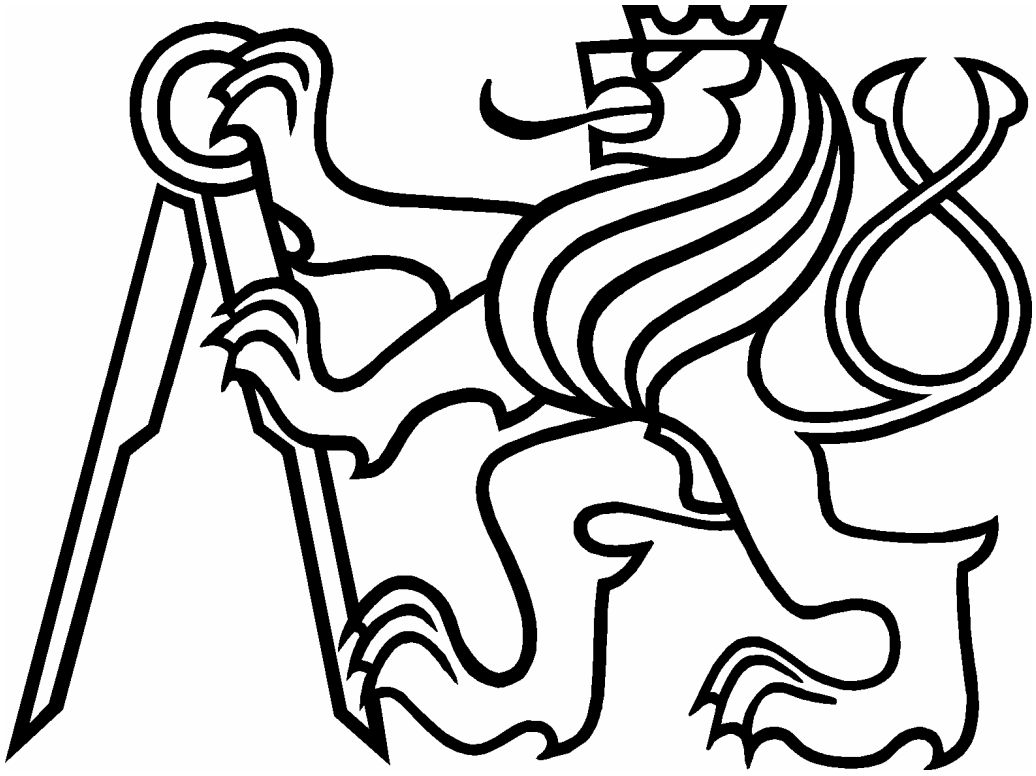
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA ELEKTROTECHNICKÁ

KATEDRA MĚŘENÍ



DIPLOMOVÁ PRÁCE

MNOHOŘÁDKOVÝ VIDEOREGULÁTOR

AUTOR: LUKÁŠ KRÁL

VEDOUCÍ PRÁCE: ING. JAN FISCHER, CSC.

PRAHA 2008

Zadání DP

Pro účely vyhodnocování polohy objektů a její následnou regulaci navrhněte a realizujte videoprocesor, který bude využívat standardní miniaturní plošnou monochromatickou CCD kameru s výstupním signálem dle standardu CCIR.

Videosenzor bude obsahovat blok digitalizace videosignálu, blok číslicového předzpracování hran pomocí logického obvodu s CPLD, blok zpracování obrazu s procesorem z řady ARM7, blok přenosu obrazu na nadřazené PC s rozhraním USB 2.0 a blok vkládání grafické informace do videosignálu.

Vytvořte a odlaďte potřebné programové vybavení pro vestavěný procesor ARM7 pro zpracování obrazu, určování polohy objektů, programy pro řízení zobrazení výsledků i programy pro PC, které budou sloužit ke konfiguraci senzoru. Ve své práci vycházejte z [1].

Videosenzor bude ve spolupráci s polohovacím mechanismem zpětnovazebně regulovat polohu objektu vzhledem k videosenzoru, případně nastavovat polohu více objektů vůči sobě navzájem.

Na tomto místě bych rád poděkoval Ing. Janu Fischerovi, CSc. za čas, který mi věnoval a za cenné rady při vedení diplomové práce. Dále bych rád poděkoval svým rodičům, sestře a přátelům za velkorysou morální i finanční podporu po celou dobu mého studia.

Anotace

Diplomová práce se zabývá návrhem kompaktního videosenzoru pro základní bezdotyková měření s následnou regulací polohy s využitím redukované obrazové informace. Jako zdroj obrazové informace je použita CCD kamera. V práci je popsána konstrukce videosenzoru, jeho programové vybavení a metody zpracování redukované obrazové informace.

Závěrem je posouzena možnost využití tohoto senzoru v průmyslovém prostředí.

Annotation

The graduation thesis is devoted to the design of compact videosensor for basic non-contacting measurements with subsequent regulation of location that is based on reduced video information usage. A CCD camera has been used as a source of visual information. The thesis describes videosensor construction, its software facility and methods of reduced video information elaboration.

The possibility of the sensor application in an industrial environment is analysed in the thesis conclusions.

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, zdrojové kódy, software atd.) uvedené v přiloženém seznamu.

V Praze dne ....

Obsah

1 Úvod a cíl práce

V průmyslových technologických procesech je sledování přítomnosti objektu, jeho polohy nebo rozměru či počtu objektů v daném místě za účelem kontroly nebo následné regulace běžnou potřebou. Využití bezkontaktního optického měření se v mnoha případech přímo nabízí díky svým výhodám; mezi ty obecné patří hlavně skutečnosti, že měřením neovlivňujeme měřený proces ani nezasahujeme do jeho průběhu a získáváme možnost měřit pohybující se objekty. Výhodou tohoto způsobu měření je i univerzálnost použití pro celou škálu měřených materiálů nebo snadná instalace videosenzoru.

S rostoucím výpočetním výkonem současných běžně dostupných mikroprocesorů se implementace algoritmů pro zpracování obrazu stává stále jednodušší. Navíc s rostoucí velikostí hradlových polí je možné proces zpracování obrazu rozdělit na několik paralelních větví, a tím ho značně urychlit.

Pod pojmem videosenzor si můžeme představit systém obsahující zdroj obrazové informace (kameru), mikroprocesor a případně ještě další obvody. Videosignál z kamery je digitalizován a zpracován tak, že výstupem z obvodu je jeden nebo více údajů, které charakterizují snímanou scénu. Videoregulátor je pak regulační obvod, jehož měřená hodnota (výstup čidla regulované veličiny - y(t)) je získána z videosenzoru.

1.1 Úvod do řešené problematiky

1.1.1 Typy senzorů

Jako zdroj obrazové informace lze použít senzor, který převádí optickou informaci na elektrický signál. Současný trh nabízí dva typy snímačů. CCD[1] kameru a CMOS[2] snímač.

[1] Charge-Coupled Device. Senzor složený z fotocitlivých prvků, které po dopadu světla generují elektrický náboj.

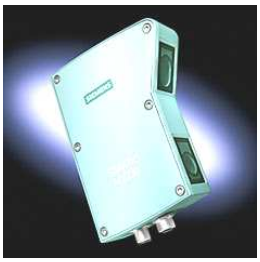
[2] Complementary Metal Oxide Semiconductor. Technologie výroby polovodičů používaná zejména pro konstrukci pamětí. Základním prvkem je buňka skládající se z tranzistoru typu P a N.

Dnes velmi moderní CMOS snímače obvykle nabízejí vyšší rozlišení snímaného obrazu a také větší variabilitu při vyčítání obrazu. Je možné volit taktovací frekvenci vyčítání, vyčítat pouze výřez scény nebo snížit rozlišení průměrováním sousedních buněk. CCD kamery tolik možností nenabízejí, výstupní signál je dán normou CCIR[3]. Manipulace s hotovou kamerou je ale mnohem snazší, není třeba osazovat snímač ani řešit mechanické připojení objektivu. Také výměna kamery v případě nevyhovujících parametrů nebo poškození je velice snadnou záležitostí.

1.1.2 Optické systémy na současném trhu

V současné době se na trhu vyskytuje mnoho typů optických měřicích systémů pro různé typy aplikací. Tyto systémy lze rozdělit do několika základních kategorií:

1.1.2.1 Optické senzory



Nejjednodušší senzory umožňující základní optická měření. Do této kategorie patří například optické závory, které umožňují detekovat přítomnost předmětu a zjišťovat v určitém časovém intervalu počet předmětů pohybujících se po dopravníkovém pásu. Do této kategorie také spadají optické měřiče vzdálenosti či rozměru, využívající ke své činnosti laser (např. triangulační laserový měřič vzdálenosti) nebo řádkový CCD snímač.

Možnosti regulace jsou však – vzhledem k jednoduchosti senzorů a většinou pouze dvoustavovému výstupu – poměrně malé.

1.1.2.2 Rozpoznávání 1D kódů, 2D kódů a textů

Tyto typy optických systémů umožňují identifikaci předmětu pomocí čárového kódu, plošného kódu nebo výrobního čísla. Slouží spíše pro aplikace spojené s evidencí a logistiku než pro měřicí účely. Ve spojení s automatizovaným

[3] konkrétně CCIR 601 definující parametry výstupního signálu černobílé kamery

měřením např. rozměru výrobků na dopravníku jsou vhodným nástrojem pro jednoznačnou identifikaci zkoumaného objektu. Pro potřeby regulace ale nenacházím u tohoto typu senzorů využití, které by se vázalo k cíli diplomové práce.

1.1.2.3 Inteligentní kamery pro univerzální aplikace

Inteligentní kamery obsahují procesor pracující na frekvenci řádově stovek MHz až jednotek GHz a disponují pamětí o velikosti stovek MB. Na procesoru je obvykle spuštěn operační systém, upravený pro účely práce v reálném čase. Kamery jsou obvykle v rozlišeních od VGA po 1600×1200 pixelů a jsou jak černobílé, tak i barevné. Jsou dodávány se softwarem umožňujícím konfigurovat nebo programovat jejich chování. Celý systém je pak většinou vybaven několika logickými výstupy a dále také některou z vysokorychlostních sběrnic (např. ethernet IEEE1394 nebo USB 2.0). Díky tomu jsou tyto typy „senzorů" naprosto univerzálním nástrojem pro měření v oblasti videometrie.

Většina smart kamer je optimalizována pro kontrolu správnosti tvaru po výrobě nebo montáži, a mají tudíž také pouze dvoustavové výstupy. Vzhledem k programovatelnosti celého systému by ale mělo být možné použití i pro účely regulace. Pro jednodušší aplikace je však takový senzor zbytečně výkonný, a tím i drahý.

1.1.2.4 Videosystémy s PC a jednou nebo několika kamerami

Potřeba komplexnějšího pohledu na průběh zpracování a komunikaci mezi jednotlivými částmi výrobního procesu vede k vytvoření systému, jehož základem je PC, upravené pro průmyslové podmínky, několik kamer a používající operační systém přizpůsobený pro zpracování v reálném čase (např. Real-time Linux). Signál je z kamer získáván buď frame-grabberem a nebo přes sběrnici ethernet. Tyto systémy používají nejnáročnější aplikace zpracování obrazu.

1.2 Cíl práce

Cílem této diplomové práce je návrh a konstrukce jednoduchého a levného měřicího systému s CCD kamerou, který bude schopen zpracovat víceřádkový videosignál, a umožňovat tak regulaci polohy na základě uživatelského nastavení.

Systém by měl splňovat následující kritéria:

- autonomnost – i bez nadřazeného systému bude schopen plně vykonávat činnost, pro kterou byl vytvořen,
- jednoduchost – co nejjednodušší realizace,
- nízké náklady na jeho vytvoření,
- univerzálnost – umožňující volit co nejširší škálu parametrů ovlivňujících způsob zpracování,
- snadnou konfiguraci – možnost jednoduché konfigurace pomocí aplikace pro PC,
- možnost zobrazit snímanou scénu a důležité údaje na PC nebo na běžném monitoru bez účasti PC.

Ve srovnání např. se smart kamerami bude zde popsané řešení obsahovat méně výkonný mikrokontrolér s pamětí menší o mnoho řádů. Takový systém je pro jednodušší aplikace plně postačující a výrobní náklady jsou menší.

2 Rozbor řešeného úkolu

Před vlastní realizací videoregulátoru je nutné si uvědomit, k jakým úkolům bude zařízení sloužit. Obecný účel je dán zadáním a cílem diplomové práce. Konkrétní účel může být ilustrován následujícím aplikačním příkladem:

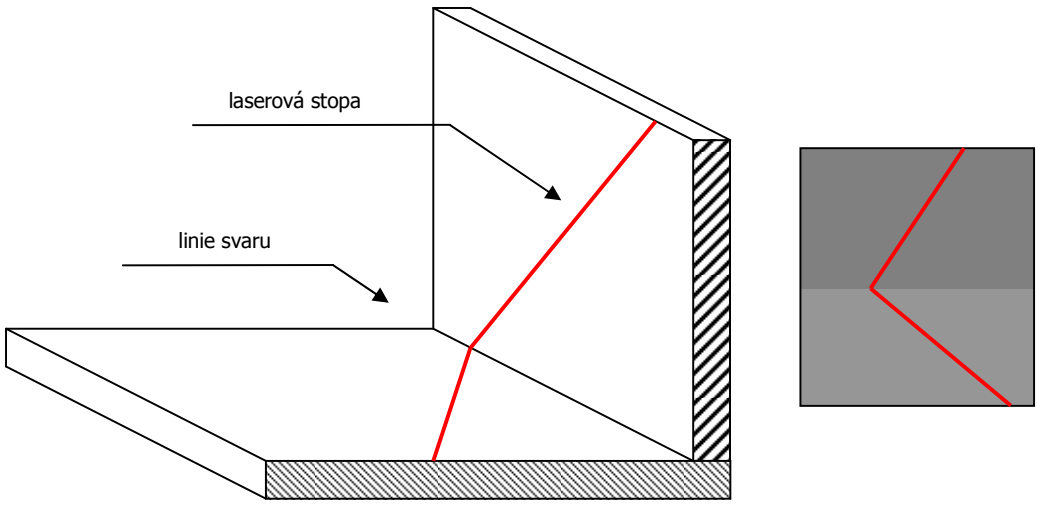
2.1 Aplikační příklad: Koutový svářecí automat

Mějme dva železné plechy, které potřebujeme svařit tak, aby svíraly úhel 90°. Pro vytvoření takového svaru je třeba navést hrot svářecí elektrody přesně do místa, kde se oba plechy stýkají.

K navedení svářečky na správné místo potřebujeme znát polohu svářečky a polohu místa styku obou plechů. Oba tyto údaje lze za jistých opatření získat pomocí kamery.

Svářečku si označíme kontrastní značkou (např. černým proužkem známé šířky, vytištěným na bílém podkladu). Tuto značku jsme schopni v obrazu rozpoznat.

Po osvícení koutu rovinným laserem z vhodného úhlu je z pohledu kamery vidět lomená světelná stopa. Místo zlomu této stopy je zároveň místem, kde se oba plechy stýkají (obr. 1).



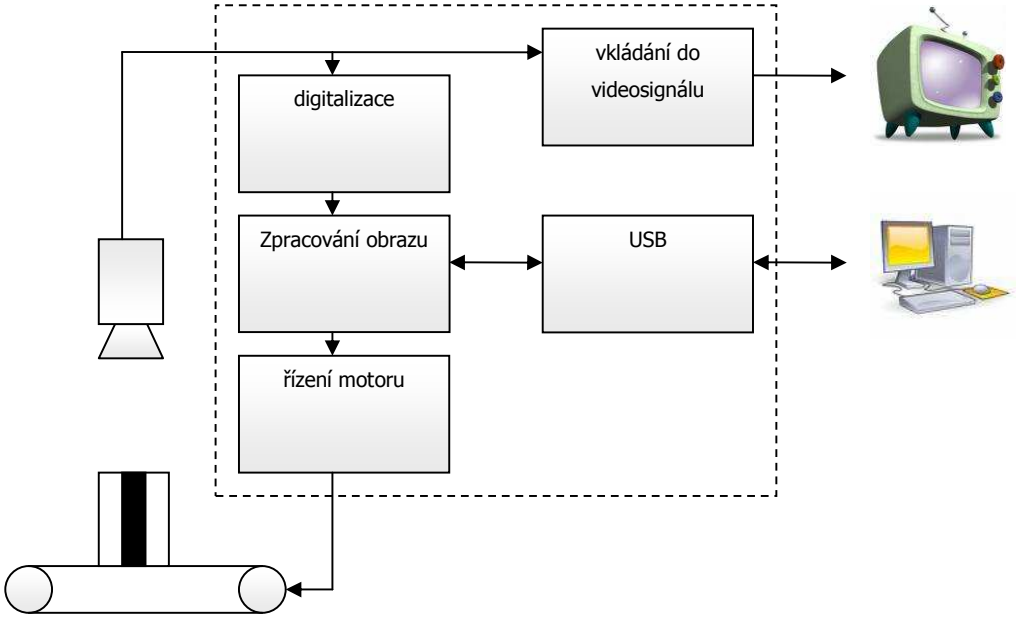
obr. 1: Kout osvícený laserovou rovinou a) z pohledu pozorovatele, b) z pohledu kamery

2.2 Požadavky na videoregulátor

Z uvedeného aplikačního příkladu jsme schopni určit, které funkce budeme pro realizaci videoregulátoru potřebovat a které by bylo vhodné implementovat do procesoru. Jsou to zejména:

- hledání značky v řádku kamery podle zadaných kritérií (poloha, šířka, počet hran),
- hledání extrémních hodnot (minimum, maximum) ve výsledcích řádkového zpracování,
- možnost odděleně definovat měřenou a žádanou hodnotu,
- regulace na základě měřené a žádané hodnoty,
- provedení žádoucí operace (např. otočení motorem) v závislosti na vypočteném akčním zásahu.

Na základě výše uvedených požadavků jsme schopni sestavit ideové blokové schéma celého zařízení (obr. 2). Popis jednotlivých bloků následuje.



obr. 2: Ideové blokové schéma

2.3 Blok digitalizace obrazu

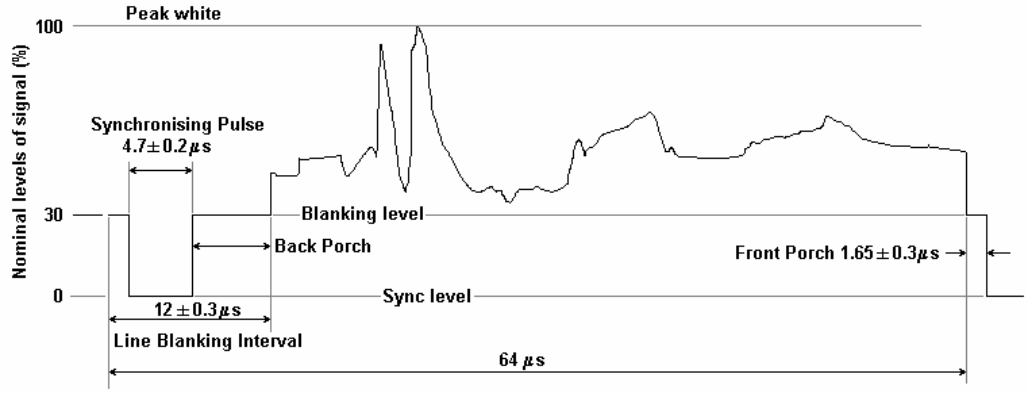
Pro vytvoření bloku, který bude správně dekódovat a digitalizovat videosignál, je třeba nejprve nahlédnout do normy předepisující přenos obrazu

z použité kamery. Jak již bylo uvedeno výše, bude jako zdroj obrazové informace použita CCD kamera s výstupním signálem podle normy CCIR 601 (norma definující parametry výstupního signálu černobílé kamery).

2.3.1 Výstupní signál monochromatické kamery

Celý snímek je přenesen 25× za sekundu. Přenos probíhá po půlsnímcích, před každým půlsnímkem je zařazena vertikální synchronizační sekvence, která oznamuje začátek přenosu půlsnímku a uvádí, o který půlsnímek se jedná. Každý půlsnímek obsahuje 312,5 řádku. V celém snímku je tedy 625 řádků, z nich aktivních je však pouze 576; ostatní řádky jsou nevyužité. Řádkový signál začíná horizontálním synchronizačním pulzem, který je následován signálem BURST. Ten v černobílém videosignálu oznamuje napěťovou úroveň odpovídající černé barvě. Po něm začíná přenos 768 obrazových bodů daného řádku.

Typická amplituda videosignálu je $1V$ na 75Ω. Spodních $0{,}3V$ je využito pro synchronizační pulzy, horních $0{,}7V$ pak pro samotnou jasovou úroveň přenášeného obrazu. Menší napětí odpovídá tmavší barvě, větší napětí barvě světlejší.



obr. 3: Jeden řádek videosignálu

Pro získání všech 768 obrazových bodů je nutné signál vzorkovat frekvencí

$$f = \frac{N}{t} = \frac{768}{52\mu s} = 14{,}769MHz$$

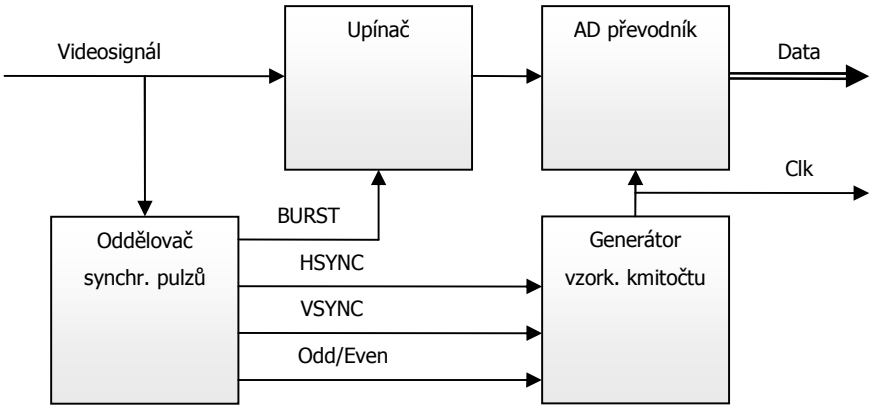
Při požadavku na menší rozlišení můžeme signál vzorkovat nižším vzorkovacím kmitočtem. Abychom dodrželi vzorkovací teorém[4], je nutné předřadit vzorkovacímu obvodu antialiasingový filtr.

2.3.2 Digitalizace videosignálu

Systém vhodný pro digitalizaci obrazu je složen minimálně z oddělovače synchronizačních pulzů, upínače, AD převodníku a generátoru vzorkovacího kmitočtu (podle obr. 4).

Oddělovač synchronizačních pulzů (např. obvod `LM1881`) zajišťuje separaci a dekódování synchronizačních pulzů z videosignálu, výstupy tohoto obvodu jsou ve standardní TTL logice. Dík signálu BURST jsme schopni upnout signál na referenční úroveň – stejnosměrný posun signálu se uloží na kapacitu a během digitalizace se tento posun od signálu odečítá.

Generátor vzorkovacího kmitočtu (tvořený časovačem mikrokontroléru) generuje hodinový signál pro AD převodník synchronně s hodinami mikroprocesoru, což zajišťuje čtení dat procesorem ve správném okamžiku.



obr. 4: Blokové schéma nejjednoduššího systému pro digitalizaci videosignálu

[4] „Přesná rekonstrukce spojitého, frekvenčně omezeného, signálu z jeho vzorků je možná tehdy, pokud byl vzorkován frekvencí alespoň dvakrát vyšší než je maximální frekvence rekonstruovaného signálu."

2.4 Blok zpracování obrazu

Srdcem celého videoregulátoru je blok zpracování obrazu, který se stará o identifikaci jeho důležitých částí. Proto je důležitý výběr typů součástek, které budou v tomto bloku použity.

