výstupu se v takovém případě objevilo hlášení "*Invalid block name TBUF=xx*". Nabízí se jednoduché řešení provést inicializaci parametrem */i* u programu *MAP2LCA*. Překlad proběhne bez jakýchkoliv chyb, avšak všechny informace o pevném umístění nožiček jsou definitivně ztraceny.

Při nahlížení do souboru .LCA byly nalezeny položky nesoucí informace o umístění třístavových budičů. Program *CorrLCA* takovéto položky automaticky v souboru .LCA vyhledává a následně ruší. Jejich uvedení

totiž není ve formátu LCA povinné. Po korekci pak může vlastní překlad proběhnout bez jakýchkoliv problémů.

Parametry: vstupní_soubor
výstupní_soubor

7.6.12 APR

Spuštění: *APR −w −A 99 $ %1*

Zautomatizuje většinu (v mnoha případech všechnu) práci s umísťováním a vzájemným spojováním bloků v .LCA projektu. Pro zlepšení optimality návrhu provádí při své práci cyklicky metodu simulovaného žíhání.

Nejprve je načten projekt uložený v .LCA souboru, který může nebo nemusí obsahovat informace o vzájemném spojení bloků. Poté je vytvořeno nové umístění bloků, spojů a uzlů v návrhu, které je zapsáno do jiného .LCA souboru. Při dokončení operace je vytvořen soubor .RPT, který obsahuje informace o provedených činnostech.
Překlad je prováděn ve třech po sobě následujících fázích:
Simulované žíhání (annealing):

V této části je spotřebována většina času. Činnost algoritmu lze ovlivnit nastavením počáteční a koncové teploty žíhání, rychlostí ochlazování a semínkem generátoru náhodných čísel. Při nastavení nízké výchozí teploty podobně jako při vysoké koncové teplotě většinou nebude nalezeno optimální rozmístění bloků. Není-li přímo specifikována rychlost ochlazování, je vypočítána optimální hodnota v rozmezí 5% až 50%.

Implicitně jsou parametry nastaveny tak, aby výsledný návrh byl optimální. Pro potřeby ladění, pokud je využito méně než 50% CLB je velmi výhodné čas podstatně zkrátit (při plné optimalitě se může na stroji 486/33 jednat i o hodiny). Zkrácení lze provést nastavením počáteční teploty na nulovou hodnotu (tím se žíhací fáze úplně přeskočí) nebo zvýšením rychlosti ochlazování.
Vyhlazení (quenching):

Představuje druhý krok fáze umísťování bloků. V této fázi jsou prováděny jen malé změny výsledného návrhu. Odpovídají nízké konstantní teplotě žíhání.
Spojování (routing):

Provádí vzájemné spojování bloků, jejichž pozice se již nemění. Router je opakovaně spouštěn pouze v případě, že se nepodaří pospojovat všechny signály. Celkem jsou k dispozici 3 routery:

r3 Router optimalizující zpoždění. Je ze všech všech tří nejpomalejší (jedná se o zlomek doby žíhání). Většinou nachází optimální spojení již při první iteraci a větší množství iterací již výsledek nezlepší.

r2 Optimalizuje spojení. Pokud *r3* nemůže vzájemně spojit méně než 5 uzlů, je vhodné ještě zkusit *r2*.

r1

Nejrychlejší router. Pro úspěšnou činnost potřebuje, aby bylo nejméně 20% CLB volných.

Parametry: -a počet Specifikuje počet pokusů o spojení jednotlivých bloků. Implicitní hodnota je 3.

-b teplota Zadání výchozí teploty pro algoritmus simulovaného žíhání.

-f Použití rychlého algoritmu žíhání. Zmenší se počet provedených přesunů a tím i celková doba činnosti.

-k rychlost Nastaví rychlost ochlazování procesu simulovaného žíhání.

-q Zakáže žíhací fázi překladu a přeskočí rovnou do vyhlazovací fáze.

-r router Definuje typ routeru.

-w Přepíše výstupní soubory bez předchozího varování.

7.7 Formáty a struktury datových souborů

V této části jsou rozebrány struktury jednotlivých formátů používaných pro reprezentaci elektrických schémat. Schéma nakreslené v OrCadu prochází při svém překladu do logického pole postupně konverzními programy přes jednotlivé formáty. Diskutovanou konverzi zachycuje obrázek: 7.6 - '*Struktura datových toků při překladu schématu*'.

V ideálním případě by se uživatel nemusel vnitřní strukturou níže popisovaných formátů vůbec zabývat. K něčemu takovému však dochází jen velmi zřídka. Uvedené informace byly získány metodou zpětného inženýrství při hledání chyb ve špatně přeloženém schématu.

Jednotlivé formáty nejsou dostatečně standardizovány a jsou u každé nové verze modifikovány. Případná nekompatibilita pak může způsobit zastavení překladu nebo v horším případě dojde k zanesení chyb do výsledné matice spojů programovatelného logického obvodu. Popisované chyby jsou pak velmi špatně zjistitelné a projevují se nekorektním chováním navrhovaného zařízení.

Testovací schéma podle 7.7 pro ukázku jednotlivých formátů bylo zvoleno co možná nejjednodušší. Z výše uvedených důvodů bylo nutno detailně se zabývat strukturou jednotlivých formátů a provádět jejich korekci do tvaru vyžadovaného na vstupu programu realizujícího další krok překladu. Na jednoduchém pokusném schématu je proveden rozbor jednotlivých formátů. Schémata pro čtení IRC snímačů a vysílání sériového kódu nejsou použita pro jejich složitost, která by zastínila nejdůležitější informace. Uvedené informace by měly posloužit každému, kdo zkouší překlad schémat z OrCadu.

7.7.1 Formát .SCH

Je základním interním formátem OrCadu. Má binární podobu a jeho přímá editace by byla velmi obtížná. Navíc při jakékoli chybě v souboru .SCH dojde ke zhroucení OrCadu a možnost opravy je tímto znemožněna. V

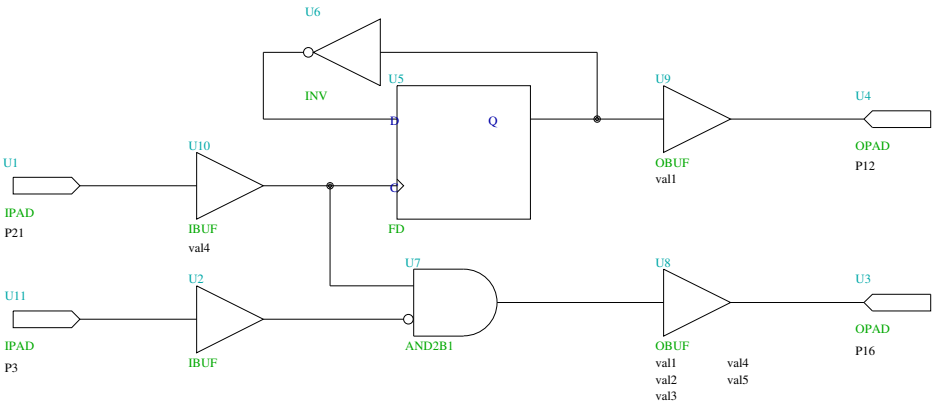


Figure 39: Testovací pokusné schéma

rámci jedné verze je dosažena 100% kompatibilita a pro nižší verze jsou distribuovány více či méně úspěšné převodní programy. K hlavním problémům dochází při používání jednotlivých knihoven. K obvodům Xilinx je distribuováno více takovýchto knihoven a jenom na naší škole se vyskytují dvě odlišné verze. Při použití jiné knihovny, která má symboly s jinou velikostí pak nedojde k vzájemnému propojení vodiče a součástky. Propojení je totiž interně kontrolováno pouze přes dotek dvou vodičů. Uvedená skutečnost si následně vynutí překreslení celého schématu, i když vodiče zůstanou na stejném místě (pokud se nepodaří sehnat původní knihovnu).

Soubory v knihovnách představují makra, jejichž vnitřní struktura musí být v dalších fázích překladu známa. Pro některé symboly v knihovně se nám nepodařilo definiční soubory nalézt. Takový symbol je pak nutno rozložit na jednodušší části nebo použít jiný s podobnou funkcí.

7.7.2 Formát .VST

Přes tento formát neprochází hlavní větev překladu. Byly však otestovány všechny možné výstupní formáty programu Netlist a pouze do VST modelu jsou uloženy přídavné informace k součástkám – tzv. *Part field*.

Informace o součástkách jsou uloženy v textové podobě. Každá elementární součástka má v tomto modelu vyhrazen jednotlivý řádek. Jednotlivé řádky jsou odděleny znaky CR LF jak je běžné u textových souborů pro PC. Celý soubor je zakončen znakem % uvedeném na samostatném řádku.

Položka začíná označením typu součástky - *Part Value*. Po něm následují doplňkové informace uzavřené do kulatých závorek. Uvnitř závorky se nacházejí informace o vstupech. Jednotlivé vstupy s odlišným významem jsou od sebe odděleny čárkou. Za středníkem pak následují podobně uspořádané informace o výstupech. Další středník odděluje

výpis jednotlivých *Part field*. Referenční označení součástky pokračuje za uzavírací závorkou následovanou středníkem.

Informace o vstupech a výstupech jsou přebírány z knihovny. Tam však žádné nejsou definovány (pro další překlad nejsou potřeba), a proto si místo těchto dvou položek OrCad doplnil nějaká, z pohledu uživatele náhodná, čísla předcházená znakem '?'. Zajímavý je způsob práce s položkami *Part Field*. První definovaná položka se napíše na první místo. Za ní je oddělena čárkou až další definovaná položka. Z této struktury však už nelze zjistit, jaké měla položka původní místo. Pro popis pinů vyplňujeme pouze **jedinou** položku, čímž je uvedený problém z části eliminován.
Struktura souboru VST:

```
IPAD(;?4;P21);U1
IBUF(;?9,?8);U2
OPAD(?7;P16);U3
OPAD(?3;P12);U4
FD(;?2,?5,?1);U5
INV(;?1,?2);U6
AND(?5,?9;?6);U7
OBUF(;?7,?6;val1,val2,val3,val4,val5);U8
OBUF(;?3,?2;val1);U9
IBUF(;?5,?4;val4);U10
IPAD(;?8;P3);U11
%
```

Jak je z uvedeného příkladu vidět, ve formátu .VST není obsažena žádná informace o vzájemném propojení nakreslených prvků. Proto tento model může mít pouze pomocný význam pro přehled o všech konstrukčních prvcích použitých ve schématu.

7.7.3 Formát .XNF

Je vstupním bodem překladového systému Xact. Má z části objektově orientovanou strukturu. Objekt začíná svým klíčovým slovem následovaným čárkou. Po ní jsou uvedeny další parametry vzájemně oddělené čárkami. Každý objekt zabírá jednotlivý řádek. Externí signály (připojené k vnějším pinům) mají jako parametr uvedeno číslo signálu, typ signálu a volitelně i pevné umístění signálu parametrem *LOC=Pxx*, kde xx označuje číslo pinu. Celý soubor je ukončen klíčovým slovem *EOF*.

Pouze objekt prvku *SYM* je uložen na více řádcích. První řádek popisuje referenční označení nazývané jméno typu a jméno prvku (symbolu). Zatímco jméno typu je jedinečné, jméno symbolu je pro všechny elementy se stejnou funkcí stejné (např. hradlo AND). Za jménem symbolu může následovat parametr *FILE=* s informacemi o jeho vnitřní struktuře.

Další řádky pak začínají klíčovým slovem *PIN*. V každé řádce je pak po sobě uvedeno označení vývodu, typ vývodu, jméno spojovacího signálu a popř. označení invertovaného signálu *INV*.

Podporované typy signálů: I Vstupní
O Výstupní

T Třístavový
B,U Pro externí signály

Klíčové slovo *SIG* popisuje symbolické jméno pinu.
např.: SIG, A, PIN=A
Struktura je objasněna následujícím příkladem:

```
LCANET, 2
PROG, PIN2XNF, 2.40, Sat Jul 15 03:51:31 1995, Created from:
pokus1.pin
PART, 3020pc68−70
SYM, U7, AND
PIN, 1, I, 1_1−000005
PIN, 2, I, 1_1−000009,, INV
PIN, O, O, 1_1−000006
END
SYM, U5, DFF
PIN, C, I, 1_1−000005
PIN, Q, O, 1_1−000002
PIN, D, I, 1_1−000001
END
SYM, U2, IBUF
PIN, O, O, 1_1−000009
PIN, I, I, 1_1−000008
END
SYM, U10, IBUF
PIN, O, O, 1_1−000005
PIN, I, I, 1_1−000004
END
SYM, U6, INV
PIN, I, I, 1_1−000002
PIN, O, O, 1_1−000001
END
SYM, U8, OBUF
PIN, O, O, 1_1−000007
PIN, I, I, 1_1−000006
END
SYM, U9, OBUF
PIN, O, O, 1_1−000003
PIN, I, I, 1_1−000002
END
EXT, 1_1−000004, I,, LOC=P21
EXT, 1_1−000008, I,, LOC=P3
EXT, 1_1−000007, O,, LOC=P16
EXT, 1_1−000003, O,, LOC=P12
EOF
```

Starý soubor XNF Liší se od nové verze pouze jinou definicí invertovaných signálů. Označení invertovaného signálu je zakončeno písmenem 'B'. Chybí zde parametr *INV*.


```
SYM, U7, AND
PIN, 1, I, 1_1−000005
PIN, 2B, I, 1_1−000009
PIN, O, O, 1_1−000006
```

7.7.4 Formát LCA

Je základním formátem pro optimalizaci struktury programovatelného logického pole. Obsahuje již návrh rozdělení logiky do jednotlivých logických elementů. Může nebo nemusí obsahovat informace o routingu. Tímto formátem jsem se musel zabývat právě z tohoto důvodu. Při postupném překladu se do něho nějakým způsobem dostaly informace o pevném přiřazení tzv. longlines a program APR již nebyl schopen tyto informace změnit.

Třístavová hradla jsou označena řádkami *NAMEBLK TBUF xx*, kde *xx* značí jejich pevné umístění. Označení je umístěno na konci souboru a je s ním nakládáno podobně jako s pevným umístěním nožiček.

Je sice možno zadat inicializaci souboru LCA, ale pak se ztrácí informace o připojení vnějších nožiček.
Ukázka souboru LCA:

```
; LCA Design=$ Part=3020PC68 −− Blocks=6 Nets=4.
; Program=APR Version=3.30 Date=Sat Jul 15 03:52:21 1995.
Design 3020PC68
Speed −70
Programorder On
Addnet 1_1−000002 AE.Y P67.O
Netdelay 1_1−000002 P67.O 1.0
NProgram AE.Y:PAD10.O
Addnet 1_1−000005 P68.I AF.C AE.K
Netdelay 1_1−000005 AF.C 4.7 AE.K 4.3
NProgram BE.20.1.3 BE.20.1.8 BF.20.1.9 BF.20.1.16 row.B.local.5:AF.C
row.B.local.4:AE.K col.E.local.4:PAD9.I
Addnet 1_1−000006 AF.Y P64.O
Netdelay 1_1−000006 P64.O 2.5
NProgram col.G.local.5:AF.Y col.G.local.5:PAD13.O
Addnet 1_1−000009 P66.I AF.A
Netdelay 1_1−000009 AF.A 1.0
NProgram AF.A:PAD11.I
Nameblk AE 1_1−000002
Editblk AE
Base F
Config F:QY Y:QY X: DY:F DX: RSTDIR: CLK:K ENCLK:
Equate F = ˜QY
Endblk
Nameblk P67 1_1−000003
Editblk P67
Base IO
Config IN: OUT:O TRI:
Endblk
Nameblk P68 1_1−000004
```

```
Editblk P68
Base IO
Config IN:I OUT: TRI:
Endblk
Nameblk AF 1_1−000006
Editblk AF
Base F
Config F:C:A Y:F X: DY: DX: RSTDIR: CLK: ENCLK:
Equate F = (C*˜A)
Endblk
Nameblk P64 1_1−000007
Editblk P64
Base IO
Config IN: OUT:O TRI:
Endblk
Nameblk P66 1_1−000008
Editblk P66
Base IO
Config IN:I OUT: TRI:
Endblk
```

7.8 Použité obvody

Pro účely zpracování dat, jejich přenosu a spojení s počítačem se ukázaly obvody XILINX jako velmi vhodné. Protože jednotlivá místa zpracování jsou od sebe značně vzdálena, bylo nutno použít tří obvodů XILINX. Vzhledem k rozsahu zapojení byly z nabídky dodavatelské firmy ASIX vybrány obvody s polem 12x12 CLB v pouzdru PLCC84. Je to jeden obvod XILINX XC3042-PC84C-70 a dva obvody XC3042A-PC84C. Obvody se liší příkonem a mezním zpracovatelným kmitočtem. Jejich zapojení a další parametry se neliší. Protože na příkonu v naší aplikaci příliš nezáleží (v rozumných mezích) a mezní kmitočty jsou vyšší než je potřeba (70 a 130MHz), jsou tyto obvody z pohledu této práce zcela záměnné. Obvody XILINX jsou umístěny na deskách XV1, XV2 a XRI.

Jako konfigurační paměť se pro obvody XC3042 používá sériová paměť PROM s typovým označením XC1736DPD8C s obsahem 36288 bitů v pouzdru DIP8. Paměť musí být u každého ze tří obvodů XILINX.

7.9 Základní zapojení obvodů XILINX

Zapojení spočívá v propojení vývodů pro napájení, programování a případně i připojení externího krystalu. Toto zapojení je stejné pro všechny tři použité programovatelné obvody. Přehledně jsou požadavky na propojení zapsány v 7.8 a závěry z toho jsou jasně čitelné na schematech desek XV1, XV2 a XRI, které tyto obvody obsahují. Na schématech je XILINX označen jako IO1 a konfigurační paměť PROM jako IO2. Napájecí napětí pro obvody XC3042 a konfigurační paměti je 5V. V mém případě používáme dvou programovacích režimů. Režim slave serial mód (M1=H, M2=H, M3=H)

XILINX vývod	PROM vývod	Nahrávací konektor K4	Režim master Připojeno na	Režim slave Připojeno na
GND 1,43	GND 5	2	0V	0V
VCC 22,64	VCC 8	1	+5V	+5V
PWRDWN 12			přes 27kΩ na VCC	přes 27kΩ na VCC
M0 32			GND	VCC
M1 31			GND	VCC
M2 33			přes 5kΩ na GND	přes 5kΩ na VCC
HDC 34			NC (2)	NC
LDC 36			NC	NC
INIT 42			NC	NC
DIN 72	DATA 1	6		
CCLK 74	CCLK 2	4		
DONE 55	CE 4	5	přes 27kΩ na VCC a Tl2 na GND	přes 27kΩ na VCC a Tl2 na GND
RESET 54			přes 27kΩ na VCC a Tl1 na GND	přes 27kΩ na VCC a Tl1 na GND
DOUT 73			NC	NC
XTAL1 57			Externí krystal	
XTAL2 53			18,432 MHz	

Poznámka:

(1) Vývody, které nejsou uvedeny v tabulce mohou být použity jako uživatelské vstupy a výstupy.

(2) NC (no conection) označuje nepřipojený vývod pouzdra.

Table 7: Zapojení obvodů XILINX XC3042

se používá pro sériový zápis konfigurace do obvodu XILINX z počítače v průběhu odladování vnitřního zapojení. Při tomto režimu je konfigurační paměť vyjmuta a nahrávání probíhá přes konektor označovaný K4 jehož zapojení vychází z firemního nahrávacího kabelu. Druhý režim nazývaný master serial mód (M1=L, M2=L, M3=L) unožňuje nahrávání konfigurace ze sériové konfigurační paměti PROM IO2. O použitém režimu rozhodujeme propojkou SV1 a vyjmutím konfigurační paměti PROM.

Pro obsluhu je důležité mít možnost provést reset obvodu pomocí tlačítka RESET Tl1 a případně vyvolat znovupřečtení konfiguračního programu z paměti sočasným stisknutím tlačítka Tl1 (RESET) a tlačítka Tl2 (PGM). Obě tlačítka jsou miniaturní a osazená přímo v desce plošných spojů. Tlačítko RESET má dlouhý dřík hmatníku.

8 Napájecí zdroj

Všechny vrtulníky mají velkou spotřebu energie. Proto jsme museli věnovat konstrukci zdroje zvýšenou pozornost. Náš předchůdce měl k dispozici pouze zdroj 20A. Ten byl velmi přetížen. Následkem toho docházelo k podstatnému poklesu napětí, zdroj často vypadával a způsoboval zkraty v síťovém okruhu.

Protože ve vybavení laboratoře nebyl k dispozici zdroj vhodných parametrů, byly zakoupeny dva impulsní měniče 5V/50A vyrobené v ZPA Děčín a použity ke kompletaci zdroje. Při tak vysokých proudech je již impulsní zdroj téměř nutný.

8.1 Popis zdroje

Zdroje byly namontovány do standardní kovové bedny za účinné pomoci školních mechaniků. Zůstala otázka možného poškození zdrojů mechanickou úpravou. Z tohoto důvodu bylo přikročeno k jejich kontrole a přezkoušení.

Zdroje jsou připevněny ke konstrukci pomocí dvou úhelníků na protějších stranách. Jeden úhelník je připevněn ke zdroji pomoci šroubů držících přední panel. Na zadních chladičích zbylo volné místo pro výkonový tranzistor (čtyři otvory). Do jednoho otvoru byl vyříznut závit a pomocí šroubu je na tomto místě připevněn druhý úhelník.

Vypracování mechaniky je celkem přijatelné. Pouze jeden ze zadních šroubů byl natolik dlouhý, že způsoboval zkrat. Při použití kratších šroubů by již ke zkratu docházet nemělo.

Činnost měničů byla zkoušena do proudu 12A (více nebylo možné z důvodů velkého odporu použitých vodičů). Oba zdroje vykazovaly konstantní napětí 4.8V (údaj školního avometu 1). Z výsledku se dá usuzovat, že oba zdroje jsou funkční.

Zdroj 1	Typ: DC C 205	vč: 274496
Zdroj 2	Typ: DC C 205	vč: 277598

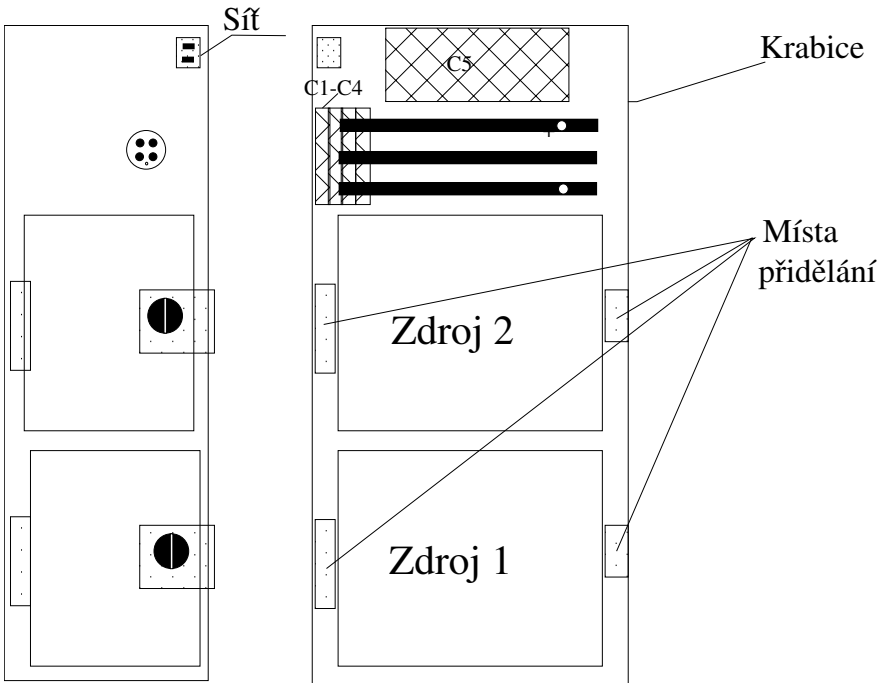


Figure 40: Rozmístění zdrojů 2x5V/50A

Zdroje byly elektricky připojeny ke konektorům na zadním panelu. Ten bylo nutno vyrobit přesně na míru. Byl přidělán síťový konektor typu 'střední žehlička', který lze používat do proudů asi 6A ze sítě, což v našem případě postačuje.

Uvnitř jsou napětí zdrojů přivedena na tři měděné profily o

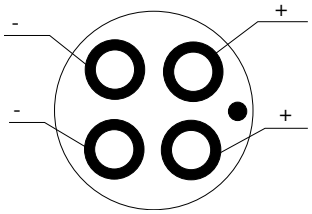


Figure 41: Výstupní konektor

průřezech 3x12. Do profilů jsou vrtány díry se závity a jsou výhodně využity pro připojení jednotlivých kabelů pomocí instalačních oček. Pro vývod napětí 10V byl použit kruhový konektor původně užívaný v letadlech pro robustní tvar jeho špiček. Přesto byly použity vždy dvojice špiček pro usměrněné napětí 10V. Zapojení konektoru ukazuje 8.1. Podle všech předpokladů by měl zdroj vydržet trvalé zatížení proudem kolem 50A (což bylo později ověřeno).

Z obavy před šířením rušivého elmg pole po přívodních vodičích byly do zdroje připojeny papírové kondenzátory s celkovou kapacitou 1μF pro každý zdroj zvlášť.

Protože použitý typ impulzních měničů nemůže pracovat naprázdno (vznikají velmi vysoká napětí ve spínacích částech zdroje), instaloval jsem do krabice ke každému zdroji jako zátěž odpor 5Ω složený z paralelní kombinace dvou odporů 10Ω/6W. Při provozu protéká každým z odporů proud 0.5A. Celková výkonová ztráta představuje 5W pro jeden zdroj. Zdroje jsou tedy naprázdno zatíženy na 2% nominálního proudu. Doporučuje se sice zatížení 5-10% celkového výkonu, ale podle našich zkušeností uvedené zatížení pro zdroje s vyšším výkonem plně postačuje. Zdroje by zbytečně odebíraly proud ze sítě a navíc by bylo potřeba někam trvale odvádět vznikající teplo. To při zatížení 5% dosahuje 12.5W na zdroj - dohromady 25W (oproti 10W).

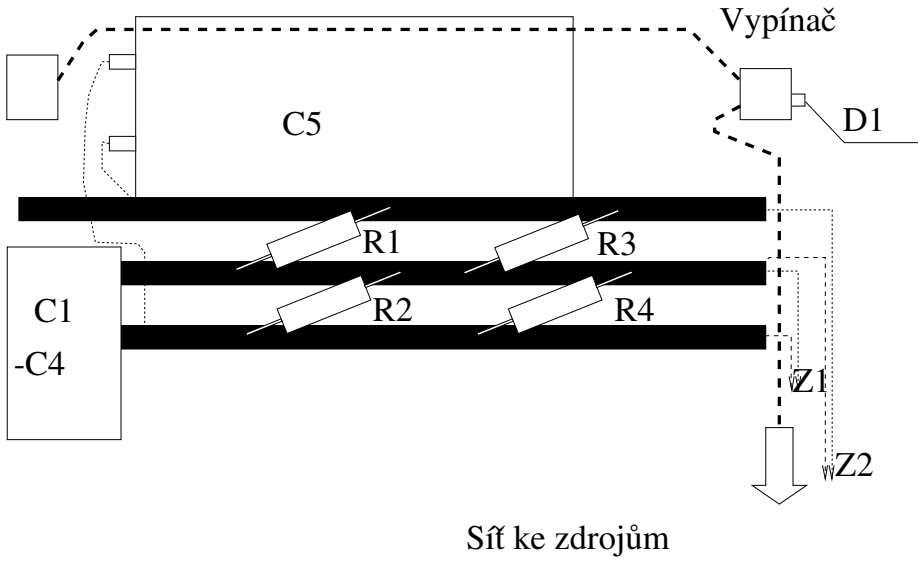


Figure 42: Rozmístění součástek

Další zkouška zdrojů byla již provedena s připojeným modelem vrtulníku. Při zapnutí zdroje však docházelo k vypadnutí nadproudové ochrany u jednoho zdroje. Protože se zdroje střídaly, usoudili jsme, že není chyba ve špatně nastavené proudové pojistce jednoho ze zdrojů.

Proudový náraz jsme zkoušeli tlumit postupným přidáváním kondenzátorů o kapacitě 20mF. Při dosažení kapacity 0.2F jsme začali odstraňovat další příčiny proudového nárazu:

- Regulátor otáček neuměl zastavit otáčení motoru. Při zapnutí proto pustil do motoru plný výkon. Po vyřazení regulátoru dochází sice také k přechodovému jevu, který se projevuje impulzem napětí na motoru, ale ten je již podstatně menší. Je způsoben přechodovým jevem při zapnutí vznikajícím v použité inverzní logice generátoru PWM.

- Do vedení jsme zařadili rozběhový odpor asi 0.04Ω. Úbytek napětí při provozu (proud 28A) je celkem snesitelných 1.1V. Při konečné instalaci tento úbytek bude na delších přívodních vodičích a do vedení budou připojeny 2 tlumivky.

- Ve finální verzi je celé zapojení regulátoru motoru provedeno odlišně. Proto by již k proudovým nárazům po zapnutí docházet nemělo.

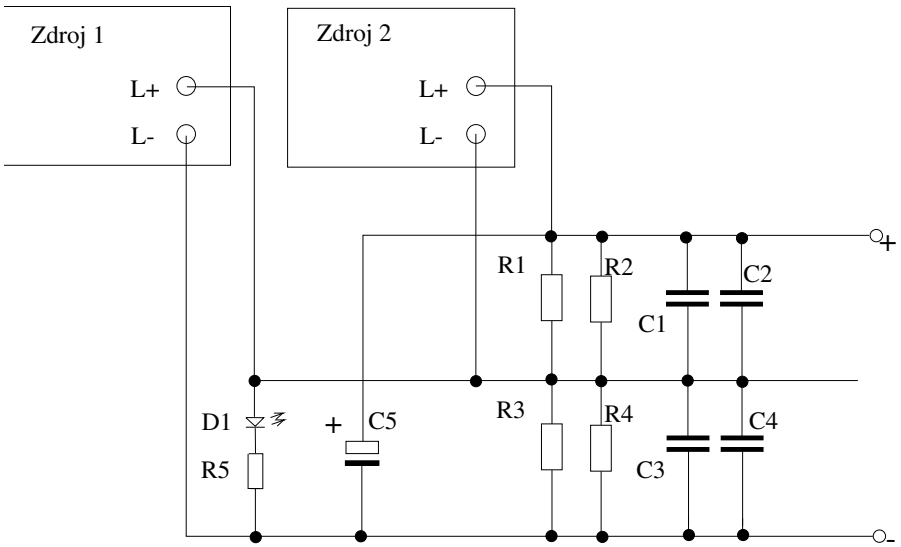


Figure 43: Vnitřní zapojení zdroje

8.2 Soupis součástek

Odpory:

R1,R2,R3,R4	10Ω/6W
R5	150Ω

Kondenzátory:

C1,C2,C3,C4	2x0.25μF/160V TC453MP
C5	100000μF/16V

Diody:

D1	LQ100

Mechanické díly:
Krabice Tesla
2xplech děrovaný 472x270 Ocel Tloušťka 0.5
3xhranol 190x12x4 Měď
8xšroub M3x12
6xšroub M6x10 Mosaz
Funkční celky:

Zdroj1,Zdroj2 Typ: DC C 205

8.3 Zkušenosti s provozem

Při provozu jsme zjistili, že nejvíce tepla vzniká uvnitř zdroje. Vzadu umístěný chladič výkonového tranzistoru zůstává relativně chladný. Proto jsme vnější horní a spodní kryt zdrojů zhotovili z děrovaného plechu. **Při provozu nesmí nikdo na tento kryt cokoli pokládat!** Hrozilo by vnitřní přehřátí a tím i pravděpodobné zničení zdrojů.
Odstraněné závady:

Došlo k proražení kondenzátoru v síťové části zdroje. To způsobilo následné přetavení ochranného odporu. Naštěstí ostatní součástky byly v pořádku. Po odstranění závady byl zdroj opět funkční. Problém jsme přičetli počáteční zahořovací fázi zdroje.

8.4 Zhodnocení činnosti zdroje

V této době je zdroj již dostatečně zahořený, protože má za sebou několik hodin činnosti. Do zdroje je vmontován další konektor. Ten v budoucnu umožní logické vypnutí zdroje z počítače, zvětšení/zmenšení výstupního napětí o delta. Poměrně zajímavá se mi jeví i možnost měřit výstupní proud pomocí analogového výstupu připojeného k A/D převodníku v počítači.

Podle orientačních měření jsme zjistili, že model při 100% výkonu odebírá ze zdroje proud 28A. Při záběru se proud pohybuje kolem maximální jmenovité hodnoty.

9 Systémy reálného času

Systémy reálného času (dále jen RT systémy) jsou určeny svojí koncepcí zejména pro spojení s okolními, většinou dynamickými systémy. Dynamický systém totiž v případě zpoždění RT systému nemůže počkat a vyvíjí svůj stav pouze v závislosti na svém vnitřním stavu a čase (vstupy se nemění). V praxi se hlavně jedná o přímé řízení technologických procesů, testování a návrh složitějších regulátorů před jejich implementací, dynamické sledování a kontrola parametrů.

Nejpodstatnější výhodou RT systémů je jejich relativně velmi snadná přeprogramovatelnost, což je obecně výhoda i všech ostatních systémů založených na programově řízených obvodech. Nutnost pracovat v reálném čase však zvyšuje požadavky jak na HW vybavení tak i na návrh SW.

Lze otestovat velké množství algoritmů a vyhledat z nich ten nejlepší (podle požadovaného kritéria). Testování a výběr algoritmů je zvláště vhodné u složitějších systémů. Model helikoptéry představuje jeden z velmi složitých systémů. Vzhledem k množství ovládacích prvků, měřených signálů a nelinearitám klade vysoké nároky na řídící systém. Právě tato skutečnost byla jedním z podnětů pro realizaci modelu takového systému.

9.1 Návrh číslicového regulátoru

Návrh číslicového regulátoru představuje opakovaný proces skládající se z několika kroků. Nejprve je potřeba vytvořit model systému podle výsledků identifikace původního systému. Po vytvoření je model simulován a jeho parametry laděny tak, aby se svým chováním co nejvíce přiblížil k původnímu systému. V další fázi je navržen model regulátoru a simulován společně s modelem systému. Nakonec je regulátor implementován, připojen k reálnému systému a odzkoušen. Výsledky pokusu obvykle vedou k potřebě změnit strukturu regulátoru, modifikovat model systému a někdy vyžadují i provedení nové identifikace.

Pod slovem identifikace rozumíme měření odezvy systému na různé vstupní signály. Tato činnost je obvykle prováděna speciálním programovým vybavením zaměřeným na sběr dat. Naměřená data jsou následně použita pro vytvoření modelu systému. Pro ověření správnosti návrhu je provedena simulace modelu. Obvykle lze obě činnosti provádět jedním nástrojem podobně jako návrh struktury regulátoru. Ať je regulátor navržen jakýmkoli způsobem, je nutné ověření jeho činnosti pomocí simulátoru. Poslední krok představuje připojení regulátoru k cílovému systému a testování výsledků na reálném systému. K tomuto účelu je potřeba potřeba převést strukturu regulátoru do jazyka RT systému, který bývá odlišný od jazyka simulátoru. Celý výše popsaný postup je zobrazen na obrázku 9.1. Plné čáry v něm určují přímé kroky návrhu. Čárkované vyznačují zpětné postupy, kterých je často zapotřebí, když regulátor ve spojení s reálným systémem nedává uspokojivé výsledky.

Aby bylo možno celou výše uvedenou činnost přivést k žádaným výsledkům, musí mít návrhář (nebo skupina návrhářů) hluboké znalosti

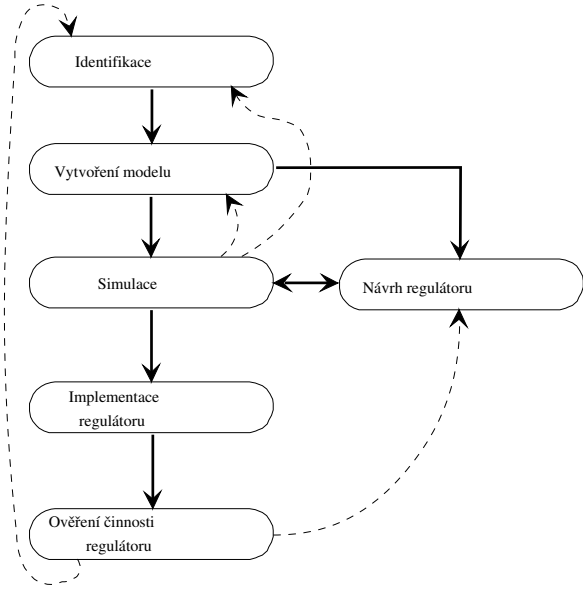


Figure 44: Fáze návrhu regulátoru

tří specializovaných programů: program pro sběr dat, simulační program a cílový řídící program. Některé programové balíky se pokoušejí automatizovat jednotlivé kroky návrhového cyklu a provazovat části mezi sebou.

9.2 Požadavky kladené na RT systém

V této kapitole bude provedeno upřesnění termínu '*Operace v reálném čase*'. RT systémy bývají velmi často připojeny k vnějším procesům. Ale každý systém, který je připojen k vnějšímu procesu nelze označit jako RT systém. Pod označením RT počítačový systém rozumíme takový systém, který splňuje následující požadavky:

- Pořadí dílčích výpočtů je přesně definováno tokem času.
- Výsledky výpočtů závisí na požadavcích přicházejících v reálném čase, nebo na čase potřebném pro výpočet.

RT systémy se dále dělí do dvou kategorií:

1) Doba běhu hlavní výkonné části musí být kratší než předem stanovené maximum.
2) Výpočet musí být proveden po vzniku libovolné události v době kratší než je maximální povolená.

Systémy splňující požadavky první kategorie jsou často nazývány programově orientované RT systémy, zatímco systémy druhé kategorie jsou nazývány obvodově orientované RT systémy. Z uvedeného je zřejmé, že

obvodově orientované systémy mají mnohem větší omezení na prostupnost požadavků systémem oproti programově orientovaným systémům.

Nyní je potřeba vzít v úvahu požadavky na RT simulační program. Číslicová řídící smyčka, ke které je simulační program připojen, má nejčastěji pevnou vzorkovací periodu a většina metod syntézy regulátorů akceptuje tuto konstantní periodu pro navrhovaný regulátor. Je potřeba pečlivě vybírat simulační algoritmus s ohledem na dobu jeho běhu.