V úvahu připadá použití klasického mikroprocesoru, signálového procesoru nebo hradlového pole. Následuje tabulka shrnující výhody, nevýhody a cenu jednotlivých řešení:

Součástka	-	Popis
mikroprocesor	výhody	nízká cena konvenční metody programování
	nevýhody	nízký výkon
	cena	$ 5
signálový procesor	výhody	rychlost zpracování – uzpůsoben pro zpracování navzorkovaných signálů
	nevýhody	cena
	cena	$ 20
CPLD[5]	výhody	cena paralelní zpracování dat
	nevýhody	malé možnosti zpracování – malý počet makrobuňek
	cena	$ 5
FPGA[6]	výhody	paralelní zpracování dat
	nevýhody	nutnost připojit externí paměť cena
	cena	$20

tab. 1: Srovnání typů součástek pro zpracování signálu

Vzhledem k cenám jednotlivých obvodů byl jako základ bloku pro zpracování obrazu zvolen klasický mikroprocesor, doplněný o hradlové pole

[5] CPLD - complex programmable logic device
[6] FPGA - field-programmable gate array

typu CPLD. Tato kombinace je v porovnání s ostatními řešeními levnější a zároveň pro jednoduché metody zpracování obrazu nabízí dostatečný výkon.

2.4.1 Předzpracování v CPLD – hranový detektor

Hrana je místo v obrazovém řádku, kde je velký rozdíl v jasu mezi sousedními pixely[7]. Pokud tedy máme v obrazu např. světlý předmět na tmavém pozadí, poloha hran v obraze zároveň identifikuje polohu předmětu. Náběžná hrana je místo, kde přechází tmavá barva do světlé. Naopak místo, kde světlá barva přechází do tmavé, budeme nazývat spádová hrana.

Obvod CPLD dokáže provést detekci hran ve videosignálu. Tím ulehčí mikroprocesoru práci s načítáním digitalizovaných dat, protože mu předává pouze informaci o tom, kde a jaké hrany se v řádku vyskytly. Pro účely diplomové práce lze použít dva základní způsoby detekce hrany v CPLD:

- komparační metodu a
- přírůstkovou metodu

Každá z metod má své výhody i nevýhody, metody fungují na principech, popsaných v následujících dvou kapitolách.

2.4.1.1 Komparační metoda hranové detekce

Komparační metoda hranové detekce je nejjednodušší metodou, při které se obraz nejprve oprahuje. Prahování je proces, při němž se pixely v obraze rozdělí do dvou skupin – na pixely tmavé a světlé. Toto rozdělení je provedeno na základě komparační úrovně, což je hodnota, která udává rozhodovací jas pixelu. Pixely, které mají jas menší než komparační úroveň, budou označené jako tmavé, ostatní jako světlé. Při výskytu tmavého a světlého pixelu v obrazu vedle sebe je toto místo označeno jako hrana.

Výhodou této metody je její jednoduchost. Nevýhodou je potřeba předem znát hodnotu prahovací úrovně. Pokud je tato hodnota nastavena napevno, je možné tuto metodu použít pouze pro snímanou scénu s předem známými světelnými podmínkami. Také je možné nasnímanou scénu nejprve analyzovat a

[7] Zkrácení anglických slov „Picture Element“ (obrazový prvek). Dále zkracováno na „px“.

teprve ze získaných informací určit hodnotu prahovací úrovně, což ale vyžaduje další paměť a výpočetní výkon. Proto automatické určení prahovací úrovně v obvodu CPLD pravděpodobně nebude možné použít.

2.4.1.2 Přírůstková metoda hranové detekce

Přírůstková metoda hranové detekce vychází z první derivace (v případě digitálního zpracování z první diference) signálu. Po výpočtu první derivace tuto hodnotu porovnáváme s referenčními hodnotami (tzv. threshold). Při překročení referenční hodnoty v kladném směru je detekována náběžná hrana, při překročení v záporném směru sestupná hrana. Úrovní referenční hodnoty lze zvolit, jak strmý přechod jasu ve snímaném obrazu už je považován za hranu.

Výhodou této metody je nezávislost na jasových podmínkách snímané scény. Nevýhodou je nepříliš významně větší nárok na paměť.

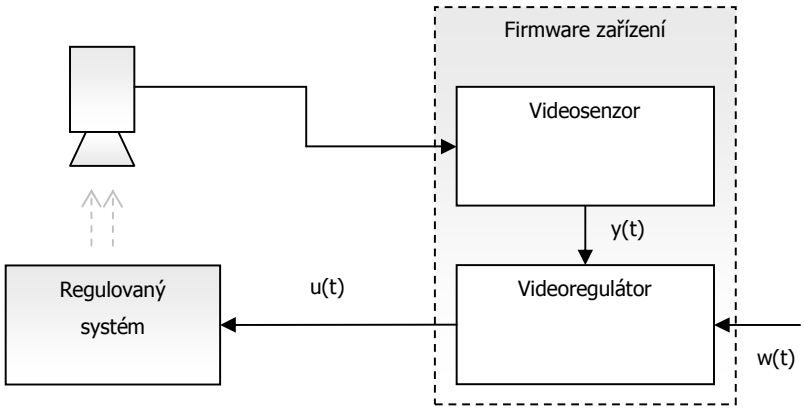
2.4.2 Zpracování obrazu v mikroprocesoru

Úkolem bloku pro zpracování obrazu v mikroprocesoru je načíst polohu hran z obvodu `CPLD`, načtené údaje vyhodnotit a nalézt v nich údaje popsané uživatelem (např. polohu tmavé značky vyskytující se v zadaném řádku, která má zadanou hodnotu). Tyto údaje blok zpracování obrazu předá dále do bloku zajišťující regulaci.

2.4.3 Regulátor v mikroprocesoru

Hlavním úkolem videoregulátoru je zajistit regulaci polohy daného předmětu (v aplikačním příkladě polohy svářečky). Tuto polohu je možné měnit motorem. Z údajů získaných z předchozích kroků je tedy třeba zjistit, na jakou stranu a o kolik stupňů je třeba otočit motorem, abychom získali požadovanou polohu motoru. O tento úkol se stará právě blok regulace. Připojením tohoto bloku se z videosenzoru stává videoregulátor.

Jelikož použitý mikroprocesor disponuje dostatečným výpočetním výkonem pro realizaci zpracování obrazu i následnou regulaci, je blok regulace pouze programovou knihovnou připojenou k řídicímu programu videosenzoru (dle obr. 5).



obr. 5: Implementace videosenzoru a videoregulátoru do jednoho zařízení

Algoritmů číslicové regulace v dnešní době existuje velice mnoho. Dle řízené soustavy lze volit různé varianty řídicích algoritmů, nebo vytvářet jejich kombinace. Nejčastěji užívaným regulátorem je však stále PSD regulátor, což je diskrétní obdoba PID regulátoru.

Vycházejme z rovnice spojitého PID regulátoru:

$$u(t) = r_0\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\,\frac{de(t)}{dt}\right),$$

kde regulační odchylka $e(t) = w(t) - y(t)$.

Pro diskretizaci integrace lze použít např. lichoběžníkovou metodu integrace, kde obsah pod funkcí mezi dvěma navzorkovanými body je nahrazen obsahem lichoběžníku.

Integrál lze poté přepsat do tvaru:

$$\int_0^t e(\tau)d\tau \approx T\left(\sum_{j=0}^{k-1}\frac{e(j)+e(j+1)}{2}\right) = T\left(\frac{e(0)+e(k)}{2} + \sum_{j=1}^{k-1} e(j)\right),$$

derivaci je třeba nahradit diferencí:

$$\frac{de(t)}{dt} \approx \frac{e(k)-e(k-1)}{T}.$$

Rovnici spojitého PID regulátoru lze pak přepsat do tvaru pro diskrétní PSD regulátor (v časovém okamžiku k):

$$u(k) = r_0\left(e(k) + \frac{T}{T_i}\left(\frac{e(0)+e(k)}{2} + \sum_{j=1}^{k-1} e(j)\right) + \frac{T}{T_d}\left(e(k)-e(k-1)\right)\right)$$

Tento typ regulátoru se nazývá polohový. Jeho nevýhodou je nutnost v každém kroku vypočíst sumu všech předchozích hodnot regulační odchylky. To by bylo z hlediska implementace v mikroprocesoru velice nepraktické. Odečtením rovnice diskrétního regulátoru v čase k-1 od téže rovnice v čase k získáme tutéž rovnici v přírůstkovém tvaru:

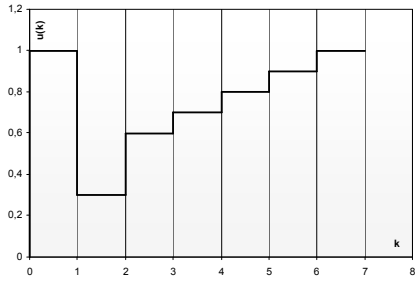
$$u(k) = r_0\left(\left(1+\frac{T}{2T_i}+\frac{T_d}{T}\right)e(k)-\left(1-\frac{T}{2T_i}+2\frac{T_d}{T}\right)e(k-1)+\frac{T_d}{T}e(k-2)\right)+u(k-1)$$

tuto rovnici lze přepsat do zjednodušeného tvaru:

$$u(k) = q_0 e(k) + q_1 e(k-1) + q_2 e(k-2) + u(k-1)\,.$$

Konstanty q_0 – q_2 nazveme koeficienty `PSD` regulátoru. Má-li mít regulátor ekvivalentní odezvu jako klasický `PID` regulátor, je třeba, aby platily následující podmínky:

- $q_0>0$ zajišťuje kladný akční zásah v prvním kroku po skokové změně regulační odchylky,
- $q_1<q_0$ říká, že akční zásah v druhém kroku po skokové změně e bude menší než v prvním kroku,
- $q_0+q_1+q_2>0$ zajistí konstantní kladný přírůstek akční veličiny v druhém a dalších krocích,
- $q_0>q_2$ zajišťuje kladnou hodnotu nárůstu akční veličiny v čase 0.



obr. 6: Typická odezva PSD regulátoru na jednotkový skok

2.5 Blok řízení motoru

Blok řízení motoru se stará o obsluhu připojeného motoru. Nejprve je třeba určit, jaké typy motorů budeme chtít řídit, v návaznosti na to lze formulovat základní požadavky kladené na tento blok:

- možnost připojení krokového motoru,
- možnost připojení stejnosměrného motoru,
- měření proudu, regulace proudu cívkou motoru,
- možnost samostatného použití v jiných aplikacích,
- jednoduchá komunikace s nadřazeným systémem a
- snadná konfigurace.

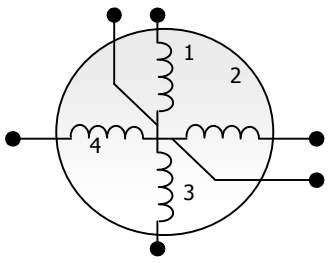
S ohledem na tyto požadavky je zřejmé, že nejvýhodnější bude realizovat blok řízení motoru jako samostatné zařízení, které bude komunikovat přes nějaké standardní rozhraní.

Zařízení tedy bude obsahovat budič motoru schopný budit jak stejnosměrný, tak krokový motor a jednoduchý mikroprocesor, který bude zajišťovat komunikaci s nadřazeným systémem a řídit budič.

2.5.1 Krokový motor

Krokový motor je točivý stroj skládající se z rotoru a statoru. Rotor je obvykle tvořen hřídelí usazenou v ložiscích a prstencem permanentních magnetů, stator je tvořen sadou (obvykle čtyř) cívek, které při průchodu proudem natočí rotor do dané polohy. Cívky statoru jsou obvykle zapojeny dle obr. 7.

Proud procházející cívkou statoru vytvoří magnetické pole, které přitáhne opačný pól magnetu rotoru. Vhodným zapojováním cívek dosáhneme vytvoření rotujícího magnetického pole, které otáčí rotorem.



obr. 7: Zapojení cívek krokového motoru

Podle požadovaného kroutícího momentu, přesnosti nastavení polohy a přípustného odběru volíme některou z dále popsaných variant řízení. Vzhledem k přechodovým magnetickým jevům je omezena rychlost otáčení motoru, a to na několik stovek kroků za sekundu (závisí na typu motoru a zatížení). Při

překročení této maximální rychlosti (nebo při příliš velké zátěži) motory začínají ztrácet kroky.

2.5.1.1 Unipolární řízení

Při unipolárním řízení proud prochází v jednom okamžiku právě jednou cívkou. Motor s tímto buzením má nejmenší odběr, ale také poskytuje nejmenší kroutící moment. Výhodou tohoto řešení je jednoduché zapojení řídící elektroniky - v podstatě stačí jeden tranzistor na každou cívku.

2.5.1.2 Bipolární řízení

Při bipolárním řízení prochází proud vždy dvěma protilehlými cívkami. Ty jsou řazeny antisériově[8]. Motor v tomto režimu poskytuje větší kroutící moment, ovšem za cenu vyšší spotřeby. Pro řízení jsou třeba dva H-můstky, pro každou větev jeden – zapojení je tedy o něco složitější.

2.5.1.3 Jednofázové

Krokový motor řízený jednofázově má vždy sepnutou jednu větev – u unipolárního řízení to znamená, že je sepnut proud jednou cívkou, u bipolárního řízení je sepnut proud oběma cívkami.

2.5.1.4 Dvoufázové

Naproti tomu při dvoufázovém řízení jsou vždy sepnuty dvě větve. V případě bipolárního řízení to znamená, že jsou sepnuty všechny čtyři cívky motoru. Kotva motoru je v tomto případě natočena do polohy mezi sepnuté cívky. Motor odebírá nejvíce proudu, ale zároveň nabízí největší možný moment.

2.5.1.5 Plný krok

Otočení motorem je provedeno stejným počtem kroků, jako má stator motoru zubů. Proud tedy protéká cívkami v pořadí:

1 – 2 – 3 – 4

[8] Cívky zapojené antisériově jsou v podstatě do série zapojené cívky, které při průchodu proudem mají navzájem opačně orientované magnetické pole.

2.5.1.6 Poloviční krok

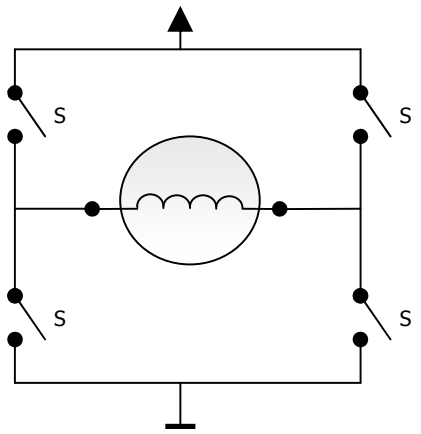
Otočení motorem provedeme dvojnásobným počtem polovičních kroků. Polovičního kroku je dosaženo tak, že střídavě spínáme proud do jedné cívky a do dvou sousedních cívek. Proud tedy protéká cívkami v pořadí:

1 – 1+2 – 2 – 2+3 – 3 – 3+4 – 4 – 4+1

Výhodou tohoto typu řízení je jemnější polohování rotoru, nevýhodou je vyšší spotřeba proudu při sepnutí dvou cívek zároveň.

2.5.2 Stejnosměrný motor

Stejnosměrný motor je točivý stroj, který má jako stator permanentní magnet. V magnetickém poli statoru se nachází rotor s cívkou, kterou protéká proud. Ten indukuje magnetické pole, které vyvolá pootočení rotoru. Proud je do cívky přiveden pomocí komutátoru, který zároveň zajišťuje přepínání polarity proudu. Při přivedení stejnosměrného proudu do motoru se motor začne konstantní rychlostí otáčet na jednu stranu. Rychlost otáčení motoru je možné ovládat pomocí PWM[9]. Rychlost i směr otáčení stejnosměrného motoru je tedy možné řídit pomocí H-můstku (ideové schéma je uvedeno v obr. 8).

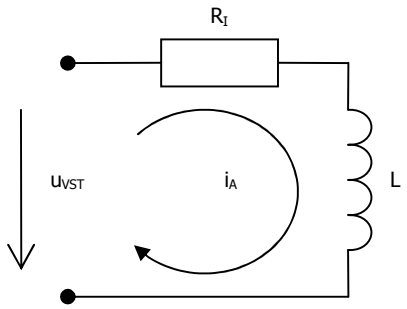


obr. 8: Ideové schéma pro řízení stejnosměrného motoru

[9] PWM – Pulse Width Modulation (pulsně šířková modulace) – způsob modulace, kdy řídíme střední hodnotu napětí tak, že po určitý čas připojíme na výstup modulátoru plné napětí a po zbytek periody toto napětí odpojíme

2.5.3 Proud vinutím motoru

Při řízení rychlosti nebo momentu motoru (ať už stejnosměrného, či krokového) je třeba znát souvislost mezi napětím a proudem v obvodu. K obvodu je připojeno známé napětí, moment motoru je však vyvolán proudem procházejícím cívkou motoru. Bližší představu o závislosti proudu na napětí získáme výpočtem obvodu (obr. 9) např. metodou smyčkových proudů.



obr. 9: Náhradní schéma motoru

Obvod jsme schopni popsat jednou diferenciální rovnicí:

$$R.i_A + L\frac{di}{dt} + u_{VST} = 0,$$

kterou lze upravit do standardního tvaru

$$\dot{i} + \frac{R}{L}i = -\frac{u}{L};\ a(t) = \frac{R}{L}, b(t) = -\frac{u}{L}.$$

Pak

$$\dot{i} = -\frac{R}{L}i,$$

$$\frac{1}{i}di = -\frac{R}{L}dt.$$

Integrací získáme

$$\int\frac{1}{i}di = \int -\frac{R}{L}dt,$$

řešením integrálu pak

$$\ln(i) = -\frac{R}{L}t + \tilde{K},$$

$i = Ke^{-\frac{R}{L}t}$.

Variace konstanty:

$i_P = K(t)e^{-\frac{R}{L}t}$,

$\dot{i}_P = \dot{K}(t)e^{-\frac{R}{L}t} - \frac{R}{L}K(t)e^{-\frac{R}{L}t}$.

Pak

$\dot{K}(t)e^{-\frac{R}{L}t} - \frac{R}{L}K(t)e^{-\frac{R}{L}t} + \frac{R}{L}K(t)e^{-\frac{R}{L}t} = -\frac{u}{L}$,

$\dot{K}(t) = -\frac{u}{L}e^{\frac{R}{L}t}$.

Integrací získáme

$K(t) = \int -\frac{u}{L}e^{\frac{R}{L}t}dt = -\frac{e^{\frac{R}{L}t}}{\frac{R}{u}}$.

Řešení celého obvodu tedy je:

$i_A = -\frac{e^{-\frac{R}{L}t}\left(e^{\frac{R}{L}t} - \frac{R}{u}K\right)}{\frac{R}{u}}$.

Z počáteční podmínky $i_A(0) = 0$ lze určit K:

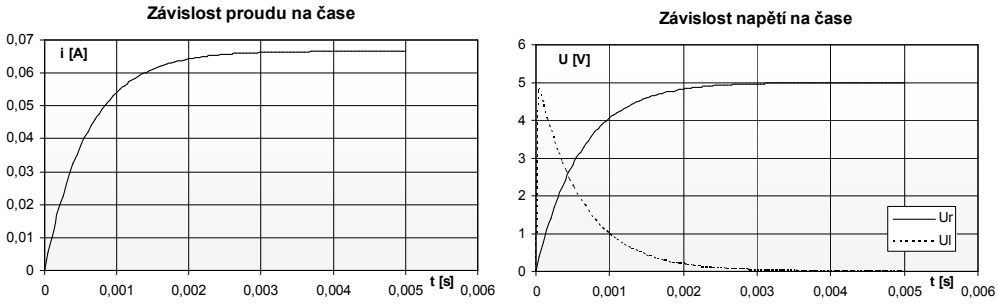
$i_A(0) = -\frac{e^{-\frac{R}{L}t}\left(e^{\frac{R}{L}t} - \frac{R}{u}K\right)}{\frac{R}{u}} = -\frac{1\left(1 - \frac{R}{u}K\right)}{\frac{R}{u}} = 0$,

$K = \frac{u}{R}$;

a tedy

$i_A = -\frac{e^{-\frac{R}{L}t}\left(e^{\frac{R}{L}t} - 1\right)}{\frac{R}{u}} = \frac{u}{R}\left(1 - e^{-\frac{R}{L}t}\right)$.

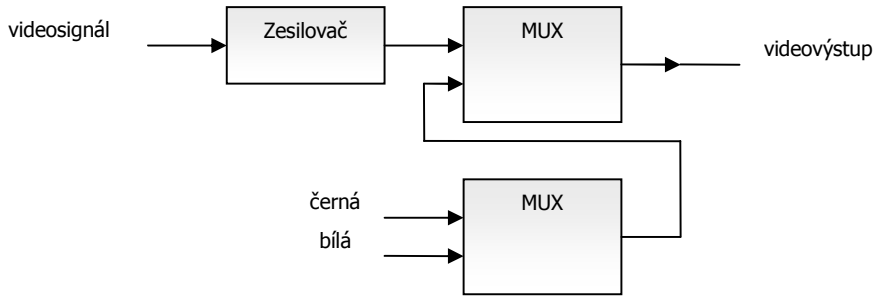
Dosazením reálných hodnot získáme časový průběh proudu obvodem a následně i napětí na jednotlivých částech při sepnutí napájení. Jako příklad lze uvést časový průběh proudu a napětí na jedné cívce krokového motoru s indukčností 45mH a vnitřním odporem 75Ω (obr. 10). Hodnoty indukčnosti a odporu odpovídají naměřeným hodnotám malého krokového motoru získaného z FLOPPY DISK mechaniky.



obr. 10: Časové průběhy napětí a proudu v RL obvodu

2.6 Blok vkládání údajů do obrazu

Pro vkládání textových informací do výstupního videosignálu bude použit klasický multiplexor (obvod 74HCT4053, dle obr. 11), který v základním režimu propouští videosignál získaný z kamery a zesílený zesilovačem. Při požadavku na vložení černého nebo bílého pixelu je přepnut do stavu, kdy se na výstup multiplexoru přenáší jedna z předem definovaných napěťových úrovní. Tyto napěťové úrovně odpovídají ve videosignálu černé a bílé. Černá barva přímo odpovídá napětí, které se ve videosignálu vyskytuje bezprostředně po synchronizačním pulzu a obvod LM1881 signalizuje její aktuální výskyt ve videosignálu nastavením výstupu BURST. Bílá barva je od černé odvozena. Multiplexor bude řízen mikroprocesorem.



obr. 11: Blokové schéma obvodu pro vkládání údajů do videosignálu

2.7 Blok komunikace přes USB

Pro kontrolu a intuitivní konfiguraci videoregulátoru je vhodné, aby uživatel viděl celou snímanou scénu. Pro přenos nekomprimovaného obrazu je však zapotřebí poměrně veliká datová propustnost použité sběrnice. Konkrétně pro přenos obrazu z CCD kamery v plném rozlišení a při 256 odstínech šedi je zapotřebí datová propustnost B:

$$B = 576 \times 768 = 11059200\,Bs^{-1} \doteq 11\,MBs^{-1}.$$

Takový datový tok klade důraz jak na použité součástky pro přenos z videoregulátoru do počítače, tak na hardware počítače použitého pro zobrazení.

2.8 Součástky zvolené pro realizaci hardwaru

Z výše uvedeného textu tedy vyplývá, že celé zařízení se bude skládat ze dvou nezávislých částí, které lze vzájemně propojit: z části videoregulátoru a z části budiče motoru. Jelikož navrhovaný hardware vychází z [1], je součástková základna velmi podobná.

2.8.1 Modul videoregulátoru

Základem bloku videoregulátoru bude mikroprocesor LPC2148 s jádrem ARM7. Procesor disponuje výkonem až 60*MIPS*[10], může využívat až 40*kB* statické paměti RAM a disponuje programovou pamětí 512*kB* FLASH.

[10] MIPS – Million Instructions Per Second – výpočetní výkon udaný v miliónech provedených instrukcí za sekundu

Pro předzpracování signálu bude modul obsahovat CPLD obvod Xilinx XC9572, který se skládá ze 1600 uživatelsky přístupných hradel seskupených do 72 makrobuněk.

Pro digitalizaci obrazu a s ní spojené oddělení synchronizačních pulzů z videosignálu budou použity obvody LM1881 s výstupními signály HSYNC, VSYNC, BURST, ODD/EVEN a AD9280, což je osmibitový paralelně kaskádní AD převodník s dobou převodu 31,25*ns*.

Komunikace videoregulátoru s nadřazeným systémem bude realizována přenosem dat po USB za pomoci obvodu CY7C68013A-56. Tento obvod se skládá z jádra mikrokontroléru 8051, který je doplněn o řadič USB 2.0 HIGH SPEED.

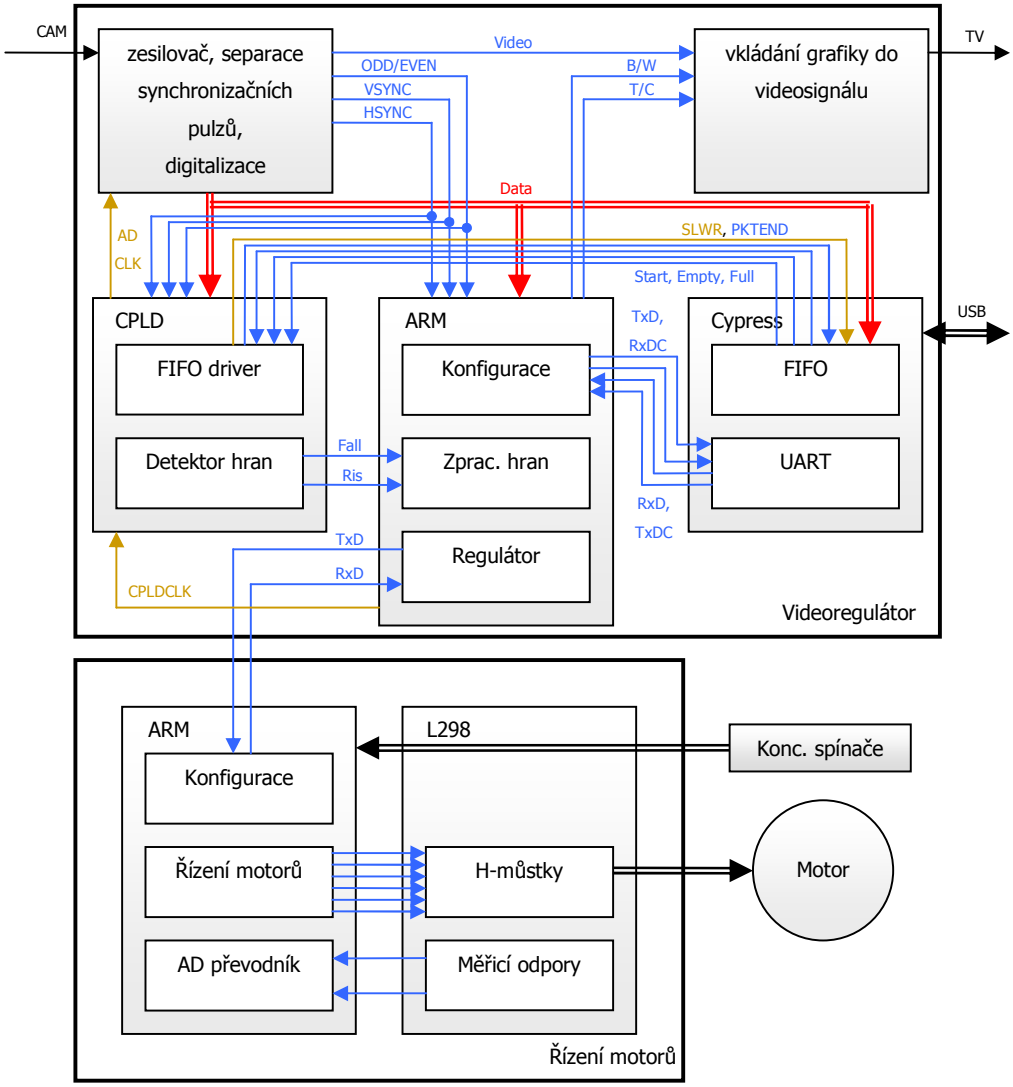
2.8.2 Modul pro ovládání motorů

Základem modulu pro ovládání motorů bude mikroprocesor LPC2103. Tento obvod má velice podobné parametry jako obvod LPC2148 použitý v modulu videoregulátoru. 8*kB* paměti RAM a 32*kB* programové paměti FLASH bude pro účely ovládání motorů plně postačovat. Za zmínku stojí, že obvod je vybaven desetibitovým AD převodníkem integrovaným na čipu mikroprocesoru, který bude možné využít pro měření proudu vinutím motorů.

Pro buzení vinutí motorů bude použit budič L298, který se skládá ze dvou H-můstků. Obvod dokáže budit napětí až do výše 46*V* a je schopen korektně pracovat až do celkového společného proudu obou můstků 4*A*. Obvod obsahuje také ochranu proti přehřátí. Výhodou je jak kompatibilita vstupních signálů s logickými úrovněmi CMOS, tak možnost připojení odporů pro měření proudu.

Veškeré součástky použité při návrhu obou modulů byly po jednotlivých blocích nejprve sestaveny a vyzkoušeny na nepájivém kontaktním poli. Tento postup zamezil vznik některých chyb při návrhu desky plošných spojů.

3 Popis realizace zařízení



obr. 12: Blokové schéma zařízení

Celé zařízení se skládá ze dvou modulů, které jsou schopny (v rámci svého účelu) fungovat samostatně, a je tedy možné je využít i k jiným účelům.

Úkolem modulu videoregulátoru je zpracovat obrazovou informaci načtenou z CCD kamery a vypočíst akční zásah. Mimo to bude modul schopen komunikovat s PC po sběrnici USB a bude schopen vkládat jednoduchou obrazovou informaci do výstupního videosignálu. Druhý modul na základě akčního zásahu vypočteného modulem videoregulátoru provede

příslušné řízení motoru a bude schopen hlídat stavy motoru jako je koncová poloha či příliš veliký proud protékající vinutím motoru.

3.1 Realizace videoregulátoru

Z navrženého blokového schématu (obr. 12, část „Videoregulátor“) je zřejmé, že se modul skládá z pěti hlavních částí:

- bloku digitalizace videosignálu,
- bloku vkládání grafiky do videosignálu,
- bloku předzpracování obrazu - hranového detektoru (CPLD),
- bloku vyhodnocení a regulace (mikroprocesor) a
- bloku komunikace s nadřazeným systémem (USB řadiče, CPLD)

Tyto bloky je třeba taktovat hodinovým signálem, který z důvodu zvýšení přesnosti měření musí být synchronizován s videosignálem.

3.1.1 Synchronizace činnosti modulu s videosignálem

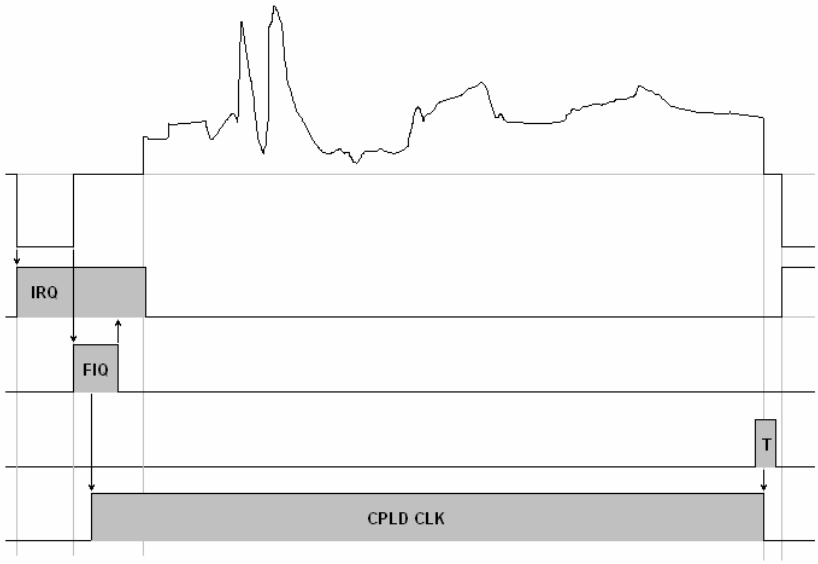
Jak mikroprocesor, tak USB řadič jsou taktovány vlastním zdrojem hodinového signálu. K řadiči USB je připojen krystal o frekvenci 24*MHz*. Frekvence krystalu je násobena interní PLL[11] jednotkou na standardní kmitočet sběrnice USB, který je 48*MHz*. Kmitočet jádra mikroprocesoru ARM 60*MHz* je interní PLL jednotkou odvozen z 12*MHz* krystalu. Stejným kmitočtem jsou taktovány i periferie mikroprocesoru.

Zejména blok digitalizace a blok vkládání grafiky je třeba taktovat kmitočtem synchronizovaným s videosignálem. Takový hodinový signál je vytvářen čítačem TIMER1 mikroprocesoru ARM, který je nastaven pro změnu svého výstupu každou druhou náběžnou hranu PCLK[12] a generuje tedy obdélníkový signál o kmitočtu 15*MHz*. Tento kmitočet odpovídá získanému rozlišení:

$$w = 52\,\mu s \times 15\,MHz = 780\,px\,,$$

což je rozlišení velmi blízké rozlišení 768 obrazových bodů na řádek, které je deklarované normou CCIR.

[11] Phase Lock Loop – obvod fázového závěsu. S děličkou frekvenci ve zpětné vazbě lze tento obvod použít jako násobičku kmitočtu.
[12] Hodinový signál periferií mikroprocesoru



obr. 13: Časový diagram spouštění hodinového signálu

Hodinový signál pro digitalizaci obrazu je spouštěn pouze v místech, kde opravdu potřebujeme digitalizovat videosignál. V místech, kde videosignál obsahuje synchronizační značky, je tento hodinový signál zastaven. Ke spuštění dojde opět po skončení synchronizační značky a je třeba dbát na to, aby oba signály byly co nejlépe sfázovány. O to se stará dvojstupňové přerušení odvozené od synchronizačního signálu.

V prvním stupni je spuštěna přerušovací rutina vyvolaná spádovou hranou signálu `HSYNC` (obr. 13). Toto přerušení je spuštěno s časovou nejistotou vůči zdroji přerušení zhruba 72*ns*; nejistota je způsobena mimo jiné tím, že procesor v době příchodu přerušení vykonává náhodnou instrukci a zpracování každé instrukce trvá různě dlouho. Přerušovací rutina je spuštěna až po vykonání právě prováděné instrukce. Po spuštění přerušovací rutiny v tomto přerušení procesor inkrementuje čítač řádku, překonfiguruje přerušení na náběžnou hranu signálu `HSYNC` a nastaví mu vyšší prioritu (`FIQ`[13]). Následně skočí do bloku instrukcí `NOP`[14], jehož délka je vypočtena tak, aby během jeho provádění bylo vyvoláno druhé přerušení od signálu `HSYNC`. Díky tomu, že bylo přerušení

[13] Fast Interrupt reQuest – v procesorech řady ARM7 je to typ přerušení s nejvyšší prioritou

[14] No OPeration – instrukce, během které se pouze inkrementuje čítač instrukcí, ale neprovádí se žádná operace s daty. Tato instrukce se vyznačuje krátkou dobou vykonání.

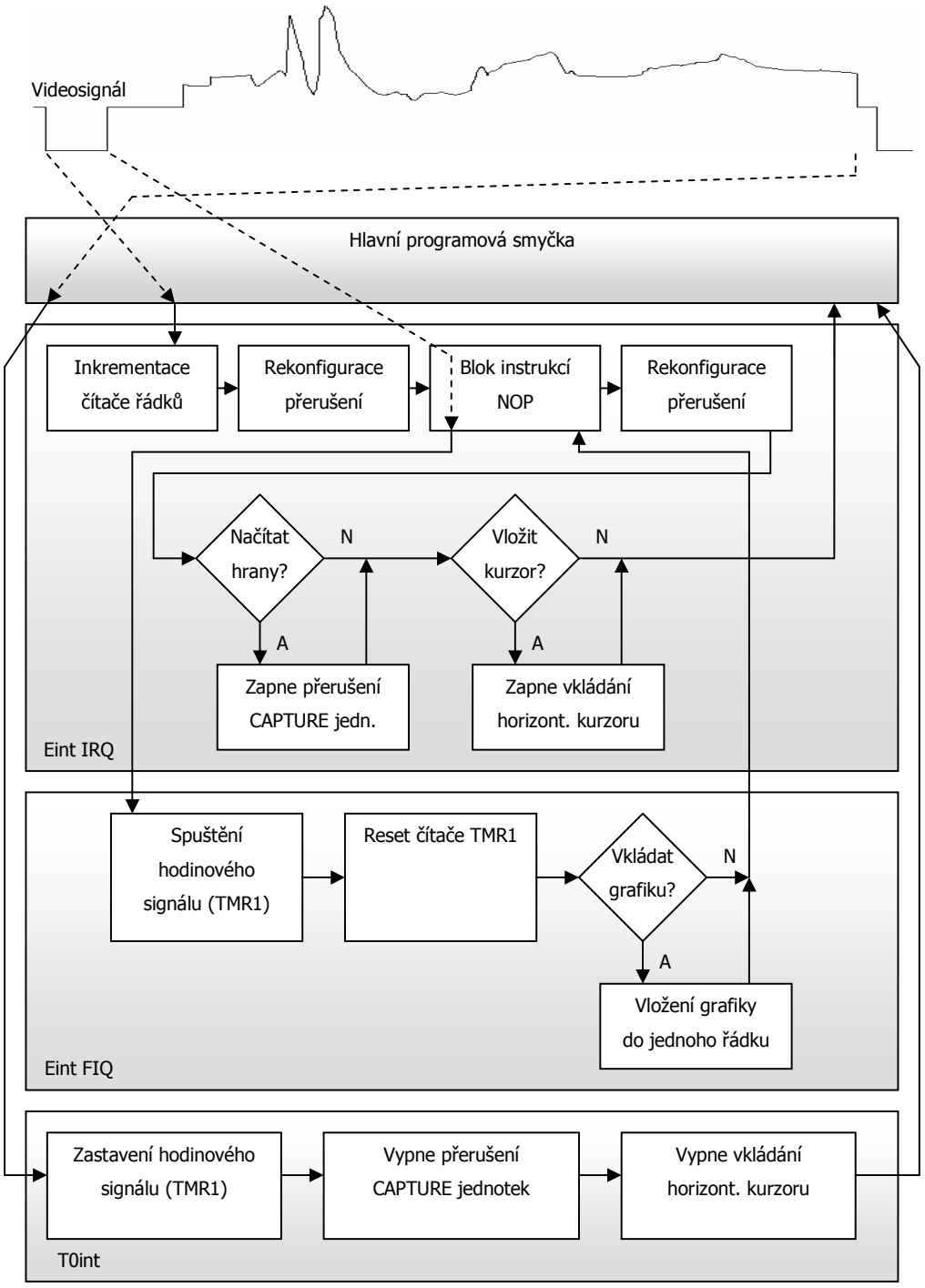
vyvoláno na instrukci NOP, je nejistota spuštění (jitter) přerušovací rutiny přibližně 17*ns*, což se blíží periodě hodinového signálu mikroprocesoru (obr. 15). V tomto přerušení dojde k aktivaci čítače TIMER1, který je nastaven pro generování obdélníkového signálu s kmitočtem 15*MHz* a zároveň je vynulován čítač TIMER0, který každou periodu PCLK inkrementuje svoji hodnotu. Na tento čítač jsou navázány CAPTURE jednotky sloužící k zachycení polohy hrany (viz 3.1.6.1).

V případě, že je tento řádek nastaven jako grafický (do řádku se vkládá informace z grafické paměti procesoru), je přímo z přerušení typu FIQ volána rutina pro zobrazení grafiky.

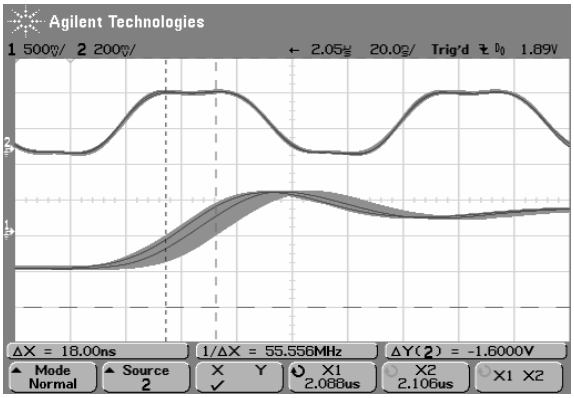
Po návratu zpět do přerušení typu IRQ jsou parametry pro vyvolání přerušení nastaveny do původního stavu, tedy na přerušení typu IRQ od spádové hrany signálu HSYNC. Dále se v přerušení testuje, zda se v daném řádku videosignálu provádí nějaké měření. Pokud ano, je zapnuto přerušení od CAPTURE jednotek CAP0.2 a CAP0.3, pokud ne, je toto přerušení deaktivováno. Nakonec je v tomto přerušení rozhodnuto, zda se mají vkládat řádkové kurzory do výstupního videosignálu.

Hodinový signál generovaný na výstupu mikroprocesoru je zastaven až na konci aktivní části videosignálu. O zastavení generování hodinového signálu se stará čítač TIMER0, který je nastaven pro vyvolání přerušení v čase 55μs od jeho spuštění (v obr. 13 je tento časový úsek označen znakem „T“). V tomto přerušení je zároveň vypnuto vkládání řádkových kurzorů do výstupního videosignálu v případě, že byl kurzor v tomto řádku aktivován.

Detailnější pohled na celý proces synchronizace běhu programu s videosignálem se uveden na obr. 14.



obr. 14: Diagram synchronizace běhu programu s videosignálem



obr. 15: Jiter vyvolání externího přerušení

3.1.2 Vzájemné propojení jednotlivých bloků

Šipky v blokovém schématu modulu naznačují propojení mezi jednotlivými bloky. Černé šipky ukazují vstupy a výstupy, modré šipky značí interní řídicí signály a červená šipka je sada datových signálů sloužící k přenosu digitalizovaného videa. Žlutě vyznačené signály jsou hodinové signály sloužící k taktování jednotlivých součástek.

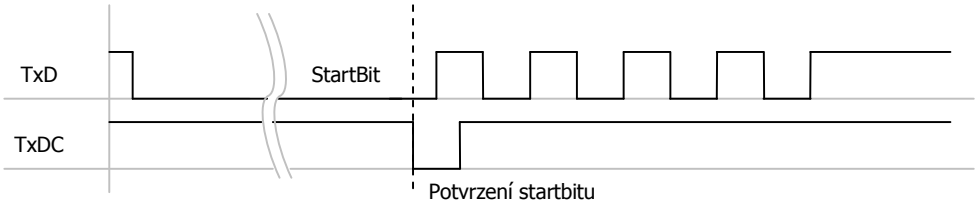
Vstupem do obvodu je signál CCD kamery. Tento signál je v bloku digitalizace videosignálu rozdělen na synchronizační pulzy HSYNC, VSYNC, BURST, ODD/EVEN a osmibitovou jasovou úroveň videosignálu. Signál BURST je přímo v bloku digitalizace využit pro signalizaci upínací úrovně. Signál HSYNC je přiveden na externí přerušení mikroprocesoru a na vstup obvodu CPLD, stejně je rozveden i signál VSYNC. Signál ODD/EVEN je přiveden na standardní vstup mikroprocesoru, konkrétně na 21. pin prány P1.

Hranový detektor implementovaný v obvodu CPLD má dva výstupy, kterými mikroprocesoru ARM signalizuje polohu spádových a náběžných hran. Tyto výstupy jsou připojeny na vstupy CAP0.2 a CAP0.3 CAPTURE jednotky, která umožňuje zaznamenat čas příchodu hrany. Celý proces je detailněji popsán v kapitole 3.1.6.1.

V obvodu CPLD je zároveň implementována řídicí logika pro přenos videosignálu přes USB. Obvod CYPRESS generuje signál VIDEO_START, který je připojen na vstup CPLD a oznamuje, že příští snímek bude přenášen. CPLD následně generuje signály SLWR a PKTEND, které řídí přenos dat. Při tom

kontroluje informace na výstupech FULL_FLAG a EMPTY_FLAG náležící FIFO paměti obvodu CYPRESS. Podrobnosti jsou uvedeny v kapitole 3.1.7.

Komunikace mezi nadřazeným systémem a mikroprocesorem přes USB je realizována sériovou linkou mezi obvodem CYPRESS a ARM. Jelikož byl použit obvod CYPRESS s pouzdrem SSOP56, který nemá vyvedenou standardní sériovou linku UART, použil jsem pro oboustranný přenos dat softwarový UART, který je doplněný o dva handshakové signály. Při vysílání obvod nastaví na vysílacím výstupu logickou nulu a čeká, až přijímač příslušným signálem potvrdí přenos. Poté začne přenos jednoho bytu. Přenos tedy probíhá standardně s tím rozdílem, že na začátku přenosu každého bytu se čeká na druhé zařízení, a startbit tak může být poměrně dlouhý (obr. 16). Tento způsob přenosu byl zvolen z toho důvodu, že na procesoru ARM již nezbývalo volné externí přerušení. Standardní UART realizovaný čistě softwarově by však příliš zatěžoval mikroprocesor, protože by se musela velmi často testovat logická úroveň na vstupu. Jelikož komunikace mezi procesorem a nadřazeným systémem není kritická, ke komunikaci dochází zřídka a nepřenášejí se velké objemy dat, je možné zvolit popsaný způsob.



obr. 16: Přenos jednoho bytu po softwarové sériové lince s potvrzováním

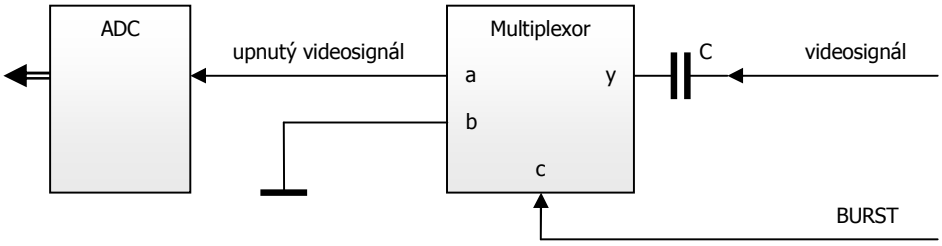
3.1.3 Blok digitalizace videosignálu

Hlavní částí bloku digitalizace videosignálu, který je realizován dle obr. 4, tvoří paralelně kaskádní AD převodník AD9280. Před vlastní digitalizací signálu je však třeba videosignál upnout na referenční úroveň.

Upínání signálu je realizováno multiplexorem řízeným signálem BURST z oddělovače synchronizačních pulzů (obr. 17). Většinu času multiplexor propojuje vstup videosignálu s AD převodníkem. Při aktivním vstupu BURST, který signalizuje napěťovou úroveň černé barvy ve videosignálu, je však

videosignál přes kapacitu C připojen na nulové napětí. Na kapacitě se tak uloží rozdílové napětí mezi úrovní reprezentující černou barvu a nulovým napětím, které je při přepnutí na AD převodník odčítáno.

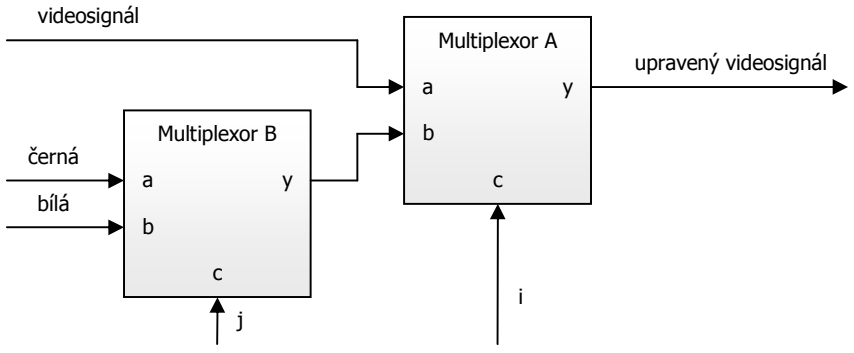
Po upnutí videosignálu lze přistoupit k převodu do digitální formy. Tento převod je řízen hodinovým signálem, jehož vlastnosti jsou popsány v kapitole **Chyba! Nenalezen zdroj odkazů.** .



obr. 17: Upínací obvod

3.1.4 Blok vkládání grafiky do videosignálu

Hardwarovou část bloku pro vkládání grafiky do videosignálu tvoří dva multiplexory řízené mikroprocesorem (obr. 18). V neaktivním stavu signálu „i" je multiplexor A přepnut tak, že na výstup je přenášen vstupní videosignál. Při aktivaci signálu „i" se výstup přepne do režimu vkládání barvy. Logická úroveň signálu „j", řídící multiplexor B, pak rozhoduje o tom, které ze dvou předdefinovaných napětí je vloženo do videosignálu. Na vstup multiplexoru B jsou připojeny dvě napětí získaná z děliče napětí a vypočtená tak, aby jednomu napětí odpovídala ve videosignálu černá a druhému bílá barva.



obr. 18: Vkládání grafiky do videosignálu

Softwarová část tohoto bloku je tvořena balíkem funkcí pro vkládání grafiky do videosignálu. Celý proces zobrazení je rozdělen do dvou kroků.