9.3 Diskrétní versus spojitá simulace

Digitální řídící smyčka nejčastěji spojuje diskrétní a spojitý dynamický systém. Regulovaný systém bývá nejčastěji spojitý, zatímco regulátor diskrétní. Vzniklý hybridní systém lze simulovat dvěma možnými způsoby: buďto provést simulaci obou systémů v původní hybridní podobě nebo použít diskrétní model spojité části a simulovat celek jako výhradně diskrétní systém.

Hlavní výhodou výhradně diskrétní simulace systému je podstatně jednodušší implementace simulátoru, která se příznivě odrazí v rychlosti simulace. Není potřeba provádět iterativní výpočty, které jsou nutné pro simulaci spojitých systémů, což umožní mnohem přesnější odhad doby běhu simulačního algoritmu. Je však navíc vyžadován model spojité části systému. Tímto je poněkud komplikována fáze návrhu modelu a model systému se stává méně obecný. I když je obyčejně možné snadno převést spojitou část systému na diskrétní, ztrácí se některé informace výběrem vzorkovací periody. Ta představuje při návrhu velmi důležitý parametr s významným vlivem na činnost regulační smyčky.

Protože je simulace hybridního systému v reálném čase mnohem složitější, dává uživateli možnost modifikovat všechny parametry regulační smyčky včetně periody vzorkování. Některé druhy simulace hybridních systémů mohou uspokojivě simulovat za podmínky, že model systému je spojitý. Daní za komplexnější přístup je požadavek zahrnout do simulátoru algoritmy numerického řešení nelineárních diferenciálních rovnic, které celý systém komplikují, snižují rychlost výpočtu a doba výpočtu se stává těžko odhadnutelnou.

Avšak dobře navržený a provedený simulátor by měl být schopen detekovat, že v systému nejsou spojité části. Po vnitřně provedené diskretizaci je pak prováděna simulace hybridního systému jako výhradně diskrétního systému, čímž jsou odstraněny nevýhody spojité simulace. Možnost simulovat spojitý systém by tedy mohla být zahrnuta do univerzálního nástroje pro simulaci v reálném čase.

9.4 Výběr platformy pro RT systém

Ideální operační systém vhodný pro podporu simulace v reálném čase by měl být víceúlohový operační systém se schopností komunikovat přes HW s okolním reálným světem. Měl by být nezávislý na připojeném HW, levný a snadno dostupný. Zatím žádný takový systém neexistuje a snad ani existovat nebude, protože některé z požadavků jsou protichůdné a nelze je současně splnit.

Pro účely simulace je využíván program MATLAB a jeho nadstavba Simulink, který se stal v současné době průmyslovým standardem. Úlohy snímání a řízení v reálném čase jsou společně implementovány v RT Toolboxu, který je koncipován jako nadstavba MATLABu pro snazší přechod od simulace k řízení v reálném čase.

10 Programové rozhraní s RT Toolboxem

Výhodnou vlastností většiny RT systémů je možnost zavést do svého jádra ovladač vnějšího zařízení. Tím je umožněna nezávislost RT systému na existenci speciálního hardware. Rozhraní je realizováno pomocí driverů, které mají pevně definovaný vnitřní formát.

Níže uvedené informace týkající se programátorského rozhraní zatím nebyly nikde publikovány a jsou bezpodmínečně nutné pro tvorbu dalších driverů k RT Toolboxu. V rámci diplomové práce bylo potřeba vytvořit driver pro řízení modelu helikoptéry.

10.1 Vzorkovací perioda a prostupnost jádra

RT Toolbox spouští časovače s jejich definovanými periodami. Na následujících řádkách bude popsán způsob dělení času CPU mezi jednotlivé časovače a zbylé procesy.

V jádru RT Toolboxu je využit pouze jeden HW časovač standardně vestavěný do PC. HW časovač definuje časový interval, ve kterém se bude periodicky volat řídící procedura jádra RT Toolboxu. Podle něho je programově určováno časování jednotlivých časovačů. Při každém intervalu HW časovače jsou zkontrolovány všechny časovače a v případě potřeby je provedena požadovaná akce. Je-li potřeba spustit ve stejném okamžiku dva procesy, proces s nižší prioritou je zpožděn do následujícího intervalu.

Priorita časovače je určena jeho identifikátorem. Časovač s nižším číslem má vyšší prioritu. Požadavek od časovače s nižší prioritou může být dočasně zmrazen, vznikne-li ve stejném okamžiku požadavek časovače s prioritou vyšší.

RT Toolbox vyžaduje určitou část času CPU pro provedení svých vnitřních operací. To normálně představuje pouze malou část celkové doby běhu procesoru. Základní vzorkovací perioda jádra RT Toolboxu je interně nastavena na 1ms a je možno jí dodatečně modifikovat při zavádění RT Toolboxu do paměti. Tato hodnota způsobí zpomalení průměrného počítače PC/AT asi o 5%. Při použití MATLABu 386 se jedná o 15-20% zpomalení způsobené přepínáním reálného a chráněného režimu procesoru. Při nastavení delší doby vzorkování není možné navrhovat rychlé řídící smyčky, ale MATLAB běží rychleji. Při volbě velmi krátké periody může dojít ke zhroucení systému i přes částečné testování vhodnosti zvolené periody. U poslední verze RT Toolboxu pro Windows je základní vzorkovací perioda počítána automaticky podle rychlosti definovaných časovačů.

Doba běhu procedur ovladače by měla být podstatně kratší, než je zvolená hlavní perioda vzorkování. V opačném případě by mohlo dojít k novému přerušení od HW časovače v době běhu driveru. Tato situace není korektně ošetřena a většinou způsobí zhroucení systému.

10.2 Používání hardwarových ovladačů

RT Toolbox umožňuje zavést do paměti současně více ovladačů. Takto je možno použít více karet pro sběr dat, což přináší výhodu zejména u speciálních zařízení, nebo zařízení skládajících se z více na sobě nezávislých částí. Pro příklad uvedu kombinaci kulička na desce jejíž polohu snímá kamera. Nahrání více driverů se provede jednoduše opakovaným zadáním příkazu *rtload* v MATLABu. Starší verze RT Toolboxu vyžaduje při opakovaném zavádění ovladače stejného jména zadání rozdílné hodnoty adresy. Hardwarové kanály jsou přiřazovány postupně po sobě, jak je ukázáno na následujícím obrázku 10.2.

Vstupních i výstupních kanálů je staticky definováno 256. Před

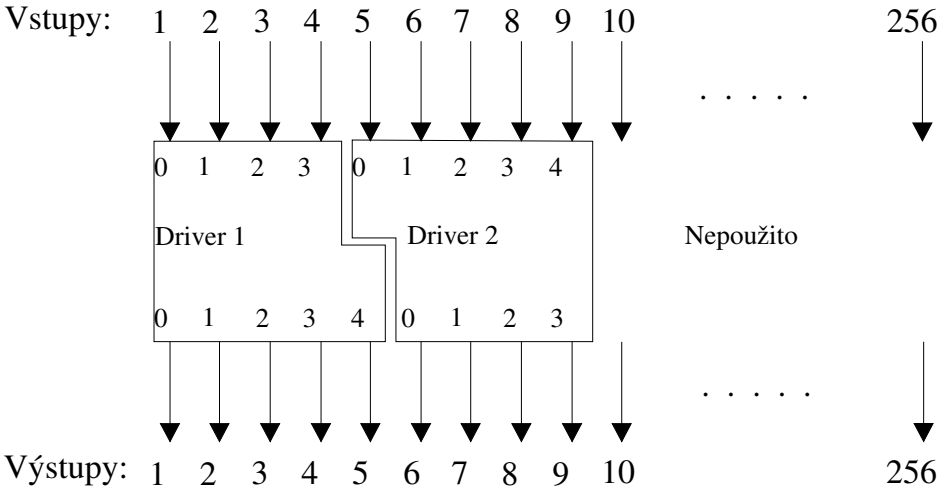


Figure 45: Způsob číslování hardwarových kanálů

zavedením ovladače jsou všechny označeny jako nepoužité. To znamená, že vstupní kanál vždy vrací nulovou hodnotu a data poslaná do výstupního kanálu se ztrácí. Do paměti lze současně zavést maximálně 8 ovladačů. Číslo ovladače 9 je interně využito pro označení nepoužitých kanálů.

Za povšimnutí stojí skutečnost, že vstupní a výstupní kanály jsou číslovány odděleně. Nepoužité kanály, které nejsou vázány k fyzickým vstupům/výstupům přes ovladač, lze výhodně využít pro definování vazby s jiným objektem.

Odkaz na kanál je prováděn přes číslo kanálu. Způsob číslování je absolutní a začíná od čísla 1. První kanál dalšího driveru daného typu bude mít pořadí 1+(počet kanálů prvního ovladače stejného typu). V ovladači je interně zavedeno číslování kanálů od 0. V nové verzi RT Toolboxu v.2.0 je povoleno navíc ještě relativní číslování. Číslo kanálu se v tomto případě skládá ze dvou číslic oddělených tečkou. První z nich definuje číslo ovladače a druhé určuje kanál ovladače. Pro zrychlení činnosti jádra je relativní

číslování odvozeno od absolutního během příkazu *rtdef*. Před použitím relativního číslování je potřeba nejprve nahrát všechny ovladače a definovat časovače. Jinak nebude tento způsob pracovat korektně.

10.3 Driver pro DOSový Matlab

Celý driver představuje fragment kódu, který začíná od adresy 0. Na uvedené adrese musí být nalezena tabulka s přesně definovaným významem jednotlivých položek. Na konci tabulky jsou obsaženy odkazy na jednotlivé procedury. Vlastní kód může zabírat maximálně 64kB paměti. Při volání *HWEnable* lze volat i služby DOSu.

Je možné psát zdrojový text i v jazyce C. Pro diskutovaný případ je nutno zapsat v Assembleru pouze vyplněnou hlavičku s odkazy do výkonné části. Při překladu jazyka C je nutno volit takový paměťový model, který má kód a data v jednom společném segmentu. Avšak všechny výkonné procedury musí být v překladači definovány jako vzdálené. Slovem výkonné jsou označeny ty, které jsou přímo volány z jádra RT Toolboxu.

Drivery pro MATLAB a pro MATLAB 386 mají totožnou strukturu. Pokud nejsou uvnitř použity speciální funkce, je možno stejný driver zavést z obou prostředí. Je také možné při zavolání funkce *HWEnable* otestovat režim, ve kterém je driver spuštěn a podle výsledku testu vybrat vhodnou část kódu. Nejjednodušší test lze provést porovnáním registrů DS a CS. V reálném režimu musí mít samozřejmě stejnou hodnotu, protože ukazují na stejné místo v paměti.

V MATLABu 386 běží RT Toolbox v okruhu 0 procesoru. S touto úrovní oprávnění jsou spouštěny i všechny drivery.

10.4 Struktura hlavičky

Na počátku hlavičky je obsažen název zařízení zakončený znakem 0. Nemá vliv na funkci driveru a slouží pouze pro informaci uživatele při zadání příkazu *RTWho*.

Při volání inicializační procedury *HWEnable* lze předat driveru maximálně 2 parametry. V případě jejich zadání z MATLABu jsou již v okamžiku volání procedury *HWEnable* nastaveny žádané hodnoty. Oba parametry nemají žádný vliv na další činnost RT Toolboxu s jednou výjimkou: nelze zavést 2x stejný driver se shodnou adresou. Parametr *Address* by měl nějakým způsobem popisovat adresu zařízení. Při psaní zdrojového textu je vhodné oba parametry nastavit na implicitní hodnoty.

Parametry *inchn* respektive *outchn* udávají počet vstupních respektive výstupních kanálů. Jejich hodnota je definitivně platná až po provedení inicializační procedury *HWEnable*. V této proceduře lze podle zadaných hodnot parametrů nastavit požadovaný počet vstupních a výstupních kanálů. Pak by se již počty kanálů neměly měnit.

Položka *functions* obsahuje odkazy na následující funkce: *HWdisable, HWenable, inword, outword, inscan* (jejich popisu je věnována následující kapitola). Funkce jsou uvedeny v požadovaném pořadí. Všechny ukazatele musí ukazovat do platné části kódu.
Celou strukturu hlavičky popisuje následující tabulka:

Položka	Délka	Popis
Hwname	32 byte	Logické jméno zařízení
option	1 word	Parametr předaný při zavádění
inchn	1 word	Počet vstupních kanálů
outchn	1 word	Počet výstupních kanálů
adress	1 word	Fyzická adresa zařízení
dmarq	1 word	Adresa kanálu DMA
dmaspeed	1 word	Rychlost DMA kanálu
reserved	4 byte	
functions	5 word	Ukazatale na jednotlivé funkce

10.5 Komunikační funkce rozhraní

Odkazy na funkce jsou uvedeny v tabulce *functions*. Přes to, že funkce jsou definovány způsobem Far, jsou v tabulce uvedeny pouze jejich offsety. Segmenty kódu musí být vzhledem k použitému modelu shodné pro celý driver a jsou doplněny podle volaného driveru.

Jednotlivé funkce jsou volány jako procedury z jazyka C, a proto je potřeba dodržet určitá pravidla při práci s registry. Nemělo by dojít ke změnám segmentových registrů ES a DS. Dále by měly být zachovány hodnoty indexových registrů SI, DI a BP. Všechny ostatní registry je možno libovolně měnit. Příklad zápisu procedury:

```
inscan      proc    c far uses bp si di
            mov     ax,−1
            ret
            endp
```

10.5.1 HWDisable

Vstupy: -
Výstupy: -

Zruší činnost ovladače karty. V praxi se může jednat o vypnutí elektronicky ovládaných zařízení. Může provést i odalokování programových zdrojů využívaných ovladačem karty. Procedury *HWEnable* a *HWDisable* by měly být při korektní práci s RT Toolboxem volány v páru po sobě.

10.5.2 HWEnable

Vstupy: AX = Datový segment driveru
DX = Údaj o rychlosti procesoru
Výstupy: AX = 0 Instalace proběhla v pořádku

!= 0 Nastala chyba při instalaci

Provede inicializaci a popřípadě i detekci přítomnosti obsluhovaného hardware. Vzhledem ke stejné struktuře driverů pro MATLAB a MATLAB 386 lze v tomto bodě přepínat potřebné části kódu. Podle nastavení proměnných *Option*, *Adress* a *DMArq* lze modifikovat činnost ovladače a dodatečně nastavit počet vstupních a výstupních kanálů.

Rychlost procesoru je měřena podle počtu průběhů jednoduchou smyčkou Loop za 1mS při zakázaném přerušení. Tento údaj by měl být kontrolován zejména výpočetně náročnějšími drivery (např. pro rozpoznávání obrazu). Nedostatečná rychlost hostitelského počítače pak povede k odmítnutí instalace.

Datový segment ukazuje na hlavičku driveru a je v reálném režimu totožný se segmentem kódu.

10.5.3 InWord

Vstupy: AX = Číslo kanálu
Výstupy: AX = Přečtená hodnota z kanálu

Funkce přečte jedno slovo ze specifikovaného kanálu. Přečtená hodnota se může pohybovat v rozsahu +maxint -maxint. Při znalosti maximálních a minimálních možných hodnot by tento interval měl být převeden do celého rozsahu slova. Uvedený rozsah je následně konvertován v prostředí MATLABu do rozsahu +1 -1.

10.5.4 OutWord

Vstupy: AX = Číslo kanálu
DX = Zapisovaná hodnota

Výstupy: -

Zapíše jedno slovo do výstupního kanálu. Interval -1 +1 je konvertován do rozsahu -maxint +maxint v registru DX. Je-li z prostředí MATLABu zapsáno číslo mimo uvedený rozsah, je výstupní hodnota limitována a do registru DX zapsána maximální hodnota.

10.5.5 InScan

Vstupy: AX = Číslo kanálu
DX = Vzorkovací perioda v μS
BX = Počet vzorků
ES:DI -> Ukazatel na data

CX:SI -> Fyzická adresa dat (pro DMA kanál)
Výstupy: AX = 0 Činnost proběhla v pořádku
= -1 Funkce není driverem podporována

Účelem této funkce je sejmout co možná nejrychleji definovaný počet po sobě následujících vzorků. V praxi nalezne použití při měření velmi rychlých přechodových charakteristik, kde nelze vystačit s použitím historických proměnných. U driveru nemusí být tato funkce implementována. V takovém případě by procedura *InScan* měla vždy vracet hodnotu -1 v registru AX.

Pokud připojené vnější zařízení je schopno pracovat přes DMA kanál, tak jej lze využít. Při volání v chráněném režimu nabývá fyzická adresa dat a ukazatel na data rozdílných hodnot.

10.6 Driver pro Matlab pod Windows

RT Toolbox od verze 2.0 podporuje pouze platformu Windows. Jedná se o novou verzi ve které došlo k větším změnám i ve struktuře ovladačů. Změny jsou hlavně způsobeny kompletním přechodem na podporu jazyka C. Všechny funkce nyní mají definován způsob předávání parametrů přes zásobník, jak je tomu obvyklé u použitého kompilátoru jazyka C. Parametry komunikačních funkcí však zůstaly z větší části zachovány.

Zdrojový text je nyní vhodné psát v jazyce C. Pro překlad byl vybrán 32 bitový kompilátor Watcom C++, který podle zkušeností generuje poměrně úsporný kód.

10.6.1 Rozšíření funkcí rozhraní

V této kapitole jsou popsány pouze změny a vylepšení, která přináší nová verze RT Toolboxu. Všechny ostatní informace zůstaly nezměněny a byly již popsány v kapitole *Komunikační funkce rozhraní*.

Disable *void Disable(void)*
Zruší činnost ovladače karty.

Enable *int Enable(int np, double *parm, int (*callback) (int fn,...))*
Provede inicializaci HW vnějšího zařízení. Nyní lze předat v proměnné *options* libovolnou strukturu MATLABu tzn. číslo, vektor nebo matici. Proměnná **parm* pak obsahuje ukazatel na celou strukturu a do proměnné *np* je uložen počet položek. Takto lze ovladači předat libovolné množství informací.
Ukazatel **callback* zpřístupňuje vnitřní funkci jádra, která poskytuje určité služby.

Input *double Input(int ch)*
Přečte hodnotu z kanálu *ch*. Nová verze neomezuje zapisovanou hodnotu na jedno slovo a nechává konverzi přímo na samotném ovladači. Přečtená hodnota z kanálu nyní není omezena intervalem <-1;1> a může nabývat libovolné hodnoty (v povoleném rozsahu typu double).

Output *void Output(int ch, double val)*

Zapíše hodnotu předanou v proměnné *val* do kanálu *ch*. Na rozsah zapisovaných hodnot, podobně jako u procedury *Input*, se nevztahují žádná omezení. Z toho vyplývá nutnost ošetřit nevhodné hodnoty mimo povolený rozsah zpřístupňovaného vnějšího zařízení.

InScan *int InScan(int ch, int len, double *spec, double *buf, int *cols)*

Sejme ze zařízení co možná nejrychleji z kanálu *ch* definovaný počet vzorků s udanou periodou vzorkování. Navzorkovaná data budou následně uložena do pole **buf*. V této verzi není již předávána fyzická adresa paměti, protože OS Windows již v sobě obsahuje funkce pro práci s kanály DMA.

Proceduře *InScan* lze předat jako parametr namísto jednoduché proměnné matici libovolné velikosti. V proměnné *len* je obsažen počet položek struktury na kterou ukazuje parametr **spec*. První číslice v předávané struktuře *spec[0]* je již vyhrazena pro periodu vzorkování.

Použitelné návratové hodnoty jsou předdefinovány v souboru RT.H. V jádře RT Toolboxu není proveden test povoleného intervalu hodnot příčin chybových hlášení a v případě použití hodnoty mimo interval dojde pravděpodobně k pádu OS (to se týká i hodnoty -1).

10.6.2 Změny ve struktuře hlavičky

Vzhledem k orientaci na jazyk C je hlavička definována jako strukturovaná proměnná. Na první adrese driveru je pevně umístěn ukazatel na strukturu hlavičky. Pozice hlavičky v driveru pak již může být libovolná. Datová struktura je definována v souboru RTDSTUB.H. Význam jednotlivých položek zůstal zachován. Odkazy na jednotlivé funkce popisují i jejich parametry.

```
typedef struct
        {
        char Name[MAXDRVNAME];
        int Inputs;
        int Outputs;
        unsigned short Address;
        void (*Disable) (void);
        int (*Enable) (int pn, double *parm, int (*callback) (int fn,...));
        double (*Input) (int ch);
        void (*Output) (int ch, double val);
        int (*InScan) (int ch, int len, double *spec, double *buf, int *cols);
        } Driver;
```

10.6.3 Grafická nadstavba ovladače

MATLAB od verze 4.0 zpřístupňuje vytváření uživatelských oken s dialogy přímo pomocí příkazů svého vnitřního jazyka. Po zadání příkazu *figure* je vytvořeno nové okno a příkazem *uicontrol* lze vytvářet jednotlivé ovládací

prvky. Nastavení ovládacích prvků je možno zjistit příkazem *get* popřípadě změnit příkazem *set*.

Většinu parametrů a nastavení RT Toolboxu lze měnit přes grafické rozhraní GUI (Graphics User Interface). Rozhraní může zpřehlednit a usnadnit komunikaci s ovladačem.

Pro ovládání pomocí GUI musí mít každý ovladač HW navíc .M soubor, který umožní jeho inicializaci v dialogovém režimu. Spustitelný .M soubor je až na koncovku pojmenován stejně jako vlastní ovladač. Po spuštění vykreslí okno s ovládacími tlačítky na obrazovku a reaguje na všechny události. V okénku je vhodné umístit řádek s aktuálním příkazem, který by bylo možno zadat i z dialogové řádky pro vykonání stejné akce. Zavádění ovladače přes grafické rozhraní tedy nepřináší oproti dialogové řádce žádné podstatné výhody.

Požadavek na zobrazení konfiguračního panelu driveru vznikne při použití příkazu *Rtload* bez parametrů. Ten otevře okno umožňující výběr ovladače. Druhou možnost představuje přímé zadání jména ovladače z dialogové řádky.

Každý .M soubor obsahující definici funkce MATLABu může obsahovat nápovědu ke své činnosti. Nápověda je zobrazena po zadání příkazu *HELP JMÉNO_SOUBORU*. V souboru je za nápovědu považována souvislá část komentáře následující po definici funkce. Stručné informace o driveru lze získat příkazem *HELP JMÉNO_DRIVERU*.

10.7 Komunikace s Matlabem

Prostředí MATLABu představuje interpreter vlastního jazyka optimalizovaného pro vědecké výpočty. Je umožněno volání vnějších procedur tzv .M souborů, nebo volání přímo spustitelných .MEX souborů přeložených do speciálního formátu.

Verze pro MATLAB a MATLAB 386 je založena na volání .MEX souborů. Nejnovější verze RT Toolboxu pro Windows má dvouúrovňové volání. Nejprve je interpretován .M soubor s definicemi grafického uživatelského rozhraní. Všechny požadavky jsou následně přesměrovány do spustitelného souboru RTTOOL.MEX, který provádí komunikaci s jádrem. Jádro je zavedeno do paměti již při spouštění Windows jako VxD ovladač. Zájemce o podrobnější informace odkazuji na literaturu [Humusoft: RT Toolbox for MATLAB User's manual].

10.7.1 rtload

Zavede jádro RT Toolboxu do paměti a provede jeho inicializaci. Většinu ostatních funkcí je možné volat až po inicializaci jádra. V okamžiku zavádění ovladače je volána procedura *Enable*.

Parametry *opt*, *addr* a *dma* (*dma* pouze v DOSové verzi) jsou interpretovány pouze zaváděným ovladačem. Není prováděna žádná kontrola předávaných hodnot. Zadání špatných hodnot může způsobit neočekávané katastrofální výsledky. HW ovladače komunikují přímo přes porty s připojeným zařízením. Nevhodně zvolené adresy portů totiž mohou odpovídat např. řadiči HD, CMOS paměti, grafickému displeji atd. Všechny

ovladače v sobě již mají předdefinovány implicitní hodnoty, čímž lze uvedenému problému částečně předejít.

Ve verzi pro DOS lze podruhé zavést stejný ovladač pouze s jinou hodnotou adresy. Ve verzi pod Windows již toto omezení neplatí a při dvou stejných zavedených ovladačích může dojít ke kolizi v přístupu k vnějšímu HW.

Modifikace: *rtload*

rtload(Základní_perioda) Pouze pro verzi běžící pod DOSem.

rtload('Jméno_ovladače')

rtload('Jméno_ovladače',parametr,adresa)

10.7.2 rtrd

Přečte obsah proměnné RT Toolboxu a zapíše jej do ordinální proměnné MATLABu. Časovač obsahuje větší množství proměnných. Většina z nich může být přepsána příkazem *rtwr*.

Pro práci je nejdůležitější stínová proměnná označená písmenem y. Umožňuje čtení výstupů systému a jsou v ní uloženy výstupní hodnoty posílané do vnějších vstupů. Obsah je aktualizován při každém hodinovém intervalu časovače a způsob aktualizace závisí na jeho typu. Všechny ostatní proměnné mají pouze upřesňující význam.

Příkaz *rtrd* je určen také pro čtení historické proměnné. Počet řádek vstupní proměnné odpovídá počtu dosud zpracovaných vzorků. Je-li zadána druhá vstupní proměnná, je do ní zapsán celkový počet zpracovaných vzorků. Ten může být i větší než je délka proměnné.

10.7.3 rtstart

Spustí všechny definované časovače. Od tohoto okamžiku budou opakovaně prováděny všechny předem definované akce včetně volání inicializovaných ovladačů. Po inicializaci příkazem *rtdef* jsou časovače zastaveny a musí být popisovaným příkazem spuštěny.

Ve verzi 2.0 je vyžadováno zadání parametru *all*. Pokud není alespoň jeden časovač spuštěn, VxD ovladač pouze pasívně komunikuje s okolím. Po zadání *rtstart all* jsou spuštěny všechny časovací a vyhodnocovací procesy, čímž se zvýší riziko pádu OS Windows. Proto není vhodné při spuštěném RT Toolboxu zavádět do paměti další programy a pracovat s nimi.

10.7.4 rtstop

Zastaví všechny definované časovače a dočasně zruší provádění všech definovaných akcí. Hodnoty časovačů zůstanou zmrazeny až do jejich opětovného spuštění příkazem *rtstart*. Pak se budou vyvíjet přesně od toho místa, kde byl časovač zastaven.

10.7.5 rtunload

Uvolní všechny ovladače z paměti počítače. Před uvolněním ovladače je volána procedura *Disable*. Všechny spuštěné časovače jsou zastaveny a historické proměnné smazány.

Ve verzi pod DOSem je při opuštění MATLABu proveden pokus o automatické uvolnění RT Toolboxu z paměti. To nemusí ve všech případech pracovat korektně a proto je lepší použít příkaz *rtunload*. U driveru pro Windows běží VxD driver dál bez možnosti ovládání (je možno s ním i nadále komunikovat přes vyhrazené přerušení, což je prakticky těžko proveditelné) a v nejlepším případě pouze zpomaluje činnost ostatních programů.

10.7.6 rtwho

Vytiskne zajímavé informace o stavu jádra RT Toolboxu na obrazovku. Jsou vypsána jména zavedených HW driverů, definované časovače spolu s jejich periodami, definované historické proměnné včetně jejich délky, vazby mezi objekty, zátěž systému v % atd.

10.7.7 rtwr

Zapíše obsah ordinální proměnné MATLABu do proměnné RT Toolboxu. U většiny proměnných lze přepsat jejich hodnotu. Zápis historické proměnné začíná vždy od jejího prvního vzorku. Operace *rtwr* přepisuje vždy celou proměnnou. Pokud je potřeba modifikovat její část, její obsah musí být nejprve přečten, modifikován a následně zapsán.

10.8 Ladění a diagnostika driveru

Testovnání jakékoliv části kódu běžícího v reálném čase je obecně velmi obtížné. Krokování již nemůže být prováděno v reálném čase. Proto je možno pouze staticky otestovat komunikační procedury s vnějším zařízením.

Do části kódu, kterou je potřeba ověřit, se přímo vloží instrukce INT 3 a ovladač se znovu přeloží. Spustí se debugger TD 286 (TD popř. TD 386 zabírá v paměti příliš mnoho místa). Z něho je zaveden do paměti program MATLAB, který se spustí klávesou F9 (run). V MATLABu normálně pracujeme s RT Toolboxem. V okamžiku, kdy procesor zpracovává námi vloženou instrukci INT 3, dojde k zastavení MATLABu v trasovacím okně debuggeru. Krokování probíhá od této chvíle analogicky jako u ostatních programů.

Ladění ovladačů pro MATLAB 386 probíhá obdobně. Pouze místo TD 286 je nutno použít Metaver Debugger nebo Watcom Debugger. Je-li použit ovladač paměti QEMM je potřeba před laděním zadat příkaz *QDPMI off*, jinak může dojít ke kolizi rozšiřujících služeb správce paměti s debuggerem.

10.8.1 Ladění Driveru pod Windows

Ovladač pro Windows běží jako proces v jádře systému Windows. Pro ladění vnitřních procesů musí být zavedeno speciální ladící jádro. Pak je možné přes sériovou linku nebo síť z jiného počítače ladit vyvíjený program.

Vzhledem k obtížnosti výše popsané metody doporučuji odladit jednotlivé procedury v testovacím programu. Když pracují správně, pak je lze snadno zkopírovat do zdrojového kódu ovladače a provést jeho opětovný překlad.

Protože procesy v jádře nelze nijak chránit, jakákoliv chyba v ovladači se projeví pádem OS. Pokud lze ještě zavolat DOS, Windows okamžitě ukončí svou činnost a přejdou do DOSu. V opačném případě je nutno provést restart počítače.

11 Programová část projektu

V této kapitole je obsažen popis a rozbor realizovaných programů, které komunikují s HW částí a provádějí její diagnostiku. Jedná se zejména o ovladače pro RT Toolbox, jejichž struktura byla diskutována v předchozí kapitole.

11.1 Propojení z hlediska PC

Propojení je realizováno přes rozšiřující kartu číslicových a analogových vstupů a výstupů PCL-812. Ta je zasunuta do rozšiřujícího slotu sběrnice ISA v základní desce počítače. Komunikace je prováděna přes skupinu 15 portů, jejichž číslování začíná od adresy *Base*.

Bázovou adresu umístění portů lze volit pomocí konfiguračních přepínačů na desce rozšiřující karty. Při připojení k počítači s jiným nastavením této adresy by pak mohl nastat problém. V takovém případě by se však nepodařilo ovladač vůbec zavést do paměti.

Před zavedením ovladače helikoptéry je prováděn test na přítomnost karty PCL-812. Test je založen na měření doby A/D převodu z libovolného kanálu. Pozice jednotlivých portů ukazuje následující tabulka:

Umístění	Čtení	Zápis
Base +0	Čítač 1	Čítač 1
+1	Čítač 2	Čítač 2
+2	Čítač 3	Čítač 3
+3	-	-
+4	A/D nižší byte	D/A nižší byte, kanál 1
+5	A/D vyšší byte	D/A vyšší byte, kanál 1
+6	D/I nižší byte	D/A nižší byte, kanál 2
+7	D/I vyšší byte	D/A vyšší byte, kanál 2
+8	-	Zrušení žádostí o přerušení
+9	-	-
+10	-	Multiplexer A/D vstupů
+11	-	Řízení PCL-812
+12	-	A/D Trigger
+13	-	D/O nižší byte
+14	-	D/O vyšší byte

Pro připojení k vnější sběrnici jsou využity následující kanály: D/O_Hi - adresa+řídící signály, D/O_Lo - datová sběrnice výstupní, D/I_Lo - vstupní datová sběrnice. Zájemce o podrobnější informace o I/O rozšiřující kartě odkazuji na literaturu:[PCL-812 Enhanced multi lab card user's manual].

11.2 Komunikace s vnějším zařízením

Pro komunikaci jsou využity dvě výstupní a jedna vstupní brána, čímž vznikne simulovaná sběrnice procesoru 8080. Tato část již byla detailně rozebrána v kapitole *Způsob připojení k PC*. Jednotlivé signály řídící sběrnice jsou zobrazeny na 11.2.

Veškerá komunikace je následně prováděna pomocí čtení a zápisu paměťových registrů připojených k vnější sběrnici. Registry v sobě obsahují všechny důležité informace o poloze vrtulníku a všechny řídící signály. Komunikace pomocí registrů je výhodná pro stejný způsob práce s jednotlivými registry.

V této části jsou vstupní respektive výstupní registry chápány z hlediska nadřízeného počítače. To znamená, že ze vstupního registru lze pouze číst a výstupní registr je určen výhradně pro zápis.

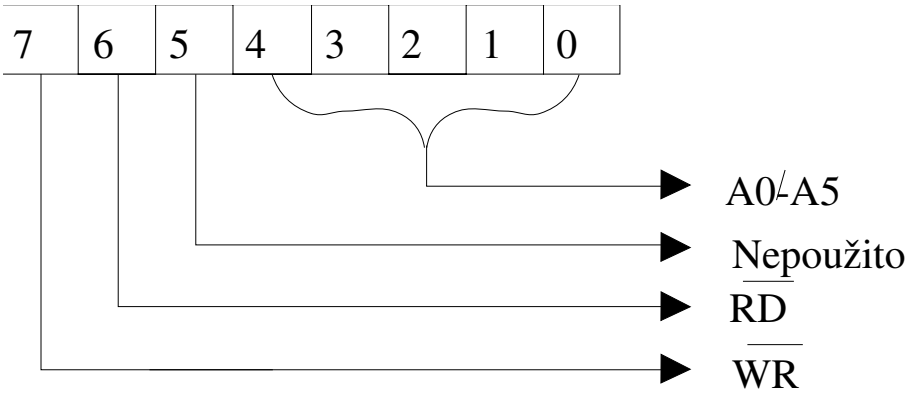


Figure 46: Umístění signálů řídící sběrnice

11.2.1 Vstupní registry

Vstupní registry obsahují všechny potřebné informace o stavu modelu včetně několika dalších určených pouze pro diagnostiku.
Obsazení vstupních registrů:

Číslo	Význam registru
0	Kurs vrtulníku
1	Příčný náklon modelu
2	Podélný náklon modelu
3	Rychlost otáčení rotoru
4	Příčný náklon vodící tyče
5	Podélný náklon vodící tyče
6	Délka vysunuté tyče
7	-
8	Kontrolní registr 1.kanálu
9	Kontrolní registr 2.kanálu
10	-
11-15	Kontrolní identifikátor 'PK&JF'

Standardní sběrnice 8080 vyžaduje předstih adresy před aktivací signálu RD. Obvod Xilinx je však naprogramován takovým způsobem, že mu nevadí snížení předstihu na nulovou dobu. Tím se podaří ušetřit v programu jeden sběrnicový cyklus zápisu do rozšiřující karty, který trvá velmi dlouhou dobu z hlediska CPU 1.4μS (za tuto dobu vykoná procesor 486/50 přibližně 280 instrukcí).

S uvedenou dobou je počítáno při vytváření vnějšího sběrnicového cyklu. Obvod Xilinx dodá data na sběrnici přibližně za 500nS. Při použití výkonnější IO karty by pak bylo nutno vložit odpovídající časové zpoždění mezi požadavek na čtení a vlastní čtení příslušného registru.

Registry 11-15 slouží pouze pro identifikaci připojeného zařízení nadřazeným počítačem. Vedlejším efektem jejich čtení je vynulování kontrolních registrů kanálů. Čtení registru provádí následující funkce:

```
int read_channel(char kanal)
{
int i;
outp(baze+14,0x90|kanal);
i=inp(baze+6);
outp(baze+14,0xD0|kanal);
return(i);
}
```

Přečtená hodnota ze snímače polohy je pouze osmibitová. Avšak rozsah možných hodnot je podstatně větší. Jedná se o nejnižších 8bitů registru polohy. Vyšší bity tohoto registru nejsou ani registrovány ani přenášeny.

Tímto způsobem je možno snížit množství dat přenášených po sériové lince. Při použití 16 bitových registrů by navíc došlo k podstatnému zvětšení počtu CLB, což by si vyžádalo větší a tedy i dražší programovatelné pole.

Řídící procesor však musí provést rekonstrukci zbylých bitů. Rekonstrukce se provádí po vyčíslení rozdílu mezi aktuální a minulou hodnotou. Je-li rozdíl větší než polovina rozsahu registru (v našem případě 128), pak je nutno provést korekci vyššího řádu. Jediným omezením pro rekonstruktor je určitá maximální doba, za kterou je potřeba provést opětovné čtení registru.

Uvedená doba je závislá na dynamice systému. Pokud vzorkovací perioda překročí uvedenou dobu, dojde k selhání rekonstrukce a všechny další údaje budou od tohoto okamžiku chybné až do opětovné inicializace celého systému.

Činnost rekonstruktoru ukazuje následující funkce:

```
unsigned char OldValues[7];
signed char Leaders[7]={0,0,0,0,0,0,0};
int check_channel(int ch)
{
int j;
j=read_channel(ch);
if(abs(j−OldValues[ch])>128)
        {
if(j>OldValues[ch]) Leaders[ch]−=1;
                else Leaders[ch]+=1;
}
OldValues[ch]=j;
return(((256*Leaders[ch]+j)));
}
```

11.2.2 Výstupní registry

Jedná se o registry dvou programovatelných čítačů/časovačů realizovaných obvodem 8253. Adresy obvodů jsou mapovány do rozsahu 0-7 externí sběrnice. Všechny z těchto níže popsaných registrů jsou určeny pouze pro zápis. Při čtení je přečten obsah vstupního registru umístěného na stejné adrese. Obsazení vstupních registrů:

Číslo	Význam registru
0	Podélná cyklika
1	Příčná cyklika
2	Kolektivní řízení
3	CWR řídící registr 1. čítače
4	Nastavení listů ocasního rotoru
5	Výkon motoru
6	-
7	CWR řídící registr 2. čítače

Je třeba si dát pozor na skutečnost, že vnitřní registry jsou 16 bitové. Zápis do obou částí se provádí pomocí dvou po sobě následujících sběrnicových cyklů. Vzhledem ke zvolenému režimu činnosti musí být vyšší polovina registru vždy naplněna hodnotou 1 a nižší polovina určuje pohyb akčního členu. Délka výstupního pulsu se pohybuje od 1mS do 2mS a řídící hodinový kmitočet je nastaven na 256kHz. Způsob zápisu do registru ukazuje následující funkce:

```
void write_channel(char kanal,char hodnota)
{
kanal=trans[kanal];
outp(baze+14,0xC0|kanal);
outp(baze+13,hodnota);
outp(baze+14,0x40|kanal);
outp(baze+14,0xC0|kanal);
outp(baze+13,1);
outp(baze+14,0x40|kanal);
outp(baze+14,0xC0|kanal);
}
```

Komunikace s výstupními registry pracuje korektně **až po nutné inicializaci** řídících registrů! Při používání driverů je tato činnost prováděna automaticky, ale při testování modelu z jiných programovacích jazyků nemůže být vynechána. Po jedné inicializaci pracují registry korektně až do výpadku napájecího napětí.