V prvním kroku se požadovaný obraz připraví jako bitmapa v grafické paměti mikroprocesoru. Pro tuto část jsou v knihovně připraveny funkce pro přípravu obrazu v grafické paměti popsané v kapitole 3.1.4.1 .

V kroku druhém se grafická paměť postupně prochází a podle jejího obsahu se nastavují řídící signály multiplexorů, čímž dochází k vykreslení požadované grafiky.

3.1.4.1 Příprava obrazu v grafické paměti

V mikroprocesoru je implementována sada funkcí pro přípravu obrazu v grafické paměti. Grafická paměť je místo v paměti `RAM` mikroprocesoru vyčleněná pro účely zobrazení grafiky. Jelikož v každém bodě je možné zobrazit tři stavy (černou barva, bílou barva a původní videosignál), každý zobrazovaný pixel zabírá v paměti dva bity. V každém bytu je tedy informace o čtyřech sousedních pixelech (obr. 19). Bit „T/C“ značí, zda bude pixel průhledný nebo bude vložena barva (transparent/color), bit „B/W“ říká, jaká barva bude vložena (black/white). Grafické paměti je přiřazeno místo v paměti `RAM` o velikosti 2kB, což vystačí pro vykreslení obrázku v rozlišení 256×32*px*.

obr. 19: Jeden byte grafické paměti

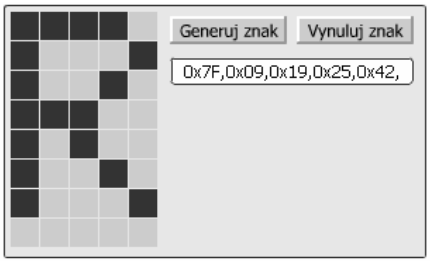
Základní a také nejjednodušší funkcí pro přípravu obrazu v grafické paměti je nastavení barvy zadaného pixelu na určitou hodnotu. V podstatě pouze zapíše požadované číslo na určité místo grafické paměti.

Další funkcí z této řady je funkce pro vložení znaku. Jednotlivé znaky jsou uloženy v programové paměti mikroprocesoru ve formě matice o velikosti 5×8 pixelů. Celkem je v paměti uloženo dolních 128 znaků `ASCII`[15] tabulky, které

[15] American Standard Code for Information Interchange – jde o kódovou tabulku, která definuje znaky anglické abecedy a jiné znaky používané v informatice.

zahrnují malou a velkou abecedu bez diakritiky, čísla a některé další často používané znaky.

Pro vytvoření matic reprezentujících grafickou podobu znaků jsem použil vlastní aplikaci napsanou ve skriptovacím jazyku ActionScript (obr. 20), která po „namalování" znaku vytvoří pětici bytů reprezentující jeden znak.



obr. 20: Aplikace pro generování matice pixelů reprezentující jeden znak

Funkce pro vložení řetězce a funkce pro vložení čísla z proměnné typu long a float jsou rozšířením funkce pro vložení jednoho znaku.

V neposlední řadě knihovna nabízí funkci, která dokáže zobrazit jednorozměrné pole typu char jako graf.

3.1.4.2 Vložení grafiky do videosignálu

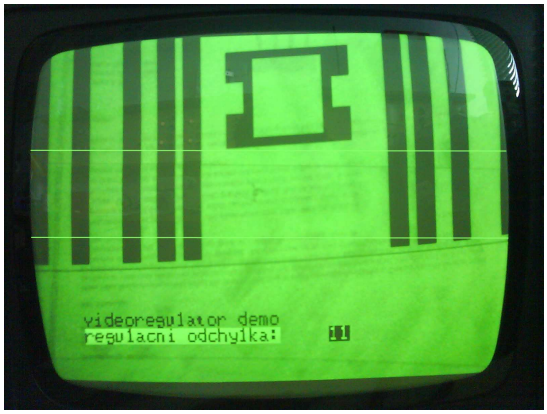
Funkce pro vložení grafiky do videosignálu je spouštěna z přerušení s nejvyšším stupněm priority (`FIQ`), pokud je číslo právě načítaného řádku v intervalu zobrazení grafiky. Tento interval je v programu mikroprocesoru definován proměnnými grPocRadek, který definuje první řádek, ve kterém se grafika do videosignálu vkládá, a grPocetRadku, který říká, kolik řádků je nastaveno jako grafických. Funkce pracuje podle jednoduchého algoritmu. Po načtení jednoho bytu z grafické paměti se tento byte rozkóduje na čtyři pixely jednoduchým posunem a maskováním dvou spodních bitů. Hodnota pixelu se přímo zkopíruje na řídicí výstupy.

První verze této funkce byla napsána přímo v assembleru mikroprocesoru, což zaručovalo optimální řešení daného problému. Komplikace však nastaly, když velikost kódu přesáhla hranici 4kB a nebylo již možné projekt slinkovat demoverzí kompileru `MDK` vývojového prostředí Keil µVision. Použil jsem tedy volně dostupný překladač `GCC`, který však nepodporuje v plném rozsahu přímý

zápis assemblerovského kódu do funkcí jazyka C (tzv. in-line assembler). Proto je nakonec finální verze této funkce napsána v jazyku C (s vloženými instrukcemi NOP, které zajišťují správné časování). Oproti rutině napsané v assembleru se funkčnost příliš nezměnila, což svědčí o kvalitní optimalizaci překladače GCC.

Experimentálně zjištěná délka trvání jednoho pixelu je 160*ns*. Při délce aktivní části jednoho řádku videosignálu 52*μs* je tedy možné do jednoho řádku vložit až 325 obrazových bodů generovaných mikroprocesorem. Z důvodu úspory místa v paměti jsou však funkce navrženy na rozlišení 255 bodů na řádek a vkládaná grafika tak na šířku nezaplňuje celou obrazovku.

Výpis zdrojového kódu funkce je uveden v příloze (Příloha 15: Funkce pro vkládání grafiky do videosignálu).



obr. 21: Příklad grafiky vložené do TV signálu

Kromě vkládání rastrové grafiky modul disponuje ještě možností vložení horizontálních kurzorů. Tyto kurzory označují úsek obrazu, ve kterém je prováděno hledání hran a následné měření. Podoba kurzoru je zřejmá z obr. 21. Jedná se o světlou vodorovnou úsečku v celé šířce obrazu. Vložení kurzorů do videosignálu má na starost přerušení od signálu HSYNC, vypnutí vkládání přerušení od časovače TIMER0.

Časová souslednost videosignálu a vkládání grafiky do videosignálu je naznačena v obr. 14.

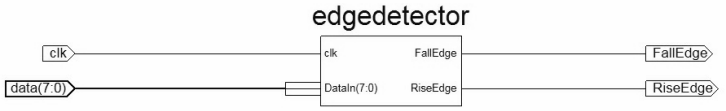
3.1.5 Blok předzpracování obrazu v CPLD

Blok předzpracování obrazu (implementovaný v obvodu XC9572XL) se stará o nalezení hran v obraze a oznámení nalezené hrany mikroprocesoru. Použitý obvod obsahuje 72 elementárních buněk, což je pro zpracování obrazu velice málo. Výhodou je však jeho cena, která se pohybuje v řádu kolem $ 5. Při vývoji vnitřní struktury obvodu proto byl kladen důraz na co největší jednoduchost návrhu.

Navržený blok hranové detekce (obr. 22) pracuje s hodnotami posledních tří vzorků získaných z AD převodníku (dva vzorky jsou uložené v registru, hodnota aktuálního vzorku je dostupná na vstupu obvodu) a hranu detekuje přírůstkovou metodou (viz 2.4.1.2). Diference je namísto klasického výpočtu, tedy $d_n = x_n - x_{n-1}$, vypočtena podle vzorce $d_n = x_n - x_{n-2}$. Po výpočtu d_n je velikost porovnána s hodnotou thresholdu a při překročení této hodnoty je hrana signalizována na příslušném výstupu.

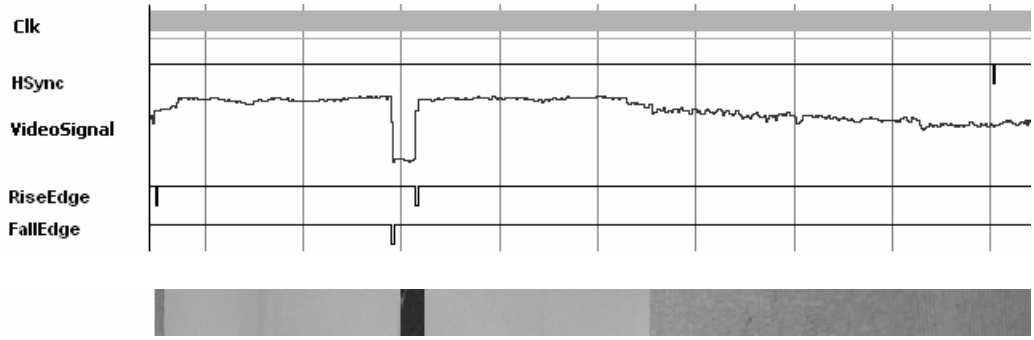
Zpoždění indikace hrany na výstupu obvodu CPLD je přibližně 360*ns*, což odpovídá pěti periodám hodinového signálu. S tímto zpožděním je třeba počítat při výpočtu regulační odchylky.

Pro vývoj byl použit software dodávaný výrobcem obvodu XILINX ISE a programovací jazyk VHDL.



obr. 22: Vstupy a výstupy hranového detektoru

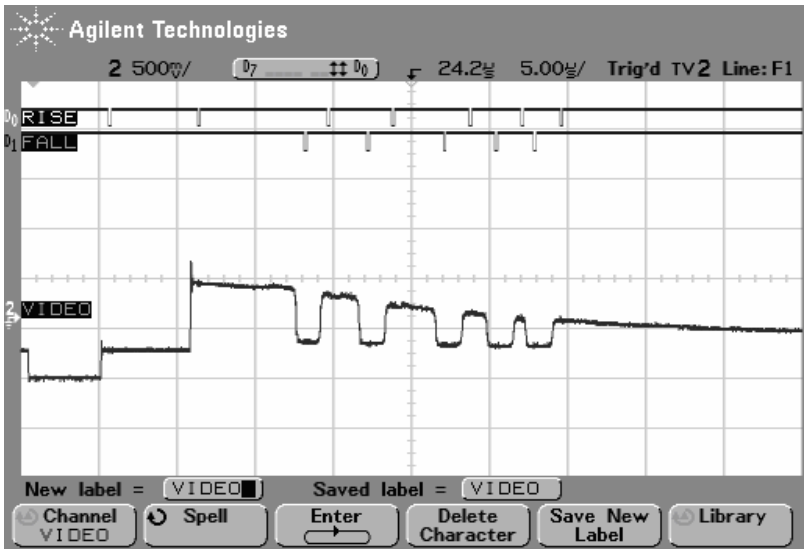
Pro ověření správnosti návrhu jsem využil simulaci v programu ModelSim XE se vstupními daty získanými z reálného obrázku pořízeného digitálním fotoaparátem (obr. 23). Jelikož je veškeré zpracování číslicové, neměla by se simulace od výsledku odchylovat.



obr. 23: Simulace hranového detektoru s reálnými vstupními daty

Na obr. 24 je uvedena reálná odezva hranového detektoru na připojený videosignál. Je zřejmé, že i když snímaná scéna nebyla nasvícena ideálně (na levé straně je scéna nasvícena správně, směrem k pravé straně jas i strmost hran klesá), hranový detektor má očekávaný výstup.

Výpis zdrojového kódu modulu hranového detektoru, který je napsán v jazyku VHDL, je uveden v příloze (Příloha 18: Modul přírůstkového hranového detektoru).



obr. 24: Odezva hranového detektoru

3.1.6 Blok vyhodnocení videosignálu a regulace

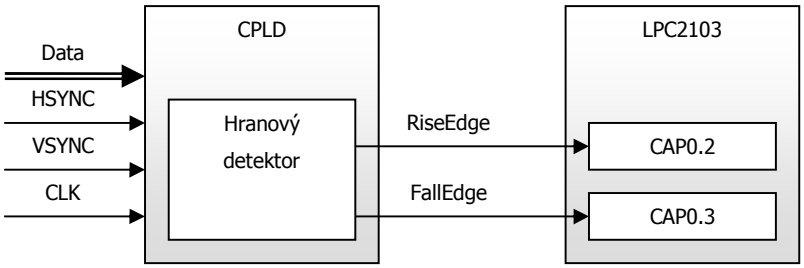
Hlavní částí modulu videoregulátoru je mikroprocesor `LPC2148`, který zastává funkci načtení zjednodušeného obrazu předzpracovaného hranovým

detektorem, vyhodnocení těchto dat a výpočet regulační odchylky. Celý proces probíhá v následujících krocích:

- načtení hran,
- nalezení důležitých částí v obrazu,
- regulace,
- odeslání výsledku.

3.1.6.1 Načtení hran mikroprocesorem

Blok hranového detektoru předává mikroprocesoru informaci o nalezené hraně ve formě krátkého impulzu v době, kdy se ve videosignálu tato hrana vyskytla. Signál je připojen na vstupy `CAP0.2` a `CAP0.3` `CAPTURE` jednotky[16] mikroprocesoru (viz obr. 25). Tato jednotka je nakonfigurována tak, že při výskytu krátkého impulzu zapíše čas výskytu impulzu do registru a vyvolá přerušení. Časem výskytu impulzu se rozumí hodnota čítače, který je inkrementován hodinovým signálem periferií mikroprocesoru a je nulován signálem `HSYNC`. Tento čas je tedy úměrný poloze hrany vzhledem h levému okraji snímané scény.



obr. 25: Propojení hranového detektoru s CAPTURE jednotkami mikroprocesoru

Přerušovací rutina vyvolaná `CAPTURE` jednotkou nejprve zjistí, na kterém vstupu jednotky došlo k události, a podle toho rozhodne, zda se jedná o náběžnou nebo spádovou hranu. Polohu a typ hrany poté zapíše do globálně přístupné fronty. Tato fronta je společná pro hrany ve všech měřených řádcích. Se zápisem hrany do fronty je zároveň vytvářen katalog řádků, který obsahuje

[16] Jednotka připojená k čítači mikroprocesoru umožňuje zaznamenat čas vzniklé události na některém z pinů mikroprocesoru a následně vyvolat přerušení.

informace důležité k dalšímu zpracování dat ve frontě. V tomto katalogu ke každému řádku nalezneme tyto informace:

- ukazatel do fronty určující první hranu, která patří danému řádku,
- počet hran patřící danému řádku,
- stav zpracování řádku – právě se načítá, načteno, zpracováno.

Během načítání je vhodné kontrolovat, zda fronta hran nebyla zcela naplněna. V takovém případě je zřejmé, že přichází více hran, než je zpracující rutina schopna zpracovat, a je tedy třeba změnit velikost fronty, nebo snímat jednodušší scénu.

Kvůli úspoře místa byl implementován algoritmus, který rozhodne, zda se v právě načítaném řádku videosignálu provádí nějaké měření. Pokud ano, jsou hrany zapisovány do fronty a zpracování proběhne normálně. Pokud však v daném řádku není definováno žádné měření, je vypnuto přerušení od CAPTURE jednotky, tudíž jsou všechny příchozí hrany ignorovány, a nedochází tak ke zbytečnému zaplňování fronty hran. Pak je možné zmenšit frontu hran a uspořit místo v paměti RAM. Pro běžná měření se fronta o velikosti 512 hran ukázala jako postačující.

Pro každý typ hrany je využita jedna CAPTURE jednotka. Tento způsob zapojení má tu výhodu, že je možné zpracovat rychlý příchod náběžné a spádové hrany těsně za sebou. Při příchodu více hran stejného typu těsně za sebou však může dojít ke ztrátě těchto hran. To je způsobeno délkou zpracování vzniklé události na vstupu CAPTURE jednotky. Přerušovací rutina pro zapsání hrany do fronty trvá od vyvolání do skončení přibližně 2,6μs. Pokud se nová hrana stejné polarity ve videosignálu vyskytne ještě před skončením zpracující funkce, nedojde k zachycení časového údaje a hrana bude ztracena. Čas 2,6μs odpovídá přibližně 38 pixelům.

Vzhledem k tomu, že bezprostředně po zapnutí hodinového signálu pro AD převodník jsou na výstupu převodníku neplatná data a nějakou dobu (konkrétně pět hodinových cyklů s přírůstkovým hranovým detektorem) trvá, než tato neplatná data opustí hranový detektor, zapíná se přerušení od CAPTURE jednotek s určitým zpožděním pozahnutí hodinového signálu.

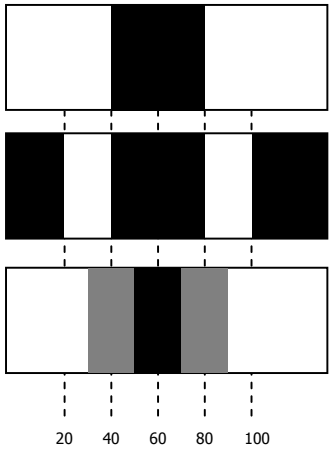
3.1.6.2 Nalezení důležitých částí v obrazu

Pro získání hodnoty akčního zásahu, což je výsledek výpočtu regulátoru, je třeba nejprve znát vstupní veličiny regulátoru - hodnotu regulované veličiny a požadovanou hodnotu. Obě tato čísla mohou být získána z předzpracovaného obrazu uloženého ve formě hran v mikroprocesoru.

Hledání důležitých částí v obrazu je rozděleno do dvou kroků. V prvním kroku je v každém řádku videosignálu hledána definovaná sekvence hran – značka. V ideálním případě je tak v každém řádku nalezena jedna značka. Ve druhém kroku je ze souboru výsledků ze všech řádků vybrán podle zadaných kritérií jeden výsledek.

Nalezení značky v řádku videosignálu

Videoregulátor umožňuje definovat až dvě značky. Pod pojmem „značka“ si lze představit sekvenci hran dané polarity, které jsou hledány v jednom řádku videosignálu. V obr. 26 jsou uvedeny příklady značek s jejich hranovým popisem. U každé hrany bude kromě polarity uvedena i vzdálenost od první hrany (v pixelech). U značky je rovněž možné uvést řádky videosignálu, na které bude hledání značky omezeno.



Tmavá značka na světlém pozadí je popsána jako spádová hrana (na poloze 0) následovaná náběžnou hranou na poloze 40.

Dva světlé proužky na tmavém pozadí budou popsány následující sekvencí hran:
náběžná 0, spádová 20, náběžná 60, spádová 80

Při použití přírůstkového hranového detektoru lze definovat značku, která používá více stupňů šedi a popsat ji vhodnou sekvencí hran:
spádová 0, spádová 20, náběžná 40, náběžná 60

obr. 26: Příklady značek a jejich hranových popisů

Hledání pak probíhá podle následujícího algoritmu: v hranách na řádku jsou nalezeny všechny sekvence hran odpovídající definované značce. Ke každé sekvenci je vypočtena odchylka od rozměrů definovaných značkou a je vybrána

sekvence hran s nejmenší odchylkou. Tato sekvence je označena jako nalezená značka a je možné k ní určit několik parametrů, které mohou být užitečné pro další výpočty. Mezi tyto parametry patří:

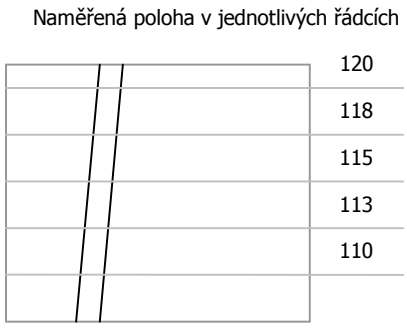
- poloha,
- šířka,
- odchylka od zadaných hodnot,
- pořadové číslo první hrany nalezené značky,
- celkový počet hran na řádku a
- pořadové číslo řádku

Ideálně tedy získáme v každém měřeném řádku jednu nalezenou značku, popsanou těmito parametry.

Výběr jednoho výsledku

Ze sady získaných hodnot z předešlého kroku je třeba získat pouze několik čísel. Tento proces jsem nazval „vyhledávání ve značkách“, i když ve skutečnosti jde o kombinaci výběru a výpočtu. U jedné značky je možné nadefinovat jedno nebo dvě vyhledávání. Základem vyhledávání je jeden ze čtyř algoritmů, které jsou nazvány průměr, směrnice, minimum a maximum. Vstupem pro tyto operace je jeden z parametrů vypočtených v předchozím kroku.

Při operaci „průměr“ je vypočtena průměrná hodnota zadaného argumentu v prohledávaných řádcích. Tato hodnota je počítána jako suma získaných čísel dělená počtem řádků, ve kterých se provádí hledání značky. Funkce slouží k zpřesnění výsledku nebo k odfiltrování špatně vypočtených výsledků.

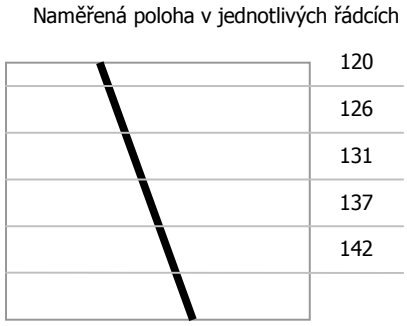


Výsledkem operace bude průměrná hodnota všech hodnot zaokrouhlená na celá čísla. V případě měření průměrné hodnoty polohy tedy:

(120 + 118 + 115 + 113 + 110) / 5 = 115

obr. 27: Příklad výpočtu průměrné hodnoty polohy

Výsledkem operace „směrnice“ je diference parametru získaného v prvním a posledním řádku. Funkce může sloužit například pro měření úhlu nebo pro měření změny šířky.

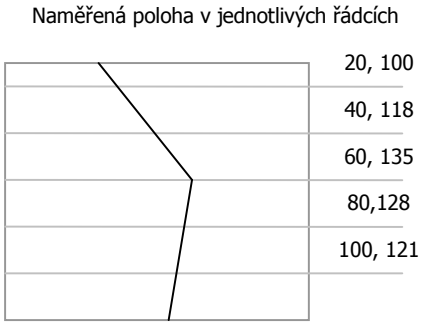


Výsledkem operace bude rozdíl první a poslední naměřené hodnoty. V případě výpočtu diference polohy bude tedy výpočet vypadat následovně:

120 - 142 = -22

obr. 28: Příklad výpočtu diference polohy

Operace „minimum“ a „maximum“ vyhledávají minimální (resp. maximální) hodnotu zadaného parametru. U těchto operací se kromě prohledávaného parametru nastavuje také výsledný parametr, a je tedy možné celý proces vyhledávání nastavit např. pro vyhledání čísla řádku, ve kterém je poloha značky největší.



Vyhledávání lze nadefinovat například pro nalezení čísla řádku v místě, kde je poloha značky maximální. V tomto případě by tedy algoritmus vracel číslo 60.

obr. 29: Příklad vyhledání extrému parametru

Přiřazení vypočtených výsledků jednotlivým vstupním veličinám regulátoru

Hodnoty získané procesem vyhledávání ve značkách je třeba přiřadit jednotlivým vstupním veličinám regulátoru.

Jeden z výsledků vyhledávání musí být přiřazen měřené veličině. Po nadefinování značky, vyhledávání ve značce a přiřazení výsledku měřené

veličině je hodnota měřené veličiny v každém okamžiku rovna výsledku vyhledávání.

Naproti tomu je žádoucí, aby žádaná veličina mohla být časově proměnná nejen v rámci jednoho výsledku, ale aby ji bylo možné také nadefinovat jako konstantu (konstantní polohu vzhledem k levému okraji scény snímané kamerou) a tuto konstantu případně měnit s časem. Proto lze žádané veličině přiřadit pole hodnot, kde každá z hodnot může být buď polohou vzhledem k levému okraji scény, nebo odkazem na některý z výsledků vyhledávání ve značkách. Každé hodnotě je také přiřazen čas, ve kterém je hodnota aktivní. Je tedy možné např. nastavit sekvenci, kde pět sekund bude žádaná veličina rovna jednomu výsledku vyhledávání, poté dvě sekundy pravému okraji snímané scény a nakonec druhému výsledku vyhledávání.

Spouštění sekvence může být buď automatické, což znamená, že se sekvence periodicky opakuje, nebo od určité spouštěcí podmínky (Trigger). Spouštěcí podmínku lze nadefinovat jako porovnání některého z výsledků vyhledávání s konkrétním číslem.

Celý mechanizmus je tak možné nastavit např. pro situaci, kdy regulátor čeká, až se v zorném poli objeví objekt (což může být reprezentováno spouštěcí podmínkou, která říká, že počet hran v daném řádku musí být větší než nějaká konkrétní hodnota) a poté navede pohyblivou část zařízení přímo k objektu (žádaná veličina bude nastavena jako poloha objektu v zorném poli, měřená poloha bude nastavena jako poloha pohyblivé části zařízení). Po určité době opět vrátí pohyblivou část do výchozí polohy.

3.1.6.3 Regulace

Ze znalosti měřené i žádané veličiny je možné prostým rozdílem vypočíst regulační odchylku. Výstupem regulačního obvodu je akční zásah. Obecný vztah mezi akčním zásahem (u) a regulační odchylkou (e) je odvozen v kapitole 2.4.3.

Regulace v mikroprocesoru je implementována podle tohoto vzorce. Koeficienty PSD regulátoru použité při výpočtu mohou však být pouze celá čísla a výsledek by tedy vycházel příliš veliký. Proto je možné nastavit ještě dělitele, kterým je možné zmenšit výslednou hodnotu akčního zásahu.

3.1.6.4 Odeslání výsledku akčnímu členu

Hodnota akčního zásahu zmenšená dělením je ihned po výpočtu odeslána po sériové lince akčnímu členu soustavy (např. modulu pro ovládání motorů). Sériová linka modulu přenáší data rychlostí 57 600bps, má nastaveno osm datových bitů, žádnou paritu a jeden stop bit. Při programování bylo předpokládáno, že na výstup bude připojen modul pro řízení motorů, který je popsaný v kapitole 3.2 této diplomové práce. Příkazy odesílané po sériové lince jsou proto přizpůsobeny tomuto modulu. V zásadě je ale možné na výstup připojit jakýkoliv modul, který bude schopný zpracovat odesílané příkazy.

Podrobněji je formát komunikace popsán v části diplomové práce zabývající se realizací modulu pro řízení motorů, konkrétně v kapitole 3.2.3.1.

3.1.7 Blok komunikace s nadřazeným systémem

Modul videoregulátoru je možné konfigurovat po sběrnici `USB` pomocí `PC`. Na straně modulu se o komunikaci přes tuto sběrnici stará obvod `CYPRESS CY7C68013A`. Ten obsahuje řadič `USB`, mikrokontrolér `8051`, paměť `FIFO` a další periferie zjednodušující použití obvodu v aplikaci.

Pro přenos dat jsou využity tři endpointy:

- EP1IN
- EP1OUT
- EP2IN

3.1.7.1 Konfigurace videoregulátoru z nadřazeného systému

Konfigurace modulu videoregulátoru a hlášení stavu počítači je realizováno pomocí endpointů EP1IN a EP1OUT. Přes každý endpoint probíhá komunikace jedním směrem (směr OUT znamená z počítače do videoregulátoru).

Vzhledem k malému množství dat přenášených těmito kanály je nastavena velikost rámce u těchto endpointů na 64*B*. Přenos je typu `BULK`.

Komunikační protokol je velmi prostý. Komunikaci vždy zahajuje PC vysláním zprávy na EP1OUT. Podle hodnoty prvního vyslaného bytu je provedena některá ze čtyř akcí uvedených v následující tabulce:

Příkaz	Popis
0x00	Testovací příkaz. Pouze blikne LED diodou připojenou na bránu řadiče.
0x04	Odešle všechna zbylá data mikroprocesoru ARM.
0x05	Žádost o vyslání dat přijatých z procesoru ARM na endpoint EP1IN.
0x07	Žádost o odeslání jednoho snímku obrazu do počítače přes EP2IN

tab. 2: Přehled příkazů USB řadiče

Zprávy začínající znakem 0x04 jsou přeposílány mikroprocesoru ARM po softwarové sériové lince (viz 3.1.2). Procesor je schopen zpracovat několik základních příkazů, pomocí kterých je možné ho konfigurovat. Komunikační protokol je velmi jednoduchý. První znak určuje, o jaký příkaz se jedná, ostatní znaky jsou již parametry příkazu. Čísla jsou uvedena v dekadickém formátu, jednotlivé cifry reprezentují ASCII znaky „0" – „9". Seznam příkazů je uveden v tab. 3.

Žádost o data přijatá řadičem USB od mikroprocesoru ARM způsobí odeslání odpovědi na endpoint EP1IN. Zpráva je vždy dlouhá 64 bytů. Prvný byte obsahuje počet platných dat, následují tři nevyužité byty a poté už samotná data doplněná nulami.

Na žádost o odeslání jednoho snímku je generovaný impulz na jednom z pinů řadiče. Tento impulz je dále zpracován obvodem CPLD (viz 3.1.7.2).

1. znak	Příkaz	Popis
s	Sériová linka	Všechna zbylá data jsou odeslána na sériovou linku UART0.
r	Results	Mikroprocesor vrátí řetězec obsahující všechny výsledky vyhledávání ve značkách.
a	Regulační veličiny	Mikroprocesor vrátí vstupy a výstup regulátoru.
z	Značka	Konfigurace značek. Bezprostředně za znakem „z" následuje číslo značky, poté čárka a pak definice hran. Každá hrana začíná znakem „r" nebo „f" podle toho, zda se jedná o náběžnou nebo spádovou hranu a je následována číslem, které uvádí její polohu vzhledem k první hraně značky. Jednotlivé hrany jsou odděleny čárkou. V tomto příkazu je také možné definovat polohu, na které se má značka vyskytovat (znak „p" následovaný číslem) a maximální možnou odchylku (znak „x" následovaný číslem)
e	Enable	Tento příkaz je schopen vypnout a opět zapnout funkci regulace. Znak „e" je následován jedním číslem. Pokud je toto číslo rovno nule, je funkce regulátoru vypnuta.

1. znak	Příkaz	Popis
f	Filtr	Tímto příkazem je možné definovat konstanty q0 až q2 implementovaného PSD regulátoru. Znak „f“ může být následován znakem „i“, který inicializuje regulátor (ten vymaže všechny dosavadní konstanty). Znak „k“ následovaný číslem přidá do regulátoru jeden kořen. Je možné přidat více kořenů tak, že do příkazu vepíšeme vícekrát příkaz pro přidání kořenu. Tyto příkazy je třeba oddělit čárkou.
v	Vyhledávání ve značce	Tímto příkazem se nastavuje vyhledávání ve značce. Argumentem je pět čísel oddělených čárkou. První číslo udává číslo značky, ke které se vyhledávání vztahuje. Druhé číslo udává číslo vyhledávání. Třetí číslo udává odkaz na parametr, podle kterého se má vyhledávat, čtvrté číslo pak matematickou operaci, která se provádí. Poslední argument udává odkaz na parametr, který bude použit jako výsledek. Odkaz na parametr má následující význam: 1- poloha, 2- šířka, 3- odchylka, 4- pořadové číslo první hrany, 5- počet hran v řádku. Hodnota určující matematickou operaci může nabývat následujících hodnot: 1- minimum, 2- maximum, 3- průměr a 4- diference.
y	Měřená veličina	Příkaz, který přiřadí výsledek vyhledávání měřené veličině. Argumentem jsou dvě čísla oddělená čárkou. První číslo udává pořadové číslo značky, druhé číslo udává pořadové číslo vyhledávání ve značce.
w	Požadovaná veličina	Příkaz pro určení požadované veličiny. Je možné nadefinovat sekvenci požadovaných veličin, kde lze kombinovat relativní polohu vzhledem k levému okraji scény a výsledky naměřené videosenzorem. Jednotlivé údaje obsahují číslo (resp. odkaz na výsledek) a čas, ve kterém se má hodnota použít. Čas je zadáván v násobcích 20ms a od hodnoty je oddělen znakem „@“. Všechny hodnoty jsou pak odděleny čárkou. Odkaz na výsledek je možné zadat ve formátu „v00“. Dvojice čísel říká, o jaký výsledek se jedná. Jsou použity kombinace 00, 01, 10 a 11.

1. znak	Příkaz	Popis
t	Spouštění sekvence požadované veličiny	Příkaz nastavuje trigger požadované veličiny. Argumentem je pět čísel oddělených čárkou. První číslo udává, zda se jedná o autotrigger (hodnota 0 znamená spouštění od zadané podmínky). Druhý a třetí argument odkazují na číslo výsledku (číslo značky a číslo vyhledávání), čtvrtý argument definuje způsob porovnání a pátý argument je konstantou, se kterou se výsledek porovnává. Způsob porovnání může nabývat následujících hodnot: 1- výsledek je menší než zadaná konstanta, 2- výsledek je roven zadané konstantě a 3- výsledek je větší než zadaná konstanta.

tab. 3: Seznam příkazů pro videoregulátor

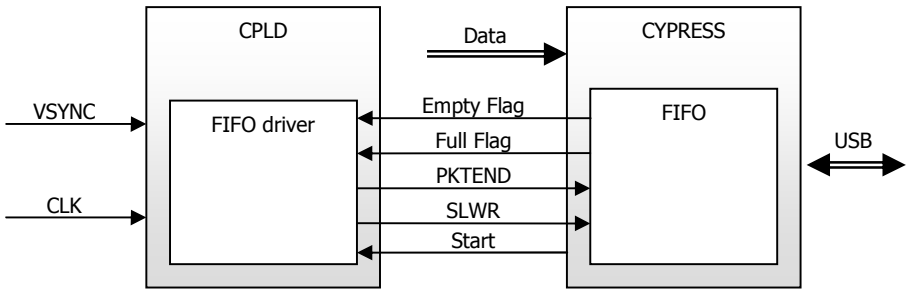
3.1.7.2 Přenos videa z modulu do nadřazeného systému

Pro přenos videa z modulu videoregulátoru do PC je vyčleněn endpoint EP8IN. Endpoint má nastavenou délku rámce 512 bytů a jako odesílací buffer slouží `FIFO` paměť obvodu `CYPRESS`. Do té je možné zapisovat data dvěma způsoby:

- synchronně a
- asynchronně.

Synchronní zápis do `FIFO` paměti nabízí vyšší zápisovou rychlost, tudíž i větší přenosovou rychlost. V tomto režimu je třeba generovat stálý hodinový signál na vstupu `IFCLK` obvodu `CYPRESS`. Vstup `SLWR` pak slouží jako hradlovací signál, kdy při náběžné hraně hodinového signálu a aktivní úrovni na signálu `SLWR` dojde k zapsání dat do paměti.

Asynchronní režim nabízí pomalejší zápis dat do `FIFO` paměti, naproti tomu však poskytuje jednodušší způsob řízení. Signál `IFCLK` je v tomto případě nevyužit, do paměti se data zapisují s náběžnou hranou signálu `SLWR`. Způsob připojení paměti FIFO je zobrazen na obr. 30.



obr. 30: Způsob připojení FIFO paměti k systému

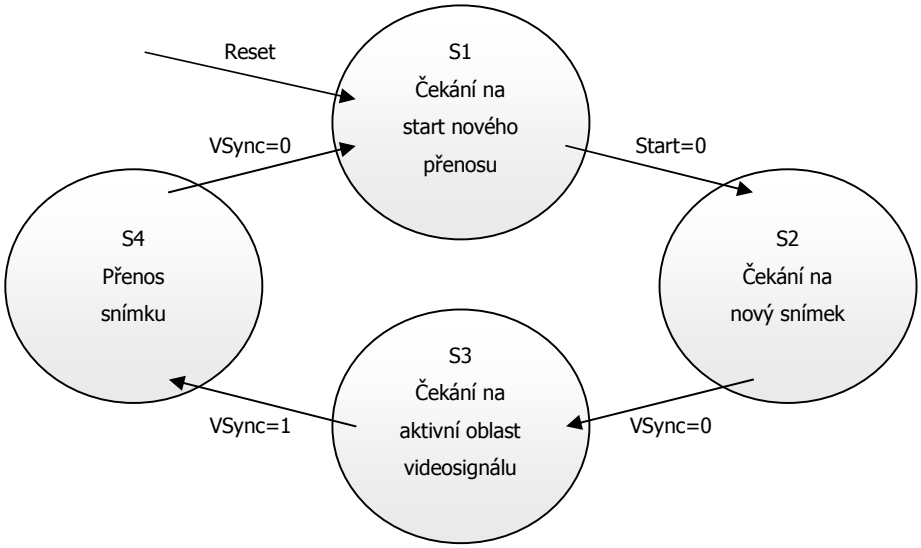
V případě modulu videoregulátoru není možné zajistit generování stálých hodin na vstupu IFCLK řadiče USB. Hodinový signál pro všechny části zařízení je generován mikroprocesorem ARM a při výskytu synchronizačního pulzu ve videosignálu je třeba na chvíli hodinový signál zastavit a synchronizovat ho s videosignálem. Z tohoto důvodu je pro přenos obrazu použito asynchronního režimu. Dle katalogového listu by kmitočet signálu SLWR v tomto případě neměl překročit mez 8,33*MHz*. To by umožňovalo přenos obrazu v rozlišení

$$r = 52\mu s \times 8{,}33MHz = 433\,px$$

na jeden obrazový řádek. Z testů však vyplývá, že při pokojové teplotě s konkrétním použitým obvodem lze dosáhnout i vyšší zápisové rychlosti – konkrétně 15MHz, což odpovídá rozlišení 780 obrazových bodů na jeden řádek videosignálu.

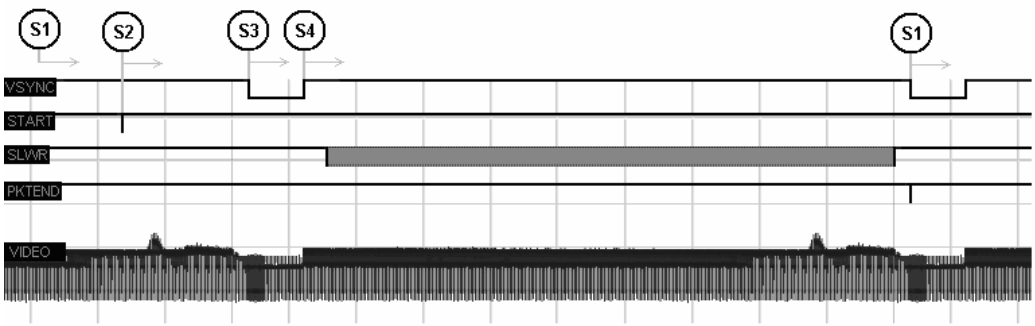
Přenos jednoho obrazového snímku je řízen obvodem CPLD, který má v sobě implementovaný stavový automat podle schématu, uvedeném v obr. 31. Po resetu se automat nachází ve stavu S1. Žádost PC o přenos snímku vygeneruje signál START, který způsobí přechod automatu do stavu S2. Od této chvíle se čeká na aktivaci a posléze deaktivaci signálu VSYNC. Tím se automat dostane přes stav S3 až do stavu S4. Pokud je automat v tomto stavu, je vnitřním čítačem hodinových pulzů čítána hodnota, která vlastně udává polohu načítaného pixelu vzhledem k levému okraji snímané scény. Pokud se hodnota čítače nalézá v předem nastaveném rozmezí, je na výstupu obvodu CPLD generován hodinový signál pro zápis dat z AD převodníku do FIFO paměti řadiče USB. Po opětovné aktivaci signálu VSYNC se generuje signál PKTEND a

dochází k přechodu automatu do výchozího stavu, kde se opět čeká na žádost o zahájení přenosu.



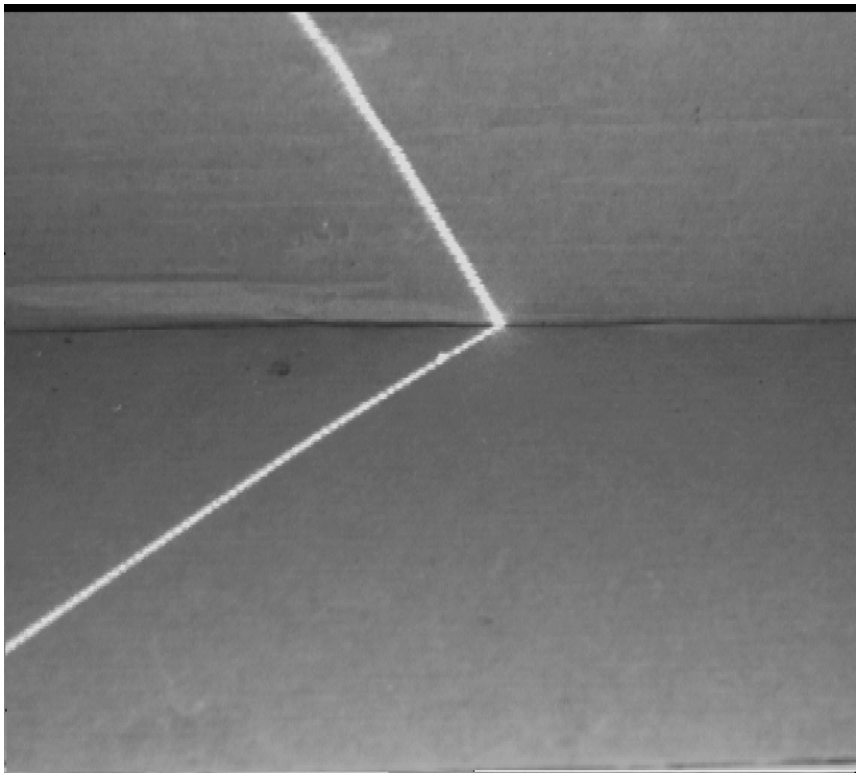
obr. 31: Schéma stavového automatu pro řízení zápisu dat do FIFO paměti

Časový průběh signálů `START`, `SLWR` a `PKTEND` v souvislosti s videosignálem a synchronizačním signálem `VSYNC` je uveden na následujícím obrázku:



obr. 32: Časové průběhy důležitých signálů při přenosu obrazu

Výpis zdrojového kódu modulu je uveden v příloze (Příloha 19: Modul pro řízení zápisu dat do paměti FIFO).



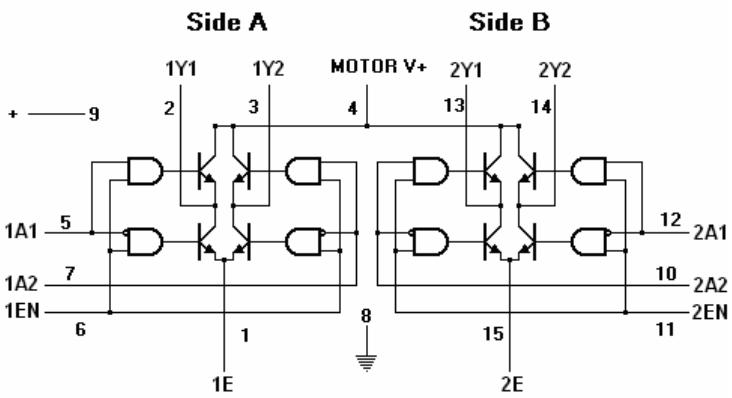
obr. 33: Příklad obrázku přeneseného po USB v rozlišení 640×576

3.2 Realizace modulu pro řízení motorů

Požadavky na modul pro řízení motorů uvedené v kapitole 2.5 vedly k navržení blokového schématu (obr. 12, část „Řízení motorů"). Podle tohoto diagramu se modul skládá ze dvou hlavních částí: řídicí a výkonové.

3.2.1 Výkonová část modulu

Výkonovým členem této desky je obvod `L298`, který obsahuje dva H-můstky (vnitřní zapojení obvodu je uvedeno v obr. 34). Každý z můstků se skládá ze dvou párů tranzistorů. Tento tranzistorový pár je schopen sepnout výstup buď na zem, napájení, a nebo ho úplně odpojit. Po připojení motoru mezi dva takto řízené výstupy jsme schopni jednoduše ovládat polaritu proudu protékajícího vinutím tohoto motoru. Navíc enablovací vstupy umožňují řízení velikosti střední hodnoty proudu pomocí `PWM`.

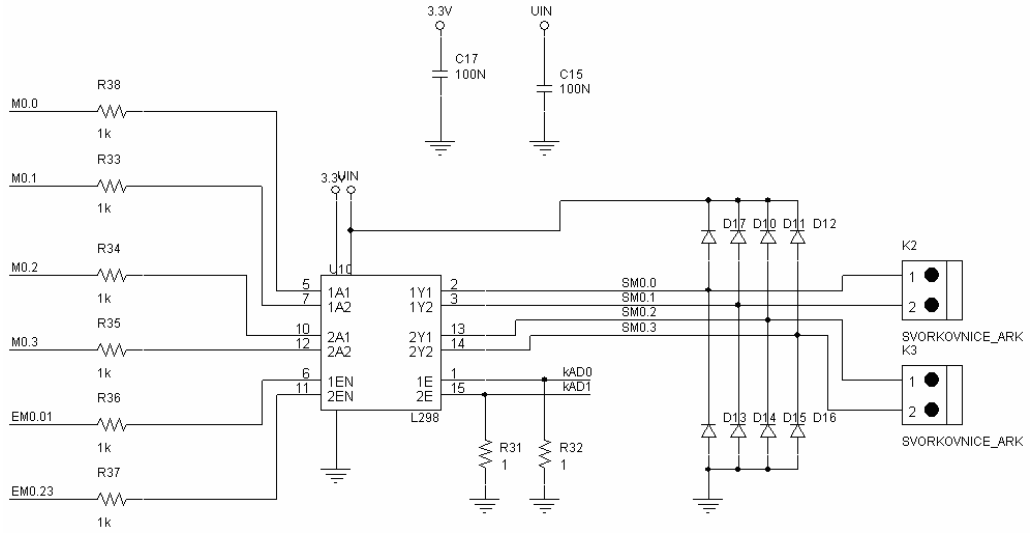


obr. 34: Vnitřní zapojení obvodu L298

Pro správnou funkci obvodu je nutné svorky označené 1E a 2E připojit na zemní vodič, a to buď přímo, nebo přes velmi malý odpor. Pro účely měření proudu bylo použito zapojení s odporem, parametry odporu jsou popsány dále (kapitola 3.2.2.3).

Obvod je ještě nutné doplnit diodami zajišťujícími ochranu proti napěťovým špičkám, které vznikají při vypnutí proudu tekoucí indukčností.

Celé schéma koncového stupně modulu pro buzení motorů je uvedeno na následujícím obrázku:



obr. 35: Zapojení koncového stupně modulu pro řízení motorů

3.2.2 Řídicí část modulu

Srdcem jednotky je mikroprocesor LPC2103 od firmy PHILIPS s jádrem ARM7. Tento mikroprocesor obsahuje řadu periferií vhodných pro realizaci tohoto úkolu, jako jsou PWM jednotky, AD převodníky či sériová linka.