Inicializaci je nutné provést u všech 6 registrů. Řídící registry pracují v režimu 1 v binárním módu. V režimu 1 se celý obvod chová jako programovatelný generátor monostabilního impulsu. Dále je nastaveno přepisování celého 16 bitového registru prováděné při každé operaci čtení nebo zápisu. Přepis registru je proveden během dvou sběrnicových cyklů. Při prvním je čtena nižší a při druhém vyšší slabika.

Po inicializaci řídících registrů je velmi vhodné okamžitě nastavit ostatní registry časovačů na počáteční hodnoty. Nejkritičtější je registr výkonu motoru, který by měl být nastaven na nulovou hodnotu. Jinak by mohlo dojít k nežádoucímu vzlétnutí modelu následovaného pádem. Ani poslední 6 registr nesmí být opomenut, i přes to, že není využit. Při zapnutí v něm může zbýt v podstatě libovolná hodnota. **Časový interval delší než je délka rámce přenosu způsobí rozpad synchronizace s následkem selhání celé komunikace!** Synchronizace a dekódování je prováděno v továrně vyrobeném přijímači pro RC ovládané modely a její rozpad není možno ani programově detekovat. Na venek lze výše popisovanou situaci poznat podle trhavých náhodných pohybů ovládacích servomotorů. O zápis do řídících registrů se stará funkce:

```
int write_CWR(char kanal,char hodnota)
{
outp(baze+14,0xC0|kanal);
outp(baze+13,hodnota);
outp(baze+14,0x40|kanal);
```


```
if(inp(baze+6)!=hodnota) return(1);
outp(baze+14,0xC0|kanal);
return(0);
}
```

V této funkci je navíc proveden test správné činnosti HW. Chyba je hlášena v případě špatného propojení datových kabelů. Úspěšné provedení tohoto testu však ještě nezaručuje skutečný zápis dat do řídícího registru.

Při použití driveru jsou již všechny výše uvedené problémy ošetřeny a uživatel se nemusí zabývat jejich řešením.

11.3 Komunikace s driverem

Driver pro helikoptéru v žádné své verzi nepodporuje DMA a s podporou DMA přenosu není počítáno ani pro další verze. Toto omezení je způsobeno sériovým přenosem dat mezi blokem snímačů a blokem registrů pro počítač.

Popisy všech modifikátorů vylepšujících činnost driveru jsou platné pouze pro verzi 2.0 pod Windows. Driver pro DOS pracuje stále v režimu dílků od IRC snímačů. Na tuto **nekompatibilitu** upozorňuji při přechodu z jedné verze RT Toolboxu do druhé.

11.3.1 Předzpracování naměřených údajů

Speciálním způsobem vyhodnocení se podařilo 4x zvýšit přesnost oproti nominální hodnotě. Na výstupu z RT driveru je výhodné mít údaje již normované do vhodného tvaru. Jako nejvýhodnější se mi jeví transformace úhlů do radiánů a délek do mm.

V modelu trenažéru jsou však použity dva různé typy IRC snímačů. Proto je potřeba každý sejmutý údaj transformovat příslušnou konstantou do správného tvaru.

$$\alpha = \frac{2*\pi}{4*540} * Dc = 0.002908882 * Dc = k_1 * Dc$$

$$\alpha = \frac{2*\pi}{4*512} * Dc = 0.0030679615 * Dc = k_2 * Dc$$

$$l = \frac{\pi * d}{2*512} * Dc = \frac{\pi * 35}{1024} * Dc = 0.107378655 * Dc = k_3 * Dc$$

Konstanta k_1 platí pro snímač typu BZ (540 dílků/otáčku) a konstanta k_2 pro typ B (512 dílků/otáčku). Vzdálenost vrtulníku od heliportu, tedy v podstatě délka vysunuté tyče, je měřena také IRC snímačem opatřeným snímacím kolečkem o ϕ35mm. Pro přepočet délky tyče platí konstanta k_3. U snímače délky je přesnost 2x snížena, protože nemá smysl měřit s přesností větší než 0.1mm, kterou není schopna udržet ani mechanická část trenažéru.

11.3.2 Předávání výsledků do Matlabu

Konfigurace ovladače Ovladači zaváděnému do paměti může být předáno několik parametrů. Ty jsou určeny pro modifikaci jeho činnosti. Způsob předávání parametrů byl již rozebrán v kapitole *Rozšíření funkcí rozhraní*.

Zde popisovaný ovladač umožňuje změnu pozice jednotlivých registrů a volbu transformace souřadného systému. Činnost je zjištěna podle délky modifikačního vektoru.

Délka vektoru	Význam jednotlivých položek
0	[]
1	[režim]
3	[D/O_Hi,D/O_Lo,D/I]
4	[režim,D/O_Hi,D/O_Lo,D/I]

Všechny ostatní vektory předané v položce *Option* nemají na činnost ovladače vliv a jsou ovladačem ignorovány. Pouze při zadání chybné transformace nebude ovladač zaveden do paměti.

Při inicializaci driveru je možné též nastavit bázovou adresu karty. Po jejím nastavení je provedena korekce adres komunikačních registrů. Nulová hodnota adresy má speciální význam. Sděluje ovladači, že není použita pro připojení karta PCL-812. Informace o umístění portů jsou v takovém případě dodatečně definovány v položce *Option*.

Kde:

režim
: Způsob transformace souřadnic (viz následující kapitola).

D/O_Hi,D/O_Lo,D/I
: Přímá hodnota komunikačních portů.

Konfigurace ovladače pomocí GUI Konfigurační panel driveru je vykreslen při použití příkazu *Rtload* bez parametrů nebo přímým zadáním jména ovladače z dialogové řádky. V případě ovladače helikoptéry je potřeba zadat příkaz *heli*. Po jeho zadání bude otevřeno okno podobné 11.3.2. Pomocí myši lze měnit nastavení jednotlivých parametrů ovladače. Všechny režimy činnosti již byly vysvětleny v kapitole *Konfigurace ovladače*.

Implicitně je nastaveno připojení přes kartu PCL-812. Bázovou adresu karty lze měnit stiskem přepínačů A4-A8, nebo přímým přepsáním hodnoty adresy v okénku Address. Při volbě alternativního způsobu připojení je překreslena prostřední část okna. Na místo jedné adresy je potřeba zadat tři různé hodnoty adres pro jednotlivé porty. Jejich význam byl popsán v kapitole *Propojení z hlediska PC*.

Po stisku tlačítka Default nebo klávesy D je provedeno nastavení implicitních hodnot ovladače. Ty jsou platné pro připojení přes kartu PCL-812 s původním nastavením bázové adresy. Po stisku tlačítka OK nebo stisku klávesy *Enter* je okno uzavřeno a zadán příkaz k zavedení ovladače. Při vzniku chyby je zobrazeno chybové hlášení *Error initializing hardware*. Hlášení může být způsobeno i nevhodným nastavením parametrů ovladače.

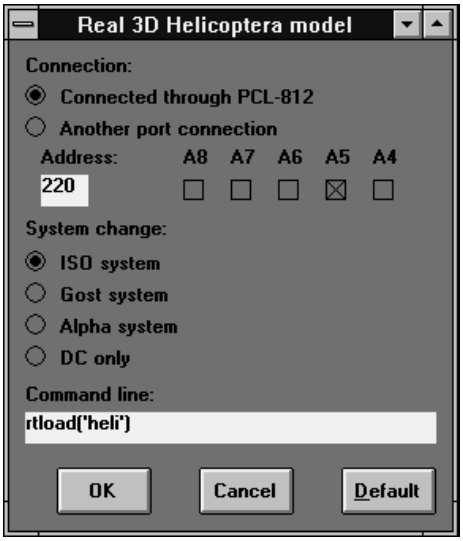


Figure 47: Konfigurace pomocí GUI

Pokud nelze ovladač zavést doporučuji spuštění diagnostického programu pro bližší lokalizaci možné závady. Volba Cancel zruší nahrávání ovladače. Stejného efektu lze dosáhnout i stiskem klávesy *ESC*.

Poslední část dialogového panelu je určena pro volbu vstupního souřadného systému. Jednotlivé souřadné systémy jsou popsány v kapitole *Vstupní kanály*.

Pro uchování informací o stavu okna je využita položka *UserData* dialogového okna. Do ní je možno zapsat libovolný vektor nebo matici. Jednotlivé položky v matici jsou využity následujícím způsobem:

1 - 2
: Přepínač způsobu připojení

3 - 14
: Manipulátory tlačítek pro zadávání adresy

15 - 18
: Přepínač souřadného systému

20 : Baze
21 : DO_Hi
22 : DO_Lo
23 : DI_Lo
24 : Číslo režimu

Používání položky *UserData* pro ukládání manipulátorů jednotlivých objektů je vhodné pouze v případě malého rozsahu programu. Při větším počtu objektů by došlo ke zvýšení obtížnosti jakékoli další modifikace kódu vzhledem k nutnosti provést přečíslování položek.

Vstupní kanály Při čtení údajů z ovladače je vhodné jejich hodnoty upravit do normalizovaného tvaru. Tím se sníží pracnost identifikace modelu a opakované provádění zbytečných přepočtů souřadných systémů. Proto má již ovladač v sobě diskutované transformace zabudovány. Rozmístění vstupních kanálů:

Číslo kanálu	režim 1	režim 2	režim 3	režim 4
1	x	x	α	α_{DC}
2	y	y*	β	β_{DC}
3	z	z*	l	l_{DC}
4	ψ	ψ	ψ	ψ_{DC}
5	δ	δ	δ	δ_{DC}
6	γ	γ	γ	γ_{DC}
7	n	n	n	$n_{DC/T}$

Kde: x,y,z souřadnice modelu

α,β	odklon tyče od svislé osy
l	vzdálenost modelu od heliportu
ψ	kurz modelu
δ	podélný sklon
γ	příčný sklon
n	otáčky hlavního rotoru

Údaje označené * jsou přepočítány do souřadného systému dle normy GOST. Údaje s indexem DC nejsou přepočítávány do standardního tvaru (radiány, metry) a zůstávají v dílcích.

Výstupní kanály Rozmístění výstupních kanálů je skoro totožné s umístěním výstupních registrů. Byly vynechány pozice řídících registrů. Jejich označení ukazuje následující tabulka:

1	Podélná cyklika
2	Příčná cyklika
3	Kolektivní řízení
4	Nastavení listů ocasního rotoru
5	Výkon motoru

11.3.3 Omezení vzorkovací periody

Vzhledem k již výše popsanému sériovému způsobu přenosu dat od snímačů do vnějších komunikačních registrů nemá smysl zbytečně rychlá doba vzorkování, která je srovnatelná s rychlostí sériového přenosu.

Základní frekvence sériového vysílače se pohybuje kolem 156kHz. Vysílač cyklicky posílá na sériovou linku pakety o délce 32bitů. Za 1 ms je odesláno přibližně 5 paketů paralelně ve dvou sériových kanálech. Úspěšnost

přenosu by se mohla teoreticky blížit ke 100%. V každém kanálu jsou cyklicky vysílány čtyři různé registry.

Celý sériový přenos byl navržen pro minimální vzorkovací periodu 1ms. Zvyšování rychlosti pod 1ms již nemá žádný praktický význam. V takovém případě jsou opakovaně čteny hodnoty z registrů a nikoliv skutečný stav modelu. Tím se zbytečně snižuje výpočetní výkon použitého počítače.

Jelikož je potřeba provádět rekonstrukci vyšších bitů polohy (poloha je udávána mod 8), je vzorkovací perioda omezena i z druhé strany. Její maximální hodnota je dána dynamikou modelu. **Překročení maximální doby** má však na regulační proces daleko horší následky. **Dojde k rozpadu rekonstruovaného stavu** a vzniku nesmyslných údajů o poloze.
Protože zatím není známa přesná dynamika modelu (identifikace modelu nebyla provedena), uvádím pro orientaci následující tabulku:

T	ψ	l	δ,γ,α,β
1ms	3750 ot/min	13.7 m/s	372 rad/s
10ms	375 ot/min	1.37 m/s	37 rad/s
100ms	37.5 ot/min	0.37 m/s	3.7 rad/s

V tabulce jsou uvedeny maximální rychlosti změn jednotlivých souřadnic, při dané periodě vzorkování. Je uvažován nejhorší případ. Při výpočtu souřadnic x,y,z je délka l přepočítávána při čtení každé souřadnice. Největší rychlost změny náklonů může nastat při posunu modelu v ose x v malé výšce a při visení (letu) již bude docházet jen k malým změnám. Podle provedených experimentů při periodě vzorkování 10ms probíhalo měření vzdáleností korektně.

Po implementaci funkce periodického volání driveru z jádra RT Toolboxu toto omezení již nebude platit. Aktualizace hodnot v registrech bude probíhat automaticky při periodickém volání vnitřní občerstvovací funkce.

11.4 Diagnostický program

Protože se jedná o velmi složité vzájemné propojení HW a SW, vyhledávání možných chyb v zapojení bude velmi obtížné. Za účelem zjednodušení právě této činnosti byl vyvinut speciální testovací program. Někdy se může jednat 'pouze' o nevhodné propojení kabelů. Tato 'banální' chyba však ve svém důsledku způsobí nefunkčnost celého zařízení.

11.4.1 OvládÁní programu

Vzhledem ke zvýšeným nárokům na přesné měření času a rychlé odezvy je program navržen pro prostředí DOSu.

Pro komunikaci s uživatelem byla zvolena dnes již klasická koncepce menu-myš. Po spuštění programu je přibližně uprostřed obrazovky otevřeno hlavní menu. Pomocí šipek lze vybírat v menu jednotlivé dílčí funkce a vybranou akci potvrdit stiskem klávesy Enter. Stejného efektu lze dosáhnout

stiskem levého tlačítka myši na požadované položce. Na posledním řádku je zobrazena nápověda povolených akcí.
Lze zvolit jednu z následujících činností:

Nastavení Umožní změnit způsob připojení k počítači.

Testování HW Provede kompletní testování HW části.

Monitorování Naměří graf závislosti přenosové rychlosti sériových kanálů na čase.

Informace Zobrazí informace o autorovi programu

Předčasné ukončení programu DOS

Konec

11.4.2 Popis prováděných testů

V této kapitole jsou popsány a rozebrány jednotlivé testy celého zařízení tak, jak jsou implementovány v programu TEST.PAS. Jejich hlavním účelem je usnadnění a zrychlení lokalizace vzniklé závady.

Testy mohou mít po svém provedení 3 možné výsledky: OK, ERROR, SKIP. Po úspěšném provedení testu je výsledný stav OK. Stav SKIP je použit v případě, nemá-li provedení testu smysl. Test je v takovém případě přeskočen (pokud všechny předchozí testy nedopadnou správně, není vhodné spouštět motor). Při zjištěné chybě je hlášen stav testu ERROR. Program v takovém případě na obrazovce nabídne možnou příčinu chyby.
Jedná se o následující testy:

Test PCL-812

Je proveden prázdný převod z jednoho analogového kanálu. Při převodu jsou kontrolovány změny hodnot v jednotlivých řídících registrech karty. Test je přeskočen v případě, pokud je propojení s modelem provedeno pomocí jiné rozšiřující karty.

Inicializace I8253

Využívá skutečnosti, že při zápisu na výstupní datovou sběrnici se po dobu sběrnicového cyklu zapisovaná data objevují i na vstupní datové sběrnici. Činnost externí sběrnice byla popsána v kapitole *Způsob připojení k PC* na straně 52.

Pokud při testu dojde k chybě, znamená to nejčastěji nesprávné připojení spojovacích kabelů. Spojovací kabely jsou bud špatně zasunuty nebo zapojeny obráceně. Chyba je také hlášena v případě špatného nastavení adresy.

Test vnějších registrů

Registry programovatelného obvodu xilinx č.11-15 obsahují identifikační řetězec 'PK&JF'. V programu je proveden pokus o přečtení těchto registrů a získané hodnoty jsou porovnány s požadovanými.

K chybě dojde v případě vypnutého napájecího napětí, chybějícího popř. vadného obvodu xilinx nebo pokud není zaveden program do obvodu.

Test sériového kanálu 1

Při komunikaci se snímači jsou přenášeny údaje do externích registrů dvěma sériovými kanály. Každý z kanálů má svůj vlastní registr, který je inkrementován po příchodu platného paketu. Je měřen počet paketů, který bude přijat během 1s.

Nejčastější závadou bývá nepřipojení propojovacích vodičů mezi vysílačem a přijímačem. Tato chyba však může znamenat také nefunkčnost

vysílače. První kanál spojuje model vrtulníku s přijímacími registry. Část je vedena vzduchem a proto mohlo dojít i k poruše vysílače.

Test sériového kanálu 2

Sériový kanál 2 je funkčně totožný s kanálem 1. Spojuje spodní díl trenažéru s externími registry přijímače.

Test Motoru

Otáčky motoru jsou snímány a předávány sériovým kanálem do počítače. Testovací program nastaví malý výkon motoru a přečte z registru naměřené otáčky. Test projde v pořádku pokud se do určité doby motor roztočí. Tento test je ze všech nejkomplexnější. Při jeho úspěšném provedení je pravděpodobně HW část v pořádku.

Naopak chyba může znamenat nepropojený kabel mezi vysílačkou a blokem registrů, nefunkční vysílač, zničené výkonové spínací tranzistory, chybu snímače otáček nebo chybnou činnost sériového kanálu 1.

11.4.3 Monitorování činnosti kanálů

Testování kanálů pouze zjistí činnost sériového kanálu v určitém okamžiku. Popisovaný režim umožňuje navíc sledovat činnost obou kanálů v čase.

Počet přijatých paketů je sčítán a vždy po 0.5s vynesen do grafu. Během měření lze přidávat rušení a zjišťovat jeho odezvu na kvalitu sériového přenosu. K tomuto účelu jsou vyhrazeny klávesy 0 .. 9, které spouští hlavní motor s odstupňovaným výkonem 0 - 90%. Stisk kterékoliv jiné klávesy má za následek vypnutí motoru. Návrat do předchozího menu lze uskutečnit stiskem klávesy Esc.

12 Závěr

Mým hlavním zadaným úkolem bylo realizovat návrh a provést optimalizaci programových algoritmů rychlé obousměrné komunikace mezi řídícím počítačem a modelem vrtulníku. Vlastní rychlost komunikace je ovlivňována mnoha faktory, z nichž optimalizace komunikační procedury s modelem má na celkovou rychlost minimální vliv. Rychlost čtení nemůže být vyšší než rychlost vzájemné komunikace a je tedy závislá na souběžně vyvíjeném HW. Další faktory, jako je vnitřní optimalita RT systému a rychlost vnější sběrnice, již nelze jednoduše ovlivnit.

Pro dokončení modelu bylo nutno vykonat mnohem větší rozsah prací. Podílel jsem se zčásti na návrhu HW a způsobu programování logických obvodů Xilinx. Hardwarová a mechanická část byla časově mnohem náročnější. Kritické mechanické části jsme si nechali vyrobit mechanikem. Výrobní doby jsou tak dlouhé, že se nám nepodařilo úplně mechanicky dokončit celou konstrukci trenažéru. Za úspěch považuji dovedení trenažéru alespoň do současného stavu.

V laboratoři automatického řízení na ČVUT FEL jsou již pro některé typy regulačních úloh používána ustálená řešení a nemá smysl na nich cokoliv měnit. Proto jsem se při řešení takových částí uchýlil pouze k popisu funkce bez diskuse o zvoleném řešení. Jedná se zejména o použití jazyku MATLAB pro zapisování regulačních úloh a o jeho nadstavbu RT Toolbox včetně formátů jeho ovladačů.

Model vrtulníku se šesti stupni volnosti představuje vyvrcholení vývoje modelů vrtulníků týmem ing. Horáčka Csc. Předchozí modely měly pouze dva stupně volnosti. Používaný způsob uchycení výrazně zhoršoval a modifikoval dynamické vlastnosti. V případě zdárného dokončení tohoto stupně by mohla následovat výstavba neupoutaného modelu.

Moje úloha představuje jen malý zlomek celkového množství prací, které je potřeba vykonat na cestě od návrhu modelu, přes jeho realizaci, identifikaci až k návrhu regulátoru. Pokud se tento základ neudělá dostatečně robustní, bude potřeba neustále se vracet k HW části a korigovat vzniklé nedostatky. S velkou lítostí jsme (spolu s autorem HW části) vyhazovali několik komponent vyrobených našim předchůdcem pro jejich ne zcela korektní funkci. Svoji část jsem se snažil udělat co nejlépe jak to bylo v daném okamžiku možné. Do jaké míry se mi to podařilo nechť posoudí ti, kteří budou s modelem dále pracovat.

Chtěl bych všem svým nástupcům popřát hodně úspěchů při návrhu velmi složitých regulačních algoritmů. Ověřování takových algoritmů by měl umožnit právě dokončený model částečně upoutané helikoptéry.

13 Příloha Výpis RT driveru

13.1 Výpis RT driveru pro DOS

Zdrojový kód programu je napsán v Turbo Assembleru. Vlastní překlad probíhá ve třech krocích. Nejprve je překládán soubor .ASM do .OBJ, poté je linkován do formátu .EXE a nakonec je provedena konverze do binárního tvaru. Soubor .EXE nelze přímo spustit, protože program nemá definován vstupní bod. Způsob překladu ukazuje dávkový soubor:

```
tasm heli /m
tlink heli
exe2bin heli.exe heli.mtb
@del *.obj
```

Zdrojový text programu:

```
.8086
.8087
_TEXT       segment byte 'CODE'
            assume      cs:_TEXT,ds:_TEXT

            org     0                   ;header definition

hwname              db      'Real Helikopter 3d Model',0

            org     20h

option              dw      0
inchn       dw      7
outchn              dw      5
address             dw      220h
dmarq       dw      1
dmaspeed    dw      35
reserved    dw      2 dup (0)

functions   dw      hwdisable
            dw      hwenable
            dw      inword
            dw      outword
            dw      inscan

      CWR_1 = 3                 ;addreses of CWR registers
      CWR_2 = 7
```

```
;*****************************************************************************
;*
;*              HWDISABLE
;*              disable hardware of the card
;*
;*              Input:          none
;*              Output:         none
;*
;*****************************************************************************
hwdisable       proc    c far
                mov     ax,5            ;implicit output values setting
                mov     dx,-7FFFh
                call    outword
                mov     ax,4
                mov     dx,-7FFFh
                call    outword
                mov     ax,3
                mov     dx,0
                call    outword
                mov     ax,2
                mov     dx,0
                call    outword
                mov     ax,1
                mov     dx,0
                call    outword
                mov     ax,0
                mov     dx,0
                call    outword
                ret
                endp
;*****************************************************************************
;*
;*              HWENABLE
;*              enable hardware of the card
;*
;*              Input:          AX = data segment alias of this code
;*                      DX = loop count per 1 millisecond
;*              Output:         AX = 0 installed OK
;*                      AX > 0 installation error
```


```
;*
;*****************************************************************************

hwenable    proc    c far uses ds
            mov     ds,ax
            mov     dseg,ax

            mov     dx,address          ;initialization CWR registers
            add     dx,14
            mov     al,8
            out     dx,al
            mov     cx,256*CWR_1 + 32h
            call    WriteCWR
            mov     cl,72h
            call    WriteCWR
            mov     cl,0B2h
            call    WriteCWR
            mov     cx,256*CWR_2 + 32h
            call    WriteCWR
            mov     cl,72h
            call    WriteCWR
            mov     cl,0B2h
            call    WriteCWR

            mov     ax,5                ;implicit output settings
            mov     dx,-7FFFh
            call    outword
            mov     ax,4
            mov     dx,-7FFFh
            call    outword
            mov     ax,3
            mov     dx,0
            call    outword
            mov     ax,2
            mov     dx,0
            call    outword
            mov     ax,1
            mov     dx,0
            call    outword
            mov     ax,0
```

```
                mov     dx,0
                call    outword

                mov     cx,5            ;test of identification string
                mov     si,offset Identif+4
IdLoop:         mov     al,cl
                add     al,10
                call    ReadChannel
                cmp     ah,byte ptr ds:si
                jne     hwenerror       ;if is not correct abort loading
                dec     si
                loop    IdLoop

                xor     ax,ax           ;driver succesfully loaded
                ret

hwenerror:      mov     ax,-1           ;error while loading driver
                ret

                endp


;*****************************************************************************
;*
;*              INWORD
;*              read single word from channel
;*
;*              Input:          AX = channel number
;*              Output:         AX = value read
;*
;*****************************************************************************

inword                  proc    c far   uses si ds
                add     dx,7FFFh
                mov     ds,word ptr cs:[dseg]
                mov     si,offset Trans
                add     si,ax
                lodsb
                mov     ah,al
```


```
                mov     dx,address              ;card address
                add     dx,14
                or      al,090h
                out     dx,al
                mov     dx,address
                add     dx,6
                in      al,dx                   ;input value
                mov     bl,al
                mov     bh,al
                mov     al,ah
                mov     dx,address
                add     dx,14                   ;card address
                or      al,0D0h
                out     dx,al

                mov     al,ah
                mov     ah,0
                mov     si,offset OldValues     ;compare with old value measured
                add     si,ax
                sub     bl,byte ptr ds:SI
                mov     byte ptr ds:SI,bh
                mov     cl,-1
                jnc     NoNeg
                mov     cl,1
                neg     bl
NoNeg:          or      bl,bl
                mov     bl,0
                js      corr
                mov     cl,0
corr:           mov     si,offset Leaders       ;estimate Hi byte of word
                add     si,ax
                add     byte ptr ds:si,cl

                mov     ah,byte ptr ds:si       ;output value store to AX
                mov     al,bh
                ret
                endp

;*************************************************************************
;*
```

```
;*              OUTWORD
;*
;*              write single word to channel
;*
;*              Input:          AX = channel number
;*                      DX = value to write
;*              Output:         none
;*
;*****************************************************************************

outword         proc    c far   uses si ds
                add     dx,7FFFh
                mov     ds,word ptr cs:[dseg]
                mov     si,offset Trans                 ;transfomation channel numbers
                add     si,ax
                lodsb
                mov     ah,al
                mov     cl,dh
                                                        ;do output write sequence
                mov     dx,address
                add     dx,14                           ;card address
                or      al,0C0h
                out     dx,al
                dec     dx                              ;DX=base+13
                mov     al,cl
                out     dx,al                           ;output value
                inc     dx
                mov     al,ah
                or      al,40h
                out     dx,ax
                xor     al,80h
                out     dx,ax
                dec     dx                              ;DX=base+13
                mov     al,1                            ;Hi byte must be set to 1
                out     dx,ax
                inc     dx                              ;DX=base+14
                mov     al,ah
                or      al,40h
                out     dx,al
                xor     al,80h
```


```
                out     dx,al
                ret

                endp

;*************************************************************************
;*
;*              INSCAN
;*              read block of words from channel
;*
;*              Input:          AX = channel number
;*                      DX = sampling period in us
;*                      BX = sample count
;*                      ES:DI = pointer to data
;*                      CX:SI = physical address of data (for the DMA)
;*              Output:         AX = 0 if success, -1 if error or not supported
;*
;*************************************************************************

inscan          proc    c far           ;This function is not supported
                mov     ax,-1
                ret

                endp

;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
;INPUT CH - channel number
; CL - value to write
WriteCwr proc near

        mov     al,ch                   ;channel
        or      al,0C0h
        mov     dx,address
        add     dx,14                   ;card address
        out     dx,al
        mov     al,cl                   ;value
        dec     dx
        out     dx,al
        mov     al,ch                   ;channel
        or      al,40h
        inc     dx
        out     dx,al
```

```
        xor     al,80h
        out     dx,al
        ret
        endp

;INPUT AL − channel number
; AH − value read

ReadChannel proc near
            mov     ah,al
            mov     dx,address      ;card address
            add     dx,14
            or      al,090h
            out     dx,al
            mov     dx,address
            add     dx,6
            in      al,dx           ;input value
            xchg    al,ah
            mov     dx,address
            add     dx,14           ;card address
            or      al,0D0h
            out     dx,al
            ret
            endp

OldValues   db      7 dup (0)           ;Old input values
Leaders     db      7 dup (0)           ;Hi byte's of input channels

trans       db      0,1,2,4,5,6,3,7     ;channels places descriptions
dseg        dw      0                   ;data segment alias
Identif     db      'PK&JF'             ;HW identification string

_TEXTends

        end
```

13.2 Výpis RT driveru pro Windows

Program je zapsán v jazyce Watcom C++ a překládán do 32 bitového kódu překladačem *WCC386*. Pro překlad je zapotřebí soubor *RTDSTUB.ASM* a include soubor *RTDRV.H* (oba jsou používány interně a nejsou součástí dodávky RT Toolboxu).

První dvojslovo ovladače musí obsahovat ukazatel na hlavičku obsaženou kdekoliv uvnitř vlastního kódu. Při linkování přeloženého driveru spolu se souborem *RTDSTUB.ASM* vznikne na počátku kódu požadovaný ukazatel. Soubor RTDSTUB.ASM

```
.386
_TEXT segment dword public 'CODE'
                assume      cs:_TEXT
                extrn   _Header: near
                org     0
headptr:        dd      _Header
_TEXT ends
                end     headptr
```

Při spuštění dávky se předpokládá instalace překladače watcom v adresáři *D:\WAT*. Je-li umístěn jinde je nutno změnit odpovídající odkazy ke knihovnám v tomto souboru a v souboru WLINK.RSP. Dávkový soubor pro překlad ovladače:

```
d:\bp\bin\tasm /l/m/mx/z rtdstub.asm rtdstub.obj
d:\wat\wcc386 -d1 -5r -7 -zq -zl -wx -onatx -Id:\wat\h -Id:\matlab\extern\include
heli
d:\wat\wdisasm -s=wheli.obj -l=wheli.lst heli d:\wat\wlink FILE
rtdstub.obj,heli.obj NAME heli.rtd @wlink.rsp
```

Definiční soubor linkeru WLINK.RSP:

```
FORMAT pharlap rex
OPTION caseexact, quiet
DISABLE 14
LIBRARY d:\wat\lib386\math387r,
d:\wat\lib386\dos\clib3r
```

Výpis programu HELI.C:

```
#define RTKERNEL
#include "rtdrv.h"
#define MINSCANPERIOD 35
#define CWR_1 3
#define CWR_2 7
#define Base Header.Address
#define k1 0.002908882                  //axis transformation constants
#define k2 0.0030679615
#define k3 0.053689327

double Input1(int ch);                  //preddeclaration input functions
double Input2(int ch);
double Input3(int ch);
double Input4(int ch);
```

```
Driver Header = {               //header definition
                "Real 3D Helicopter model",
                32,
                18,
                0x220,
                Disable,
                Enable,
                Input1,
                Output,
                InScan
                };

static unsigned short int DO_Hi=0x220+14,DO_Lo=0x220+13,DI_Lo=0x220+6;
static int Option = 0;
unsigned char trans[8]={0,1,2,4,5,6,3,7};
unsigned char transI[7]={4,5,6,0,1,2,4};
unsigned char OldValues[7];
signed char Leaders[7]={0,0,0,0,0,0,0};
/*****************************************************************************
;*
;*              DISABLE
;*              disable hardware of the card
;*
;*              Input:          none
;*              Output:         none
;*
;*****************************************************************************/
void Disable(void)
{
if(Base!=0)                             /* Disable analog part PCL-812 card */
{
outp(Base+11,0);                        /* set disabled mode */
outp(Base+8,0);                         /* reset any interrupt request */
inpw(Base+4);                           /* dummy read from A/D */
outpw(Base+4,0x800);                    /* reset analog outputs */
outpw(Base+6,0x800);
}
write_channel(0,128);                   /* initialize digital outputs */
write_channel(1,128);                   /* to default value */
```


```
write_channel(2,128);
write_channel(3,128);

write_channel(4,0);                         /* stop motor */
write_channel(5,0);
}

/*****************************************************************************
;*
;*              ENABLE
;*              enable hardware of the card
;*
;*              Input:          np = number od parameters
;*                      parm = pointer to parameter array
;*                      callback = ptr to kernel callback
;*              Output:         0 if success, error code otherwise
;*
;*****************************************************************************/
int Enable(int np, double *parm, int (*callback) (int fn,...))
{
int i;
usearg(callback);

switch (np)                             /* read parameters */
{
case 0: break;

case 1: {                               /* 1 argument */
        Option=parm[0];                 /* set input transformation */
        ChangeInput();
        break;
        }
case 2: {                               /* 2 arguments */
        Option=parm[0];                 /* set input transformation */
        Base=parm[1];                   /* and baze address */
        ChangeInput();
        break;
        }
case 3: {                               /* 3 arguments */
        Base=0;
        DO_Hi=parm[0];                  /* set all port locations */
        DO_Lo=parm[1];
        DI_Lo=parm[2];
```

```
        break;
        }
case 4: {                           /* 4 arguments */
        Option=parm[0];             /* set input transformation */
        Base=0;
        DO_Hi=parm[1];              /* and set all port locations */
        DO_Lo=parm[2];
        DI_Lo=parm[3];
        ChangeInput();
        break;
        }
default:return(ERR_MANYINARG);      /* incorrect number of arguments */
}

if(Base!=0)                         /* Test present PCL−812 card */
{

Disable();                          /* disable hardware */
outp(Base+11,1);                    /* enable software trigger */
if (!(inp(Base+5) & 0x10))          /* if data ready, error */
        return(ERR_INITHW);

outp(Base+12,0);                    /* try the conversion */
for (i=0;i<0x10000;i++)
        if (!(inp(Base+5) & 0x10)) break;

if (i==0x10000)                     /* if no conversion, error */
        return(ERR_INITHW);

inpw(Base+4);

if (!(inp(Base+5) & 0x10))          /* if data still ready, error */
        return(ERR_INITHW);
}

Header.Inputs=7;                    /* set number of I/O */
Header.Outputs=5;

outp(Base+14,8);                    /* Test of Main HW of Helicoptera model*/
if(write_CWR(CWR_1,0x32)) return(ERR_INITHW);

write_CWR(CWR_1,0x72);              /* initialize serial communicaton */
write_CWR(CWR_1,0xb2);
write_CWR(CWR_2,0x32);
write_CWR(CWR_2,0x72);
write_CWR(CWR_2,0xb2);
```


```
write_channel(0,128);                   /* initialize digital outputs */
write_channel(1,128);
write_channel(2,128);
write_channel(3,128);
write_channel(4,0);

write_channel(5,0);                     /* Not used, must be initialized too. */

for(i=0;i<7;i++) OldValues[i]=read_channel(i);
return(ERR_OK);
}

/*****************************************************************************
;*
;*              INPUT
;*              input from the card
;*
;*              Input:          ch = channel number
;*              Output:         input value
;*
;*****************************************************************************/

double Input1(int ch)           //ISO system
{
switch(ch)
        {
        case 0:return(k3*check_channel(6)*sin(k1*check_channel(5)));
        case 1: return(k3*check_channel(6)*cos(k1*check_channel(5))*

                sin(k1*check_channel(4)));
        case 2:return(k3*check_channel(6)*cos(k1*check_channel(5))*
                cos(k1*check_channel(4)));
        case 3:return(k2*check_channel(0));
        case 4:return(k1*check_channel(1));
        case 5:return(k1*check_channel(2));
        case 6:return(k4*check_channel(3));
        }
return(0);
}
double Input2(int ch) //Ghost System
{
switch(ch)
        {
        case 0:return(k3*check_channel(6)*sin(k1*check_channel(5)));
        case 1:return(k3*check_channel(6)*cos(k1*check_channel(5))*
```

```
                cos(k1*check_channel(4)));
        case 2:return(-k3*check_channel(6)*cos(k1*check_channel(5))*
                sin(k1*check_channel(4)));
        case 3:return(k2*check_channel(0));
        case 4:return(k1*check_channel(1));
        case 5:return(k1*check_channel(2));
        case 6:return(k4*check_channel(3));
        }
return(0);
}

double Input3(int ch)           //alpha system
{
switch(ch)
        {
        case 0:return(k1*check_channel(4));
        case 1:return(k1*check_channel(5));
        case 2:return(k3*check_channel(6));
        case 3:return(k2*check_channel(0));
        case 4:return(k1*check_channel(1));
        case 5:return(k1*check_channel(2));
        case 6:return(k4*check_channel(3));
        }
return(0);
}

double Input4(int ch)           //Dc only
{
if(ch>=7) return(0);
return(check_channel(transI[ch]));
}
/*****************************************************************************
;*
;*              OUTPUT
;*              output from the card
;*
;*              Input:          ch = channel number
;*                      val = value to output
;*              Output:         none
;*
;*****************************************************************************/
void Output(int ch, double val)
{
short sval;
```


```
if(val> .97) val=.97;                   /* Limit output values range */
if(val<−.97) val=−.97;
sval=val*128+128;
write_channel(ch,sval);
}

/*****************************************************************************
;*
;*              INSCAN
;*              read block from channel
;*
;*              Input:          ch = channel number
;*                      len = sample count
;*                      spec = driver specific options
;*                      spec[0] = sampling period
;*                      buf = address of buffer
;*              Output:         0 if success, error code otherwise
;*
;*****************************************************************************/
int InScan(int ch, int len, double *spec, double *buf, int *cols)
{
usearg(ch);
usearg(len);
usearg(spec);
usearg(buf);
usearg(cols);

return(ERR_INVFUNCTION);                /* This function is not supported */
}

/***********************Help procedures**************************/

void ChangeInput(void)                  /* modify input procedure */
{
if(Base!=0)
        {

        DO_Hi=Base+14;                  /* set communication port */
        DO_Lo=Base+13;
        DI_Lo=Base+6;
        }
switch (Option & 0x7)
{
```

```
case 1: Header.Input=Input1;break;          /* set input transformation */
case 2: Header.Input=Input2;break;
case 3: Header.Input=Input3;break;
case 4: Header.Input=Input4;break;
}
}

void refresh(void)                          /* This will be callback procedure */
{                                           /* and will periodicaly refresh */
check_channel(0);                           /* all registers */
check_channel(1);
check_channel(2);
check_channel(3);
check_channel(4);
check_channel(5);
check_channel(6);
}

int write_CWR(char kanal,char hodnota)      /* write walue to CWR register */
{                                           /* of IC 8253 */
outp(DO_Hi,0xC0|kanal);
outp(DO_Lo,hodnota);
outp(DO_Hi,0x40|kanal);
if(inp(DI_Lo)!=hodnota) return(1);
outp(DO_Hi,0xC0|kanal);
return(0);
}

                                            /* write walue to digital channel */
void write_channel(char kanal,char hodnota)
{
kanal=trans[kanal];
outp(DO_Hi,0xC0|kanal);
outp(DO_Lo,hodnota);
outp(DO_Hi,0x40|kanal);
outp(DO_Hi,0xC0|kanal);
outp(DO_Lo,1);
outp(DO_Hi,0x40|kanal);
outp(DO_Hi,0xC0|kanal);
}

int read_channel(char kanal)                /* read walue from digital channel */
{
int i;
outp(DO_Hi,0x90|kanal);
i=inp(DI_Lo);
outp(DO_Hi,0xD0|kanal);
```


```
return(i);
}

int check_channel(int ch)                    /* read walue from digital channel */
{                                            /* and estimate high byte */
int j;
j=read_channel(ch);
if(abs(j−OldValues[ch])>128)

        {                                    /* high byte was changed */
        if(j>OldValues[ch]) Leaders[ch]−=1;
                else Leaders[ch]+=1;
        }
OldValues[ch]=j;
return(((256*Leaders[ch]+j)));
}
```