Ačkoliv by počet PWM jednotek i počet kanálů AD převodníku integrovaných na čipu mikroprocesoru plně postačoval k ovládání dvou budičů L298, je na modulu nakonec použit pouze jeden tento obvod. Jednotka tedy bude umožňovat připojení jednoho krokového nebo dvou stejnosměrných obvodů, což bude pro účel diplomové práce plně postačovat.

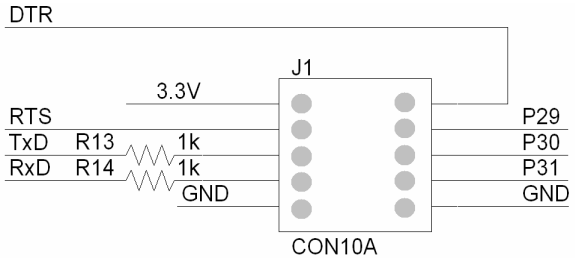
3.2.2.1 Komunikace s nadřazeným systémem

Jako hlavní zdroj informací o tom, co má modul vykonat, je nadřazený systém. Ten po sériové lince předává příkazy popsané níže (kapitola 3.2.2.5). Komunikace probíhá v režimu MASTER-SLAVE, nadřazený systém je vždy MASTER.

Konektor sériové linky obsahuje kromě signálů TxD a RxD také signál RTS, který po přivedení logické úrovně 0 resetuje celý modul a signál DTR, který je

připojen na pin procesoru P0.14. Díky těmto signálům je možné modul programovat pomocí ISP/IAP[17] např. aplikací Philips LPC2000 Flash Utility.

Dále jsou na konektor vyvedeny tři obecné vstupy/výstupy mikroprocesoru, které zatím nejsou využity, ale v případě potřeby mohou posloužit jako další signalizační nebo komunikační kanál.



obr. 36: Konektor pro propojení s nadřazeným systémem

3.2.2.2 Uživatelská tlačítka

Kromě příkazů odeslaných na sériovou linku modulu je k ovládání možné využít dvě ovládací tlačítka, pomocí kterých je možné řídit rychlost jednoho motoru. Tlačítka jsou přes malý odpor připojená na vstupy mikroprocesoru, proti zákmitům jsou chráněna jednak malým kondenzátorem přidaným k tlačítku paralelně, jednak programem, který provádí více čtení stavu tlačítka s krátkými mezerami a krátké impulsy ignoruje.

Na stisk tlačítka modul reaguje zvýšením (resp. snížením) rychlosti motoru o rychlostní jednotku.

3.2.2.3 Měření proudu vinutím motoru

Jedním z požadavků na modul je možnost měření proudu cívkou motoru. Připojením malého odporu mezi zem a svorku SENSE obvodu L298 a následným měřením napětí na tomto odporu jsme schopni vypočíst proud, který protéká cívkou motoru. Odpor musí být malý, aby nedošlo k přílišnému omezení proudu cívkou motoru, zároveň ale musí vyvolat dostatečně veliký úbytek napětí s ohledem na vlastnosti použitého AD převodníku. K měření velikosti

[17] In-System Programming a In-Application Programming umožňují programovat flash paměť zapájeného mikroprocesoru.

napětí na odporu je použit AD převodník integrovaný na čipu mikrokontroléru. Tento převodník v použitém zapojení nabízí na rozsahu 0 – 3,3V desetibitové rozlišení. Z toho vyplývá, že nejmenší měřitelné napětí U_{LSB} je:

$$U_{LSB} = \frac{3,3}{2^{10}} = 3,22mV$$

Při použití jednoohmového měřicího odporu lze tedy měřit proud protékající cívkou motoru s krokem 3,22mA až do velikosti 3,3A.

V mikroprocesoru je implementován algoritmus pro hlídání maximální úrovně středního proudu protékajícího cívkou motoru. Po překročení zadaného maxima (které je volitelné) dojde k nastavení nulového proudu vinutím daného motoru. Při vyšší zátěži motoru stoupá proud vinutím motoru a při správném nastavení lze touto funkcí docílit vypnutí motoru i v případě, že nemáme k dispozici koncové spínače a chceme proud vinutím motoru vypnout v koncové poloze mechanizmu, nebo v případě, že k ovládanému mechanizmu není omezen přístup a hrozí např. přiskřípnutí prstu.

Pro určení velikosti střední hodnoty proudu je použito průměrování ze 128 naměřených vzorků. Odběry vzorků jsou prováděny asynchronně vůči periodě `PWM` jednotky s poměrem period 30000:7552. Tento poměr odpovídá zhruba čtyřem odměrům proudu v jedné periodě `PWM` a při použitém počtu průměrovaných hodnot zaručuje rovnoměrné vzorkování v celé periodě.

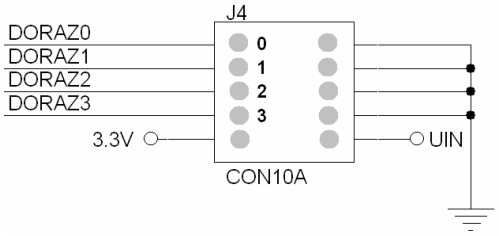
3.2.2.4 Připojení koncových spínačů

Pokud jsou na ovládaném mechanizmu k dispozici koncové spínače, je vhodné informaci z nich využít k vypnutí motoru, pokud se mechanizmus nachází v koncové poloze. Modul proto obsahuje konektor, ke kterému je možné připojit až čtyři koncové spínače (obr. 37).

Pomocí konfiguračních příkazů je možné nastavit, ke kterému motoru připojený koncový spínač patří a na jaké straně a také jakou logickou úrovní signalizuje koncovou polohu.

Na modulu jsou osazeny pull-up odpory (10$k\Omega$), zajišťující definovanou úroveň na vstupu mikroprocesoru při odpojeném koncovém spínači. Pokud tedy koncový spínač signalizuje koncovou polohu logickou „1“, je nutné zajistit, aby

v neaktivním stavu byl výstup koncového spínače v logické „0“ a nebyl odpojen. V případě signalizace koncové polohy logickou nulou stačí jednoduchý spínač spínající proti zemi.



obr. 37: Konektor pro připojení koncových spínačů, číslování pinů

3.2.2.5 Ovládání motorů mikroprocesorem

Ovládání motorů probíhá v závislosti na aktuálně nastavených parametrech a také v závislosti na stavech tlačítek, koncových spínačů a zjištěné velikosti proudu protékajícího motorem.

Při požadavku na otáčení motoru zvolenou rychlostí je nejprve na základě nastaveného typu motoru vybrána funkce pro ovládání motoru, která je posléze volána z přerušení časovače každé dvě milisekundy. Před každým zavoláním je kontrolován stav koncových spínačů a velikost proudu.

Rychlost otáčení motorem je zadávána v rychlostních pseudojednotkách. Skutečná rychlost otáčení pak závisí na konstrukci připojeného motoru a zadané rychlostní konstantě. Zadaná rychlost je násobena rychlostní konstantou, vypočtené číslo je pak postoupeno funkci pro ovládání daného typu motoru. Např. při nastavené rychlostní konstantě 500 u stejnosměrného motoru odpovídá hodnota rychlosti 15 plnému napětí na výstupu modulu. Při stejné rychlostní konstantě a zadané rychlosti a připojení krokového motoru bude tento motor posunut o 500 kroků za sekundu.

Program dále zohledňuje minimální rychlost motoru, kterou je také možné konfigurovat. Tento parametr je přičítán k zadané rychlosti motoru ještě před násobením rychlostní konstantou a je výhodné ho použít zejména při ovládání stejnosměrného motoru. U těchto motorů totiž při nižší střídě `PWM` jednotky dochází k tomu, že mají příliš malý moment a neotáčejí se.

Funkce pro ovládání stejnosměrného motoru

Funkce nejprve provede rozklad čísla reprezentujícího rychlost na znaménko a absolutní velikost. Podle znaménka provede přepnutí výstupů mikroprocesoru týkajících se daného motoru, čímž způsobí sepnutí tranzistorů v obvodu `L298` tak, aby na výstupu byla nastavena správná polarita napětí. Zároveň odpovídajícím způsobem nastaví střídu korespondující `PWM` jednotky podle následujícího vzorce:

$$s = \frac{k(r+m)}{7551},$$

kde „k" je rychlostní konstanta nastavená danému motoru, „r" je hodnota zadané rychlosti a „m" je hodnota minimální rychlosti (viz dále).

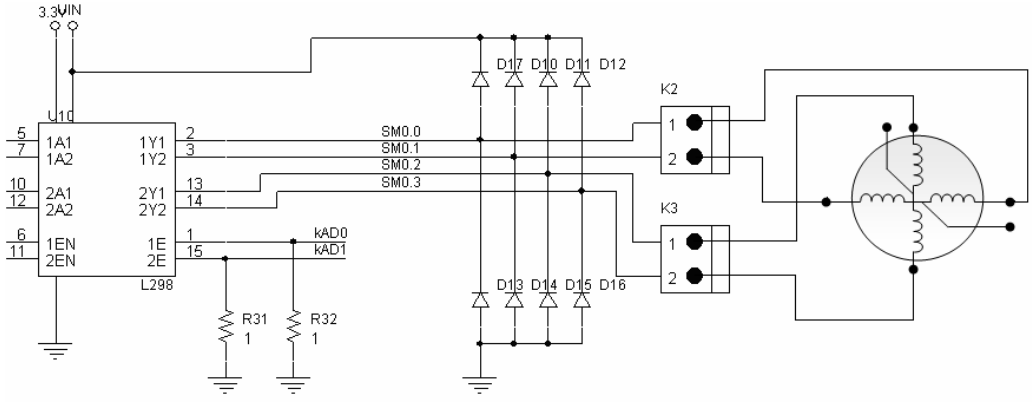
Střída je přímo úměrná zadané rychlosti. Při nastavení hodnoty 1 je tedy střída (podle hodnoty rychlostní konstanty) např. jen 5%, což nestačí na to, aby se stejnosměrný motor vůbec začal točit. K tomuto účelu je do programu přidána možnost nastavit minimální hodnotu rychlosti, která se vždy k zadané rychlosti přičte. Při nastavení minimální rychlosti např. na hodnotu 5 a rychlosti 1 je motoru nastavena rychlost 6. To by v našem příkladu odpovídalo 30% činiteli plnění `PWM` jednotky, což už je pro roztočení testovaného motoru postačující.

Výpis zdrojového kódu funkce je uveden v příloze (Příloha 16: Funkce pro ovládání stejnosměrného motoru).

Funkce pro ovládání krokového motoru

Krokový motor funguje na zcela jiném principu než motor stejnosměrný. Z toho důvodu je i jeho ovládání odlišné.

Abychom mohli měřit proud, který krokový motor odebírá, musí být motor připojen bipolárně (viz 2.5.1.2, dle obr. 38). V tomto schématu středové vodiče nejsou vůbec připojené do obvodu, a veškerý proud tedy teče přes můstkový budič.



obr. 38: Připojení krokového motoru k modulu

Důležitým prvkem ve funkci pro ovládání krokového motoru je střadač – registr, ke kterému se při každém spuštění funkce přičte aktuální rychlost. Střadač má omezenou délku a každé jeho přetečení (resp. podtečení) způsobí posun motoru na následující (resp. předchozí) krok. Číslo kroku je následně indexováním do předdefinované tabulky převedeno na stav výstupů mikroprocesoru.

Při řízení krokového motoru je třeba, aby cívkou protékal větší proud pouze v době, kdy se změnil krok, a je tedy třeba pootočit rotorem. V případě, kdy motor stojí, není třeba cívky motoru budit maximálním možným proudem. V krajním případě je dokonce možné proud vypnout úplně. Jelikož je modul schopen pomocí PWM velikost střední hodnoty proudu řídit, stará se funkce také o postupné snižování proudu v případě, že motor stojí.

Výpis zdrojového kódu funkce je uveden v příloze (Příloha 17: Funkce pro ovládání krokového motoru).

3.2.3 Ovládání modulu pro řízení motorů

Po připojení napájení nebo resetu je třeba modul nakonfigurovat podle toho, jaký typ motoru bude použit a na jakém portu navržené desky budou dostupné výstupní signály. Jak jsem již zmínil výše, mikroprocesor je schopen obsloužit dva krokové nebo čtyři stejnosměrné motory, firmware mikroprocesoru je tomuto faktu uzpůsoben. Na navržené desce je ale vyvedena pouze polovina řídicích signálů a jeden budič, takže je možné připojit pouze dva stejnosměrné

nebo jeden krokový motor. Stejnosměrné motory mohou být tedy nakonfigurovány na PORT0 a PORT1, krokový motor (vzhledem k nutnosti ovládat více vinutí) může být konfigurován pouze na PORT0.

3.2.3.1 Ovládání modulu přes sériovou linku

Veškerá konfigurace i ovládání jednotky lze provést přes sériovou linku UART. Sériová linka modulu přenáší data rychlostí 57600*bps*, má nastaveno osm datových bitů, žádnou paritu a jeden stop bit, tedy stejné parametry, jaké má nastaven výstup videoregulátoru.

Komunikační protokol je vzhledem ke snaze o co nejjednodušší zpracování velmi prostý. Po znaku „c“, který uvozuje příkaz (command), je jeden znak určující, ke kterému motoru se příkaz vztahuje, následovaný znakem identifikujícím typ příkazu. Dále je uveden řetězec obsahující parametry příkazu. Příkaz je ukončen znakem CR (0x0D). Modul na platný příkaz odpovídá znaky „ok:“, na neplatný příkaz znaky „error(E):“, kde místo znaku „E“ je uvedeno číslo chyby (viz tab. 4). Tato sekvence je následována přijatým příkazem.

Číslo chyby	Popis chyby
0	Příkaz nezačíná úvodní znakem „c“
10	Chybné číslo motoru
20	Neznámý příkaz
30	Parametr mimo povolený rozsah
31	Neznámý typ motoru
32	Neznámý typ dorazu

tab. 4: Seznam chybových hlášek modulu pro řízení motorů

Parametrem příkazu je vždy celé číslo. Číslo se odesílá v dekadickém formátu, jednotlivé cifry jsou reprezentovány ASCII znaky „0“ – „9“. Číslu může být předřazeno znaménko „+“ nebo „-“, v případě vynechání znaménka je číslo interpretováno jako kladné.

V následující tabulce je uveden seznam příkazů, které je modul schopen zpracovat. Místo znaku „X“ je v příkazu uvedeno číslo motoru v rozsahu 0 – 3.

Příkaz	Popis
cXm[typ motoru]	Nastaví typ motoru. Proměnná [typ motoru] je jeden znak, který může nabývat hodnot „k“ nebo „s“ podle toho, zda se jedná o krokový nebo stejnosměrný motor.
cXp[port]	Nastaví port motoru. Proměnná [port] může nabývat hodnot 0 – 3. Pro použitém zapojení má však význam pouze 0 – 1. Při použití stejnosměrného motoru a při použití krokového motoru je možné zadat pouze 0
cXk[rychlostni konst.]	Nastaví rychlostní konstantu motoru.
cXi[proud]	Hodnota [proud] určuje maximální povolený proud protékající motorem. Při překročení této hodnoty dojde k vypnutí motoru. Zadaná hodnota se vypočte jako skutečná velikost proudu v mA vynásobená hodnotou 40.
cXn[minRychlost]	Příkaz umožňuje nastavit minimální rychlost otáčení motorem. U stejnosměrného motoru může při nízkém činiteli plnění PWM jednotky dojít k tomu, že se motor vůbec neotáčí. Tomu lze zabránit zadáním této hodnoty.
cXs[+-][rychlost]	Kombinace proměnných [+-] a [rychlost] určuje směr a rychlost otáčení motorem. Znaménko může být „+“ a „-“, kladné znaménko je možné vynechat.
cXd[doraz][pin]	Příkaz pro konfiguraci koncových spínačů. Hodnota [doraz] uvádí typ dorazu a může nabývat hodnot „p“ nebo „n“ podle toho, zda se jedná o kladný nebo záporný doraz. Kladný doraz je aktivován při otáčení motoru kladným směrem. Hodnota [pin] udává číslo pinu (dle obr. 37), na který je koncový spínač připojen. Nastavením obou dorazů na stejnou hodnotu říkáme, že koncový spínač je společný pro obě koncové polohy. V tomto případě je potřeba zajistit, aby nebyl koncový spínač aktivován po resetu modulu.
cXds[polarita]	Příkaz definující polaritu dorazu. Polarita může nabývat hodnot 0 a 1. Při použití koncového spínače proti zemi s pull-up odpory je vstup aktivní v logické nule a je tedy třeba zadat hodnotu 0.
cXz[zpRizeni]	Příkaz nastavuje způsob řízení motoru. U krokového motoru „0“ znamená s polovičním krokem, „1“ s plným krokem jednofázově, „2“ s plným krokem dvojfázově. U stejnosměrného motoru nemá tento parametr vliv.

tab. 5: Seznam příkazů pro konfiguraci modulu pro řízení motorů

Typická sekvence pro nastavení stejnosměrného motoru na `PORT0`, s rychlostní konstantou 450, maximálním proudem 200mA a dorazy na pinech 0 a 1 má tedy následující podobu:

```
c0ms                // stejnosměrný motor
c0p0                // na portu 0
c0k450              // rychlostní konstanta 450
c0i8000             // maximální proud 200mA
c0ds0               // koncové spínače spínají na zem
c0dp0               // pozitivní doraz na pinu 0
c0dn1               // negativní doraz na pinu 1

c0s-11              // otáčet motorem rychlostí 11 v záporném směru
```

kód 1: Příklad konfigurace modulu pro řízení motorů přes UART

V zařízení nasazeném do průmyslového prostředí by bylo vhodné odeslanou zprávu nějakým způsobem zabezpečit, např. na konec zprávy přidat kontrolní součet. V laboratorních podmínkách však i nezabezpečená komunikace funguje bezchybně.

3.2.3.2 Ovládání modulu tlačítky

Na desce jsou osazena tři tlačítka. Jedno tlačítko slouží jako `RESET`, další dvě uživatelská tlačítka („Btn1“ a „Btn2“) umožňují ovládat rychlost motoru č. 0. V tomto režimu je třeba jednotku buď využívat v základní konfiguraci, nebo provést příslušnou konfiguraci přes sériovou linku.

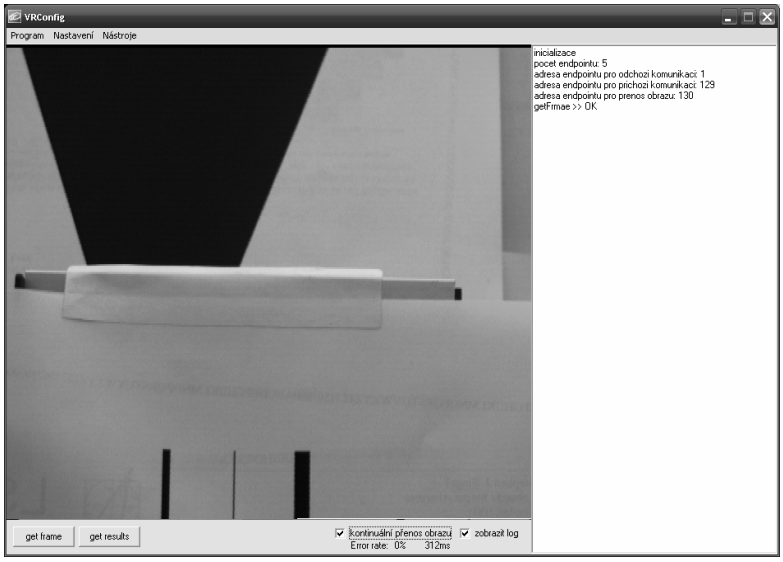
V základní konfiguraci je modul nastaven pro ovládání malého krokového motoru. Rychlostní konstanta je nastavena na 500, maximální proud na 125mA, koncový spínač společný pro obě koncové polohy je umístěn na pinu 0 a je aktivní v logické „0“.

Modul lze jednorázově přenastavit do režimu ovládání stejnosměrného motoru. K tomu je třeba stisknout jedno z funkčních tlačítek, toto tlačítko podržet a stisknout tlačítko RESET. Při stisknutí tlačítka „Btn1“ bude modul nakonfigurován pro ovládání stejnosměrného motoru na portu 0, při stisku „Btn2“ pro ovládání stejnosměrného motoru na portu 1. Minimální rychlost je nastavena na hodnotu 5. Ostatní parametry zůstanou zachovány.

3.3 Konfigurační aplikace pro PC

Konfigurační aplikace VRConfig umožňuje nastavit základní parametry modulu videoregulátoru přes USB. Dále také nabízí možnost přenosu a zobrazení scény snímané kamerou připojenou k videoregulátoru a tento obrázek případně uložit ve formátu BMP[18].

Aplikace je vytvořena ve vývojovém prostředí Borland C++ Builder 6. Pro komunikaci přes USB využívá knihovnu „CyAPI.lib“, která je dodávána s ovladačem k obvodu CYPRESS. Tento ovladač je třeba mít nainstalovaný.



obr. 39: Hlavní okno aplikace VRConfig

Po spuštění aplikace se program pokusí spojit se se zařízením. Pokud neuspěje, je vypsána informativní hláška a program je ukončen. Pokud je komunikace navázána, zobrazí se hlavní okno aplikace (obr. 39) složené ze čtyř částí:

- snímané scény,
- hlavního menu,
- loga a
- panelu s nejpoužívanějšími příkazy

[18] BMP nebo také Microsoft Windows Bitmap je souborový formát pro ukládání nekomprimované rastrové grafiky.

3.3.1 Zobrazení snímané scény

Snímaná scéna zabírá největší část hlavního okna aplikace. Ihned po navázání komunikace je zařízení vyslán příkaz „žádost o přenos snímku“, který vyvolá přenos následujícího snímku ze zařízení do PC. Pro přenos nového snímku slouží tlačítko s názvem „get frame“ a nebo checkbox „kontinuální přenos obrazu“, který žádá o nový snímek ihned po dokončení přenosu a vykreslení stávajícího snímku.

Při kontinuálním přenosu obrazu jsou na hlavním okně aplikace zobrazeny dvě hodnoty. Jedna udává počet chybně přenesených snímků, které byly detekovány v posledních sto pokusech o přenos snímku. Druhá hodnota udává, jak dlouho trval přenos a vykreslení posledního snímku.

Nasnímanou scénu je možné uložit do souboru jako obrázek `BMP`. K tomu slouží příkaz „Nástroje → Uložit obrázek“ přístupný z hlavního menu. Je tedy možné modul videoregulátoru společně s konfiguračním programem využít jako jednoduchý framegrabber.

3.3.2 Konfigurace

Hlavním cílem aplikace je však konfigurace modulu videoregulátoru. V aplikaci VRConfig je připraveno několik základních dialogových oken pro konfiguraci následující části zařízení:

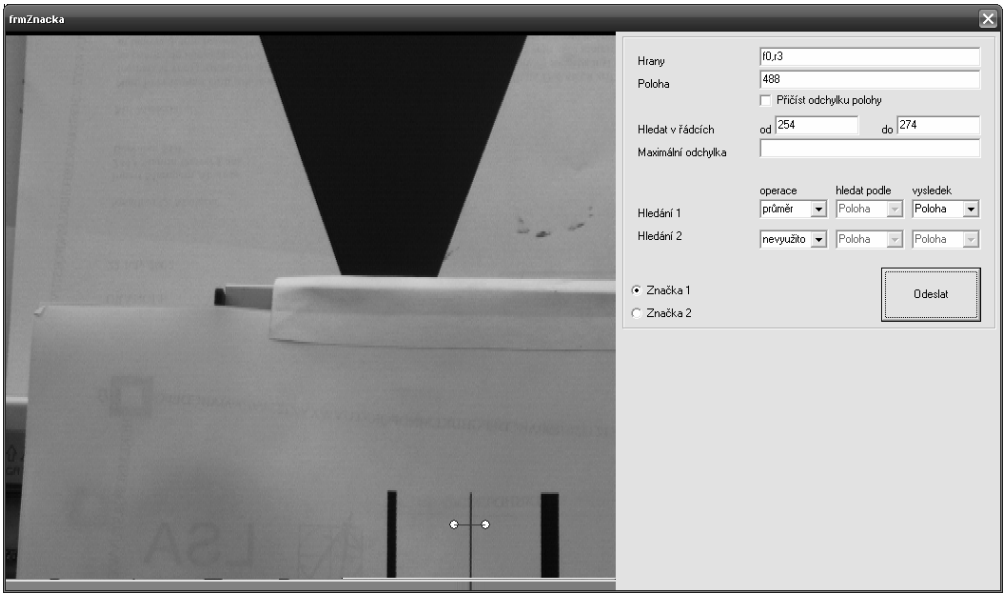
- Značky
- Regulace
- Motory

Je také možné se zařízením komunikovat pomocí terminálu dostupného pod položkou menu „Nástroje → Přímá komunikace s mikroprocesorem“ (viz 3.3.3). Nastavení každé části je přístupné z hlavního menu pod položkou „Nastavení“.

3.3.2.1 Nastavení značek

Dialogové okno pro konfiguraci značek (obr. 40) se skládá z oblasti zobrazení snímané scény, oblasti zobrazení vybrané značky a formuláře pro zadání údajů.

Značky je možné nadefinovat pouze zadáním hodnot do připraveného formuláře. V tabulce tab. 6 je popsán význam jednotlivých formulářových polí.



obr. 40: Dialog konfigurace značek

Název pole	Popis	Příklad
Hrany	Pole obsahuje definici značky. Značka je definována sadou hran určité polarity, které zaujímají definovanou vzájemnou polohu. Náběžná hrana je označena písmenem „r“, spádová hrana písmenem „f“. Za tímto písmenem následuje poloha vůči předchozí hraně (v pixelech). První hrana by měla mít nastavenou polohu 0. Jednotlivé hrany jsou odděleny čárkou.	f0, r10
Poloha	Pokud očekáváme značku na určité poloze vůči levému okraji snímané scény, je možné vyplnit pole „Poloha“ touto vzdáleností (v pixelech). Při zaškrtlém checkboxu „Přičíst odchylku polohy“ bude rozdíl polohy značky od zadané polohy započítán do odchylky značky.	300
Řádky	pole „Hledat v řádcích“ definuje dva řádky, mezi kterými bude značka vyhledávána.	od 20, do 50
Maximální odchylka	Definuje maximální odchylku, při jejímž překročení bude značka považována za neplatnou a nebude zahrnuta do procesu vyhledávání ve značkách.	100
Značka 1, 2	Tímto výběrem lze určit, zda se odeslané hodnoty týkají značky č. 1 nebo značky č. 2.	Značka 1, 2
Hledání 1, 2	V těchto řádcích se definuje vyhledávání ve značkách	

Název pole	Popis	Příklad
Hledání, Operace	Operace definuje, jaká matematická operace bude provedena se sadou výsledků nalezených hran. Je možné zvolit jednu ze čtyř možností: minimum, maximum, průměr a směrnice.	
Hledání, hledat podle	Tato položka je aktivní pouze při výměru operace minimum nebo maximum. Definuje parametr, ve kterém se hledá zadaný extrém.	
Hledání, výsledek	Definuje parametr, který bude použit jako výsledek operace. V případě minima a maxima je to hodnota zadaného parametru v řádku, ve kterém je hodnota parametru zadaná v poli „hledat podle" extrémní. V případě směrnice a průměru se z tohoto parametru vypočte zadaná operace.	

tab. 6: Popis formulářových polí v dialogovém okně „Značky"

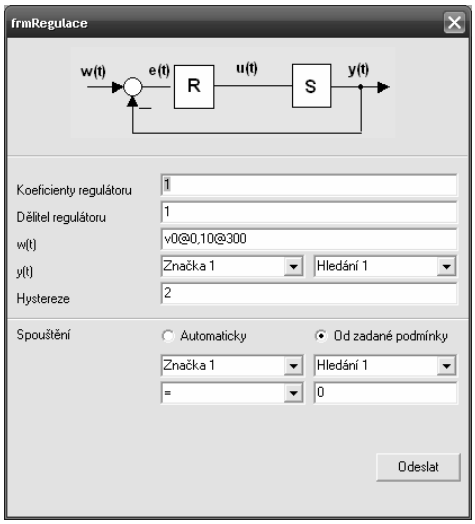
Celý proces zdlouhavého vyplňování údajů je však možné obejít pomocí „klikacího rozhraní". Pokud je na snímané scéně vidět značka, kterou chceme vyhledávat, stačí ji označit myší (stisknout tlačítko myši před značkou, posunout kurzor myši v horizontálním směru za značku a tlačítko uvolnit). Takto označená část snímku se převede na odpovídající sekvenci hran a zapíše se do příslušného pole formuláře. Zároveň se také nastaví poloha vůči levému okraji snímané scény. Rozmezí řádků lze jednoduše definovat označením řádků ve snímané scéně, které chceme prohledávat (stisknout tlačítko myši, přesunout kurzor vertikálním směrem a tlačítko uvolnit). Celé nastavení značek se tak zjednoduší na výběr, zda chceme odchylku od zadané polohy započítat do celkové odchylky a případné zadání maximální povolené odchylky.

3.3.2.2 Nastavení regulace

Dialogové okno pro nastavení regulátoru (obr. 41) umožňuje nastavit koeficienty regulátoru a dělitel výsledku (viz 3.1.6.3). Dále umožňuje přiřazení výsledků z vyhledávání konkrétním veličinám regulátoru. V neposlední řadě slouží tento dialog k nastavení spouštění sekvence žádané veličiny w. Význam jednotlivých formulářových polí je popsán v tabulce.

Název pole	Popis	Příklad
Koeficienty regulátoru	Toto pole obsahuje koeficienty PSD regulátoru q0, q1 a q2. Význam koeficientů je uveden v kapitole 2.4.3 Koeficienty jsou oddělené čárkou a mohou nabývat hodnot -32768 až 32767	60,-100,41
Dělitel regulátoru	Hodnota, kterou je dělený výsledek po výpočtu výstupu filtru	75
w	Zde se zadává požadovaná veličina. Je možné nadefinovat sekvenci požadovaných veličin, kde lze kombinovat relativní polohu vzhledem k levému okraji scény a výsledky naměřené videosenzorem. Jednotlivé údaje obsahují číslo (resp. odkaz na výsledek) a čas, ve kterém se má hodnota použít. Čas je zadáván v násobcích 20ms a od hodnoty je oddělen znakem „@“. Všechny hodnoty jsou pak odděleny čárkou. Odkaz na výsledek je možné zadat ve formátu „v00“. Dvojice čísel říká, o jaký výsledek se jedná. Jsou použity kombinace 00, 01, 10 a 11.	v10@0,500@200
y	V tomto poli se zadává, který z výsledků vyhledávání bude přiřazen měřené veličině y.	
Hystereze	Hystereze definuje rozmezí, ve kterém je měřená veličina považována za stejnou jako požadovaná veličina. Hodnota je udána v pixelexh.	3
Spouštění	V části „spouštění“ je možné nastavit podmínku, po jejímž splnění dojde k opakování sekvence požadované veličiny (viz 3.1.6.3). Podmínka je definována odkazem na výsledek, porovnávacím znaménkem a hodnotou, se kterou je výsledek porovnáván. Je také možné zvolit automatické spouštění, které zaručí, že sekvence požadovaných veličin se bude neustále opakovat.	

tab. 7: Popis formulářových polí v dialogovém okně „Regulace“



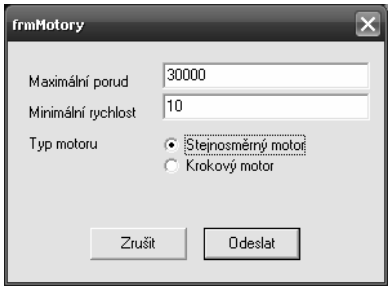
obr. 41: Dialog konfigurace regulátoru

3.3.2.3 Nastavení motoru

Pro konfiguraci typu a vlastností motoru připojeného k modulu pro řízení motorů slouží dialog konfigurace motoru. V tomto dialogu nalezneme následující formulářová pole:

Název pole	Popis	Příklad
Maximální proud	Pole, ve kterém se nastaví maximální proud, který je povolený pro daný typ motoru. Při překročení zadaného proudu dojde k vypnutí motoru. Proud je zadáván v násobcích 40mA.	
Minimální rychlost	Minimální rychlost je hodnota použitelná zejména pro stejnosměrný motor, kde vlivem nízké hodnoty PWM modulátoru dojde k tomu, že by se motor měl točit, ale netočí se. Zadáním tohoto minimální rychlosti lze tento efekt eliminovat.	
Typ motoru	Lze vybrat, zda je k modulu připojen krokový nebo stejnosměrný motor.	

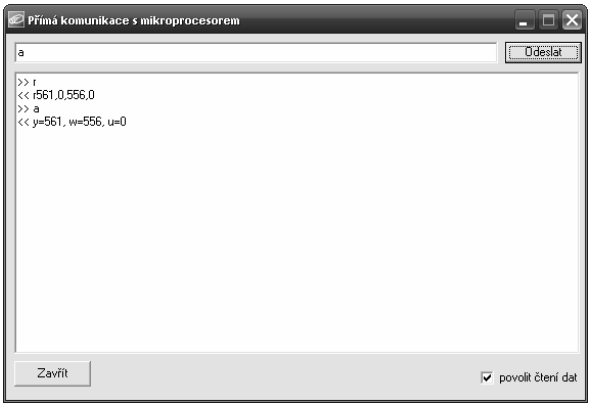
tab. 8: Popis formulářových polí v dialogovém okně „Motor“



obr. 42: Dialog konfigurace motoru

3.3.3 Přímá komunikace s mikroprocesorem

Aplikace nabízí také jednoduchý nástroj pro přímou komunikaci s mikroprocesorem (obr. 43). Tento nástroj je využitelný zejména při ladění, protože je možné přímo napsat příkaz, který je odeslán do mikroprocesoru. Zároveň je vidět, jaký řetězec mikroprocesor vrátil. Pokud je zobrazeno okno přímé komunikace s mikroprocesorem, jsou všechny příkazy posílané z mikroprocesoru do aplikace přesměrovány do výpisového okna, a nedojde tedy k jejich zpracování aplikací.



obr. 43: Okno přímé komunikace s mikroprocesorem

Seznam dostupných příkazů je uveden v tab. 3.

4 Výsledky práce

4.1 Zachycení hran

Správná funkce videoregulátoru je podmíněna korektním zachycením detekovaných hran. Cílem proto bylo vyvinout takový systém, který bude schopen zachytit hrany umístěné ve snímané scéně velmi blízko sebe. Zachycení detekované hrany mikroprocesorem se zajišťuje pomocí dvojice `CAPTURE` jednotek; každá jednotka slouží k zachycení hran jedné polarity. Uložení hrany do paměti RAM mikroprocesoru proto zahrnuje následující úkony:

- vyvolání přerušení na základě události na `CAPTURE` jednotce,
- zjištění, jaká `CAPTURE` jednotka vyvolala přerušení, a tedy o jaký typ hrany jde,
- uložení polohy hrany z `CAPTURE` registru fronty hran,
- inkrementace ukazatele na poslední hranu,
- test, zda nedošlo k přetečení fronty hran a
- návrat do hlavní programové smyčky.

Celá tato sekvence trvá přibližně 2,6μs.

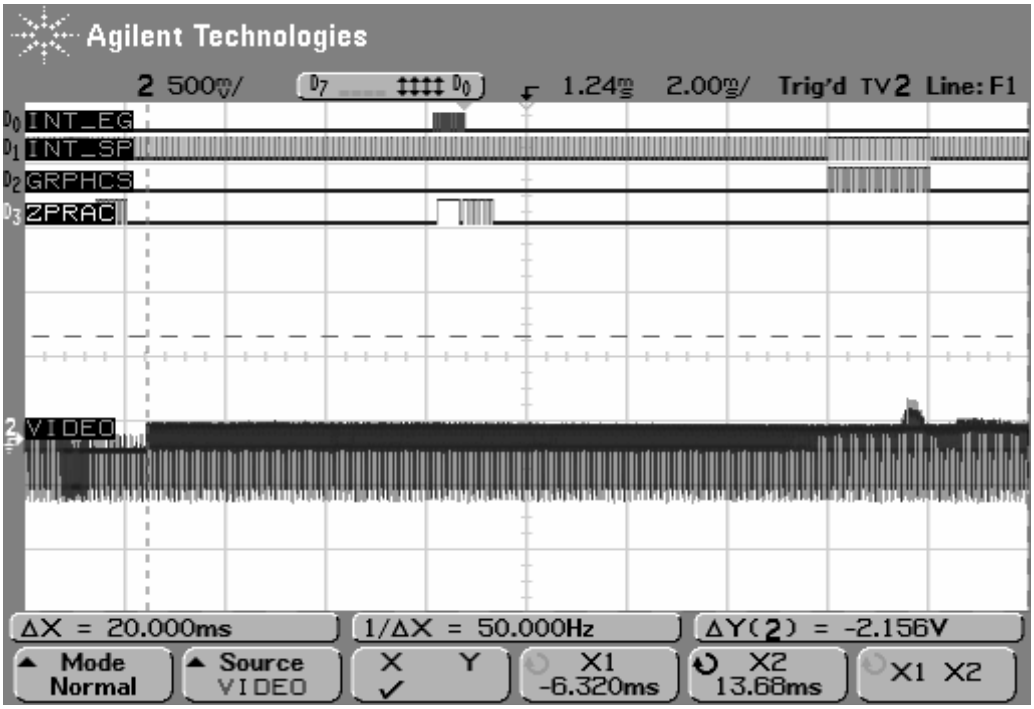
Z výše uvedeného je zřejmé, že systém je schopen zachytit dvě hrany bezprostředně vedle sebe, pokud jsou opačné polarity. Dvě hrany stejné polarity musí být pro korektní zachycení ve videosignálu vzdáleny minimálně 2,6μs, což odpovídá rozlišení 38 pixelů. Při ideálním rozložení hran je tedy možné zachytit až 40 hran na řádek. To je však pouze teoretické maximum; snímaná scéna by musela obsahovat jeden pixel široké kontrastní proužky se vzájemnou vzdáleností 38px. Běžně dosažitelné maximum je přibližně 20 hran na řádek.

Tento způsob zachycování hran je vhodný například pro zachycení tenké laserové stopy promítnuté na jednobarevné stínítko, což by odpovídalo situaci nastíněné v aplikačním příkladu.

4.2 Nalezení značek a výpočet výsledků

Hledání značek v zachycených hranách je také velice důležitou částí v procesu celkového zpracování redukované obrazové informace. Rutina pro zpracování obrazu je volána z hlavní programové smyčky, má tedy nejnižší prioritu zpracování a je přerušována jak zachycováním hran, tak synchronizačními pulzy videosignálu. Délka zpracování tedy závisí nejen na počtu přijatých hran a počtu hran, kterými je definována značka, ale také na době, po kterou mikroprocesor stráví v přerušení.

Na obr. 44 je vidět časový diagram v jednom půlsnímku videosignálu. Signál `INT_EG` zobrazuje dobu, kdy dochází k načítání hran. Na signálu `ZPRAC` je vidět doba, po kterou jsou načtené hrany zpracovávány. Signál `INT_SP` udává dobu, po kterou mikroprocesor strávil v přerušení od synchronizačních signálů. Konečně signál `GRPHCS` zobrazuje časový úsek, kdy dochází k vykreslování grafiky a také není možné provádět žádné zpracování.



obr. 44: Délka trvání zpracování

Během tohoto testu bylo v jednom řádku videosignálu načítáno 16 hran se střídající se polaritou a byla měřena délka zpracování jednoho řádku. Hrany

byly měřeny v řádcích 117 – 127. Doba zpracování byla měřena pro značky definované dvěma a čtyřmi hranami, u každé nalezené značky byl proveden výpočet průměrné polohy značky vůči levému okraji snímané scény a čísla řádku s minimální odchylkou nalezené a definované značky.

Zpracování jednoho řádku při značce definované dvěma hranami trvalo 59μs, při značce definované čtyřmi hranami trvalo 64μs. Uvážíme-li, že doba trvání aktivní části videosignálu je 52μs a doba potřebná pro zachycení 16 hran je přibližně 25μs, zjistíme, že na zpracování zbývá pouze 27μs, což pro zpracování nestačí. Výpočet je tedy dokončen v řádcích, kde se neprovádí měření a nedochází ani k přerušení od příchozích hran.

Tento způsob zpracování tedy vede k tomu, že nelze měřit v celé části obrazu, ale v závislosti na počtu přijatých hran pouze v určitém úseku obrazu. Zároveň je třeba mít v paměti mikroprocesoru připravenou dostatečně velikou frontu pro uložení hran. Během testování modulu se ukázalo, že pro většinu aplikací postačí fronta s možností uložení 512 hran.

4.3 Regulace

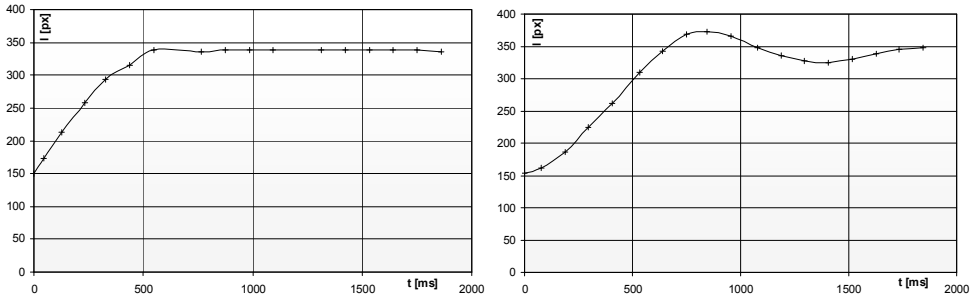
V modulu videoregulátoru je implementován algoritmus pro výpočet akčního zásahu ze známé regulační odchylky podle rovnice diskrétního regulátoru v přírůstkovém tvaru:

$$u(k) = q_0 e(k) + q_1 e(k-1) + q_2 e(k-2) + u(k-1).$$

Regulovaný systém nemá známé parametry, v každé aplikaci může být použito jiné polohovací zařízení používající od těch nejpomalejších krokových motorů s velmi jemným krokem až po poměrně rychlé stejnosměrné motory. Nelze tedy předem určit koeficienty regulátoru a je na uživateli, aby tyto koeficienty pomocí aplikace VRConfig nastavil.

Při řešení diplomové práce nebyl kladen důraz na návrh regulátoru pro konkrétní systém. Během ladění aplikace a testů bylo naměřeno několik časových průběhů znázorňujících odezvy systému na skokovou změnu požadované veličiny s různě nastavenými regulátory (obr. 45). Tyto průběhy uvádím pouze pro ilustraci.

Při návrhu regulátoru je třeba pamatovat na to, že perioda, s níž se aktualizují výsledky měření (a tedy i „perioda vzorkování“ použitá při návrhu regulátoru) je 20*ms*. Tato perioda je dána zpracovávaným videosignálem, v němž přenos jednoho půlsnímku trvá právě 20*ms*, a informace ze snímku tedy nelze získat častěji.



obr. 45: Příklad odezvy systému na skokovou změnu požadované veličiny

4.4 Přenos videosignálu přes USB

Přenos videosignálu z modulu videoregulátoru do PC je realizován po sběrnici USB v nekomprimované podobě. Použitý obvod CY7C68013A-56 nabízí v osmibitovém asynchronním režimu, který je pro přenos použit, teoretickou maximální rychlost zápisu dat do paměti FIFO přibližně 8*MB/s*. Při testech se však ukázalo, že v podmínkách laboratoře videometrie a s konkrétním použitým obvodem lze tuto rychlost až dvojnásobně překročit. Prototyp modulu videoregulátoru i konfigurační aplikace VRConfig jsou tedy nastaveny pro přenos obrazu v rozlišení 640×576*px*, přičemž levý a pravý okraj snímané scény v šířce 70*px* nejsou přenášeny. Toto rozlišení si žádá zápis do FIFO paměti s kmitočtem 15*MHz* a datovou propustnost USB přibližně 9,2*MB/s*.

U sériové výroby modulu by však jistě nebylo vhodné překračovat parametry součástky uvedené výrobcem, a proto by bylo třeba rychlost přenosu snížit na povolenou mez. Jelikož je hodinový signál pro zápis do paměti FIFO obvodu CYPRESS generován obvodem CPLD, lze v hradlovém poli jednoduše implementovat děličku kmitočtu dvěma. Horizontální rozlišení přenášeného

obrazu by bylo poloviční a nároky na přenosovou rychlost by se také o polovinu snížily.

Příklad obrázku přeneseného po USB v rozlišení 640×576 je uveden v kapitole 3.3.1.

4.5 Vykreslování grafiky do videosignálu

Možnost zobrazit výsledky měření a regulace v průmyslovém prostředí vedla k realizaci bloku pro vkládání grafiky do videosignálu. Ten umožňuje zobrazit nejdůležitější výsledky na běžném monitoru se vstupem kompatibilním s normou CCIR.

Stávající verze firmware mikroprocesoru umožňuje zobrazit grafiku na TV monitoru v rozlišení 256×32*px*, přičemž udávané rozlišení neodpovídá rozlišení videosignálu signálu, ale je zhruba poloviční (tj. jeden pixel vkládané grafické informace zabírá přibližně dva pixely na televizní obrazovce).

Vkládání připravené bitmapy do videosignálu je proces náročný na časování a na rychlost a žádá si plný výpočetní výkon mikroprocesoru. Rutina pro vložení je tedy volána z přerušení FIQ, které je vyvoláno s velmi malou chybou časového posunu vůči videosignálu a má nejvyšší prioritu. Z výše uvedeného je zřejmé, že během vkládání grafiky do videosignálu nelze provádět zachycení hran ani žádné další výpočty.

Časování a spouštění vkládání obrazové informace do videosignálu je znázorněno na obr. 14.

5 Závěr

Cílem práce bylo navrhnout a realizovat mnohořádkový videoregulátor pro základní bezdotyková měření polohy a rozměru objektu a následnou regulaci podle naměřených hodnot. Zdrojem dat pro měření je redukovaná obrazová informace. Pro videoregulátor byly zadány následující parametry:

- zdrojem obrazové informace je monochromatická CCD kamera s výstupem dle normy CCIR,
- předzpracování obrazové informace je provedeno hranovým detektorem implementovaným v obvodu CPLD,
- jako řídicí jednotka je použit mikroprocesor řady ARM 7,
- spolupráce videoregulátoru s PC je realizována pomocí sběrnice USB 2.0 high speed,
- je implementováno zobrazování výsledků měření na TV monitoru.

Při řešení této diplomové práce bylo dosaženo následujících výsledků.

Na základě analýzy dané problematiky byl řešený úkol rozdělen do dvou částí – části pro zpracování a vyhodnocení snímané scény a části pro ovládání motorů. Každá část diplomové práce vedla k realizaci jednoho modulu. Tyto moduly je možné navzájem spojit a vytvořit tak jeden regulační systém.

Modul pro zpracování a vyhodnocení snímané scény zpracovává načtenou informaci v několika krocích.

Prvním krokem je digitalizace videosignálu. S digitalizací je velmi úzce spojen blok generování hodinového signálu, který by měl být co nejlépe synchronizován s videosignálem. Bylo dosaženo konstantní doby reakce na externí přerušení mikroprocesoru, odchylka od této konstantní doby činila nejvýše 17*ns*. To je zároveň doba periody hodinového signálu jádra mikroprocesoru a tedy také teoretická mez, které lze dosáhnout.

V dalším kroku je provedena hranová detekce digitalizovaného signálu přírůstkovou metodou v obvodu CPLD a zachycení detekovaných hran capture jednotkou mikroprocesoru ARM. Díky použití dvou nezávislých capture kanálů lze zachytit dvě hrany s opačnou polaritou, které jsou na snímaném obrazu těsně

vedle sebe. Dvě hrany stejné polarity musí být pro úspěšné zachycení vzdáleny nejméně 38*px*.

Dalším krokem je prohledání zachycených hran v definovaných řádcích za účelem nalezení definované značky, následný výpočet souhrnných výsledků a přiřazení těchto výsledků jednotlivým vstupům regulátoru.

Posledním krokem pak je ze znalosti vstupních veličin vypočíst akční zásah regulátoru a tuto hodnotu odeslat v definovaném formátu po sériové lince.