13.3 Výpis grafické nadstavby driveru

Program HELI.M je zapsán v jazyce MATLABu. Může být spuštěn pouze od verze MATLABu 4.0 pod Windows. Nižší verze neobsahují rozšiřující funkce pro práci s dialogovými okny. Jeho činnost byla podrobně popsána v kapitole *Konfigurace ovladače pomocí GUI*.

```
function heli(command)
% Heli − helicopter driver setup.
%
%       Heli displays GUI for setting parameters of driver for
%       Real 3D helicopter model
%

%       See also RTLOAD.
%

%       29−06−96 Jaroslav Fojtik
%       Copyright (c) 1995 F&FSoft.

if nargin==0
% Define figure parameters
drvname=rtdname_('heli.rtd');
figh=findobj('Type','figure','Name',drvname);
if ~isempty(figh)
figure(figh);
return;
end;
pos=get(0,'DefaultFigurePosition');
pos(3)=280; pos(4)=310;
```

```
figh=figure('Color',get(0,'DefaultUIControlBackgroundColor'),...
'MenuBar','none','Visible','off','NextPlot','new',...
'Name',drvname,'NumberTitle','off',...
'Position',pos,'KeyPressFcn','heli @key');
% Define host ports
x=10; y=265;
uicontrol('Style','text','Position',[x y+16 100 20],'HorizontalAlignment','Left',...
'String','Connection:');
uic(1)=uicontrol('Style','radio','Position',[x y 200 20],'String',' Connected
through PCL−812',...
'CallBack','rbutton;heli @cpcl812;','Value',1);
uic(2)=uicontrol('Style','radio','Position',[x y−20 200 20],'String',' Another
port connection',...
'CallBack','rbutton;heli @canother;');
% Define Address control
x=20; y=205;
uic(3)=uicontrol('Style','text','Position',[x y+18 75 20],'HorizontalAlign-
ment','Left',...
'String','Address:');
uic(4)=uicontrol('Style','edit','BackGroundColor',[1 1 1],'Position',[x y 30
20],'String',220,...
'CallBack','heli @addr;');
for i=1:5;
uic(4+i)=uicontrol('Style','check','Position',[x+60+30*i y 20 20],...
'CallBack','heli @addrb;');
uic(9+i)=uicontrol('Style','text','Position',[x+58+30*i y+18 20 20],'String',sprintf('A%1d',9−i));
end;
set(uic(8),'Value',1);
% Define system mode control
x=10; y=160;
uicontrol('Style','text','Position',[x y+18 100 20],'HorizontalAlignment','Left',...
'String','System change:');
uic(15)=uicontrol('Style','radio','Position',[x y 200 20],'String',' ISO sys-
tem',...
'CallBack','rbutton;heli @system1','Value',1);
uic(16)=uicontrol('Style','radio','Position',[x y−20 200 20],'String',' Gost
system',...
'CallBack','rbutton;heli @system2');
uic(17)=uicontrol('Style','radio','Position',[x y−40 200 20],'String',' Alpha
system',...
'CallBack','rbutton;heli @system3');
uic(18)=uicontrol('Style','radio','Position',[x y−60 200 20],'String',' DC
only',...
'CallBack','rbutton;heli @system4');
% Define Command line control
x=10; y=55;
uicontrol('Style','text','Position',[x y+18 100 20],'HorizontalAlignment','Left',...
```


```
'String','Command line:');
uic(19)=uicontrol('Style','text','BackGroundColor',[1 1 1],'HorizontalAlign-
ment','Left',...
'Position',[x y pos(3)−20 20]);
% Define buttons
x=30; y=10;
uicontrol('Style','push','Position',[x y 60 30],'String','OK',...
'CallBack','heli @ok');
uicontrol('Style','push','Position',[x+90 y 60 30],'String','Cancel',...
'CallBack','close');
uicontrol('Style','push','Position',[x+180 y 60 30],'String','&Default',...
'CallBack','heli @revert');
% Count port address
uic(20)=544;
uic(21)=uic(20)+13;
uic(22)=uic(20)+14;
uic(23)=uic(20)+6;
uic(24)=1;
set(figh,'Visible','on','UserData',uic);
drawnow;
else
uic=get(gcf,'UserData');
% Call if OK pressed
if strcmp(command,'@ok')
cline=get(uic(19),'String');
close;
eval(cline);
uic(1)=0;
% Call if Revert to Default pressed
elseif strcmp(command,'@revert')
uic(20)=544;
if get(uic(1),'Value')~=0,
set(uic(4),'String','220');
else
set(uic(1),'Value',1);
set(gcf,'UserData',uic);
heli('@cpcl812');
uic=get(gcf,'UserData');
end;
set(uic(15),'Value',1);
set(gcf,'UserData',uic);
heli('@system1');
uic=get(gcf,'UserData');
command='@addr';
% Call if system changed
elseif strcmp(command,'@system1')
```

```
set(uic(16                ,'Value',0);
set(uic(17),'Value',0);
set(uic(18),'Value',0);
uic(24)=1;
elseif strcmp(command,'@system2')
set(uic(15),'Value',0);
set(uic(17),'Value',0);
set(uic(18),'Value',0);
uic(24)=2;
elseif strcmp(command,'@system3')
set(uic(15),'Value',0);
set(uic(16),'Value',0);
set(uic(18),'Value',0);
uic(24)=3;
elseif strcmp(command,'@system4')
set(uic(15),'Value',0);
set(uic(16),'Value',0);
set(uic(17),'Value',0);
uic(24)=4;
% Call if connection changed
elseif strcmp(command,'@cpcl812')
v=get(uic(2),'Value');
if v~=0,
for i=3:8;
close(uic(i));
end;
set(uic(2),'Value',0);
x=20; y=205;
uic(3)=uicontrol('Style','text','Position',[x  y+18  75  20],'HorizontalAlign-
ment','Left',...
'String','Address:');
uic(4)=uicontrol('Style','edit','BackGroundColor',[1  1  1],'Position',[x  y  30
20],...
'CallBack','heli @addr;','String',dec2hex(uic(20)));
for i=1:5;
uic(4+i)=uicontrol('Style','check','Position',[x+60+30*i y 20 20],...
'CallBack','heli @addrb;');
uic(9+i)=uicontrol('Style','text','Position',[x+58+30*i y+18 20 20],'String',sprintf('A%1d',9-i));
end;
command='@addr';
end;
elseif strcmp(command,'@canother')
v=get(uic(1),'Value');
if v~=0,
for i=3:14;
close(uic(i));
```


```
end;
set(uic(1),'Value',0);
x=20; y=205;
uic(3)=uicontrol('Style','text','Position',[x  y+18  75  20],'HorizontalAlign-
ment','Left',...
'String','DO_Hi:');
uic(4)=uicontrol('Style','edit','BackGroundColor',[1  1  1],'Position',[x  y  30
20],...
'CallBack','heli @dio;','String',dec2hex(uic(21)));
uic(5)=uicontrol('Style','text','Position',[x+70  y+18  75  20],'Horizonta-
lAlignment','Left',...
'String','DO_Lo:');
uic(6)=uicontrol('Style','edit','BackGroundColor',[1  1  1],'Position',[x+70  y
30  20],...
'CallBack','heli @dio;','String',dec2hex(uic(22)));
uic(7)=uicontrol('Style','text','Position',[x+140  y+18  75  20],'Horizonta-
lAlignment','Left',...
'String','DI_Lo:');
uic(8)=uicontrol('Style','edit','BackGroundColor',[1 1 1],'Position',[x+140 y
30  20],...
'CallBack','heli @dio;','String',dec2hex(uic(23)));
end;
% Call if Address bits changed
elseif strcmp(command,'@addrb')
j=16;
a=0;
for i=1:5
a=a+get(uic(10−i),'Value')*j;
j=2*j;
end;
a=a+512;
uic(20)=a;
uic(21)=uic(20)+13;
uic(22)=uic(20)+14;
uic(23)=uic(20)+6;
set(uic(4),'Value',a);
set(uic(4),'String',dec2hex(a));
end;
% Call if Address field changed
if strcmp(command,'@addr')
a=hex2dec(get(uic(4),'String'));
uic(20)=a;
uic(21)=uic(20)+13;
uic(22)=uic(20)+14;
uic(23)=uic(20)+6;
a=fix(a/16);
for i=1:5
```

```
set(uic(10−i),'Value',rem(a,2));
a=fix(a/2);
end;
%Call if port location changed
elseif strcmp(command,'@dio')
set(uic(4),'String',dec2hex(hex2dec(get(uic(4),'String'))));
set(uic(5),'String',dec2hex(hex2dec(get(uic(5),'String'))));
set(uic(6),'String',dec2hex(hex2dec(get(uic(6),'String'))));
% Call if key pressed
elseif strcmp(command,'@key')
ch=get(gcf,'CurrentCharacter');
if ch==13
cline=get(uic(19),'String');
close;
eval(cline);
uic(1)=0;
elseif ch==27
close;
uic(1)=0;
elseif upper(ch)=='D',
heli @revert;
uic=get(gcf,'UserData');
end;
end;
if uic(1)~=0, set(gcf,'UserData',uic);end;
end;
%Display new Status line if changed
if uic(1)~=0,
set(uic(19),'String','rtload("heli")');
if uic(20)~=544,
set(uic(19),'String',['rtload("heli",',num2str(uic(20)),')',]);
end;
if uic(24)~=1,
set(uic(19),'String',['rtload("heli",',num2str(uic(20)),',',num2str(uic(24)),') ',]);
end;
if get(uic(2),'Value')==1,
set(uic(19),'String',['rtload("heli",0,[',num2str(uic(24)),',',num2str(uic(21)) ,',',...
num2str(uic(22)),',',num2str(uic(23)),'])',]);
end;
end;
```

14 Příloha Programy pro konverzi schémat

Programy jsou napsány v jazyce Pascal. Nemají žádné speciální nároky na kompilaci. Mohou být přeloženy do reálného i chráněného režimu a po drobných úpravách i do prostředí Windows. V posledních dvou variantách však nevidím žádný přínos, neboť HW požadavky obou programů jsou ve srovnání s ostatními programy zanedbatelné.

14.1 Výpis programu CorrXNO

Program slouží pro opravu formátu XNF. Jeho činnosti je věnována samostatná podkapitola. Z jednotky *WIND2* je použita funkce *Assign-ToString*, která umožní pracovat s proměnnou typu string stejně jako se souborem.

```
{$i−}
{$i MODIF.INC}
Program CorrXno(Input,Output);
uses wind2,dos,{$ifdef crtx}crtx;{$else}crt;{$endif}
type Tpin=record {struktura popisující atributy pinu}
        oznaceni:string[4];
        nazev:string[12];
        nozicka:string[5];
        end;
FnameStr=string[79];
var a,b:string;
i,j:integer;
txt,txt2:text;
piny:array[1..100] of Tpin;
maxpin:byte;
Label 40;
Function GetcharTXT(var txt:Text):char;          {přečti jeden znak ze souboru}
var pismeno:char;
begin
read(txt,pismeno);
GetcharTXT:=pismeno;
end;

Function Upcases(s:string):string;          {převeď celý řetězec na velká písmena}
var i:byte;
begin
for i:=1 to length(s) do s[i]:=UpCase(s[i]);
Upcases:=s;
end;
Procedure ChangeExt(var a:string;NewExt:ExtStr);

var i:Byte;                                 {změna koncovky v názvu souboru}
begin
```

```
for i:=length(a) downto 1 do                {ořezej původní koncovku}
begin
if a[i]='.' then begin;dec(i);break;end;
if (a[i]='\')or(a[i]='/') then begin;i:=0;break;end;
end;
if i<2 then i:=Length(a);
a[0]:=chr(i);

a:=a+'.'+NewExt;                    {přidej na konec novou koncovku}
end;
Function ReadExt(const a:string):ExtStr;

var i:Byte;                                 {extrahuj příponu ze jména souboru}
s:ExtStr;
begin
s:='';

for i:=length(a) downto 1 do                {čti ji od konce z řetězce jména}
begin
if a[i]='.' then break;
if (a[i]='\')or(a[i]='/') then break;
if length(s)>=3 then break;

s:=a[i]+s;                                  {a ukládej do proměnné s}
end;
ReadEXT:=s;
end;
Function readFirstWord(var txt:text):string;{přečti první slovo ze souboru}
var c:char;
s:string;
label 1;
begin
s:='';

repeat                                      {dojede na počátek slova}
if eof(txt) then GoTo 1;
c:=GetcharTXT(txt);
until ((c<>' ') and (c<>#10)) and (c<>#13);
while ((c<>' ') and (c<>#10)) and (c<>#13) do
begin
s:=s+c;
c:=GetcharTXT(txt);
readFirstWord:=s;

if eof(txt) then                            {není-li konec souboru, načti další znak}
        begin
        s:=s+c;
        Goto 1;
        end;
```


```
end;
1:
readFirstWord:=s;
end;

Function readFirstInt(var f:text):Word;                {přečte integer ze souboru}
var c:char;
i:word;
begin
readFirstInt:=0;

repeat                                                 {dojede na počátek čísla}
if eof(f) then exit;
c:=GetcharTXT(f);
until (c >= '0') and (c <= '9');

i:=0;                                                  {tady se načítá číslo po cifrách}
repeat
i:=10*i + ord(c)−ord('0');
c:=GetcharTXT(f);
until not((c >= '0') and (c <= '9'));
readFirstInt:=i;
end;
Function readFirstInt2(var txt:Text;var i:Word):char;

var c:char;                                {Přečte integer ze souboru a vrátí znak za ním}
begin
I:=0;
readFirstInt2:='?';

repeat                                     {dojede na počátek čísla}
if eof(txt) then exit;
c:=GetcharTXT(txt);
until (c >= '0') and (c <= '9');

repeat                                     {tady začne načítat čílo}
i:=10*i + ord(c)−ord('0');
c:=GetcharTXT(txt);
until not((c >= '0') and (c <= '9'));
readFirstInt2:=c;
end;
Procedure Analyze(var s:string);           {provede rozbor jednoho řádku .XNF
souboru}
var b:string;
i:byte;
begin
AssignToString(s,Input);
reset(Input);
readfirstWord(Input);
b:=readfirstWord(Input);
if b[length(b)]=',' then dec(b[0]);
```

```
for i:=1 to maxpin do                    {prohledej všechny piny s definovanou pozicí}
        begin
        if piny[i].nazev=b then          {při shodě jmen doplň údaje o umístění pinu}
                begin
                s:=',, LOC='+piny[i].nozicka;
                exit;
                end;
        end;
close(Input);
s:=";
end;
Procedure LoadTab(const NameTab:string); {nahraje tabulku pozic pinů do
paměti}
label 10;
begin
assign(txt,NameTab);
reset(txt);

while not(eof(txt)) do                   {každý řádek popisuje jiný pin}
begin

inc(maxpin);                             {načti údaje o poloze pinu}
piny[maxpin].oznaceni:=UpCases(readfirstword(txt));
piny[maxpin].nazev:=";
piny[maxpin].nozicka:=readfirstword(txt);
if IOresult<>0 then

                begin
                dec(maxpin);
                goto 10;
                end;
end;
if (maxpin>1)and(piny[maxpin].oznaceni=")then dec(maxpin);
close(txt);

if IOresult<>0 then                      {při chybě vypiš varovnou zprávu}
        begin
10:     writeln('Se souborem ',NameTab,' nelze pracovat!');exit;
        end;
end;

                                         {podle .VST souboru vytvoř .TAB soubor}
Procedure RebuildVstTab(var NameVst:string);
var name:string[20];
Pval3rd:string[20];
uname:string[20];
```


```
c:char;
i:byte;
Label 1;
begin

assign(txt,NameVst);                {otevři vstupní .VST soubor}
reset(txt);

changeExt(Namevst,'TAB');           {otevři výstupní .TAB soubor}
assign(txt2,NameVst);
reWrite(txt2);

while not(eof(txt)) do                      {postupně procházej soubor po řádcích}
begin
name:=";
pval3rd:=";
uname:=";

i:=0;                                       {hledej řádky typu:IPAD(;?17;p29);U21}
c:=getchartxt(txt);

while c<>'(' do                             {načti název typu součástky}
begin
if ((c=#10)or(c=#13))or((c=' ')or(eof(txt))) then goto 1;
name:=name+UpCase(c);
c:=GetcharTXT(txt);
end;
if ((name<>'IPAD')and(name<>'OPAD'))and(name<>'PAD') then goto
1;

c:=GetcharTXT(txt);                         {načti 3 part value − pozici pinu}
while i<1 do
begin
if ((c=#10)or(c=#13))or((c=')')or(eof(txt))) then goto 1;
c:=GetcharTXT(txt);
if c=';' then inc(i);
end;
c:=GetcharTXT(txt);
while ((c<>')')and(c<>';'))and((c<>#10)and(c<>#13)) do
begin
if eof(txt) then goto 1;
Pval3RD:=Pval3rd+UpCase(c);
c:=GetcharTXT(txt);
end;

while c<>')' do                             {přečti slovo za koncem závorky}
begin                                       { − referenční číslo součástky}
if ((c=#10)or(c=#13))or(eof(txt)) then goto 1;
c:=GetcharTXT(txt);
end;
```

```
c:=getcharTXT(txt);
if c<>';' then goto 1;
readln(txt,uname);

writeln(txt2,uname,' ',pval3rd);                {do souboru .TAB zapiš další řádek}
{ writeln(uname,' ',pval3rd);}
1:
end;

close(txt);                                     {na konci uzavři oba soubory}
close(txt2);
if IOresult<>0 then
        begin
        writeln('Se souborem ',NameVst,' nelze pracovat!');exit;
        end;
end;
begin
maxpin:=0;
writeln(#10#13'<<< CorrXno >>> Corector pad"s placing');

if paramcount<4 then begin                      {tiskne nápovědu na obrazovku}
        writeln('Málo parametrů: stary.xno novy.xno tabulka.tab|schema.vst
stary.pin');exit;
        end;
a:=upcases(paramstr(3));

if ReadExt(a)='TAB' then LoadTab(a);            {při zadání .TAB tabulku rovnou nahraj}
if ReadExt(a)='VST' then begin                  {jinak musí být vytvořena z .VST souboru}
                         RebuildVstTab(a);
                         ChangeExt(a,'TAB');
                         LoadTab(a);            {nahraj nově vytvořenou tabulku}
                         end;

assign(txt,paramstr(4));
reset(txt);

while not(eof(txt)) do                          {postupně prochází .PIN soubor}
begin
readln(txt,a);
a[0]:=#5;
if a='(SYM,' then
        begin
        readln(txt,a);
        b:=copy(a,8,length(a));
        readln(txt,a);
        a:=copy(a,8,length(a));
        if ((a='IPAD')or(a='OPAD'))or(a='PAD') then
```


```
                begin                           {po nalezení proložky pinu}
                readln(txt,a);
                a:='1_1-'+copy(a,9,6);
                for i:=1 to MaxPin do{zjisti, je-li jeho číslo obsaženo v tabulce}
                        begin           {a zapamatuj si označení externího signálu}
                        if piny[i].oznaceni=b then piny[i].nazev:=a;
                        end;
                end;
        end;

end;
close(txt);

if IOresult<>0 then             {po chybě při práci se souborem .PIN ukonči program}
        begin
        writeln('Se souborem ',ParamStr(4),' nelze pracovat!');exit;
        end;

assign(txt,paramstr(1));                        {otevři opravovaný .XNF soubor}
reset(txt);
if IOresult<>0 then
        begin
        writeln('Se souborem ',ParamStr(1),' nelze pracovat!');exit;
        end;

assign(txt2,paramstr(2));                       {otevři korigovaný .XNF soubor}
rewrite(txt2);
if IOresult<>0 then
        begin
        writeln('Se souborem ',ParamStr(2),' nelze pracovat!');exit;
        end;

Readln(txt,a);                          {postupně po řádcích kopíruj .XNF soubor}
while copy(a,1,3)<>'EXT' do             {až do poslední části popisující externí signály}
begin
writeln(txt2,a);
Readln(txt,a);
if copy(a,1,4)='PIN,' then

        begin                           {oprav všechny invertované vnitřní signály}
        assignToString(a,Input);
        reset(Input);
        ReadFirstWord(Input);
        if Upcase(ReadFirstInt2(Input,word(i)))='B' then a:=a+',, INV';
        end;
if eof(txt) then goto 40;                       {při předčasném konci opusť program}
```

```
end;

while a<>'EOF' do                    {procházej seznam externích signálů}
begin                                {až do koncové značky EOF}
write(txt2,a);
write(#10#13,a);

Analyze(a);                          {analyzuj a popř. oprav načtený řádek}
write(a);

writeln(txt2,a);                     {a opravený ho zapiš zpátky do souboru}
Readln(txt,a);

if eof(txt) then goto 40;            {při předčasném konci opusť program}
end;
writeln(txt2,a);

40:;                          {na konci programu uzavři otevřené soubory}
close(txt);
close(txt2);
if IoResult<>0 then writeln('Nastala chyba při práci se souborem ',Param-
Str(2));

end.                                 {návrat do DOSu}
```

14.2 Výpis programu CorrLCA

Program slouží pro opravu formátu LCA. Jeho činnosti je věnována samostatná podkapitola.

```
{$i-}
Program CorrLca(Input,Output);
uses wind2,dos,crtx;
var a,b:string;
i,j:integer;
txt,txt2:text;
maxpin:byte;
Function GetcharTXT(var txt:Text):char;          {Přečte 1 znak ze souboru}
var pismeno:char;
begin
read(txt,pismeno);
GetcharTXT:=pismeno;
end;

Function Upcases(s:string):string;          {Provede UpCase pro celý řetězec}
var i:byte;
begin
for i:=1 to length(s) do s[i]:=UpCase(s[i]);
Upcases:=s;
end;
```


```
Function readFirstWord(var txt:text):string;          {Přečte 1 slovo ze souboru}
var c:char;
s:string;
label 1;
begin
s:=”;

repeat                                                {dojede na počátek slova}
if eof(txt) then GoTo 1;
c:=GetcharTXT(txt);
until ((c<>’ ’) and (c<>#10)) and (c<>#13);

while ((c<>’ ’) and (c<>#10)) and (c<>#13) do         {načítej slovo až do konce}
begin
s:=s+c;
c:=GetcharTXT(txt);
readFirstWord:=s;
if eof(txt) then

        begin                                         {konec souboru=konec slova}
        s:=s+c;
        Goto 1;
        end;
end;
1:
readFirstWord:=s;
end;

Function readFirstInt(var f:text):Word;               {Přečte integer ze souboru}
var c:char;
i:word;
begin
readFirstInt:=0;

repeat                                                {dojede na počátek čísla}
if eof(f) then exit;
c:=GetcharTXT(f);
until (c >= ’0’) and (c <= ’9’);

i:=0;                                                 {tady začíná číslo}
repeat
i:=10*i + ord(c)−ord(’0’);
c:=GetcharTXT(f);
until not((c >= ’0’) and (c <= ’9’));
readFirstInt:=i;
end;

begin                                                 {Počátek programu CorrLca}
maxpin:=0;
writeln(#10#13’<<< CorrLca >>> Corector tbuf”s placing’);
```

```
if paramcount<2 then begin                          {Tisk nápovědy}
        writeln('Málo parametrů: stary.lca novy.lca');exit;
        end;

assign(txt,paramstr(1));                            {Otevři vstupní soubor}
reset(txt);
if IOresult<>0 then
        begin
        writeln('Se souborem ',ParamStr(1),' nelze pracovat!');exit;
        end;
assign(txt2,paramstr(2));
rewrite(txt2);

if IOresult<>0 then                                 {Otevři výstupní soubor}
        begin
        writeln('Se souborem ',ParamStr(2),' nelze pracovat!');
        close(txt);
        exit;
        end;

Readln(txt,a);                                      {Kopíruj oba soubory}
while UpCases(copy(a,1,12))<>'NAMEBLK TBUF' do

begin                                 {Pokud se vyskytne řetězec 'NAMEBLK TBUF',}
writeln(txt2,a);                      {tak vynech celou řádku}
Readln(txt,a);
if eof(txt) then break;
end;

close(txt);                                         {kopírování hotovo − uzavři soubory}
close(txt2);
if IoResult<>0 then writeln('Nastala chyba při práci se souborem',

                    ParamStr(1));
end.
```

15 Příloha Výpis testovacího programu

Program TEST.PAS je napsán v jazyce Pascal. Je možno jej kompilovat do reálného nebo chráněného režimu. Vzhledem k potřebě měřit časové intervaly nedoporučuji pracovat v prostředí Windows a to ani v DOSovém okně. Program je k HW části modelu přiložen v přeloženém tvaru TEST.EXE.

Pro maximální usnadnění komunikace s uživatelem je použita knihovna WIND2.TPU, která provádí správu otevřených oken na obrazovce. Vzhledem k jejímu rozsahu a složitosti není součástí této práce. Pro další úpravy testovacího programu jsem ochoten případnému zájemci knihovnu poskytnout.

Pro zpřístupnění ovladače myši je využívána knihovna MOUSE.TPU, pro kterou platí to, co již bylo řečeno o knihovně WIND2.
Výpis programu TEST.PAS:

```
{$i modif.inc}
uses {$ifdef crtx}crtx,{$else}crt,{$endif}
{$ifdef mouse}mouse,{$endif}
wind2,dos,graph,graf6,PCL812;
var jas:byte;
DO_Hi,DO_Lo,DI_Lo:Word;
Base:Word;
const CWR_1=3;
CWR_2=7;
trans:array[0..6] of byte=(0,1,2,4,5,6,0);
OldValues:array[0..6] of byte=(0,0,0,0,0,0,0);
Leaders:array[0..6] of byte=(0,0,0,0,0,0,0);
k1=0.002908882;                    {kalibrační konstanty souřadného systémy}
k2=0.0030679615;
k3=0.053689327;
Function CheckWindows:Byte;
var r:registers;
begin
CheckWindows:=0;
r.CX:=0;
r.AX:=$160A;
intr($2f,r);
if r.AX<>0 then exit;
CheckWindows:=r.CX;
end;
Function Tg(x:real):Real;
begin
Tg:=sin(x)/cos(x);
end;

Function GetInterval:Longint; assembler;          {aktuální stav počítadla času}
asm
```

```
        MOV   AH,0
        INT   1Ah
        MOV   AX,DX
        MOV   DX,CX
end;

procedure speed(spid:Word); assembler;        {změna rychlosti interního časovače}
asm
cli
mov al,$34
out $43,al
mov dx,spid
mov al,dl
out $40,al
mov al,dh
out $40,al
sti
end;

Function write_CWR(kanal:Byte;hodnota:Byte):Boolean;              {přepiš CWR registr}
begin
write_CWR:=False;
Port[DO_Hi]:=$C0 or kanal;
Port[DO_Lo]:=hodnota;
Port[DO_Hi]:=$40 or kanal;
if(Port[DI_Lo]=hodnota) then write_CWR:=True;
Port[DO_Hi]:=$C0 or kanal;
end;

Function Dummy_write(hodnota:Byte):Boolean;       {přepiš CWR registr}
begin
Dummy_write:=False;
Port[DO_Hi]:=$C0 or 8;
Port[DO_Lo]:=hodnota;
Port[DO_Hi]:=$40 or 8;
if(Port[DI_Lo]=hodnota) then Dummy_write:=True;

{       else write(hodnota,':',i,' ');}
Port[DO_Hi]:=$C0 or 8;
end;

Procedure write_channel(kanal,hodnota:Byte);                {zapiš byte do kanály}
begin
kanal:=trans[kanal];
Port[DO_Hi]:=$C0 or kanal;
Port[DO_Lo]:=hodnota;
Port[DO_Hi]:=$40 or kanal;
Port[DO_Hi]:=$C0 or kanal;
Port[DO_Lo]:=1;
Port[DO_Hi]:=$40 or kanal;
Port[DO_Hi]:=$C0 or kanal;
end;
```


```
Function read_channel(kanal:Byte):Byte;                    {přečti byte z kanálu}
begin
Port[DO_Hi]:=$90 or kanal;
read_channel:=Port[DI_Lo];
Port[DO_Hi]:=$D0 or kanal;
end;

Function Check_channel(kanal:Byte):integer;                {odhadni vyšší byte
přečteného slova}
var j:integer;
begin
j:=read_channel(kanal);
if(abs(j−OldValues[kanal])>128) then

        begin                                              {došlo ke změně vyššího byte}
        if(j>OldValues[kanal]) then dec(Leaders[kanal])
                        else inc(Leaders[kanal]);
        end;
OldValues[kanal]:=j;
Check_channel:=256*Leaders[kanal]+j;
end;

Procedure WriteHelpLine(help:string);                      {tiskni spodní řádek
nápovědy}
var a:Byte;
begin
a:=TextAttr;
TextAttr:=16*lightgray+Black;
while length(help)<maxX do help:=help+' ';
writeXY(1,maxY,help);
TextAttr:=a;
end;

Procedure Header(const s:string);                          {záhlaví programu na
obrazovce}
var vm,vx:Word;
posx:integer;
begin
vm:=windmin;vx:=windmax;
TextAttr:=Green;
AktualniRamecek:=3;okno(1,1,MaxX,3);
posX:=(MaxX−length(s)−2) div 2;
if posX<=1 then posX:=2;
WriteXY(posX,2,s);
windmin:=vm;windmax:=vx;
end;

Function OsetriError(chyba,pricina:string):Byte;           {tisk hlášení o vzniklé
chybě}
var key:char;
```

```
ax,AoldX,x,y:byte;
begin
{$ifdef mouse}HideMouse;DelButtons;{$endif}
ChangeWindBox(2,TextAttr);
OpenWindow(30,12,37,11);
TextAttr:=16*Red;
InWindBox(3);
writeln(chyba,#13#10#10);
aOldx:=1;ax:=1;
y:=lo(windmax)−lo(windmin);
pricina:=pricina+' ';
for x:=1 to length(pricina) do
        begin
        if pricina[x]=' ' then

                begin
                if x−AoldX>y then
                begin
                writeln(copy(pricina,AOldX,Ax−AOldX));
                AoldX:=AX+1;
                end;
                ax:=x;
                end;
        end;
writeln(copy(pricina,AOldX,Length(pricina)−AOldX+1));
TextAttr:=16*Green+Yellow;
GotoXY(2,9);write (' Přeskočit ');
GotoXY(15,9);write(' Zrušit ');
GotoXY(25,9);write(' Opakovat ');
{$ifdef mouse}ShowMouse;{$endif}
Key:=#0;
repeat
inputsJob;
{$ifdef mouse}
if (MouseInArea(ActiveWindowˆ.X+2,ActiveWindowˆ.Y+9,11,1))=LeftButton

                then key:='P';
if (MouseInArea(ActiveWindowˆ.X+15,ActiveWindowˆ.Y+9,8,1))=LeftButton

                then key:='Z';
if (MouseInArea(ActiveWindowˆ.X+25,ActiveWindowˆ.Y+9,10,1))=LeftButton

                then key:='O';
{$endif mouse}
if keypressed then key:=upcase(chr(lo(readkeyEx)));
until key in ['P','Z','O'];
{$ifdef mouse}
while MousePos(x,y)=LeftButton do;
```


```
{$endif}
delay(1);
if key='P' then OsetriError:=1;
if key='Z' then OsetriError:=2;
if key='O' then OsetriError:=3;
CloseWindow;
ChangeWindBox(1,TextAttr);
end;

Procedure VriteInfo;                                {tiskne informace o programu}
var x,y:byte;
begin
OpenWindow(21,10,40,12);
InWindBox(1);
{$ifdef mouse}HideMouse;{$endif}
WriteHelpLine('Stiskni něco pro pokračovnání.');
TextAttr:=16*Red+Yellow;
InWindBox(1);
write(#10#13' Program TEST je určen pro ověření'+
#10#13' funkčnosti HW části modelu'+
#10#13' helikoptéry.'+
#10#10#13' SW design (c) 1995 Jaroslav Fojtík'+
#10#13' HW design (c) 1995 Pavel Krsek'+
#10#13#10,' ');
TextAttr:=Green;
write(' OK ');
{$ifdef mouse}
ShowMouse;
repeat
InputsJob;
if (MouseInArea(ActiveWindow^.X+17,ActiveWindow^.Y+9,4,1))=LeftButton
        then
        begin
        while MousePos(x,y)=LeftButton do;
        Press(Enter);
        end;
until keypressed;
{$endif}
ReadKeyEx;
CloseWindow;
ChangeWindBox(1,TextAttr);
if LastKeyPressed=Esc then LastKeyPressed:=0;
end;

Procedure Configure;                            {nastavení konfigurace zařízení}
const jas:byte=0;
var s:String;
j:word;
code:integer;
begin
ChangeWindBox(2,TextAttr);
repeat
```

```
if base=0 then OpenWindow(19,8,28,6)
          else OpenWindow(19,8,28,4);
InWindBox(1);
write('Připojení: ');
if Base<>0 then begin
                    write('Přes PCL812'#10#13'Bázová adresa:',Base);
                    DO_Hi:=Base+14;
                    DO_Lo:=Base+13;
                    DI_Lo:=Base+6;
                    end
          else write('Jiné'#10#13'Di_Lo:',DI_Lo:11,
          #10#13'DO_Lo:',DO_Lo:11,#10#13'DO_Hi:',DO_Hi:11);
menu(jas,nil);
if LastKeyPressed=Enter then
begin
if jas=0 then if base=0 then base:=DI_Lo-6
                              else base:=0
          else
          begin
          gotoXY(15,jas+1);
          TextAttr:=16*Brown+Yellow;
          clreol;
          readln(s);
          val(s,j,code);
          if code<>0 then OsetriError('Špatně zadané číslo',")
                    else
                    begin
                    case jas of 1:if base=0 then DI_Lo:=j
                                            else base:=j;
                              2:DO_Lo:=j;
                              3:DO_Hi:=j;
                              end;
                    end;
          end;
end;
CloseWindow;
until LastKeyPressed=Esc;
ChangeWindBox(1,TextAttr);
if base<>0 then
          begin
          end;
LastKeyPressed:=0;
end;
```


```
Function InitTx:boolean;                {inicializuje obvody 8253}
var Ok:Boolean;
begin
Ok:=True;
ok:=ok and write_CWR(CWR_1,$32);

ok:=ok and write_CWR(CWR_1,$72);        {inicializace ridicich registru}
ok:=ok and write_CWR(CWR_1,$b2);
ok:=ok and write_CWR(CWR_2,$32);
ok:=ok and write_CWR(CWR_2,$72);
ok:=ok and write_CWR(CWR_2,$b2);
if Ok then
        begin

        write_channel(0,128);           {inicializace digitálních výstupů}
        write_channel(1,128);
        write_channel(2,128);
        write_channel(3,128);
        write_channel(4,0);
        write_channel(5,0);
        end;
InitTx:=Ok;
end;
Procedure testy;
begin
write('Zkouška PCL−812 '#10#13'Inicializuji I8253'#10#13+
        'Test vnějších registrů'#10#13'Test sériového kanálu 1'#10#13+
        'Test sériového kanálu 2'#10#13'Test Motoru');
end;

Procedure TestHW;                       {provádí jednotlivé testy}
var ok,allOk:Boolean;
j,je0,je255:byte;
kanaly:array[0..15]of char;
MarkTime:Longint;
oldSec,oldsec100:word;
prenosu:Word;
AcLine:Byte;
i:longint;
Label 0,1,2,3,4,5,10;
begin
AllOK:=True;
ChangeWindBox(2,TextAttr);
OpenWindow(24,9,33,8);
InWindBox(1);
TextAttr:=Blue;
Testy;
TextAttr:=Green;
WriteHelpLine('Čekejte na provedení všech testů ....');
```

```
0:
AcLine:=1;
GotoXY(1,AcLine);
writeln('Zkouška PCL−812');
GotoXy(25,AcLine);
if base=0 then write('Skip')
else
begin
ok:=true;

Port[Base+11]:=0;                       {zakázaný režim}
Port[Base+8]:=0;                {reset všech požadavků na přerušení}
i:=Portw[Base+4];                       {prázdné čtení z A/D}
Portw[Base+4]:=$800;                    {reset analogových výstupů}
Portw[Base+6]:=$800;
asm cli; end;

Port[Base+11]:=1;                       {povolení programovávho spouštění}
if (Port[Base+5] and $10)=0             {data stále připravena − chyba}
                then ok:=false;

asm cli; end;
Port[Base+12]:=0; {pokus o konverzi}
for i:=0 to $10000 do if (Port[Base+5] and $10)=0 then break;
asm sti; end;
if (i=$10000) then Ok:=False; {nepodařilo se převést − chyba}
i:=Port[Base+4];
if (Port[Base+5] and $10)=0 {data stále připravena, chyba}
        then Ok:=False;
if not(Ok) then
begin
AllOK:=False;
j:=OsetriError('Chyba karty PCL−812',
         'Pravděpodobně je nastavena špatná adresa, nebo karta není za-
sunuta v počítači!');
if j=3 then Goto 0;
if j=2 then Goto 10;
end;
if OK then write('OK')
        else write('Error');
end;
inc(AcLine);
1:
GotoXY(1,AcLine);
writeln('Inicializuji I8253');
ok:=InitTx;
oldSec:=0;
if Ok then
```


```
begin
for prenosu:=0 to 8000 do
        begin
        if not(Dummy_write(Lo(prenosu))) then

                begin;inc(oldSec);Ok:=False;end;
        end;
end;
InitTx;
if not(Ok) then
begin
AllOK:=False;
if OldSec>0 then
         j:=OsetriError('Nastala chyba při kontrole kabelu',
         'Pravděpodobně mají některé zhoršenou průchodnost signálu, nebo
dochází k jejich rušení!')
        else j:=OsetriError('Nastala chyba při zápisu',
         'Pravděpodobně jsou prohozeny nebo špatně připojeny propojovací
kabely!');
if j=3 then Goto 1;
if j=2 then Goto 10;
end;
GotoXy(25,AcLine);
if OK then write('OK')
else write('Error');
inc(AcLine);
2:
GotoXY(1,AcLine);
writeln('Test vnějších registrů');
for j:=0 to 15 do kanaly[j]:=chr(read_channel(j));
Ok:=copy(kanaly,12,5)='PK&JF';
if not(Ok) then
begin
AllOK:=False;
je0:=0;
je255:=0;
for j:=0 to 15 do
        begin
        if kanaly[j]=chr(0) then inc(je0);
        if kanaly[j]=chr(255) then inc(je255);
        end;
if je0=16 then j:=OsetriError('Obvod neni připojen',

                'Pravděpodobně neni zapojeno napajeci napeti.')
        else if je255=16 then
                j:=OsetriError('Obvod nekomunikuje',
                        'Buď chybí konfigurační paměť, nebo Xilinx. '+
                        'Dále může být prohozena konfigurační přípojka pro externí
```

```
nahrávání.')
                else j:=OsetriError('Chybný identifikační řetězec',
                'Špatné připojení, připojeno jiné zařízení, možnost vnitřní
chyby.');
if j=3 then Goto 1;
if j=2 then Goto 10;
end;
GotoXy(25,AcLine);
if OK then write('OK')
else write('Error');
inc(AcLine);
3:
GotoXY(1,AcLine);
writeln('Test sériového kanálu 1');
prenosu:=0;
MarkTime:=GetInterval;
while GetInterval=MarkTime do if keypressed then break;
read_channel(15);
je0:=0;
repeat
j:=read_channel(8) and $F;
if j<je0 then inc(j,15);
inc(prenosu,j−je0);
je0:=j and $F;
until abs(GetInterval−MarkTime)>18;
Ok:=prenosu>2000;
if not(Ok) then
        begin
        AllOK:=False;
        if prenosu<100 then j:=OsetriError('Kanál 1 vůbec nepracuje',
                'Pravděpodobně není připojen spojovací kabel.')
            else j:=OsetriError('Kanál 1 pracuje špatně',
                'Špatný kontakt spojovacího kabelu.');
      if j=3 then Goto 3;
      if j=2 then Goto 10;
      end;
GotoXy(25,AcLine);
if OK then write('OK')
else write('Error');
inc(AcLine);
4:
GotoXY(1,AcLine);
writeln('Test sériového kanálu 2');
prenosu:=0;
MarkTime:=GetInterval;
while GetInterval=MarkTime do if keypressed then break;
read_channel(15);
```


```
je0:=0;
repeat
j:=read_channel(9) and $F;
if j<je0 then inc(j,15);
inc(prenosu,j−je0);
je0:=j and $F;
until abs(GetInterval−MarkTime)>18;
Ok:=prenosu>2000;
if not(Ok) then
          begin
          AllOK:=False;
          if prenosu<100 then j:=OsetriError('Kanál 2 vůbec nepracuje',

                    'Pravděpodobně neni připojen spojovací kabel.')
                    else j:=OsetriError('Kanál 2 pracuje špatně',
                    'Špatný kontakt spojovacího kabelu.');
          if j=3 then Goto 4;
          if j=2 then Goto 10;
          end;
GotoXy(25,AcLine);
if OK then write('OK')
else write('Error');
inc(AcLine);
5:
GotoXY(1,AcLine);
write('Test motoru');ClrEol;
GotoXy(25,AcLine);
if AllOk then
begin
ok:=False;
for i:=10 to 80 do
          begin

          write_channel(4,i);                              {rozbeh motoru}
          GotoXy(25,AcLine);
          write(i:2);
          delay(80);
          if read_channel(3)>5 then break;
          end;
write_channel(4,0);                              {zastaveni motoru}
ok:=i<80;
if not(Ok) then
          begin
          j:=OsetriError('Motor se nepodařilo roztočit',

                    'Špatná funkce RC vysílače, RC vysílač přepnut na režim
ručního řízení, nepropojený kabel s RC.');
          if j=3 then Goto 5;
```

```
        if j=2 then Goto 10;
        end;
GotoXy(25,AcLine);
if OK then write('OK ')
      else write('Error');
end
else write('Skip');
WriteHelpLine('Stiskni něco pro pokračovnání.');
readkeyEX;
10:
CloseWindow;
ChangeWindBox(1,TextAttr);
LastKeyPressed:=0;
end;

Procedure MonitorChannels;                {měření propustnosti kanálů v čase}
var i:word;
j1,bj1,j2,bj2:Byte;
prenosu1,prenosu2:word;
StopTime:Longint;
CSSmallFont:Byte;
ComunicationOk:Boolean;
label 5,10;
begin
ComunicationOk:=InitTx;
{$ifdef Graph}
CSSmallFont:=InstallUserFont('csli');
if GraphResult <> grOk then CSSmallFont:=SmallFont;
OpenWindow(1,1,MaxX,MaxY);
if not(OpenGraph) then
        begin
        goto 10;
        end;
SetTextStyle(CSSmallFont, HorizDir, 5);
if GraphResult<>0 then
        begin
        SetTextStyle(SmallFont, HorizDir, 5);
        CSSmallFont:=SmallFont;
        end;
NastavXY;
Scale(0,0,100,8000);
Osy;
SetTextStyle(CSSmallFont, VertDir, 5);
OuttextXY(DolniX+TextHeight('P')+4,TextHeight('Počet přenosů')+6,'Počet
přenosů');
SetTextStyle(CSSmallFont, HorizDir, 5);
OuttextXY(GetMaxX−50,GetMaxY−DolniY−20,'Čas');
if ComunicationOk then
```


```
        begin
        SetTextStyle(CSSmallFont, HorizDir, 6);
        SetColor(Brown);
        i:=GetMaxX−TextWidth('0 .. 9 − 0% .. 90% výkonu motoru.')−10;
        OuttextXY(i,20,'Spuštění motoru:');
        OuttextXY(i,40,'0 .. 9 − 0% .. 90% výkonu motoru.');
        end;
For i:=1 to 200 do
        begin
        prenosu1:=0;bj1:=0;
        prenosu2:=0;bj2:=0;
        StopTime:=GetInterval;
        while GetInterval=StopTime do if keypressed then break;
        read_channel(15);
        repeat
         j1:=read_channel(8) and $F;
         if j1<bj1 then inc(j1,15);
         inc(prenosu1,j1−bj1);
         bj1:=j1 and $F;
         j2:=read_channel(9) and $F;
         if j2<bj2 then inc(j2,15);
         inc(prenosu2,j2−bj2);
         bj2:=j2 and $F;
        until abs(GetInterval−StopTime)>10;
        if odd(i) then

                begin
                GrafColor:=Green;Bod(i/2,prenosu1*2.0229);
                GrafColor:=Yellow;Bod(i/2,prenosu2*2.0229);
                end
        else begin
                GrafColor:=Yellow;Bod(i/2,prenosu2*2.0229);
                GrafColor:=Green;Bod(i/2,prenosu1*2.0229);
                end;
        while keypressed do
                begin
                if readkeyEx=Esc then Goto 5;
                if ComunicationOk and (chr(Lo(LastKeyPressed)) in ['0'..'9'])
                        then write_channel(4,(Lo(LastKeyPressed)−ord('0'))*25)
                        else write_channel(4,0);
                end;
        end;

{readkey;}
5:
```

```
CloseGraph;
NastavXY;
10:
DirectVideo:=True;
CloseWindow;
{$endif}
{$ifdef mouse}ShowMouse;{$endif}
SetCursor(Hide);
LastKeyPressed:=0;

write_channel(4,0);                              {zastaveni motoru}
end;
Procedure MerPolohu;
var i,j:byte;
nuly:array[0..4] of longint;
ComunicationOk:Boolean;
begin
ComunicationOk:=InitTx;
WriteHelpLine('Esc−předchozí menu, 0..9 %výkonu motoru');
ChangeWindBox(2,TextAttr);
if Base=0 then OpenWindow(30,11,24,9)
          else OpenWindow(30,11,24,13);
InWindBox(1);
for i:=0 to 6 do
          begin
          Leaders[i]:=0;
          OldValues[i]:=read_channel(i);
          If OldValues[i]>128 then dec(Leaders[i]);
          end;

for i:=0 to 3 do nuly[i]:=0;              {stredni poloha pakovych ovladacu}
for j:=0 to 9 do
begin
for i:=0 to 3 do
          begin
          inc(nuly[i],inA_D(i));
          end;
delay(10);
end;
for i:=0 to 3 do nuly[i]:=nuly[i] div 10;
repeat
GotoXY(1,1);
{$ifdef mouse}
with ActiveWindow^ do HideCursorBox(x,y,x+Dx−1,y+Dy−1);
{$endif}
writeln(' x =',k3*check_channel(6)*sin(k1*check_channel(5)):10:2,' mm');
writeln(' y =',k3*check_channel(6)*cos(k1*check_channel(5))*sin(k1*check_channel(4)):10:2,'
mm');
writeln(' z =',k3*check_channel(6)*cos(k1*check_channel(5))*cos(k1*check_channel(4)):10:2,'
mm');
writeln('psi=',k2*check_channel(0):10:5,' rad');
```


```
writeln(' δ =',k1*check_channel(1):10:5,' rad');
writeln(' τ =',k1*check_channel(2):10:5,' rad');
write(' n =',read_channel(3)/(0.227*19):10:5,' ot/s');
if Base<>0 then
        begin
        writeln(#10#13'R1 =',inA_D(0)−nuly[0]:10);
        writeln('R2 =',inA_D(1)−nuly[1]:10);
        writeln('R3 =',inA_D(2)−nuly[2]:10);
         write('R4 =',inA_D(3)−nuly[3]:10);
        end;
{$ifdef mouse}ShowMouse;{$endif}
if ComunicationOk and (chr(Lo(LastKeyPressed)) in ['0'..'9'])

                        then write_channel(4,(Lo(LastKeyPressed)−ord('0'))*25)
                        else write_channel(4,0);
if keypressed then readkeyEx;
until LastKeyPressed=Esc;
write_channel(4,0);
CloseWindow;
ChangeWindBox(1,TextAttr);
LastKeyPressed:=0;
end;
Procedure ZobrazPolohu;
var x,y,z:real;
xx,yy:integer;
CSSmallFont:Byte;
i:integer;
color:byte;
const a11=1;a12=0.5;a13=0;
a21=0;a22=−0.5;a23=−1;
label 1,10;
Procedure PlotXYZ(x,y,z:real);
begin
xx:=round(1.5*(a11*x+a12*y+a13*z))+getMaxX div 2;
yy:=round(1.5*(a21*x+a22*y+a23*z))+getMaxY div 2;
PutPixel(xx,yy,color);
end;
begin
InitTx;
for i:=0 to 6 do
        begin
        Leaders[i]:=0;
        OldValues[i]:=read_channel(i);
        If OldValues[i]>128 then dec(Leaders[i]);
        end;
{$ifdef Graph}
CSSmallFont:=InstallUserFont('csli');
if GraphResult <> grOk then CSSmallFont:=SmallFont;
OpenWindow(1,1,MaxX,MaxY);
```

```
if not(OpenGraph) then goto 10;
NastavXY;
SetTextStyle(CSSmallFont, HorizDir, 5);
if GraphResult<>0 then
        begin
        SetTextStyle(SmallFont, HorizDir, 5);
        CSSmallFont:=SmallFont;
        end;
1:
SetBKColor(Black);Cleardevice;
SetColor(Yellow);
OuttextXY(150,10,'Zobrazení polohy helikoptéry v prostoru');
SetColor(Brown);
OuttextXY(0,GetMaxY−TextHeight('E')−1,'ESC−konec, C− smaže obra-
zovku');

color:=green;                   {kreslení os}
for i:=−100 to 100 do
        begin
        plotXYZ(0,0,i);
        plotXYZ(0,i,0);
        plotXYZ(i,0,0);
        end;
SetColor(white);
PlotXYZ(90,0,0);OuttextXY(xx+4,yy+3,'x');
PlotXYZ(0,90,0);OuttextXY(xx+4,yy+3,'y');
PlotXYZ(0,0,90);OuttextXY(xx+4,yy+3,'z');
color:=white;
repeat
x:=k3*check_channel(6)*sin(k1*check_channel(5));
y:=k3*check_channel(6)*cos(k1*check_channel(5))*sin(k1*check_channel(4));
z:=k3*check_channel(6)*cos(k1*check_channel(5))*cos(k1*check_channel(4));
PlotXYZ(x,y,z);
if Keypressed then
        begin
        ReadkeyEx;
        if upcase(chr(lo(lastkeypressed)))='C' then goto 1;
        end;
until LastKeyPressed=Esc;
CloseGraph;
NastavXY;
10:CloseWindow;
{$ifdef mouse}ShowMouse;{$endif}
{$endif}
SetCursor(Hide);
LastKeyPressed:=0;
end;
Procedure PohniServama;
var Ok:Boolean;
```


```
a:array[0..5] of byte;
i:integer;
PM:PMenuItems;
const jas:Byte=0;
begin
Ok:=InitTx;
ChangeWindBox(2,TextAttr);
OpenWindow(25,13,28,7);
for i:=0 to 3 do a[i]:=128;
a[4]:=0;
Repeat;
WriteHelpLine('Esc−předchozí menu, Enter−zadání hodnoty, Šipky </>
přidej/uber + Ctrl Rychle');
InWindBox(1);
Write('Příčná cyklika'#10#13'Kolektivní řízení'#10#13'Podélná cyk-
lika'#10#13+
        'Nastavení listů ocasu'#10#13'Výkon motoru');
for i:=0 to 4 do
        begin
        gotoXY(23,i+1);
        write(a[i]:3);
        Write_Channel(i,a[i]);
        end;
PM:=nil;
NewMenuHandler(PM,0,GInvAttr);
inc(PM^.Flags,VystupSipkami);
PM^.OtherEvents:=Ie(Ie(Ie(PM^.OtherEvents,29696,nil),29440,nil),2864,nil);
menu(jas,PM);
if LastKeyPressed=Enter then
        begin
        SetCursor(Low);
        gotoXY(23,jas+1);
        ClrEol;
        read(i);
        if i>255 then i:=255;
        if i<0 then i:=0;
        a[jas]:=i;
        SetCursor(Hide);
        end;
if LastKeyPressed=Cur_Right then if a[jas]<255 then inc(a[jas]);
if LastKeyPressed=Cur_Left then if a[jas]>0 then dec(a[jas]);
if LastKeyPressed=29696 then if a[jas]<245 then inc(a[jas],10)

                        else a[jas]:=255;
if LastKeyPressed=29440 then if a[jas]>10 then dec(a[jas],10)

                        else a[jas]:=0;
if LastKeyPressed=2864 then a[jas]:=0;
until LastkeyPressed=Esc;
```

```
Write_Channel(4,0);
CloseWindow;
LastKeyPressed:=0;
end;
var i:integer;
q:integer;
c:char;
begin
Base:=$220;
DO_Hi:=$220+14;
DO_Lo:=$220+13;
DI_Lo:=$220+6;
textcolor(Brown);
Writeln(#13'<<<Test>>> Testovací program HW části modelu he-
likoptéry (c)1995 F&TSoft');
c:='D';
if CheckWindows > 0 then
        begin
        textcolor(LightRed);
        write('Výstraha!  V okně Windows program nebude pracovat ko-
rektně'#10#13+
         'Pokračovat Ano/Ne?');
        repeat
         c:=upcase(Readkey);
        until c in ['Y','A','N'];
        NormVideo;
        if c='N' then exit;
        c:='W';
        end;
Speed(0);
OpenWindow(1,1,MaxX,MaxY);
TextAttr:=Green;

Fillarea(1,1,MaxX,MaxY,'▒');          {inicializuje DeskTop}
TextAttr:=Green+Blink;
if c='W' then WriteXY(MaxX-14,MaxY-2,'WindowsMode');
TextAttr:=Green;
Header('Testovací program HW části modelu helikoptéry');
{$ifdef mouse}
InitMouse;
ShowMouse;
InputsJob:=MouseMoveWindow;
{$endif}
SetCursor(Hide);
openwindow(17,6,21,10);

repeat                                  {smyčka hlavního menu programu}
{$ifdef mouse}HideMouse;{$endif}
inwindbox(1);
write('Nastavení'#13#10'Testování HW'#13#10'Monitorování kanálů'+
        'Měření polohy'#10#13'Zobrazení polohy'#10#13'Ovládání serv'#10#13+
```


```
        'Informace'#10#13'Konec');
WriteHelpLine('Vybírejte šipkami Esc−Konec, Enter−Volba');
{$ifdef mouse}ShowMouse;{$endif}
menu(jas,nil);
if LastKeyPressed=Enter then case jas of
0:Configure;
1:TestHw;
2:MonitorChannels;
3:MerPolohu;
4:ZobrazPolohu;
5:PohniServama;
6:VriteInfo;
7:LastKeyPressed:=Esc;
end;
until LastKeyPressed=Esc;
CloseWindow;
CloseWindow;
SetCursor(Low);
write_channel(4,0);
NormVideo;
end.
```

16 Příloha Popis desky XV1 a zdroje ZD1

Deska XV1 obsahuje obvody pro zpracování signálů ze tří inkrementálních snímačů úhlu natočení, které snímají polohu modelu v jednom kloubu. Tato deska je určena pro umístění na helikoptéře a proto musí obsahovat také obvody pro snímání otáček hlavního rotoru a pro generování pulsně šířkového signálu ovládajícího motor. Pulsně šířkový signál je generován na základě šířky pulsu z pátého kanálu původního přijímače RCRx, který je určen pro ovládání motoru. Zpracované údaje ze snímačů jsou sériově vysílány k počítači do přijímacího obvodu XRI.

16.1 Ovládací prvky a obsluha

Hlavním kontrolním prvkem je dioda D1, která indikuje přítomnost napájecího napětí. Destička by po zapnutí měla okamžitě pracovat. Pouze v případě nahrávání konfigurace z počítače, nebo potížích s nahráváním konfigurace, je nutno použít tlačítko RESET Tl1 (má delší dřík hmatníku) a tlačítko PROGRAM (D/P) Tl2. Uživetele bych odkázal na kapitolu 7.3, kde je základní použití vysvětleno. V případě programování z počítače bude nutno manipulovat i s propojkou SV1 a pamětí. Tyto operace by měla dělat pouze osoba znalá problematiky a s rozvahou.

16.2 Obvodové zapojení

Základem obvodového řešení je využití programovatelného obvodu XILINX, který v sobě realizuje celé zapojení pro snímání a přenos dat. Vnější zapojení odpovídá standartnímu zapojení obvodů XILINX popsanému v kapitole 7.9. Obvod XILINX XC3042 IO1 je spojen s paticí pro sériovou konfigurační paměť XC1736 IO2 a s konektorem K4 pro nahrávání z počítače. V době odladování je patice paměti prázdná a obvod XILINX je v režimu SLAVE nahráván kabelem od počítače. Později po odzkoušení se konfigurace zapíše do konfigurační paměti. Paměť se dá do patice a obvod XILINX se může nadále konfigurovat v režimu MASTER z této paměti. Přepínání režimu je zajištěno propojkou (jumper) SV1. Schema zapojení je na 16.2 a připojení snímačů k vývodům obvodu XILINX je v 16.2.

Připojení inkrementálních snímačů je provedeno dvacetivývodovým samořezným konektorem. Snímače jsou spojeny ohebnou kabeláží s propojovací destičkou na kloubu a z ní je pak signál veden dvacetižilovým vodičem ke konektoru. Značení propojovací destičky je na 16.2. Součástí základního zapojení je i zdroj stejnosměrného napětí 5V s obvodem 7805. Tento zdroj je určen pro napájení obvodu XILINX a inkrementálních snímačů. Do vrtulníku jde dvěma vodiči napájení 10V pro hlavní hnací motor. Zdroj zajišťuje napájení 5V aniž by musel být do vrtulníku veden další vodič (každý vodič něco váží a zhoršuje tak dynamické vlastnosti).

Deska XV1 určená pro vrtulník musela být doplněna o obvod

Význam signálu	Snímač	Propojovací destička	Konektor desky XV1	Vývod obvodu XILINX
GND	Všechny	1A,1B,1C	1,6,11,12 K3	1, 43
+5V	Všechny	5A,5B,5C	4,9,16 K3	22, 64
ChA	Příčný	NC (1)	3 K3	29
ChB	náklon	NC	5 K3	28
Index		NC	2 K3	30
ChA	Podélný	2A	7 K3	21
ChB	náklon	2B	8 K3	20
Index		2C	10 K3	19
ChA	Kurs	3A	13 K3	16
ChB	(index	3B	14 K3	15
Index	nemá smysl)	3C	15 K3	14
OT	Otáčky (2)			39
MOT	Kanál 5 RCRx			13
RIZ	Rizení motoru			76
TXDATA	Výstup modulovaných dat		K2	35
DPWM	Data PWM kód			58
DNRZ	Data NRZ kód	Spojeno s INMO		59
INMO	Vstup modulátoru			63
INF	Vstup nosné	Spojeno s FOUT		37
INVI	Invertor input			46
INVO	Invertor output			47
FOUT	Výstup 18,432MHz			50

Poznámka:

(1)

NC označuje nepřipojení signálu do danného místa.

(2)

Ve druhé části tabulky je v kolonce snímač uváděn význam a v kolonce propojovací destičky je zapsáno přímé propojení vývodů.

Table 8: Připojení snímačů k desce XV1

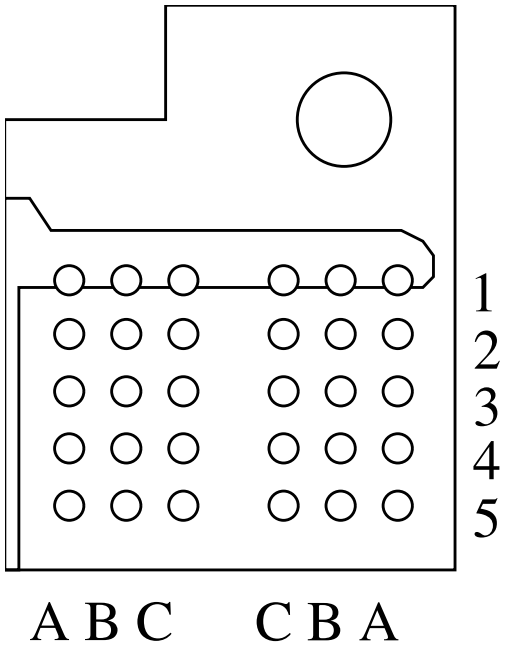


Figure 48: Propojovací destička

Číslo vývodu	Význam signálu	Barva vodiče
1	GND	modrá
2	Index	bílá
3	ChA	fialová
4	+5V	červená
5	ChB	zelená (tmavo)

Table 9: Připojení inkrementálního snímače

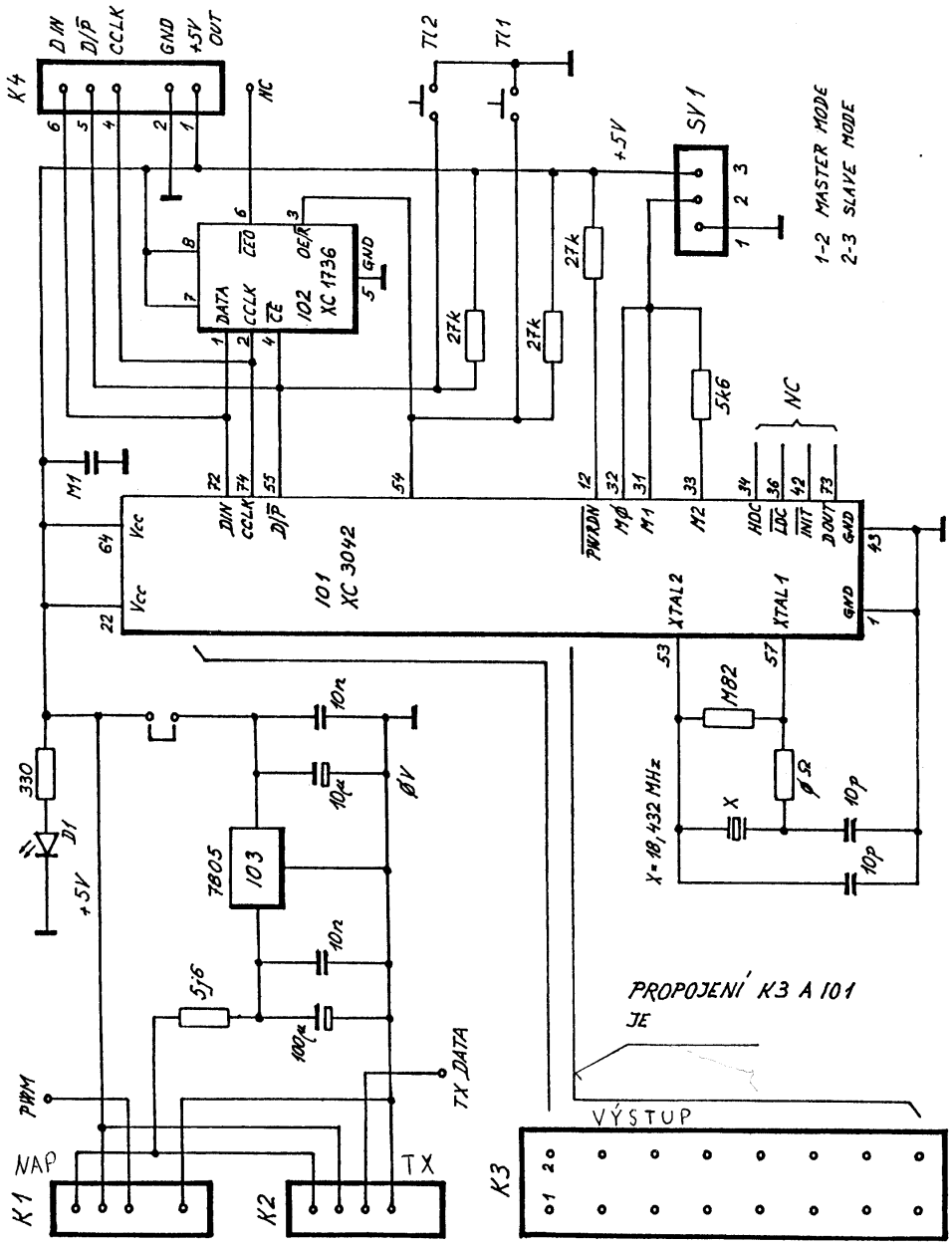


Figure 49: Schema zapojení desek XV1 a XV2

pro úpravu signálu ze snímače otáček a o obvod pro buzení spínače motoru. Oba obvody jsou na 16.2 včetně připojení příslušných konektorů. Signál ze snímače otáček je upraven do úrovně signálů TTL tranzistorovým zesilovačem, který je osazen tranzistorem T1 BC149B . Tranzistorem T2 stejného typu je osazen i obvod pro buzení spínače motoru.

Pro úplnost je na 16.2 schéma spínacího obvodu motoru, který navrhl a realizoval již můj předchůdce Pavel BENEŠ. Bylo nutno udělat jen určité úpravy v oblasti buzení. Později bylo nutno nahradit původní trojici spínacích tranzistorů typu BUZ11 jedním velmi výkonným tranzistorem FET s typovým označením IRFP054. Tento obvod je umístěn přímo na motoru a na jeho vstup PWM je připojen výstup budícího obvodu označený také PWM. Budící obvod upravuje TTL signál z obvodu XILINX přivedený na jeho vstup MOT.

Motor je řízen podle signálu z pátého kanálu RCRx, který musí být zaveden do obvodu XILINX. Protože signál z přijímače může mít i větší napětí než odpovídá TTL úrovním je do série vřazen ochranný odpor 15KΩ (využívá se ochranných diod na vstupech obvodu XILINX). Zapojení s konektorem je na 16.2.

16.3 Připojení desky XV1

Konektor K1 spojuje desku XV1 se spínačem motoru a přivádí na desku napájecí napětí 10V. Tímto konektorem je ke spínači vedeno i jeho řízení. Konektor K2 spojuje desku s vysílačem dat Tx a přivádí mu kromně dat (signál TXDATA) i napájení. Konektory na desce jsou zajištěny proti vytažení umělohmotovým páskem. Přijímač RCRx a snímač otáček jsou připojeny vodiči opatřenými na konci modelářskými konektory v černé barvě. Jeden konektor má dutinky a druhý špičky, čímž je zajištěna jejich nezáměnnost.

16.4 Vnitřní zapojení obvodu XILINX na desce XV1

Základním obvodem pro všechna tři zapojení je generátor hodin na 16.4. Z krystalového oscilátoru s kmitočtem 18,432MHz jsou za pomoci děliče a jednoduchého obvodu generovány dvoufázové hodiny s kmitočtem 2,304MHz. Obvod pro generování dvou fází hodin reprezentovaný KO U1 a U2, zajišťuje generování hodin bez ohledu na stav po RESETu. Vyvedeny jsou i některé další výstupy děliče, které se používají v jiných částech zapojení a bylo by zbytečné mít děliče duplicitní.

Na blokovém schematu 16.4 lze jednoduše rozpoznat základní koncepci systému. Kolem datové a adresové sběrnice jsou soustředěny jednotlivé obvody pro vyhodnocení snímačů. Zpracovaná data ze snímačů jsou sběrnicí vedeny do vysílacího shift registru a odtud jsou již sériově vysílány. Výstup sériových dat jde posléze přes modulátor v němž je vytvořena modulace pro bezdrátový přenos (AM s činitelem modulace 100% neboli A1).

Časování celého systému zajišťuje časová základna 16.4, která se odvozuje z hlavních hodin (2,304MHz). Čítač U8 definuje délku jednoho

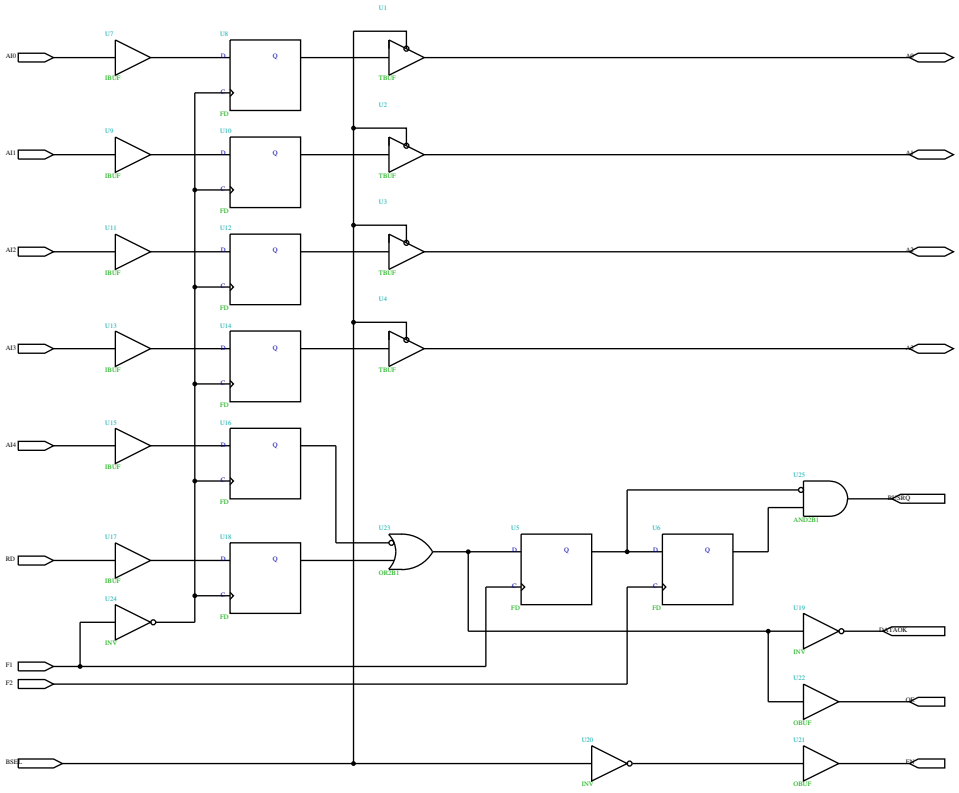
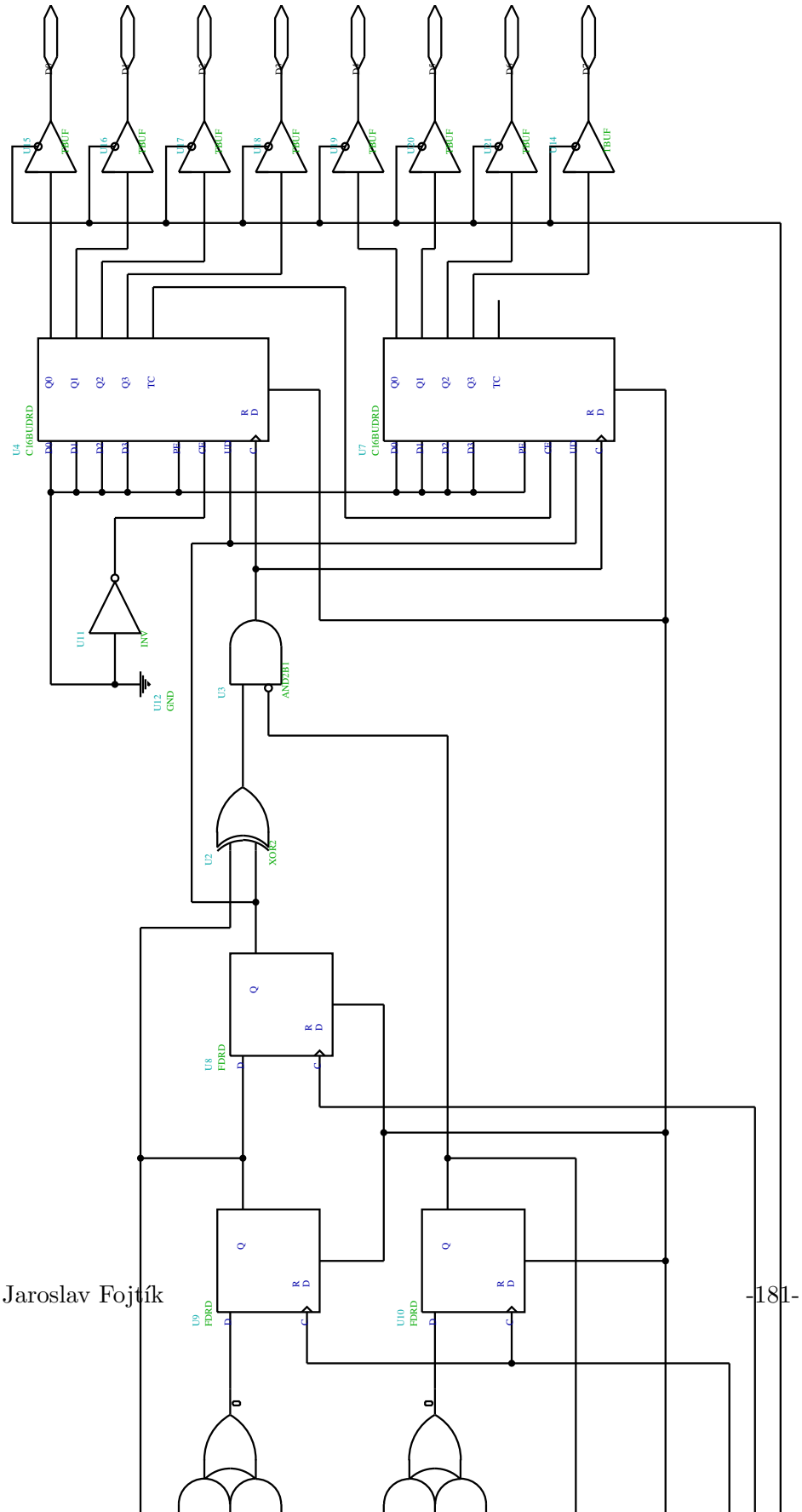


Figure 50: Oscilator a generator hodin


U5
IBUF
U6
IBUF
U7
IBUF
U8
IBUF
U9
IBUF
U10
IBUF
U11
IBUF
U14
TBUF
U4
C16BUDRD
Q0
Q1
Q2
Q3
TC
R
D
U7
C16BUDRD
U11
INV
U12
GND
U3
AND2B1
U2
XOR2
U8
FDRD
U9
FDRD
U10
FDRD

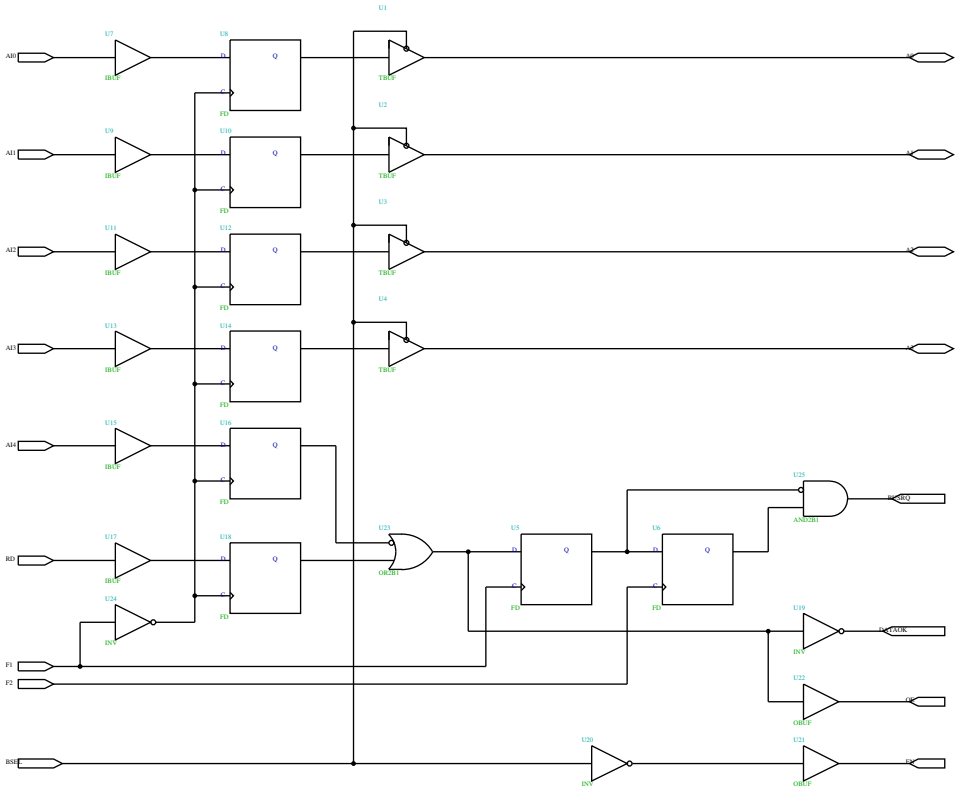


Figure 52: Časová základna pro vysílač

vysílaného bitu. V době, kdy tento čítač napočítá do 11, je inkrementováno počítadlo bitů U1. Vysílací shift registr je posouván okamžikem kdy se v čítači U8 objeví číslo 15. Nová data se zapisují do registru při stavu čítače U8 = 14 a za předpokladu, že prvních 5b čítače U1 (počítadlo bitů) bude ve stavu log. nuly. Čítač bitů udává vlastně bit, který se začne následujícím posunutím vysílat. Další dva bity udávají vysílané slovo a tvoří tak adresu zdroje. Tato adresa je také společně s daty vysílána. Zkrácení čítače U1 umožní vysílat pouze čtyři datová slova. Signály T1 a T2 jsou určeny pro řízení modulátoru při vytváření pulsně šířkové datové modulace.

Součástí výstupního shift registru 16.4 jsou obvody vytvářející 32bitové slovo, které obsahuje
adresu data a další kontrolní bity. Zajmavé je řešení výstupu pomocí vysílacího KO U12. Tento obvod spolu s multiplexerem U11 zajistí stálé vysílání požadovaného bitu. Běžně by bylo složité zajistit synchronnost hodin vysílacích (posouvají reg.) a hodin pro nahrání dat. Posouvání proto probíhá v rámci celého registru (32b+1b), ale při zápisu se do KO U12 kopíruje jeho minulý stav a tím zajistí nepřerušenost vysílání a nová data se začnou vysílat až se správným posouvacím hodinovým impulsem.

Data z shift registru jdou do modulátoru 16.4, kde je tvořen datový signál NRZ a volitelně i signál s digitální šířkovou modulací. Výstup dat DNRZ je připojen do vstupu modulátoru INMO. Druhý vstup modulátoru INF slouží pro nosný signál (zde 18,432MHz). Výstup pak jde přímo do vysílacího VF zesilovače.

Základem zapojení jsou obvody pro vyhodnocení fázového signálu z inkrementálních snímačů 16.4. Základem pro vyhodnocení je 6b obousměrný čítač U4, U7. Samotný signál lze použít jako dva nejnižší bity čítače v Grayově kódu (převod na binární je jednoduchý). KO U8, U9, U10 detekují změnu na signálu FA v době, kdy FB je v log. nule. Tato změna je hodinovým pulsem pro čítač a směr změny (nástup, sestup) udává směr počítání. Celý obvod je synchronní. KO spolu s multiplexery U1, U14 brání v současné změně signálu FA a FB. Takový přechod je považováno za chybu a čítač zůstává bez pohybu. Zvláštní pozornost je nutno věnovat synchronizaci celého obvodu, aby v době čtení dat (F2 v log. 1) nedocházelo ke změně údaje čítače.

Protože na vrtulníku je třeba získat i informace o otáčkách rotoru, byl vytvořen jednoduchý otáčkoměr 16.4. Základem je čítač U7 čítající po určitou dobu (227 ms) pulsy přicházející od snímače otáček. Na jednu otáčku rotoru připadá 10 pulsů a čítač reaguje na každou hranu (tj. 20 pulsů), pak jeden krok čítače v měřící periodě představuje 13,2 ot/min nosného rotoru. To představuje i nejnižší přímo měřitelné otáčky. Protože na čítači se údaj mění a je třeba zajistit asynchronnost čtení dat do vysílacího čítače, je výsledek vždy po skončení čítání (čtyřikrát za sekundu) přepsán do registru U1, kde je k dispozici pro další zpracování. Děliče U2 a U3 tvoří časovou základnu pro měření otáček.

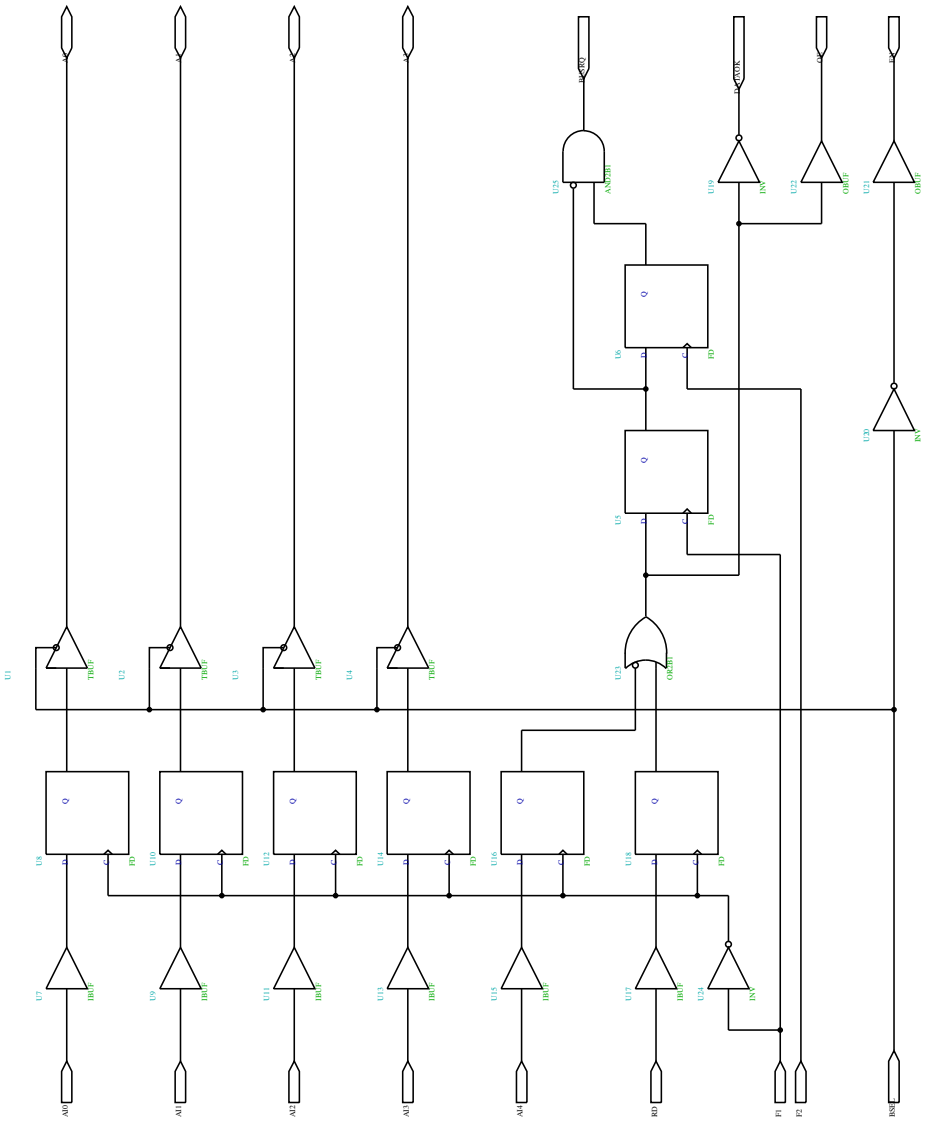


Figure 53: Vysílací posuvný registr

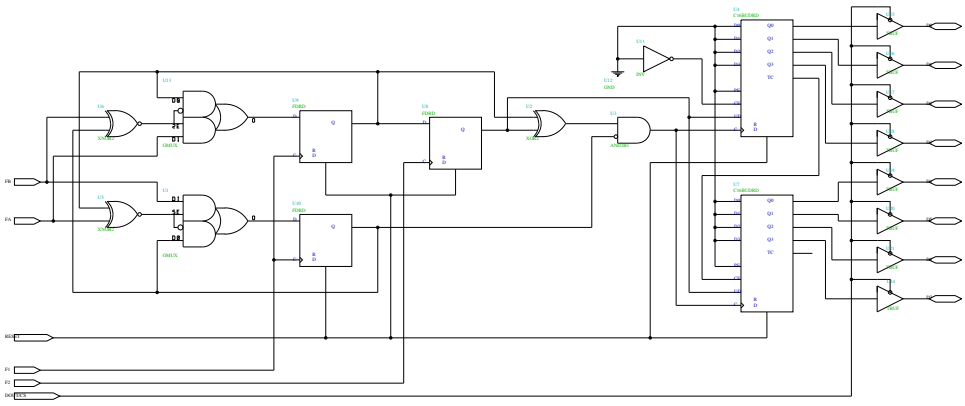


Figure 54: Generátor kódu a modulátor

Posledním obvodem je generátor PWM signálu pro řízení motoru. Úkolem je z délky pulsu přijatého z přijímače RCRx (1 až 2ms jednou za 20ms) vytvořit PWM signál se zajištěnou možností vypnutí motoru. Základní frekvence řídícího PWM signálu je přibližně 1kHz. Obvod je rozdělen do dvou částí. První část 16.4 zajistí změření délky pulsu a její převedení na číslo včetně odečtení ofsetu (1ms představuje 0). Druhá část 16.4 obsahuje čítač U1, který je poháněn kmitočtem 144kHz a registr uchovávající požadovanou délku pulsu. Puls sám je generován komparátorem porovnávající oba údaje (požadavek a čítač). Zapojení je jednoduché, ale pro naše účely vyhovující.

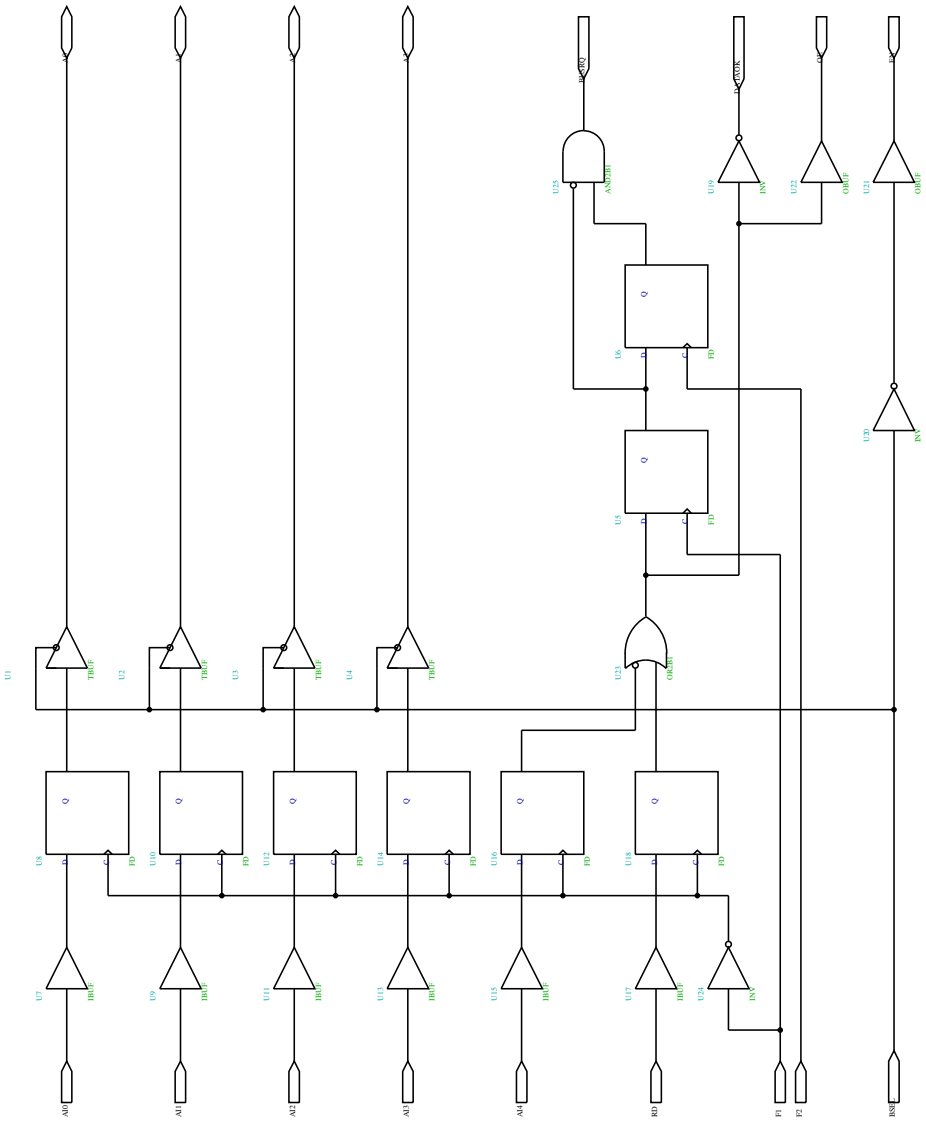


Figure 55: Zpracování dat z IRC snímačů

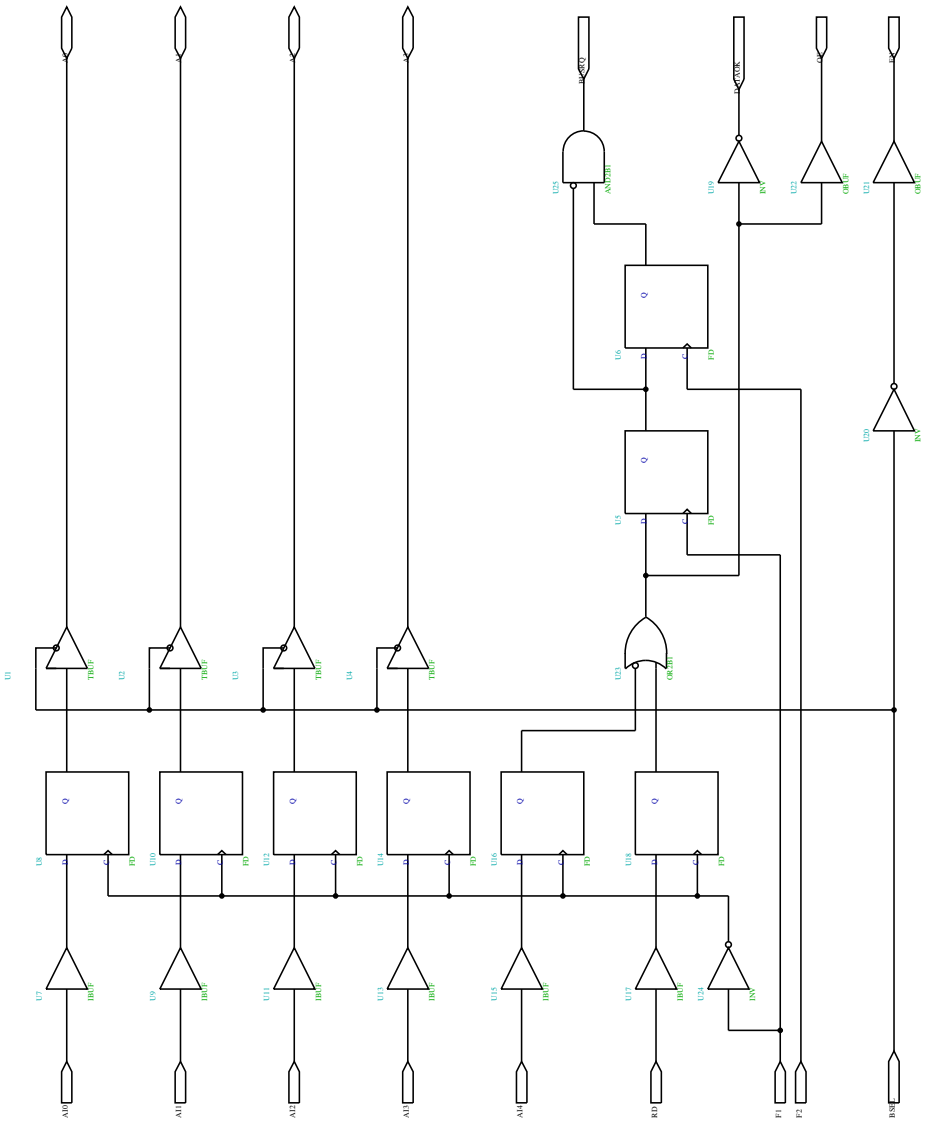


Figure 56: Měření otáček vrtule

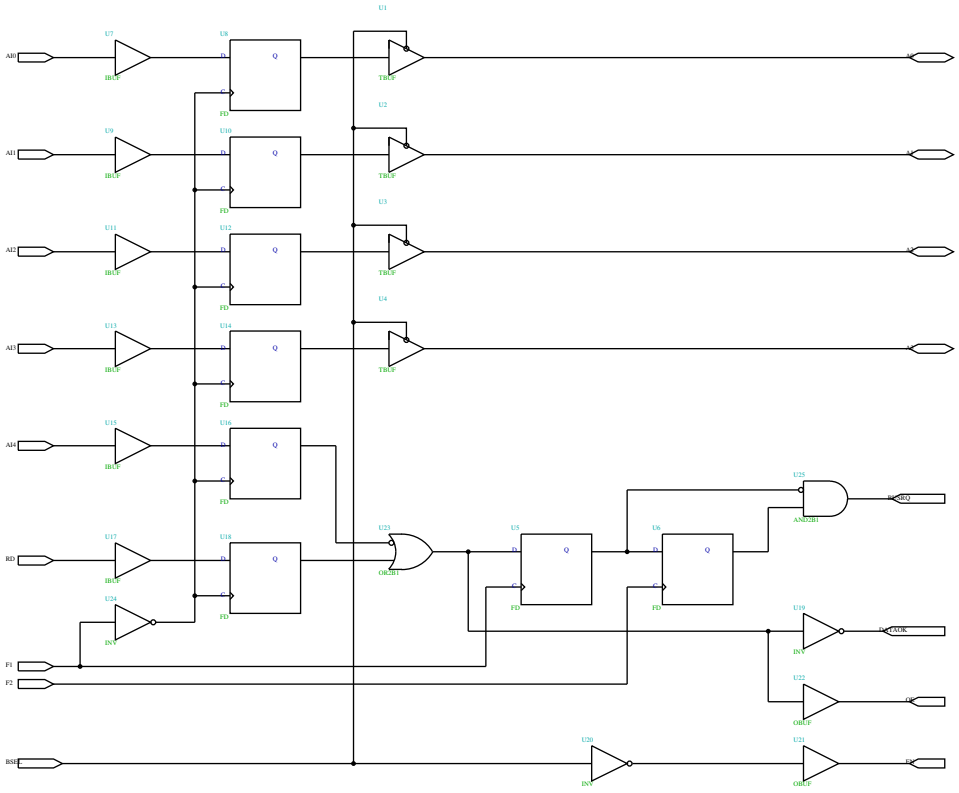


Figure 57: Měření délky pulsu z RX pro servo

16.5 Zdroj napájení ZD1

Původní přijímač dálkového řízení RCRx a serva vyžadují napájením 4,7 až 7V. Běžně se používá napájení 6V, při kterém odebírá přijímač a serva bez pohybu asi 15mA. Problém nastává při pohybu, kdy odběr serva dočasně vzrůstá na 300mA. Odběr roste s rychlostí pohybu a odporem kladeným tomuto pohybu. Vememe-li v úvahu, že musíme napájet čtyři pohybující se serva, vyjde nám proudové zatížení zdroje 1,2 až 1,5A. Na vrtulník je přivedeno napájení 10V/50A. Není tedy problém s odběrem, ale je nutné stabilizovat napájecí napětí, tak aby vyhovovalo požadavkům přijímače. Rozmezí přípustného napájecího napětí umožňuje použít zdroj 5V, který může být z hlediska obvodového řešení jednodušší než například zdroj 6V.

Schema zapojení zdroje je na 16.5. Základem konstrukce je integrovaný stabilizátor 78S05 (5V/2A), který je doplněn o filtrační kondensátory a sériový filtrační rezistor. Na filtračním odporu se realizuje část napěťové ztráty a tím se zmenšuje výkonové zatížení stabilizátoru.

Zdroj ZD1 je pevným vedením spojen s obvodem spínání motoru. Od spínače motoru je přivedeno napájení (10V). K výstupu (5V/2A) zdroje je připojen také přijímač RCRx. Připojení přijímače je realizováno modelářským konektorem (dutinky) červené barvy, který zakončuje kabel ze zdroje. K záměně při připojování může na straně RCRx dojít, ale nic nebude poškozeno, pouze nám nebude pracovat jedno servo. Na přijímači jsou konektory dostatečně popsány a při troše pozornosti nemusí k záměně dojít.

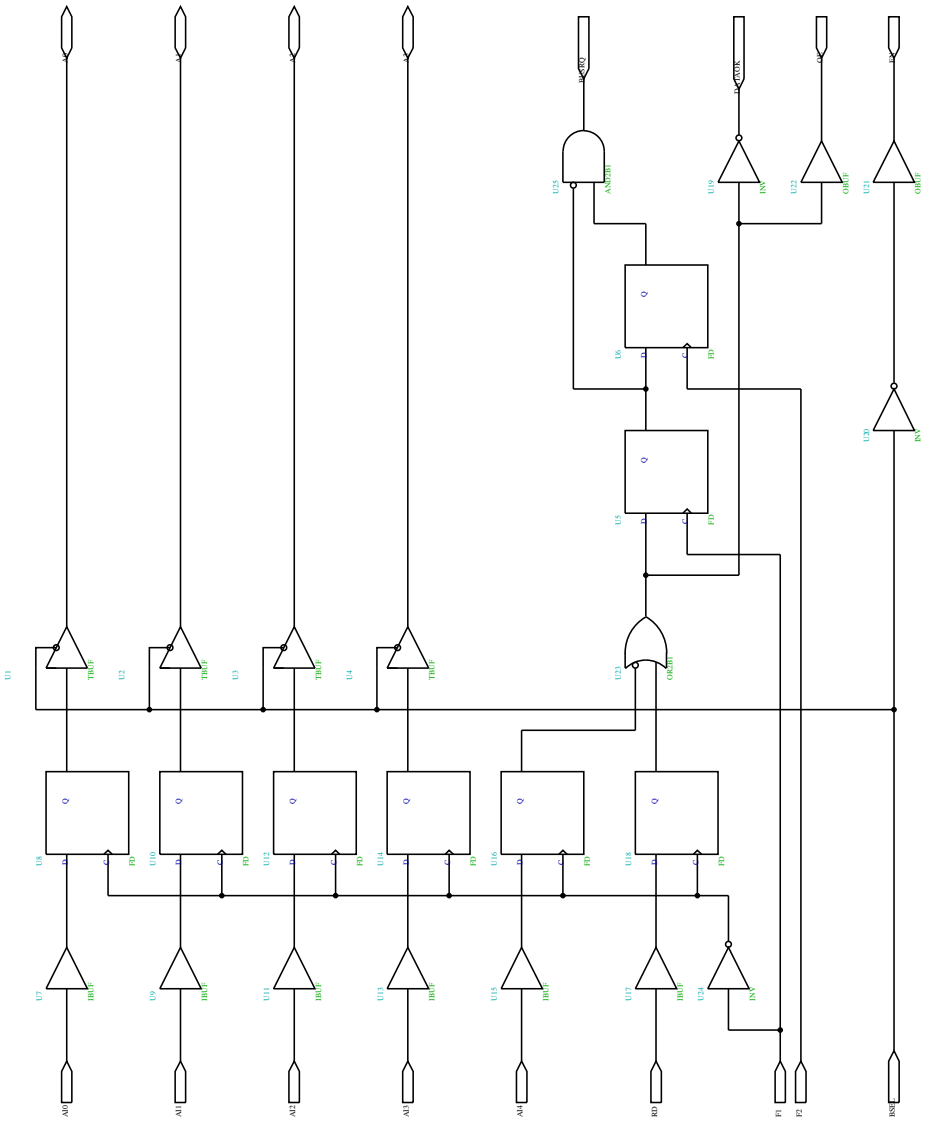


Figure 58: Generátor PWM pro ovládání motoru

17 Příloha Popis desky XV2

Deska XV2 zpracovává údaje inkrementálních snímačů z kloubu II pod heliportem. Získané údaje převádí na sériový kód a předává je tak vodičem do přijímacího obvodu na desce XRI. Napájena je z počítačového zdroje 12V a proto musí být také vybavena stabilizovaným zdrojem 5V/1A, který zajišťuje napájení nejen samotných obvodů na desce XV1, ale i inkrementálních snímačů. Desky plošných spojů XV1 a XV2 jsou stejné a je tu i možnost záměny po menších úpravách zapojení.

17.1 Ovládací prvky a obsluha

Deska XV2 se na první pohled neliší od desky XV1. Obě desky jsou zapojeny stejně, alespoň pokud jde o ovládací a kontrolní prvky. Pokyny pro obsluhu jsou stejné jako v případě desky XV1.

17.2 Obvodové zapojení

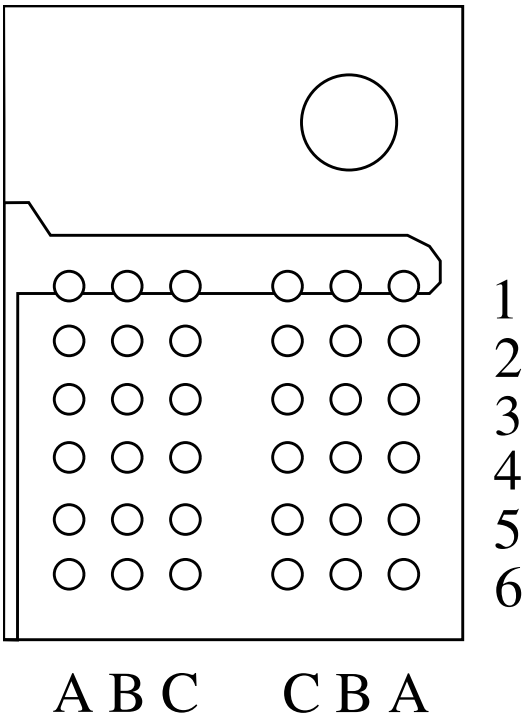


Figure 59: Propojovací destička

Číslo vývodu	Význam signálu	Barva vodiče
1	GND	modrá
2	Index	bílá
3	ChA	fialová
4	+5V	červená
5	ChB	zelená (tmavo)

Table 10: Připojení inkrementálního snímače

Význam signálu	Snímač	Propojovací destička	Konektor desky XV1	Vývod obvodu XILINX
GND	Všechny	1A,1B,1C	1,6,11,12 K3	1, 43
+5V	Všechny	5A,5B,5C	4,9,16 K3	22, 64
ChA	Příčný	NC (1)	3 K3	29
ChB	náklon	NC	5 K3	28
Index		NC	2 K3	30
ChA	Podélný	2A	7 K3	21
ChB	náklon	2B	8 K3	20
Index		2C	10 K3	19
ChA	Vysunutí	3A	13 K3	16
ChB	(index	3B	14 K3	15
Index	nemá smysl)	3C	15 K3	14
FOUT	Výstup kmitočet 18.432MHz (2)			50
TXDATA	Výstup modulovaných dat		K2	35

Poznámka:

(1)

NC označuje nepřipojení signálu do danného místa.

(2)

Ve poslední části tabulky je v kolonce snímač uváděn význam a případně v kolonce propojovací destičky je zapsáno přímé propojení vývodů.

Table 11: Připojení snímačů k desce XV2

Obvod XILINX je zapojen stejně jako na desce XV1 včetně připojení inkrementálních snímačů polohy a jeho zapojení tak vychází ze standardního zapojení popsaného v 7.9. Na desce XV2 nejsou ovšem zapojeny obvody pro úpravu signálu ze snímače otáček a spínání motoru, protože tyto činnosti deska XV2 nemusí vykonávat. Bude však zapotřebí, aby výstup sériových dat byl opatřen tranzistorovým zesilovačem, pro výkonové buzení asi šest metrů dlouhého kabelu k počítači (desce XRI). Šestimetrový kabel by neměl být napájen přímo z vývodu obvodu XILINX, i když právě jeho vývody jsou zapojeny jako výkonové budiče. Zapojení desky XV2 je stejné jako zapojení na 16.2. Připojení inkrementálních snímačů a ostatních pracovních vývodů nezachycených na schematu je v 17.2, která se trochu liší od stejné tabulky pro desku XV1. Jak již bylo řečeno obě desky jsou zaměnitelné po menší úpravě spočívající v přestavbě obvodů pro úpravu signálu.

17.3 Připojení desky XV2

Sériová data nesoucí informaci o poloze jsou asi šestimetrovým vodičem vedena do interface u počítače na desku XRI. Stejným kabelem se vede naopak z desky XRI napájecí napětí 12V pro desku XV2. Připojení k desce XV2 je zajištěno konektorem K2. Konektor K1 je zde nevyužit a má pouze umožnit záměnnost desek XV1 a XV2. Inkrementální snímače polohy jsou připojeny k desce dvacetivývodovým samořezným konektorem. Jednotlivé snímače kloubu II (pod heliportem) jsou svedeny ohebnou kabeláží k propojovací destičce podobně jako v případě kloubu I. Od propojovací destičky vede již plochý dvacetižilový kabel ke konektoru na desce XV2. Způsob značení propojovací destičky je na 17.2.

17.4 Vnitřní zapojení obvodu XILINX na desce XV2

Vnitřní zapojení obvodu XILINX 17.4, který zpracovává data z kloubu pod heliportem (kloub I), je stejné jako zapojení obvodu na desce XV1 pouze s některými zjednodušeními. Na heliportu není zapotřebí zpracovávat údaj o otáčkách ani řídit otáčky motoru. Protože mezi deskou XV2 a počítačem (resp. deskou XRI) je metalický spoj není nutno provádět modulaci sériového datového signálu jako v obvodu XV1.

Jednotlivé bloky zpracovávající signál z inkrementálních snímačů polohy jsou seskupeny kolem sběrnice. Data jsou po sběrnici přenášena do vysílacího registru, který je převádí na sériový signál. Časování zajišťuje časová základna vysílače. Generátor hodin 16.4, obvody pro zpracování signálu ze snímačů 16.4, vysílací registr 16.4 i časová základna 16.4 jsou stejné jako příslušné části v případě obvodu na desce XV1. Rozdílem je existence pouze tří zdrojů dat (tři snímače polohy). Časová základna není krácena, takže se čtvrté slovo vysílá, ale obsahuje nespecifikovaná data. Na přijímací straně je toto slovo podle adresy rozpoznáno a zapomenuto. Toto řešení je jednodušší než zbytečně předělávat zapojení.

Třetí snímač, který snímá vysunutí spojovací tyče, má příliš velkou

citlivost. Jedna perioda fázových signálů odpovídá přibližně 0,22mm vysunutí tyče. Velká citlivost není nutná a naopak je nevhodná, protože 8 bitů určuje polohu absolutně v menším rosahu (zde 55mm). Proto je jednomu obvodu pro zpracování signálů ze snímačů snížena citlivost 17.4. Jednotka na výstupu pak odpovídá jedné periodě fázových signálů snímače a ne jedné čtvrtině periody jako v případě zpracování signálu z ostatních inkrementálních snímačů.

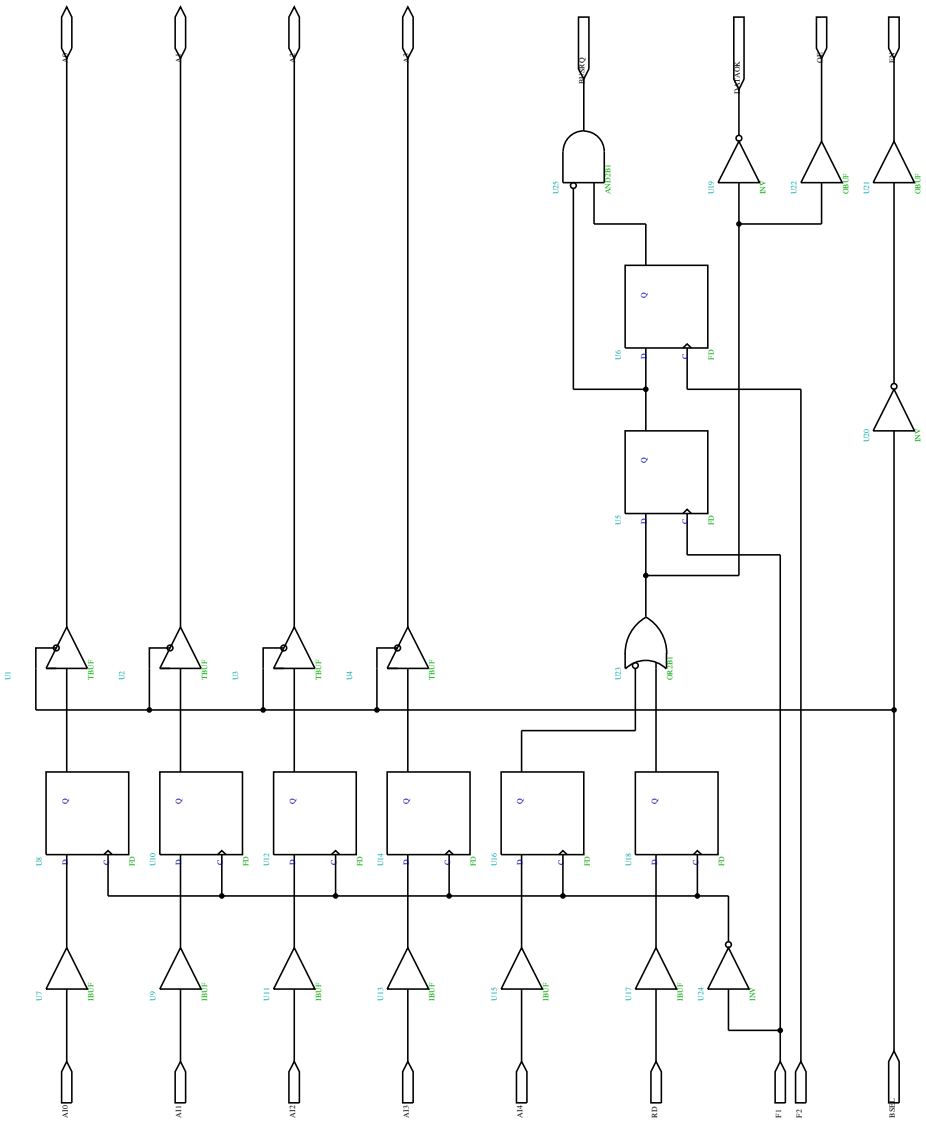


Figure 60: Blokové schema vysílače na heliportu

18 Příloha Popis desky XRI a UN

Deska XRI má dva základní úkoly. První část obvodů vytváří symulovanou sběrnici systému I8080, přes kterou je k počítači připojen nejen přijímací obvod XILINX, ale i vysílací časovače VT. Funkce této sběrnice a jejích obvodů byla poměrně detailně popsána v kapitole 6.2 a není třeba se k ní vracet. Druhou část obvodů na desce XRI tvoří programovatelné logické pole XILINX s podpůrnými obvody. Obvod XILINX přijímá data ze dvou sériových linek, uchovává je v registrech a asynchronně je poskytuje počítači k dalšímu zpracování. Především této části se bude věnovat následující popis.

Deska UN je tvořena universálním plošným spojem a je určena k instalaci dvou konektorů a pro případné další rozšíření systému. Deska UN je součástí interface a konektorem propojena s deskou XRI a deskou VT, protože konektor pro připojení vysílače RCRx je jedním z konektorů na universální desce UN.

18.1 Ovládací prvky a obsluha

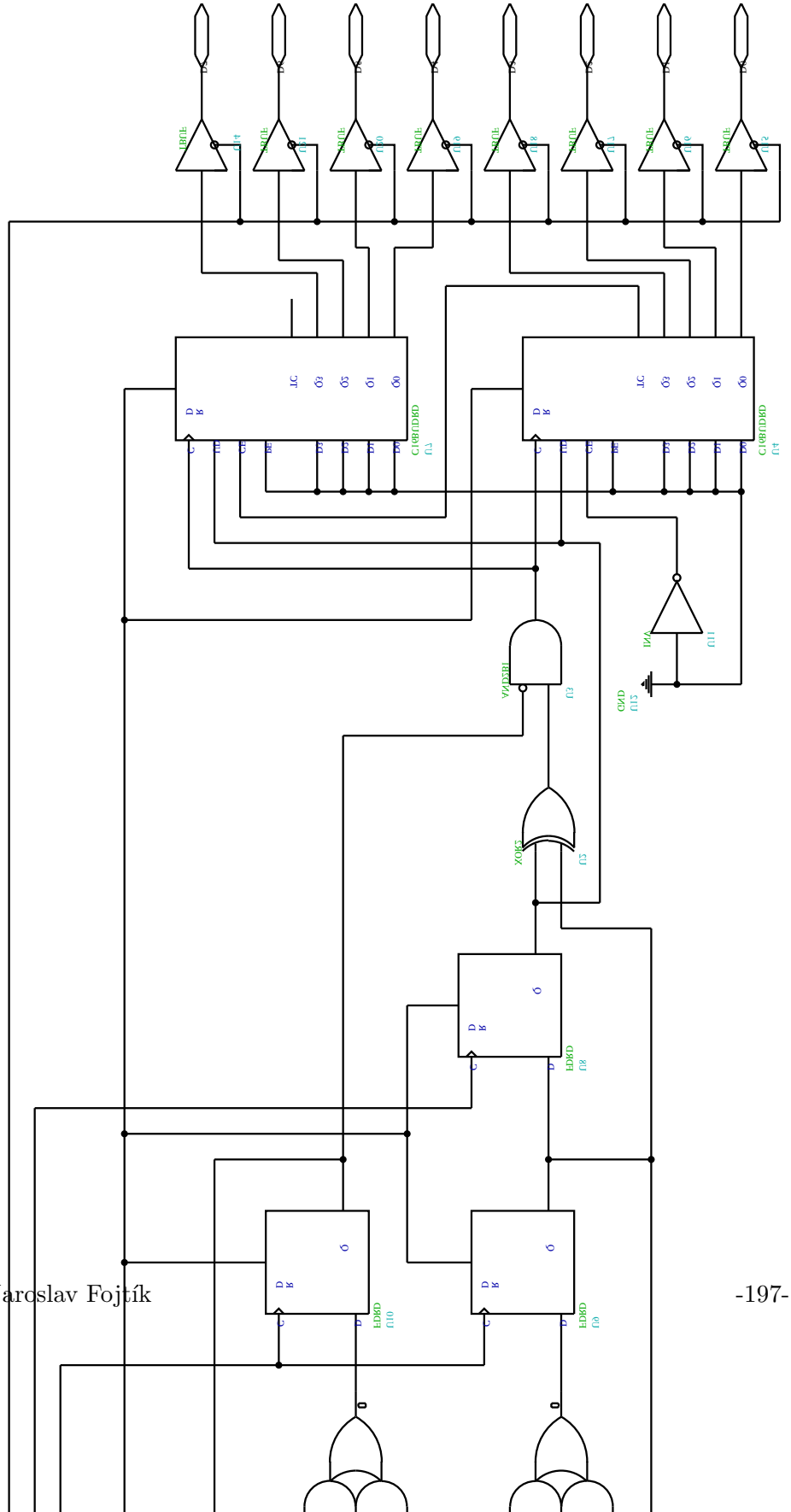
Deska XRI je hlavní součástí interface. Kromě desky XRI je jeho součástí deska VT a naposled přidaná deska UN. Základním ovládacím prvkem je na zadním panelu interface umístěný vypínač, který vypíná obě napájecí napětí (5V, 12V). Přes desku XRI je napájen nejen interface, ale i vysílač RCTx, deska XV2 pod heliporten a datový přijímač Rx. Na předním panelu jsou dvě žluté kontrolní LED diody D9 a D10 pro obě napájecí napětí a osm diod D1 až D8 připojených k vnitřní sběrnici obvodu XILINX, které dávají představu o jeho správné funkci.

Na desce XRI jsou po odkrytování přístupné pojistky P1 pro zdroj 5V a P2 pro 12V. Pro ovládání nahrávání konfigurace jsou jako v případě desek XV1 a XV2 k dispozici tlačítka RESET Tl1 (s delším dříkem hmatníku) a PROGRAM (D/P) Tl2.

18.2 Ovodové zapojení

Obvody IO6 až IO9 tvoří symulátor sběrnice I8080. Ke sběrnici je pevnými vodiči připojena deska vysílacích časovačů VT umístěná v interface nad deskou XRI. Základem přijímacích obvodů je standardní zapojení obvodu XILINX XC3042 IO1 spolu s pamětí IO2 a krystalem 18,423MHz. Integrovaný obvod 74HCT373 IO5 je latch s třístavovým výstupem, který odděluje interní sběrnici obvodu XILINX od sběrnice systému. Data mohou postupovat pouze jedním směrem a to do počítače. Oddělovač je ovládán signály OE a EN generovanými obvodem IO1. Signál OE uvolňuje výstup dat na systémovou sběrnici a je podmíněn průběhem čtecího cyklu na této sběrnici (signál RD), aby nemohlo dojít ke kolizi při nahrávání konfigurace obvodu XILINX IO1.

Protože v programovatelném logickém poli XILINX musel být re-



Číslo vývodu IO1 (XILINX)	Konektor K3 (Cannon 9)	Zdroj nebo cíl signálu
51	2	Sériová data z desky XV2 (Kloub II pod heliportem)
52	4	Sériová data z Rx a tím desky XV1 (Kloub I pod vrtulníkem)
Ground 0V	1, 3, 5	
Zdroj +5V	7, 8	Napájení desek XV1 a Rx
Zdroj +12V	6, 9	

Table 12: Připojení sériových datových kanálů k desce XRI

alizován složitý obvod jehož součástí jsou i paměťové registry (zabere asi 90 až 95% obvodu), muselo být počítáno s možností nedostatku místa v obvodu. Protože data jsou přenášena vlastně z obvodu XILINX do registrů a z nich pak do počítače, bylo jednoduché vřadit mezi systémovou sběrnici a přijímací obvod XILINX paměť. Paměť tvoří dva obvody IO3 a IO4 typu MH7479 s maticí 16x4b. Dohromady tak tvoří paměť 16 bytů, která je řízená přímo z obvodu XILINX. Po dokončení a odzkoušení vnitřního zapojení se ukázalo, že pamětí nebude třeba s čímž jsem počítal. Paměti tak nejsou osazeny v paticích a naopak jsou přemostěny drátovými spojkami což šetří minimálně proud 120mA ze zdroje 5V, které by paměti odebíraly.

Schéma zapojení desky XRI je na 18.2. Na schematu jsou napsána čísla vývodů na obvodu XILINX pro všechny přívody. V 18.2 je uvedeno propojení desky XRI se zdroji dat (deska XV1 a XV2). Deska UN tvoří pouze propojení dvou konektorů K6 a K7. Jejím úkolem je také vytvořit prostor pro případná další rozšíření systému. Na desku UN je z desky XRI vedeno napájení a z desky VT modulační signál pro vysílač. Deska UN je tvořena převážně prázdným universálním plošným spojem a lze ji jednoduše popsat pomocí 18.2.

Konektor K6 A/I připojení počítače (samořezný 20pinů)	Konektor K7 připojení RCTx (Cannon 15 pinů)	Význam signálu
	7, 8	Napájení +12V
	5, 6, 13, 14	Zem napájení 0V
	15	Modulační signál pro RCTx
2, 4, 6, 8	1, 2, 3, 4	Zem analogových signálů
1	9	Analogový vstup A/D 0
3	10	Analogový vstup A/D 1
5	11	Analogový vstup A/D 2
7	12	Analogový vstup A/D 3

Table 13: Zapojení universální desky UN

18.3 Připojení desky XRI a celého interface

Pod vypínačem jsou dva hlavní připojovací dvacetivývodové konektory D/O K1 a D/I K2, kterými se připojuje interface k počítači. Připojení pomocí těchto dvou konektorů je přesně popsáno v kapitole 6.3.3. Na zadním panelu je ještě jeden stejný konektor K6 desky UN pro připojení analogových vstupů ke kartě PCL812, který má později sloužit jako vstup údaje z křížových ovladačů. Konektor (K5) Cannon s patnácti vývody slouží pro připojení vysílače RCTx. K vysílači je veden modulační signál z desky VT a napájení. Zpět od vysílače mají být vedeny analogové údaje z křížových ovladačů. Posledním konektorem je devíti vývodový Cannon K3, kterým je vedeno napájení k heliportu pro desku XV2 a Rx. Spolu s napájením jdou i dva sériové datové spoje od desky XV2 a desky XV1 přes bezdrátový spoj. Pohled na přední i zadní panel interface je na 20.2 a 20.2, které jsou na straně 213.

18.4 Vnitřní zapojení obvodu XILINX na desce XRI

Základem zapojení je opět stejný generátor hodin 16.4 vycházející z krystalového oscilátoru s kmitočtem 18,432MHz. Tento hodinový obvod zajišťuje postačující stabilitu a minimální rozdíl v časování přijímače a vysílače, což je jedna z podmínek dobrého dekódování sériového signálu.

Středem zapojení příjmacího obvodu XRI je datová (8b) a adresová sběrnice (4b). Ke sběrnicím jsou připojeny paměťové registry, v nichž se uchovávají přijatá data. Nutnou součástí jsou dva přijímací obvody složené z majoritního filtru a přijímacího registru. Tyto obvody jsou stejné a liší se pouze adresou, na níž zapisují přijatá data (0 až 3 a 4 až 7). Přístup počítače je zajištěn přes takzvaný výstupní registr a vnější obvody, které byly popsány již dříve. Blokové zapojení je na 18.4.

Protože sběrnice je sdílena více zařízeními musí existovat obvod

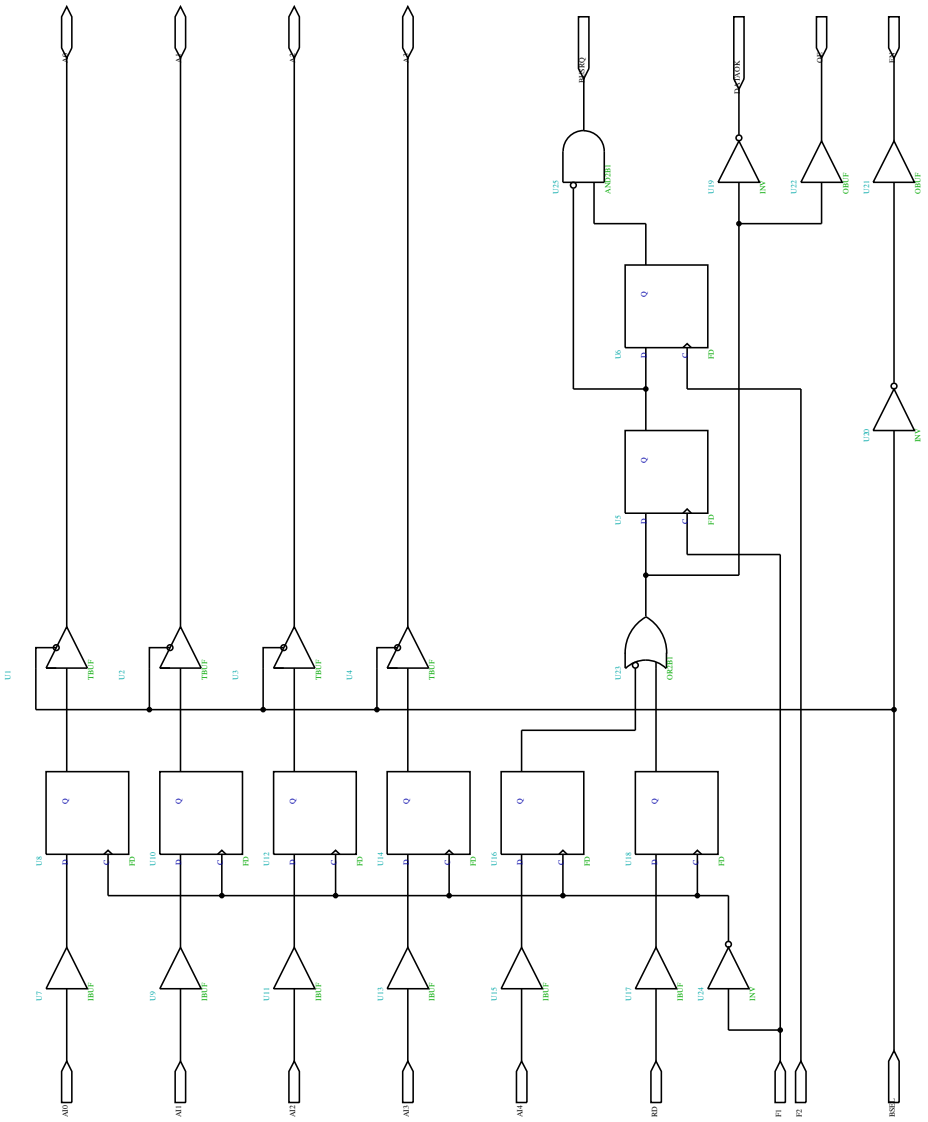


Figure 62: Blokové schema přijímače

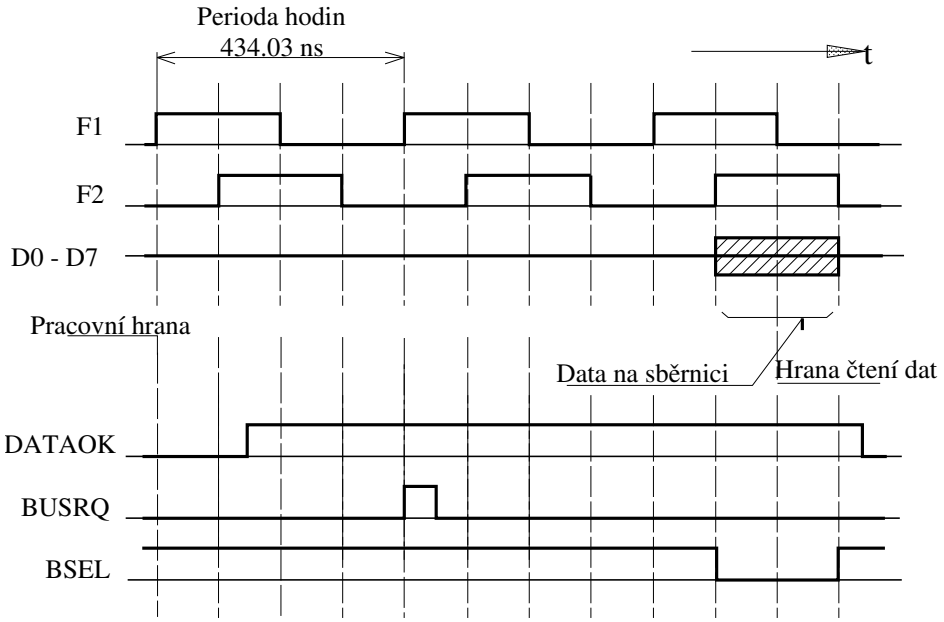


Figure 63: Časování sběrnice s požadavkem na přenos (využito jen v XRI)

správy sběrnice. Tento obvod rozhoduje o tom, kdo bude využívat sběrnici. Obecně se dá říci, že ten, kdo požádá o sběrnici jako první ji také dostane. Pouze v případě více žádostí se rozhoduje podle připojení požadavků k obvodu správy sběrnic (pevně stanovená priorita). Na obvodu správy sběrnic jsou tři trojice řídících signálů označených pořadovým číslem, které roste s klesající prioritou. Požadavek generuje zařízení signálem DATAOK v log.1 a nástupnou hranou signálu BUSRQ. Signál DATAOK musí být v log.1 po celou dobu přenosu jinak je požadavek okamžitě zapomenut. Předání sběrnice je detekováno signálem BSEL (aktivní v log.0). Obvod správy sběrnic generuje i signál MIN (memory in) indikující zápis do registrů. Tento signál je vlastně logickým součtem BSEL2 a BSEL3, které označují přenos z přijímacích registrů. Přehledně je časování požadavku na sběrnici a jejího využití zachyceno na 18.4.

Na 18.4 je blokové schéma paměťové části obvodu. Každý snímač má odpovídající paměťový registr 18.4 do něhož je ukládán údaj o jeho stavu (adresy 0 až 6). Kromě paměťových registrů je připojena i pevná paměť s kontrolním řetězcem PK&JF 18.4 (adresa 11 až 15). Později byly přidány dva kontrolní čítače přenosů (4b) 18.4, které počítají počet přenosů z jednotlivých přijímacích registrů do paměťových registrů. Každý čítač je určen pro jeden přijímací kanál. Čítače přenosů jsou nulovány přečtením kteréhokoliv znaku z kontrolního řetězce. Pro zápis a čtení z registrů slouží řídící signály. Každý registr má vstup CS (chip select) aktivní v log.1. Tento

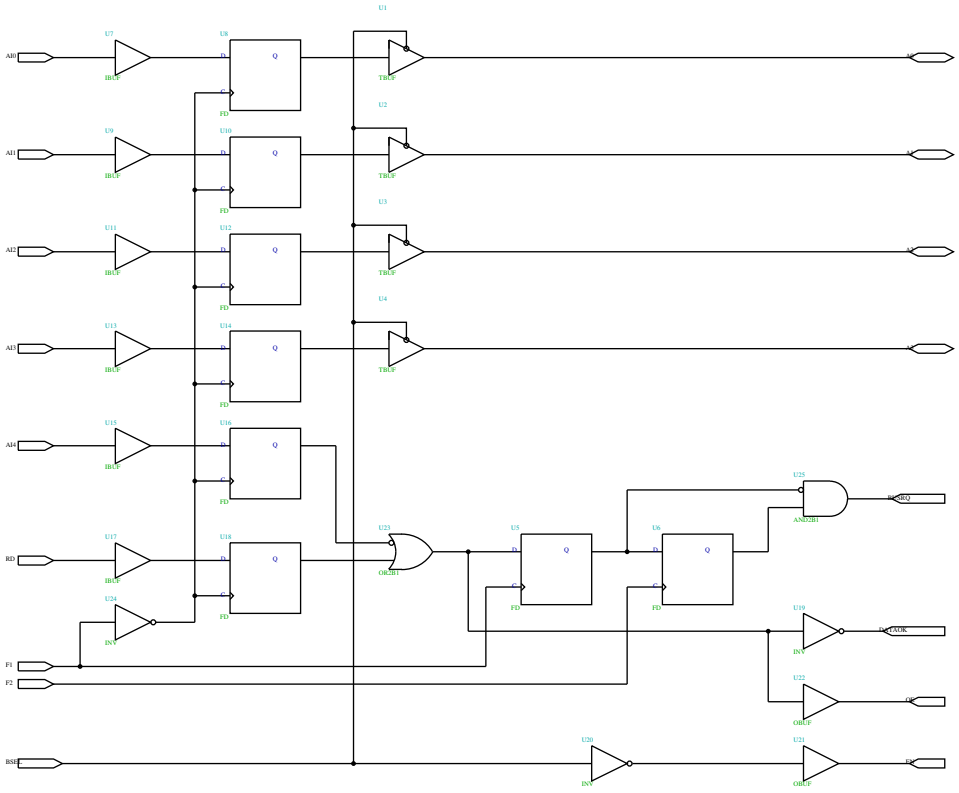


Figure 64: Správa sběrnic přijímače

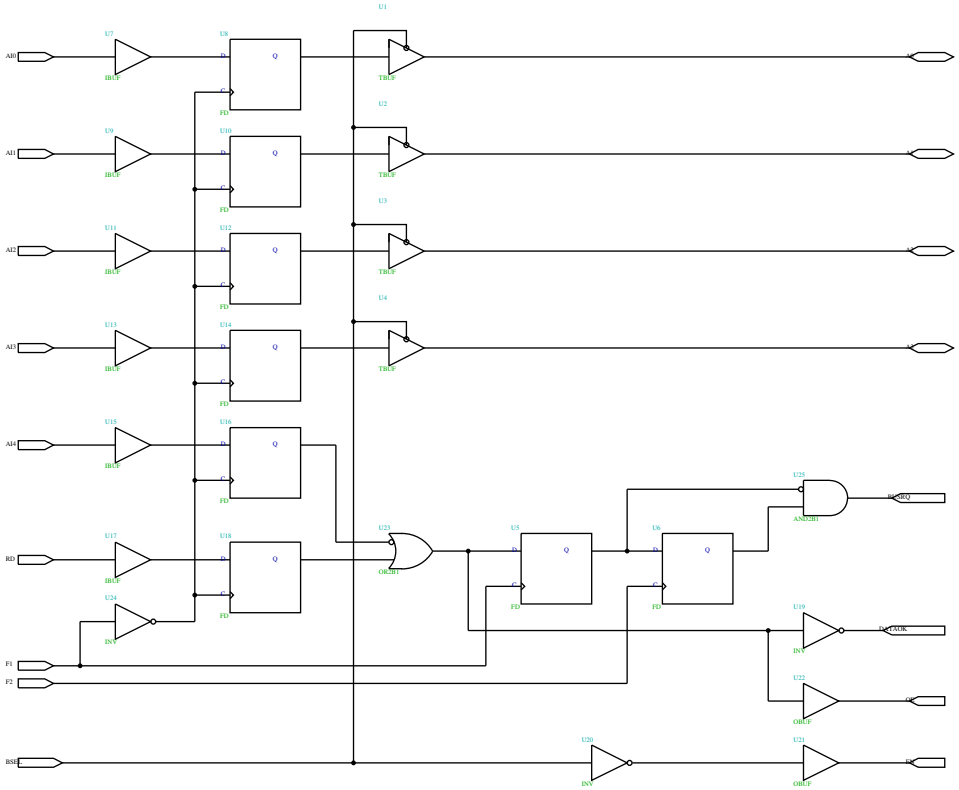


Figure 65: Datový paměťový registr

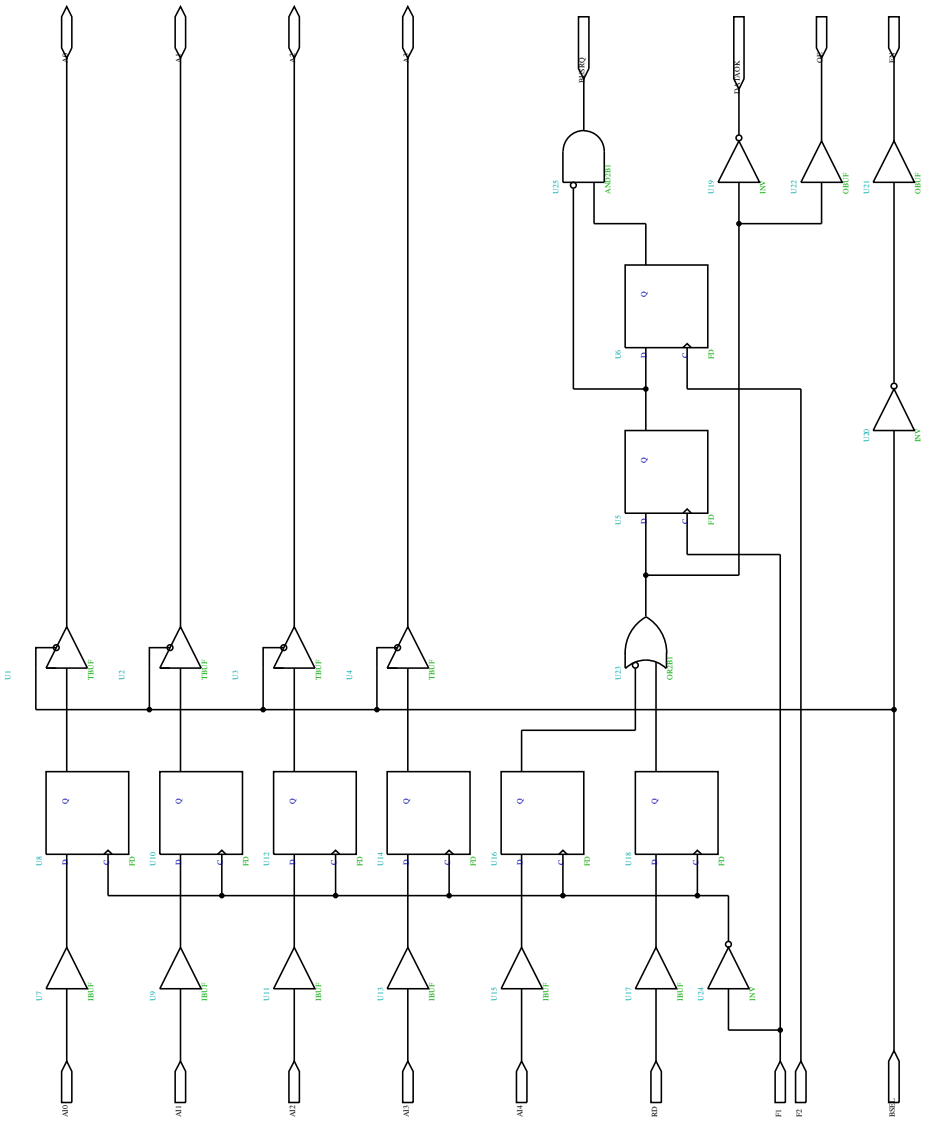


Figure 66: Blokové schema paměti přijímače

signál určuje, o který registr půjde. Signál DIN (aktivní v log.1) a DOUT (aktivní v log.0) rozhodují o zápisu či čtení dat. Zápis dat do registru se provádí sestupnou hranou F1 za předpokladu, že CS=1 a DIN=1.

Data přicházejí sériovou linkou do majoritního filtru 18.4 tvořeného

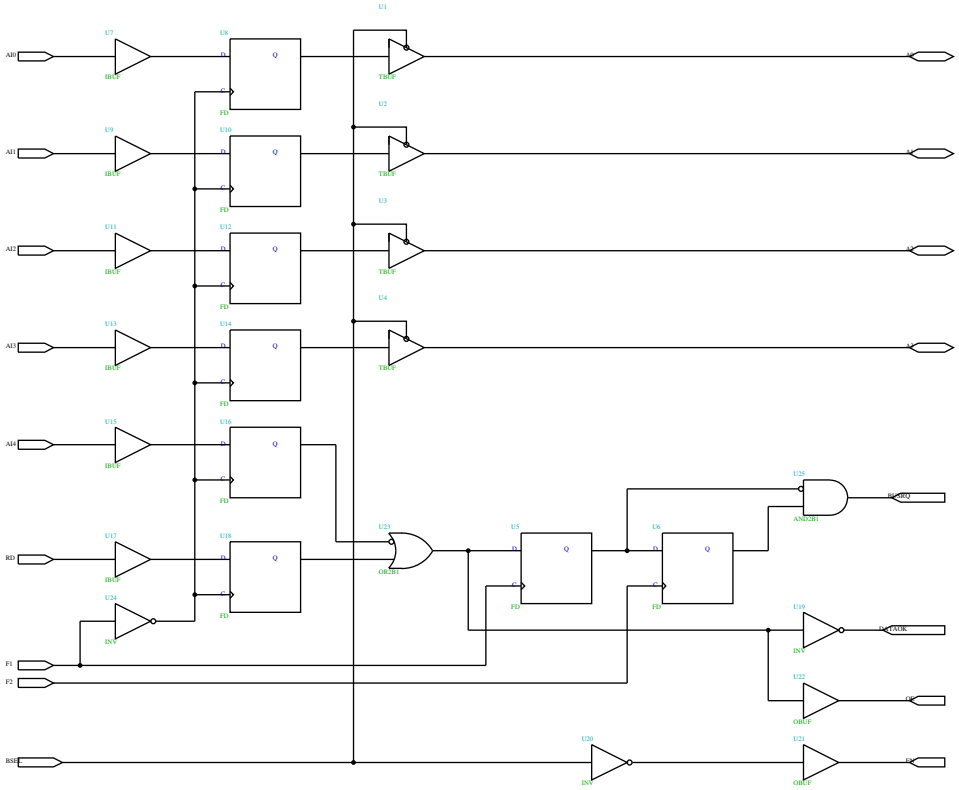


Figure 67: Kontrolní obvod ROM (PK&JF)

třemi klopnými obvody typu D. Klopné obvody jsou seřazeny do posuvného registru posouvaného rychlostí základních hodin (2,304MHz). Výstupem filtru je potom funkce většiny ze stavu jednotlivých kaskád registru. Součástí je i detektor náběžné hrany signálu tvořený klopným obvodem U8. Náběžná hrana datového signálu se používá k synchronizaci přijímacího časovače. To umožňuje dekódování jak signálu v kódu NRZ tak i signálu s pulsně šířkovou modulací.

Data jdou z filtru do přijímacího posuvného registru 18.4 dlohého 32b (délka vysílaného slova). Součástí schématu je i časovací obvod představovaný čítačem U1 a dekodéry jeho stavu. Tento čítač je nástupnou hranou přijatého signálu nulován. Délka jednoho bitu je právě 16 period základních hodin (F1=2,304MHz) a shoduje se tak s dobou čítání čítače

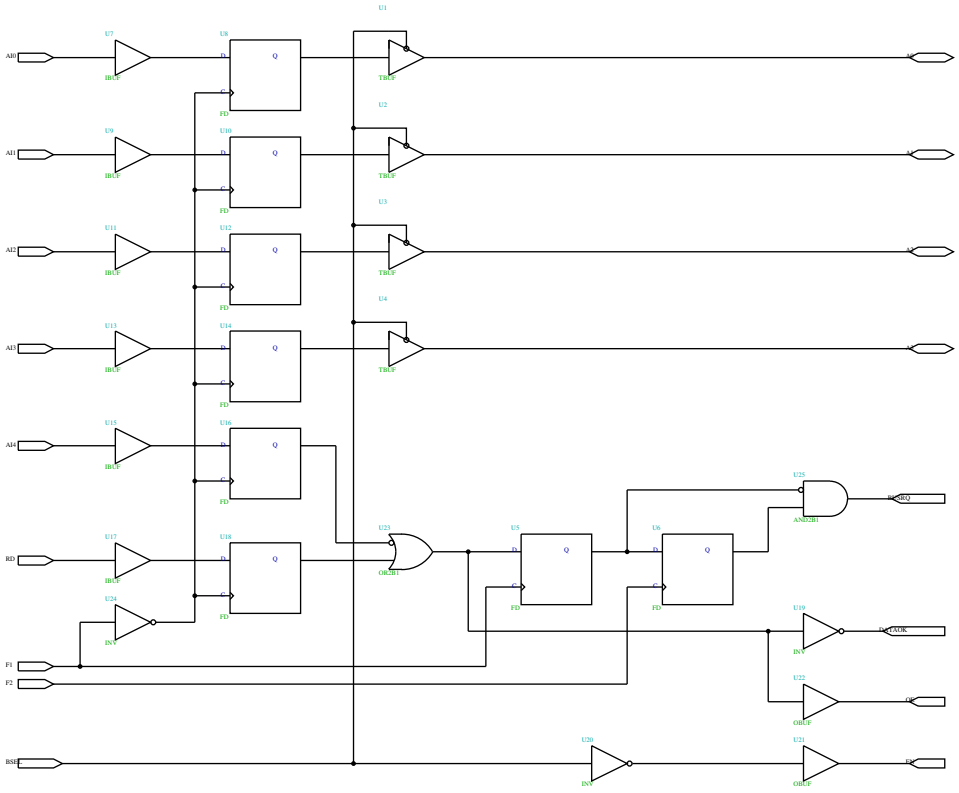


Figure 68: Kontrolní čítač uskutečněných přenosů

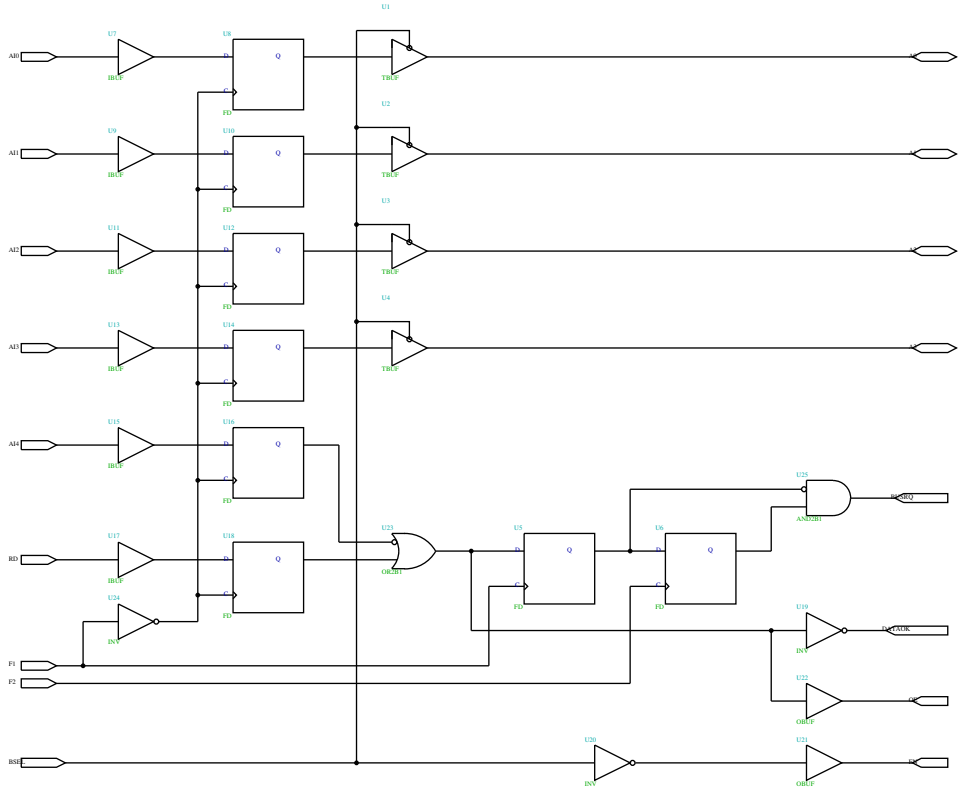


Figure 69: Majoritní 3 bitový filtr a detektor hran

U1. Vzorkování vstupních dat se provádí v polovině bitu, neboli čítač je posouván vždy když čítač U1 napočítá do 0111B. Posuvný registr se posouvá stále s neměnným rytmem. Je ovšem nutno zjistit, kdy bylo přijato správné slovo. V případě, že je v posuvném registru celé slovo, je nutno ho přenést dál dříve než dojde k opětnému posunutí. Správnost dat udává signál DATAOK (aktivní v log.1). Tento signál je testován požadavkem na přenos BUSRQ (aktivní nástupnou hranou), který se generuje vždy následující takt po posunutí dat v registru (U1 ve stavu 1000B). Když budou data správná vygeneruje se touto posloupností žádost o sběrnici, která bude vyřízena do tří taktů hodin F1. Řídící obvod dá signál k přenosu BSEL (aktivní v nule). Protože na vnitřní sběrnici přistupuje více obvodů musí být všechny tyto obvody vybaveny třístavovými oddělovači.

Výstupní registr 18.4 má za úkol dekódovat a oddělovat adresu dodanou počítačem od vnitřní sběrnice. Zároveň generuje žádost o přenos dat po sběrnici na základě signálu RD a dekódované adresy. Po předání sběrnice jsou předána data na vnitřní sběrnici, která je vyvedena ven z obvodu XILINX. Data jsou zaznamenána ve vnějším reguistru a oddělovači IO5 na desce XRI. Řízení IO5 zajišťuje také výstupní registr signály EN a OE.

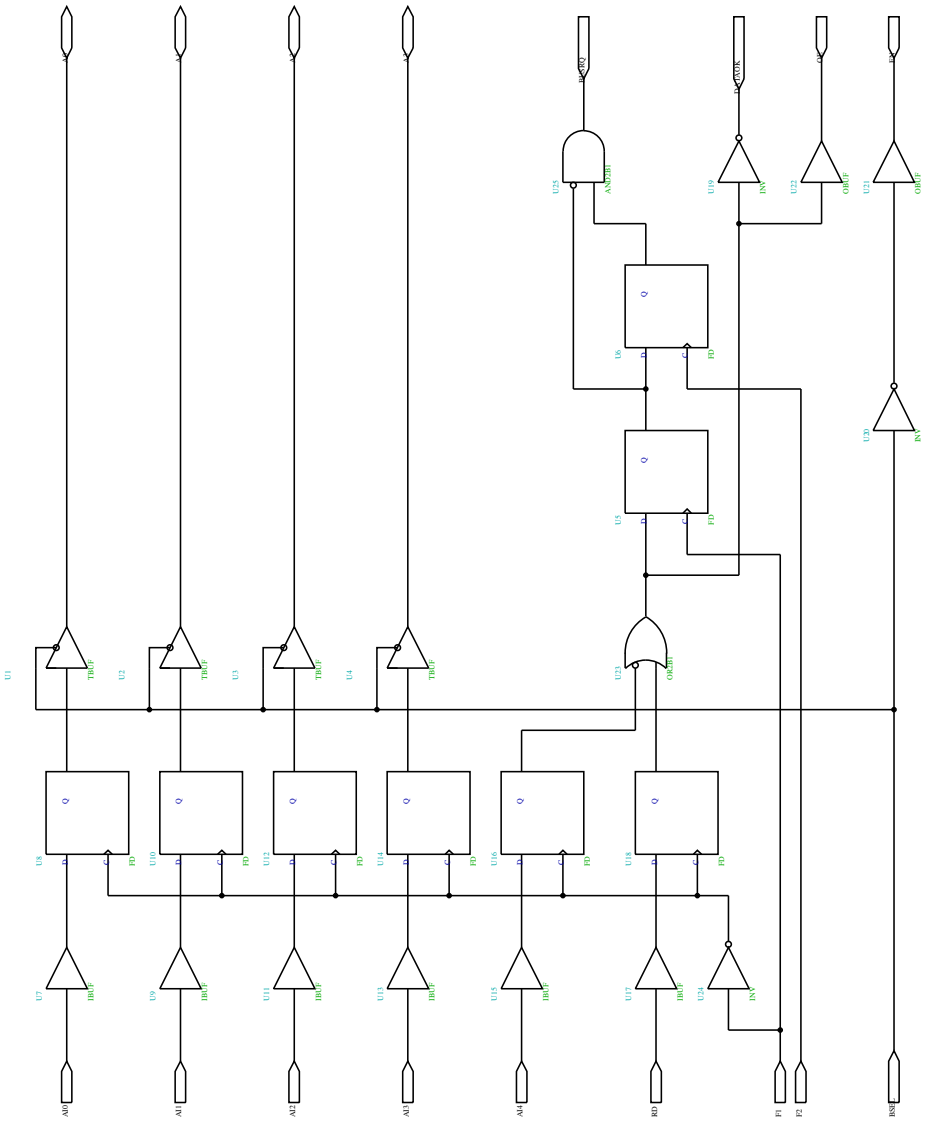


Figure 70: Přijímací posuvný registr a časovací obvody přijímače

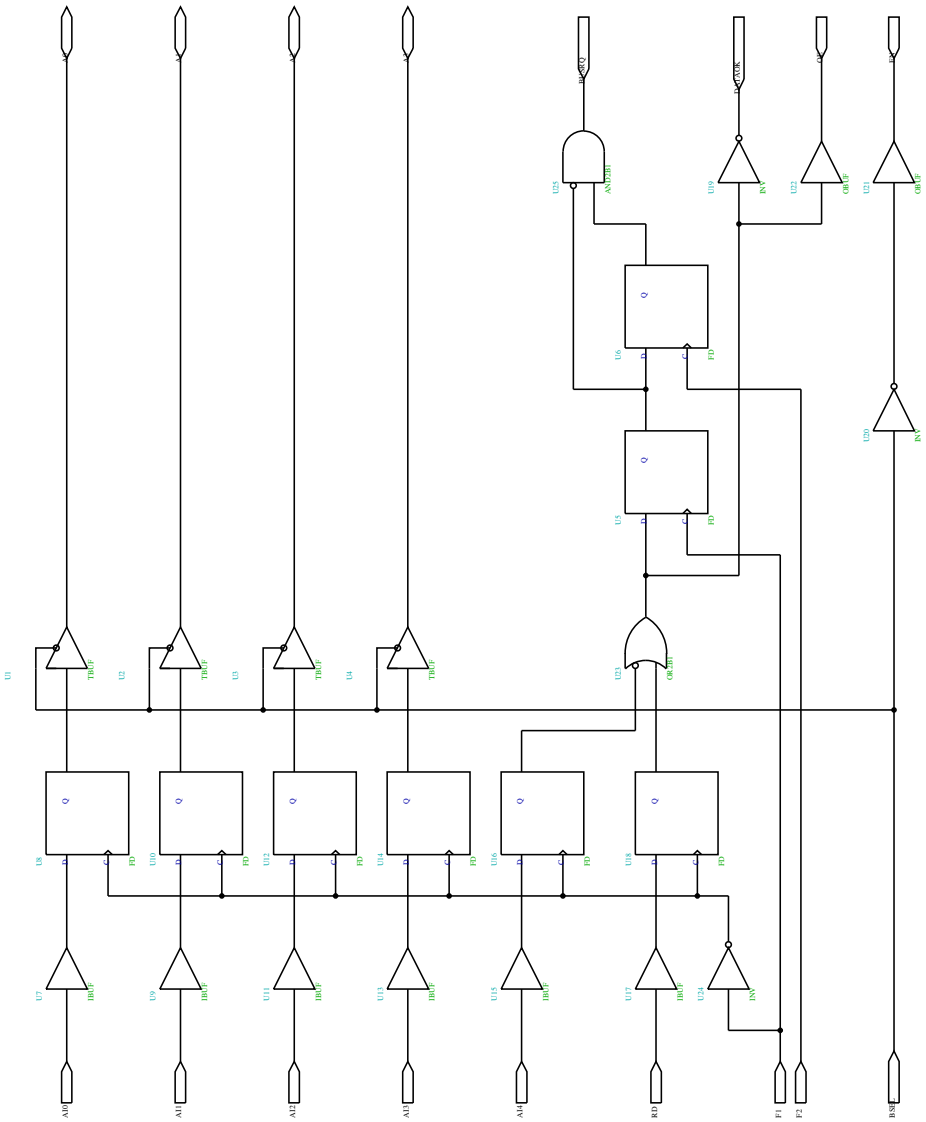


Figure 71: Výstupní registr

19 Příloha Bezdrátový datový spoj

Popis koncepce celého zařízení je v kapitole 6.5 a zde jsou uvedeny pouze schémata se stručným komentářem. Bezdrátový datový spoj se skládá z přijímače Rx a vysílače Tx. Obě tyto části jsou realizovány jako zapojení na univerzální desce.

19.1 Deska Tx (vysílač)

Deska Tx obsahuje vlastně pouze VF zesilovač, protože z desky XV1 již přichází modulovaný signál s kmitočtem nosné 18,432MHz. S deskou XV1 je spojen vysílač třívodičovým kabelem opatřeným na konci konektorem. Z desky XV1 se přivádí datový signál a napájecí napětí 10V. Konektor na desce spojuje vysílač s jeho vysílací cívkou Lant, která je umístěna pod vrtulníkem. Schéma zapojení vysílače (deska Tx) je na 19.1. Samotná deska je umístěna v zadní části trupu pod místem upevnění ocasní trubky.

19.2 Deska Rx (přijímač)

Oproti popisu v kapitole 6.5.3 byl na vstupu přijímače použit jednoduchý integrovaný obvod. Důvodem jeho použití byla snaha, aby přijímač měl dostatečný zisk, protže za VF zesilovačem následuje jednoduchá diodová detekce a je nutno mít již dostatečný rozkmit signálu. Napájecí napětí přijímače je 12V s vyjímkou NF zesilovače, který musí být napájen 5V z důvodů kompatibility se vstupy obvodu XILINX. Schema zapojení přijímače je na 19.2.

Deska Rx je umístěna těsně pod heliportem v krabici společně s deskou XV2. Přijímací cívka se stejně jako v případě vysílače připojuje konektorem. Druhý konektor spojuje přijímač s deskou XRI. Z desky XRI jsou přes tento konektor vedeny i obě napájecí napětí.

20 Příloha Mechanické řešení desek

20.1 Desky plošných spojů

Deska plošných spojů XV1, XV2 a XRI byly navrženy a nakresleny ručně. Návrh počítal s možností úprav zapojení, které jsou při vývoji složitých zařízení běžné. Integrované obvody XILINX jsou vyvedeny do propojovacího pole a s konektory jsou spojeny drátovými propojkami což umožní provádět jednoduše úpravy. Plošným spojem je rozvedeno napájení a případně realizován zdroj. To platí především pro desku XV1 a XV2, které jsou z hlediska plošného spoje totožné.

Deska XRI musela být vzhledem k počtu spojů reralizována prakticky celá na plošném spoji ovšem s tím, že jsou na ní propojovací pole a zůstává tak možnost přerušit příslušný spoj a spojení provést vodičem. Tím je zachována pozdější možnost jednoduchých úprav. Na 20.2 jsou orientační výkresy rozmístění nejdůležitějších součástí všech tří desek. Protože desky XV1 a XV2 jsou stejné, je uveden pouze jeden výkres pro obě společně.

Zdroj ZD1 a desky přijímače Rx a vysílače Tx jsou řešeny zapojením na univerzálních deskách plošných spojů. Obdobná je i deska UN, která má tuto vlastnost již v názvu. Protože se jedná o doplňkové obvody, neuvádím jejich rozmístění součástek.

20.2 Umístění desek

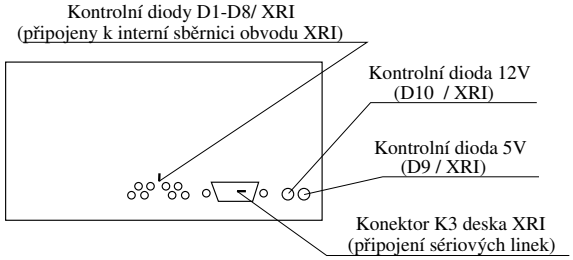


Figure 72: Interface - přední panel

Deska UN VT a XRI jsou společně umístěny v umělohmotové krabici nazývané interface. Rozmístění těchto desek je na 20.2.

Na vrtulníku jsou desky XV1 a ZD1 upevněny společně s přijímačem RC soupravy RCRx. Přímo na motoru je umístěn jeho spínač. Na něj vede také hlavní napájecí přívod. Od spínače se napájení rozvádí po jednotlivých částech systému. Základem pro upevnění desky XV1 se stala nosná destička přimontovaná vertikálně ve směru letu na přídi modelu. K této desce je z jedné strany přišroubována deska XV1 a z druhé strany pak zdroj ZD1 a přijímač RCRx. Celá sestava je na 20.2.

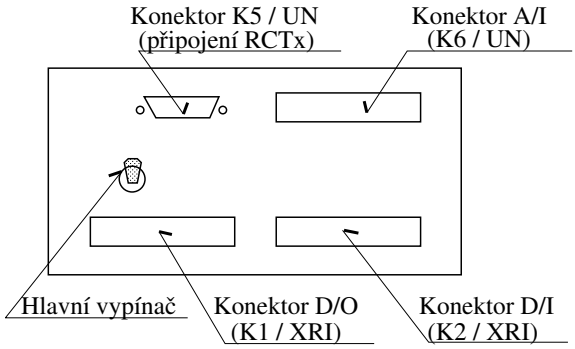


Figure 73: Interface - zadní panel

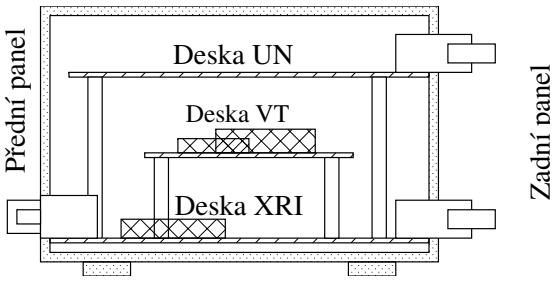


Figure 74: Interface řez

Deska XV2 a Rx jsou v umělohmotové krabici, která je připevněna těsně pod heliportem. Rozmístění desek v této krabici je na 20.2.

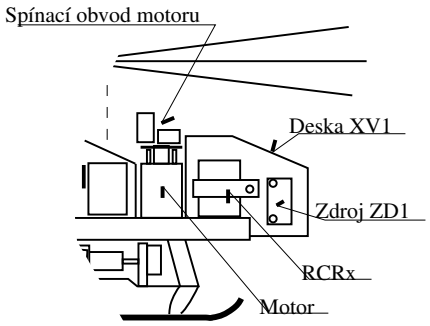


Figure 75: Umístění desek na vrtulníku

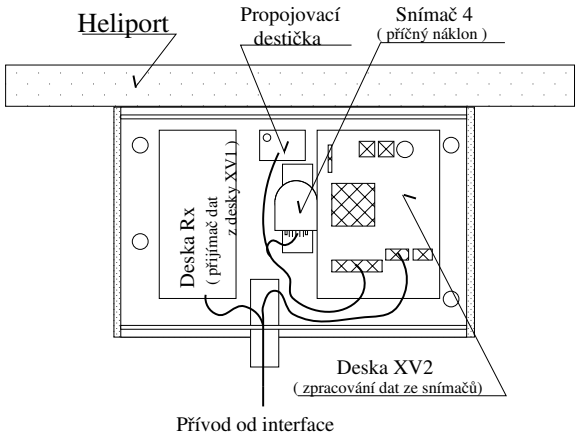


Figure 76: Umístění desky XV2 a Rx pod heliportem

21 Příloha Realizovaná zařízení

21.1 Přijímač s vnějšími registry

Přijímač je vestavěn do víceúčelové krabice o rozměrech 180x160x85 určené pro amatérské výrobky.

Figure 77: Pohled na přední panel

Na přední straně je umístěn propojovací konektor k desce II pojmenované podle 6.5.4. Dvě žluté diody napravo signalizují přítomnost napětí 5V a 12V. Čtveřice diod nalevo od konektoru je pevně připojena k vyšším 8 bitům vnitřní sběrnice a umožňuje opticky kontrolovat její činnost.

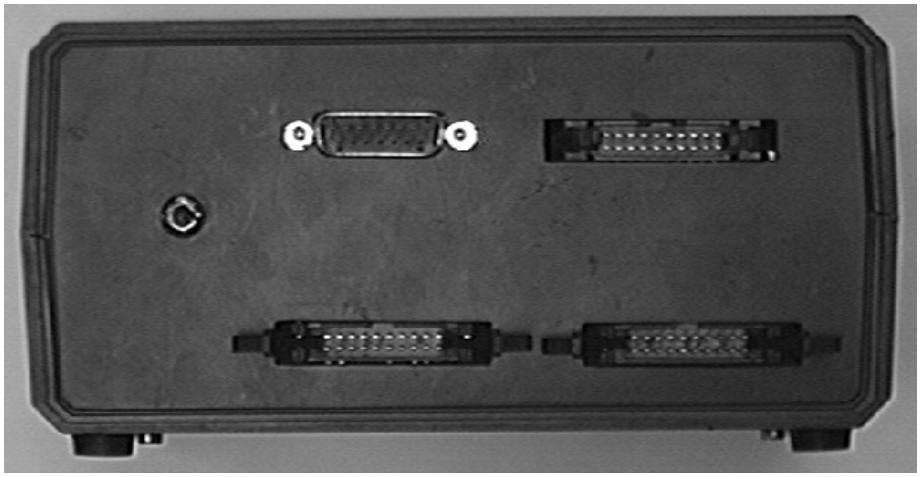


Figure 78: Pohled na zadní panel

S IO kartou je zařízení povinně propojeno dvěma 20 žilovými plochými kabely: jeden přivádí digitální vstupy a druhý propojuje digitální výstupy. Oba jsou umístěny v dolní části zadního panelu. Volitelně může být propojen ještě třetí kabel pro snímání poloh ovládacích páček na vysílačce. Nalevo je umístěn hlavní vypínač. 15 pinový konektor Canon je určen pro napájení a předávání kódovaného signálu do vysílačky.

Celé zařízení lze rozebrat povolením čtyř šroubů umístěných v umělohmotných nožičkách. Krabička se rozpadne na dvě části. Pohled na vrchní desku je zobrazen na 21.1. Vrchní deska není z větší části osazena součástkami a v případě potřeby na ni lze přidat jakékoliv elektrické zapojení. Pro usnadnění dodatečných úprav byl vrchní tištěný spoj použit univerzální. Vrchní deska je se spodní spojena částí konektoru FRB a mechanicky je připevněna pomocí šroubů ke třem distančním sloupkům. Mechanické připevnění je přizpůsobeno snadné vyjímatelnosti vrchní desky.

Po jejím vyjmutí se otevře pohled na hlavní část celého zařízení. V levém dolním rohu se nachází hlavní vypínač a pod ním dvě pojistky pro napětí 5V a 12V. V levém horním rohu je umístěn programovatelný obvod Xilinx. Nalevo jsou vidět dvě tlačítka a pod nimi sériová paměť umístěná v patici. Pravou polovinu zabírá původní deska programovatelných časovačů osazená dvěma obvody 8253. Povolením dvou šroubů ji lze uvolnit a částečně vyjmout. Je pevně spojena svazkem kabelů se základní spodní deskou.

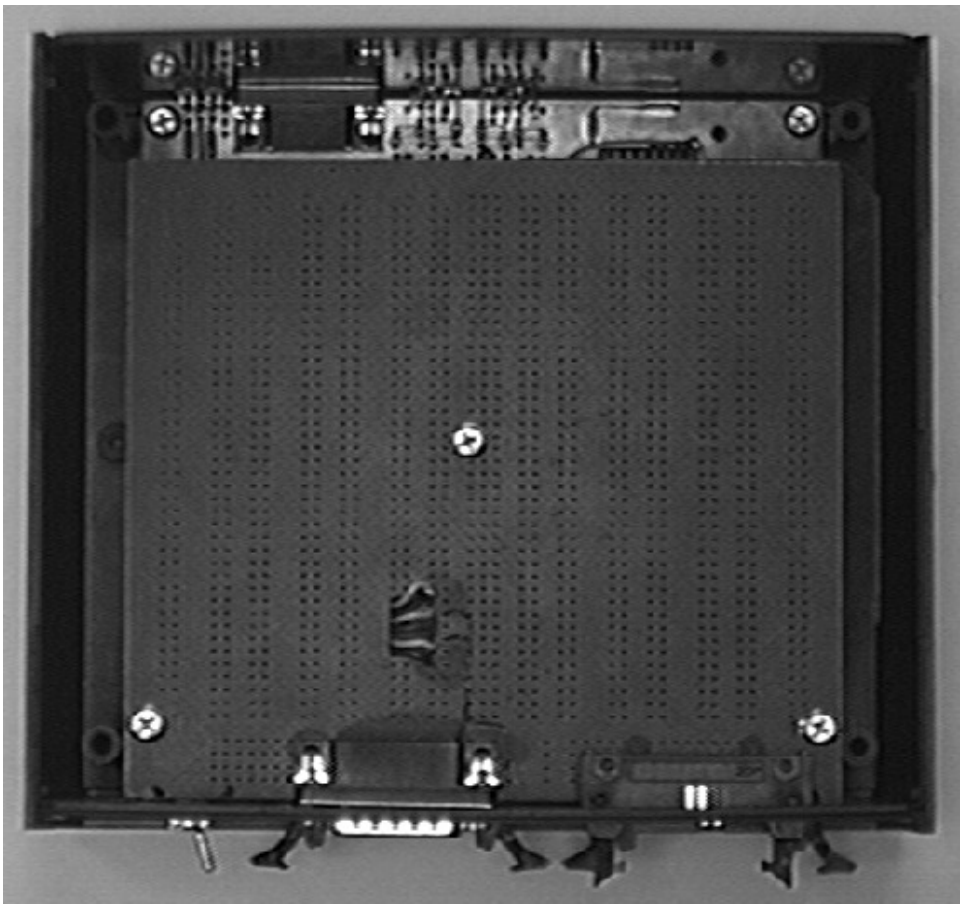


Figure 79: Pohled na vnitřní část zařízení

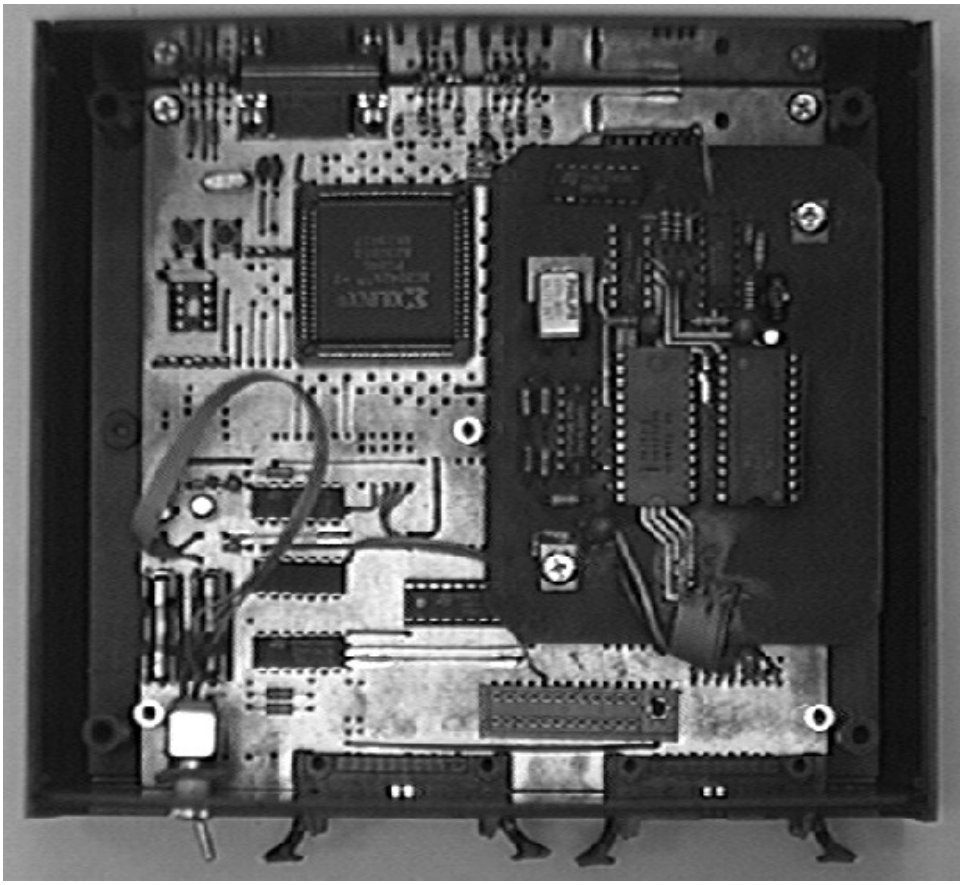


Figure 80: Vnitřní část po odkrytí vrchní desky

Po jejím vyjmutí si lze na následujícím obrázku prohlédnout celou spodní desku. Spodní deska je připevněna čtyřmi šrouby k spodnímu dílu krabice a po jejich vyšroubování ji lze od krabice oddělit. Musí být vysunuta současně s předním a zadním panelem.

21.2 Model vrtulníku

Na obrázku 21.2 je zobrazen celkový pohled na model vrtulníku. Jednotlivé části vrtulníku a jejich funkce již byly rozebrány v kapitole *Popis modelu*.

Po sejmutí krytu kabiny si lze na 21.2 prohlédnout HW část vyhodnocovače polohy horního Cardanova kloubu a vysílače sériového kódu. V této části je také prováděno měření otáček hlavního rotoru a řízení motoru.

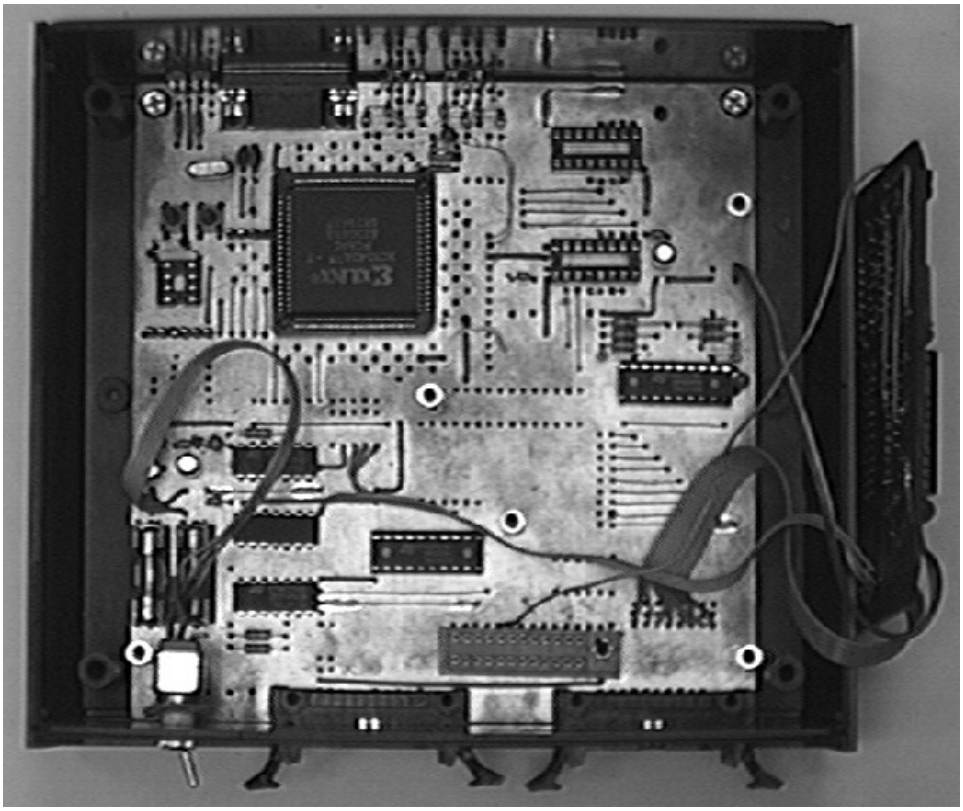


Figure 81: Pohled na spodní desku s plošnými spoji

Figure 82: Pohled na model helikoptéry

Původní přijímač bezdrátového ovládání polohy serv zůstal zachován a je umístěn za deskou s plošnými spoji.

Detailní pohled na spodní Cardanův kloub obsahuje 21.2. Hřídel IRC snímače viditelného uprostřed obrázku je uzpůsobena k pevnému spojení s vodící tyčí trenažéru. Problém přívodů napájecího napětí zatím nebyl uspokojivě dořešen.

21.3 Vysílač pro řízení modelu

Pro ovládání serv byl použit upravený původní vysílač. Pohled do vnitřní části je zobrazen na 21.3. Způsob úpravy byl popsán v kapitole *Komunikace PC-model*. Prostředkem horní části prochází prutová anténa. Nalevo od antény se nachází přepínač pro volbu ruční řízení/řízení počítačem. Napravo je umístěn konektor pro přivedení kódovaných sériových dat z počítače. V dolní části vysílače byl doplněn tištěný spoj se zesilovačem a oddělovačem signálu z počítače.

21.4 Přistávací plošina vrtulníku

Přistávací plošina je vyrobena ze dřeva. Uprostřed je vyříznut kruhový otvor pro vodící tyč. Při provozu se vodící tyč čtvercového profilu zasunuje do spodního kloubu. Pro skutečné experimenty bude vhodné na její

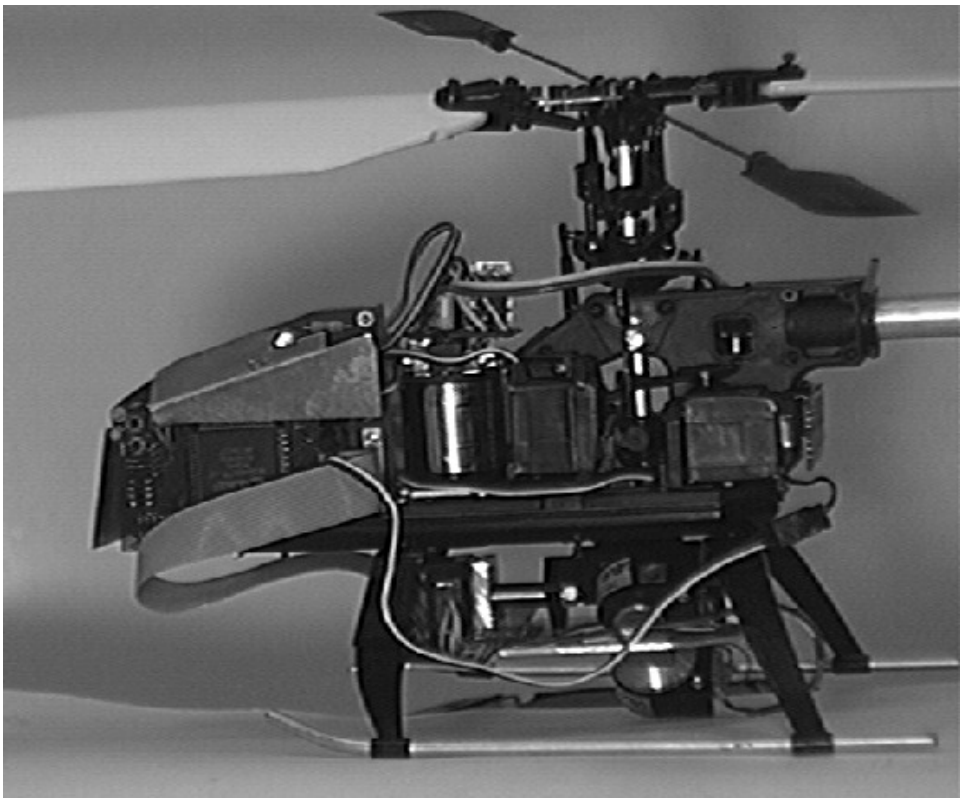


Figure 83: Pohled na část vyhodnocování otáček a řízení motoru

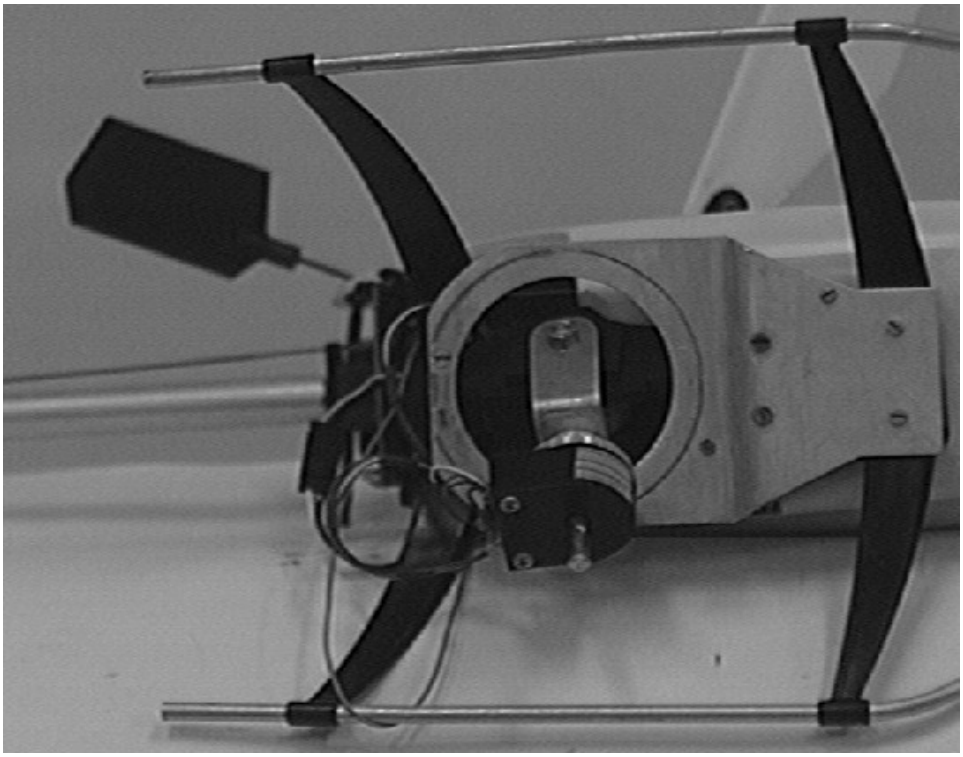


Figure 84: Pohled na horní Cardanův kloub

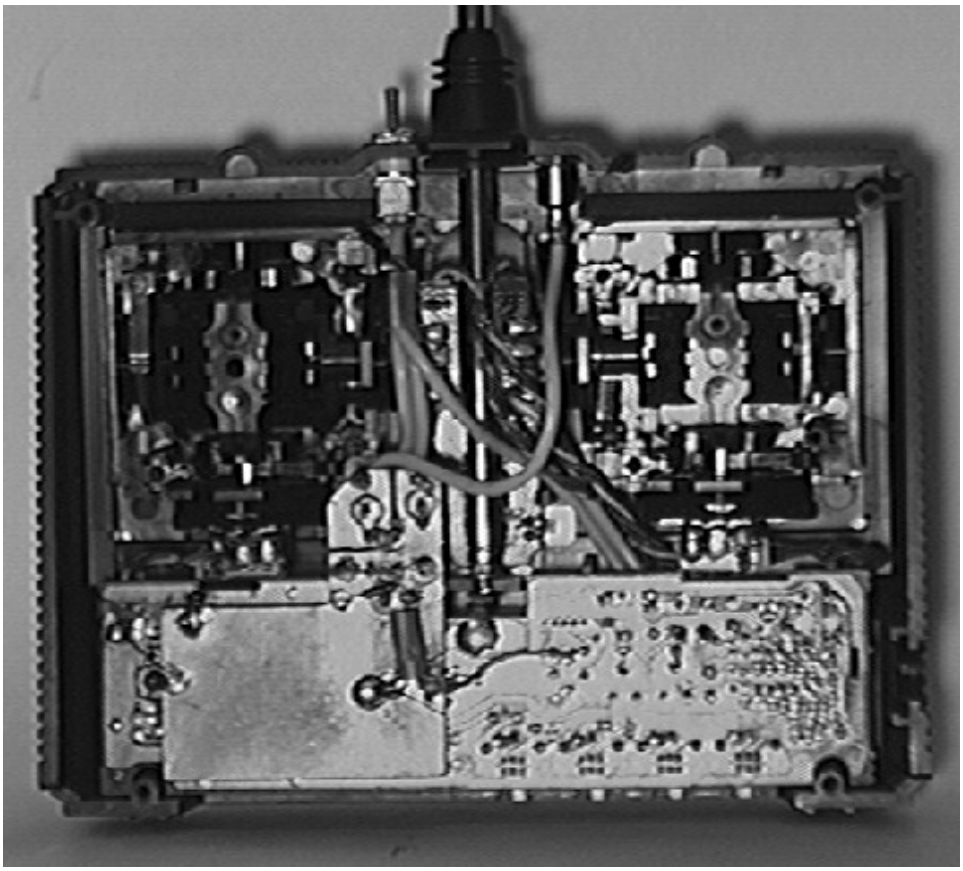


Figure 85: Pohled do útrob vysílače

povrch nalepit měkkou gumu pro utlumení možných pádů modelu. Pohled na heliport včetně spodního Cardanova kloubu je na 21.4. Na obrázku jsou dobře viditelné IRC snímače polohy kloubů, které jsou přidělány k mechanice kloubů. Ty jsou elektricky propojeny s vyhodnocovací částí, která je umístěna v černé krabici pro amatérské výrobky. Celý kloub je mechanicky vyvážen závažími. Na obrázku je zobrazen detailní pohled na přistávací plošinu. Cekový pohled na mechanickou část trenažéru je nakreslen na 4.5.2.

Na 21.4 je zachycena vyhodnocovací logika IRC snímačů. V této

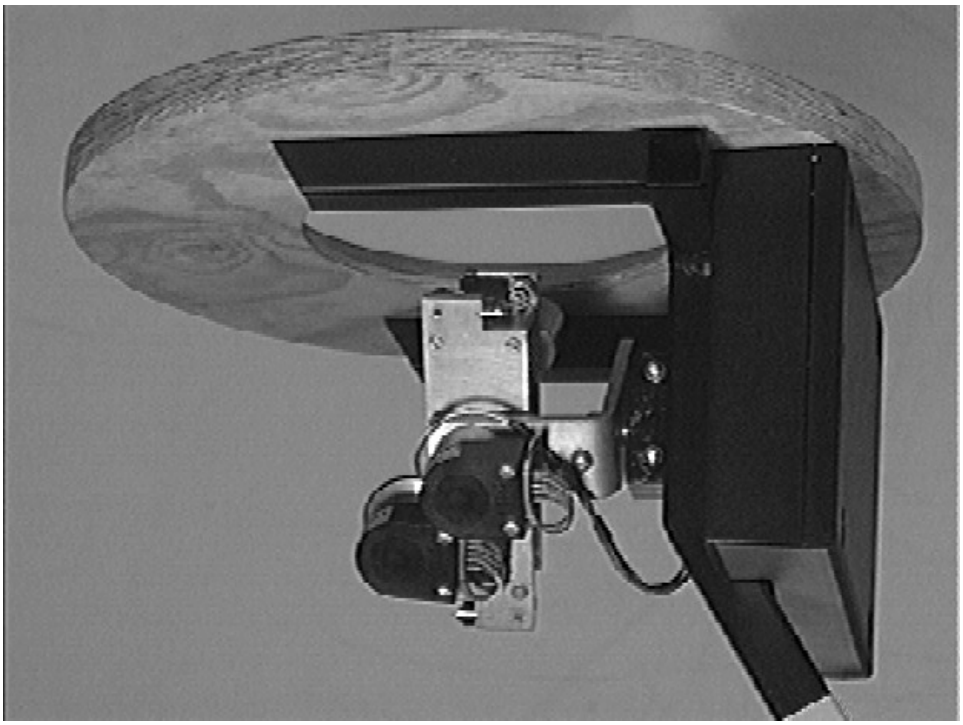


Figure 86: Pohled na heliport a spodní Cardanův kloub

části je snímána poloha spodního Cardanova kloubu a délka vysunutí vodící tyče. Informace o poloze jsou posílány sériovým kódem do přijímacích registrů. Celá logika je soustředěna do programovatelného logického pole Xilinx. V celkové sestavě na 6.5.4 je zařízení nazváno DESKA II. Pod přistávací plošinou je také umístěn opakovač rádiově vysílaných dat z modelu helikoptéry. Oba sériové kanály jsou vedeny kabelem do spodní desky označené jako INTERFACE.

Přistávací plošina je již z větší části dokončena. Při praktických pokusech bylo zjištěno, že vodící gumové kolečko pro měření vysunuté části tyče občas prokluzuje. Napájecí vodiče budou procházet vodící tyčí a přívod napájení si nevyžádá žádnou změnu v mechanické konstrukci spodního

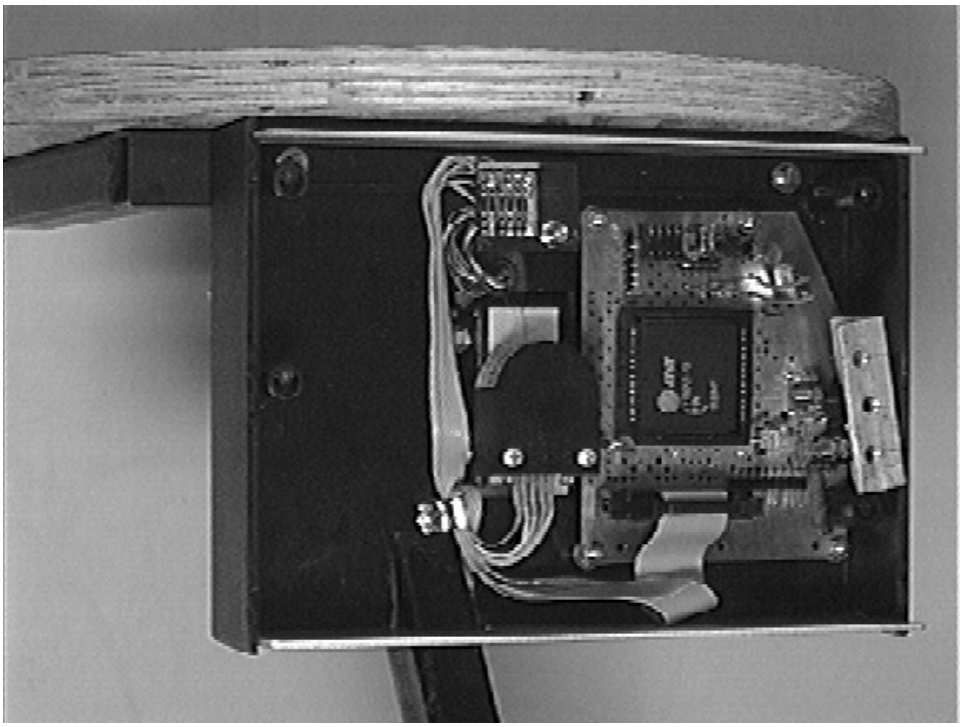


Figure 87: Pohled na vyhodnocovací logiku IRC snímačů

kloubu.

22 Příloha Použitá literatura

[1] Pavel Beneš : Diplomová práce ČVUT FEL - Řízení helikoptéry 1993

[2] Xilinx inc. : Xact Design Implementation Reference Guide 1991

[3] Humusoft : Real Time Toolbox for MATLAB User's manual 1994

[4] Sobota Branislav: Abeceda OrCADu, Kopp 1994

[5] Houška Jan : Minimum Thesis ČVUT FEL - Real-Time Simulation of Dynamic Systems 1995

[6] Krsek Pavel : Diplomová práce - Řídící systém modelu helikoptéry 1995

[7] Xilinx inc. : The programmable Logic Data Book, USA 1994

[8] Advantech Co. Ltd : PCL-812 Enhanced multi lab card user's manual 1989

[9] Kopp : ABC Programátora v jazyce C, Kopp 1992

[10] Valášek Pavel : Mikroprocesor 8080 a základní obvody, Skriptum ČSVTS Praha 1986

[11] Xilinx inc. : Xact Design Interface Macro Libraries 1991

[12] Svoboda,V.: Vrtulníky. Naše vojsko, Praha 1979.

[13] Wojciechowski,J.: Dialkové ovládanie elektronických modelov. Alfa, Bratislava 1978.

[14] Hrdina,Z.; Vejražka,F.: Digitální radiová komunikace. Skriptum ČVUT, Praha 1994.

[15] Vítovec,J.: Telemetrie a přenos dat. Skriptum ČVUT, Praha 1985.

[16] Kocourek,P.: Přenos informace. Skriptum ČVUT, Praha 1994.

[17] Janeček,J.: Počítačové sítě. Skriptum čVUT, Praha 1990.

[18] Martin,J.: Telecommunications and the Computer. Prentice Hall 1976.

[19] Frejlach,K.: Paket radio (Packet - Radio). České Budějovice 1994.

[20] Stříž,V.: Malý katalog výkonových polem řízených tranzistorů POWER, MOS, DMOS, SIPMOS, VMOS, HEXFET. AMATÉRSKÉ RÁDIO A/1/93.

[21] SIEMENS AG : SIPMOS Tranzistors Application Notes 1983. Siemens AG 1983.

[22] TESLA a.s.: Katalog, Polovodičové součástky 1984/85. Rožnov pod Radhoštěm 1983.

[23] GM Electronic s.r.o.: Katalog, Sočástky pro elektroniku. Praha Květen 1994.

[24] FK technics s.r.o.: Katalog, Aktivní elektronické prvky 94. Praha 1994.

23 Příloha Rejstřík