Vzhledem k povaze signálu `CCD` kamery je možné získat naměřené výsledky jednou za 20*ms* v případě, že výsledky počítáme v každém půlsnímku.

Modul pro ovládání motorů má za úkol na základě informace přijaté z nadřazeného systému řídit motor připojený k modulu. Motor může být stejnosměrný nebo krokový, a to až do celkového odběru 2*A* na jednu fázi. Modul je schopen kontrolovat stav jednoho nebo dvou koncových spínačů přiřazených motoru a zároveň sledovat, zda nedošlo k překročení maximálního proudu protékajícího motorem.

Domnívám se, že cíle stanovené v úvodu diplomové práce a všechny body zadání byly splněny. Výsledky práce ukazují, že po dokončení vývoje videosenzoru bude možné použít zařízení v praxi.

6 Seznamy

6.1 Seznam použité literatury

[1] Dobruský, J.: Kompaktní videosenzor. Diplomová práce ČVUT-FEL, Praha 2007

[2] Česák, P.: Autonomní videoprocesor pro zpracování videosignálu a řízení osvětlovačů. Diplomová práce ČVUT-FEL, Praha 2005

[3] Souček, P.: Použití rozhraní USB2.0 pro rychlý přenos obrazu. Bakalářská práce ČVUT-FEL, Praha 2007

[4] Fischer, J.: Optoelektronické senzory a videometrie. Skripta ČVUT, Praha 2002

[5] Internetové stránky výrobce procesorů řady LPC 21xx - http://www.keil.com

[6] Internetové stránky výrobce USB řadiče - http://www.cypress.com

[7] Internetové stránky výrobce CPLD - http://www.xilinx.com

[8] http://www.wikipedia.org/

[9] http://www.robotika.cz/

6.2 Seznam obrázků

6.3 Seznam tabulek

6.4 Seznam zdrojových kódů

6.5 Seznam příloh

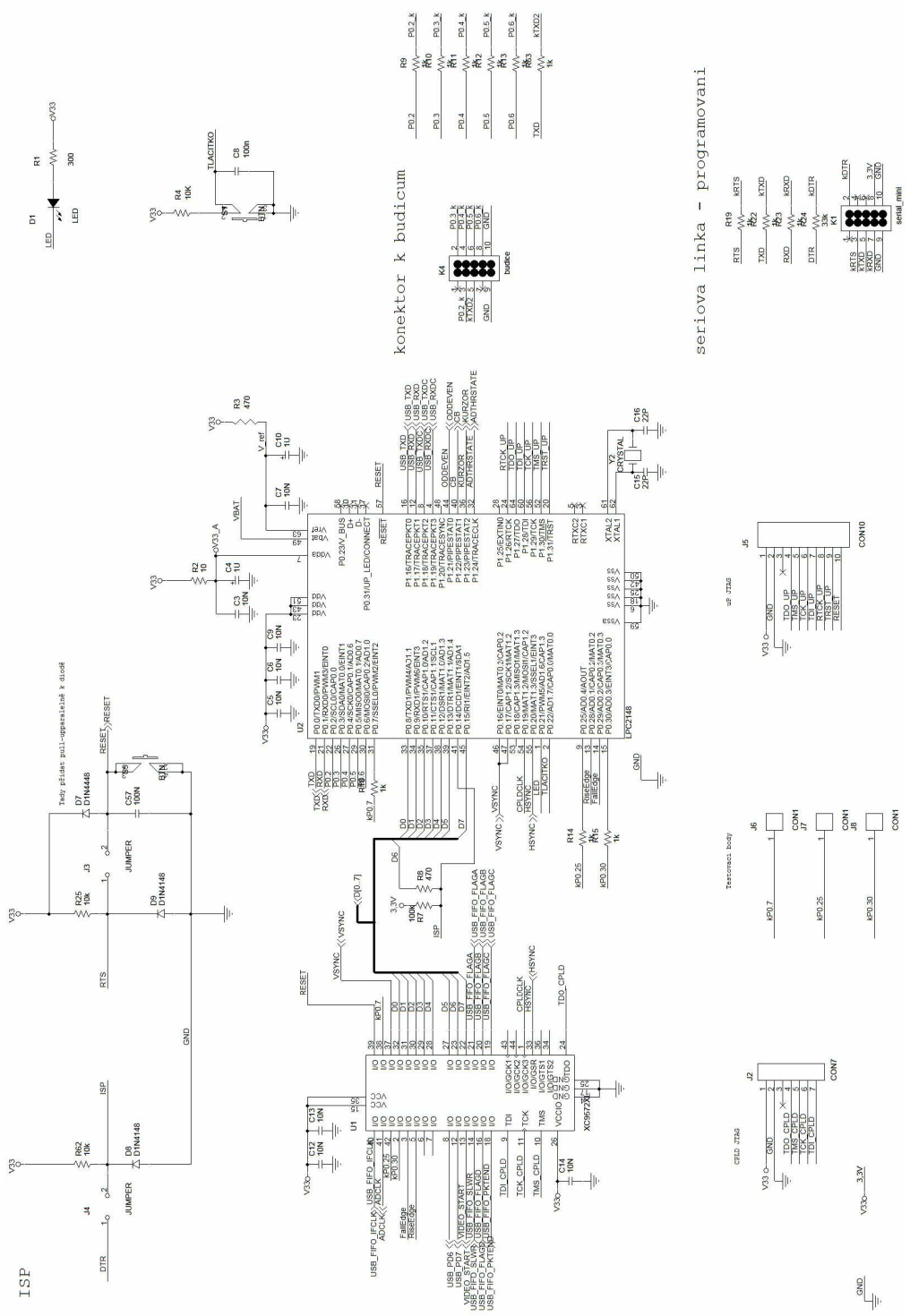
7 Přílohy

Chyba	Nepřekřížené signály sériové linky UART na konektoru pro spojení s modulem pro řízení motorů
Navrhované řešení	Opravit ve schématu i v návrhu DPS
Chyba	Obrácená polarita blokovacího kondenzátoru C27
Navrhované řešení	Je třeba tento kondenzátor zaletovat na obráceně, než je uvedeno v podkladech.

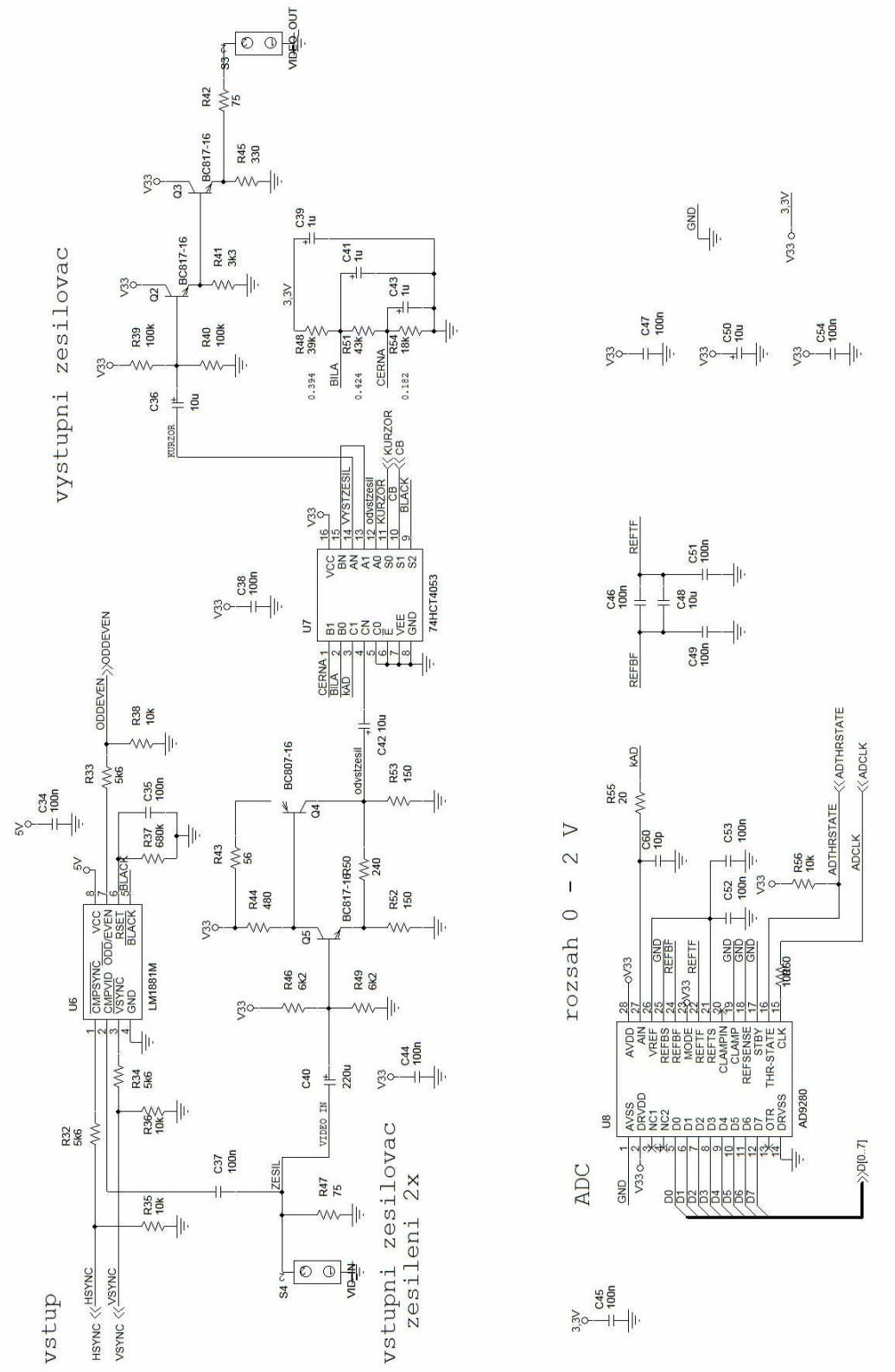
Příloha 1: Tabulka chyb v modulu videoregulátoru

Chyba	Nepřipojený vstup napěťové reference AD převodníku na mikroprocesoru
Navrhované řešení	Opravit ve schématu i v návrhu DPS
Chyba	Špatné pouzdro pro tranzistory
Navrhované řešení	Opravit návrh DPS
Chyba	Špatné pouzdro pro měřicí odpory.
Navrhované řešení	Opravit návrh DPS. Bylo by vhodnější použít větší pouzdro s většími otvory.
Chyba	Malé vrtací otvory pro diody.
Navrhované řešení	Opravit návrh DPS.

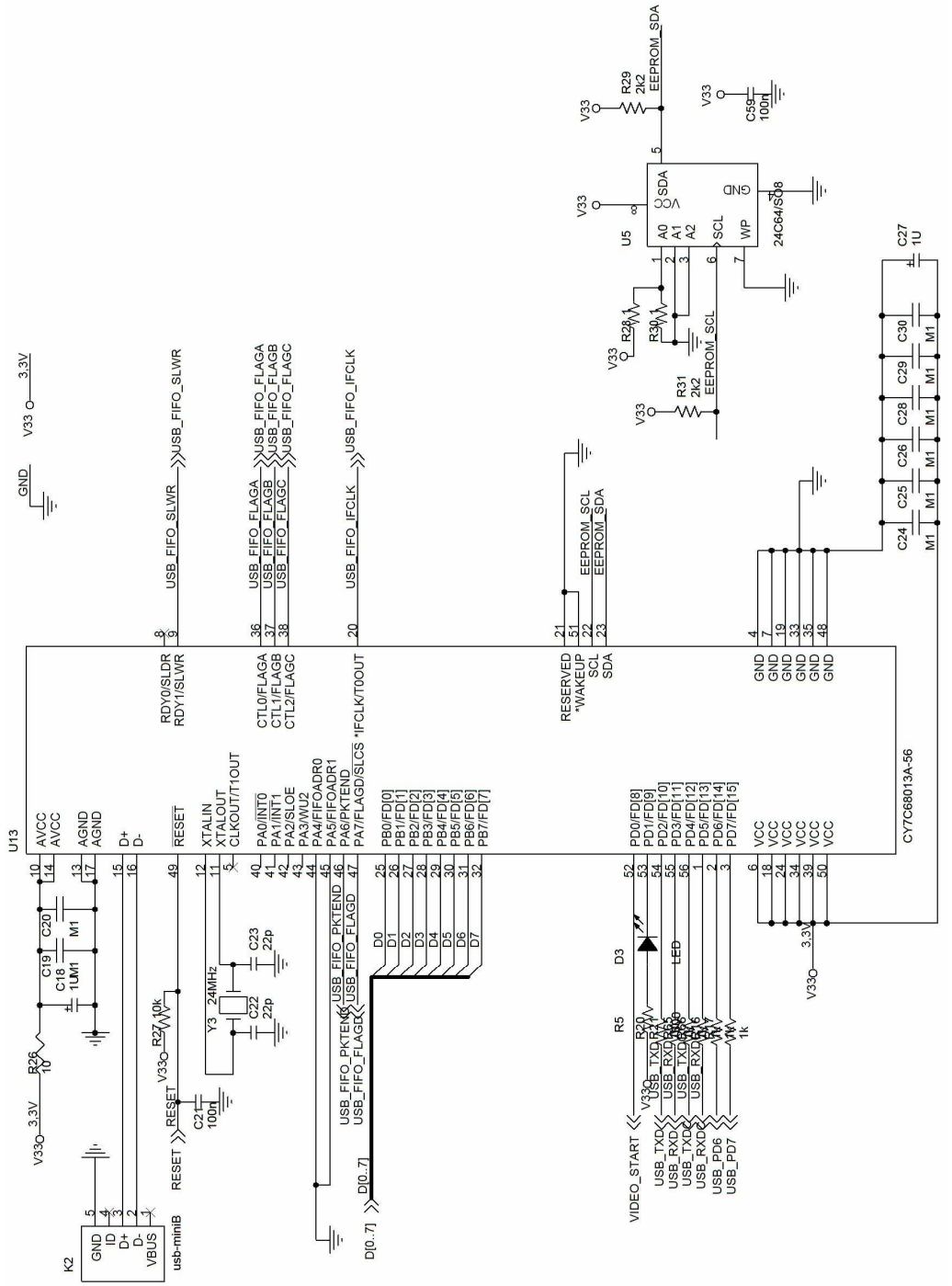
Příloha 2: Tabulka chyb v modulu pro řízení motorů



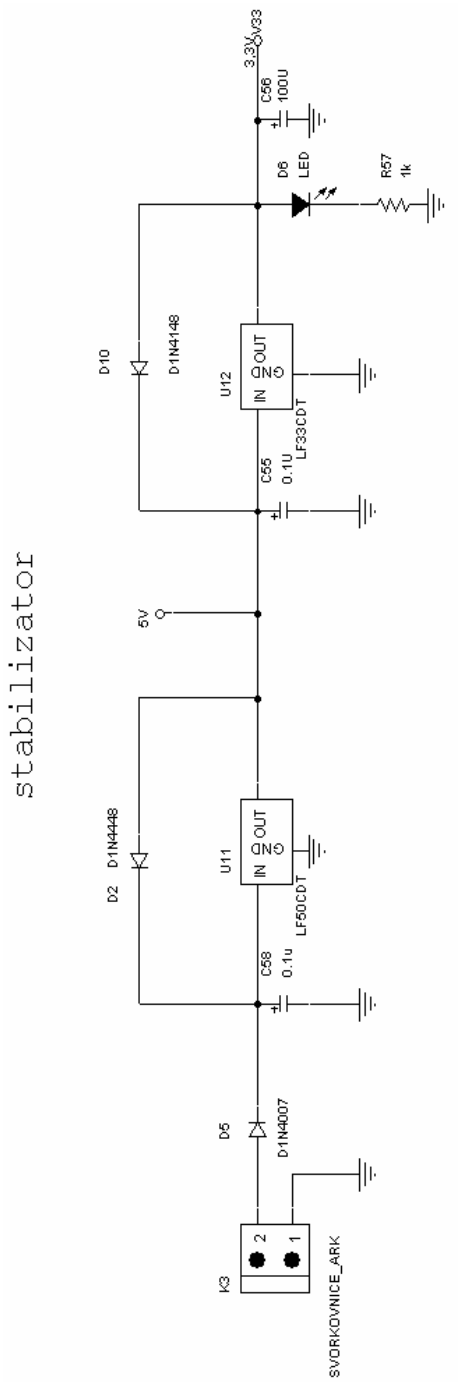
Příloha 3: Videoregulátor, mikroprocesor a CPLD



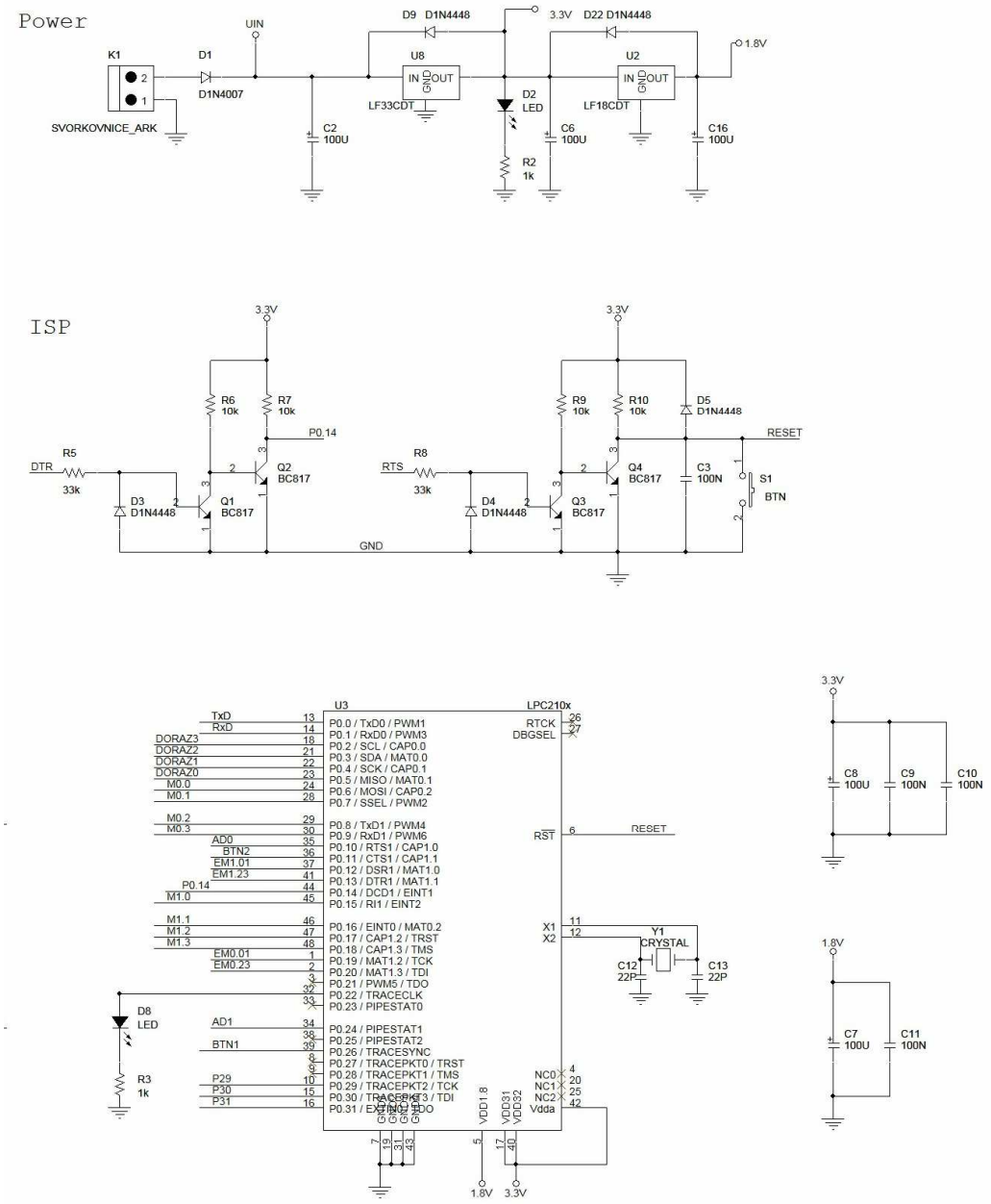
Příloha 4: Schéma – Videoregulátor, digitalizace obrazu



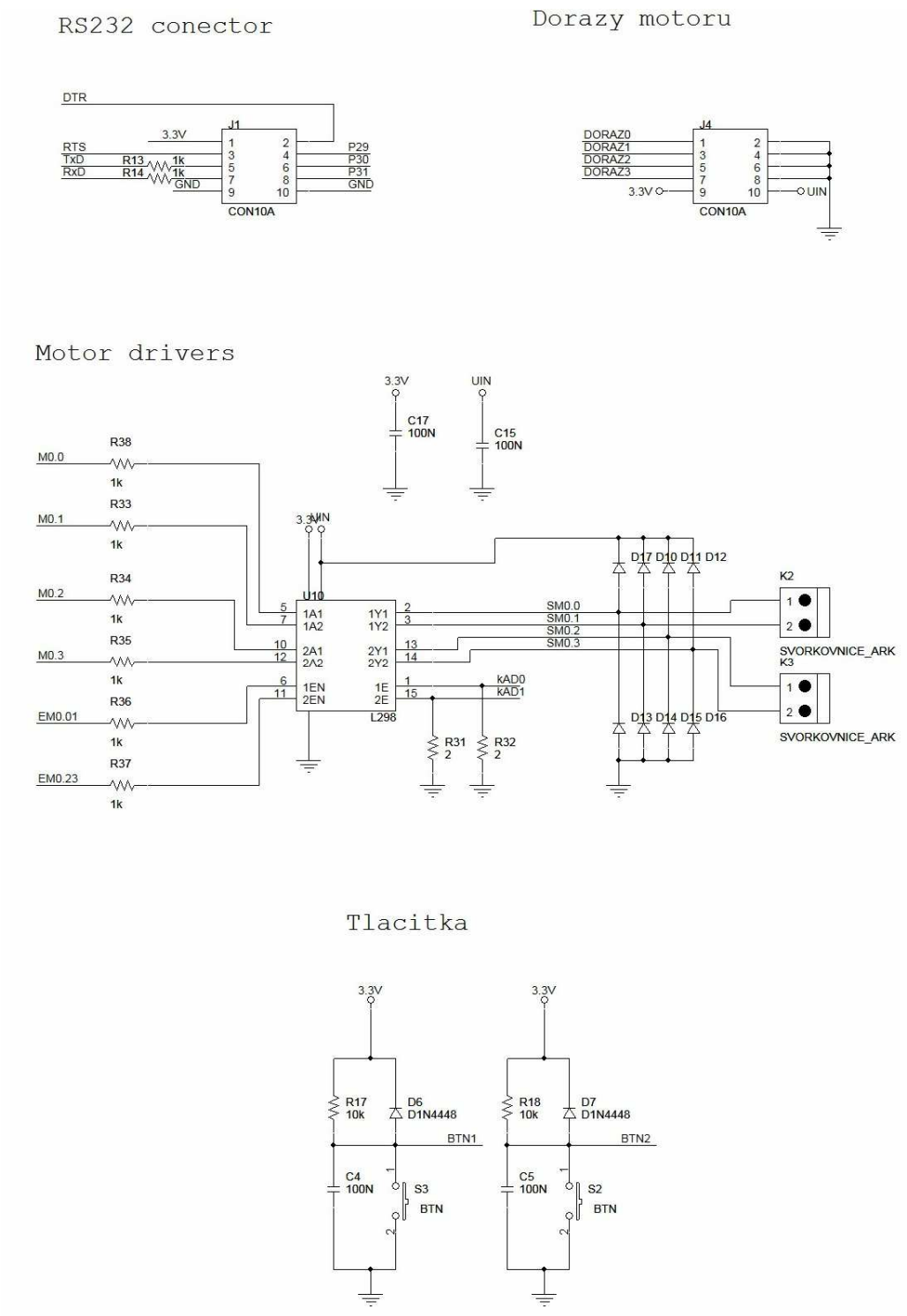
Příloha 5: Schéma – Videoregulátor, řadič USB



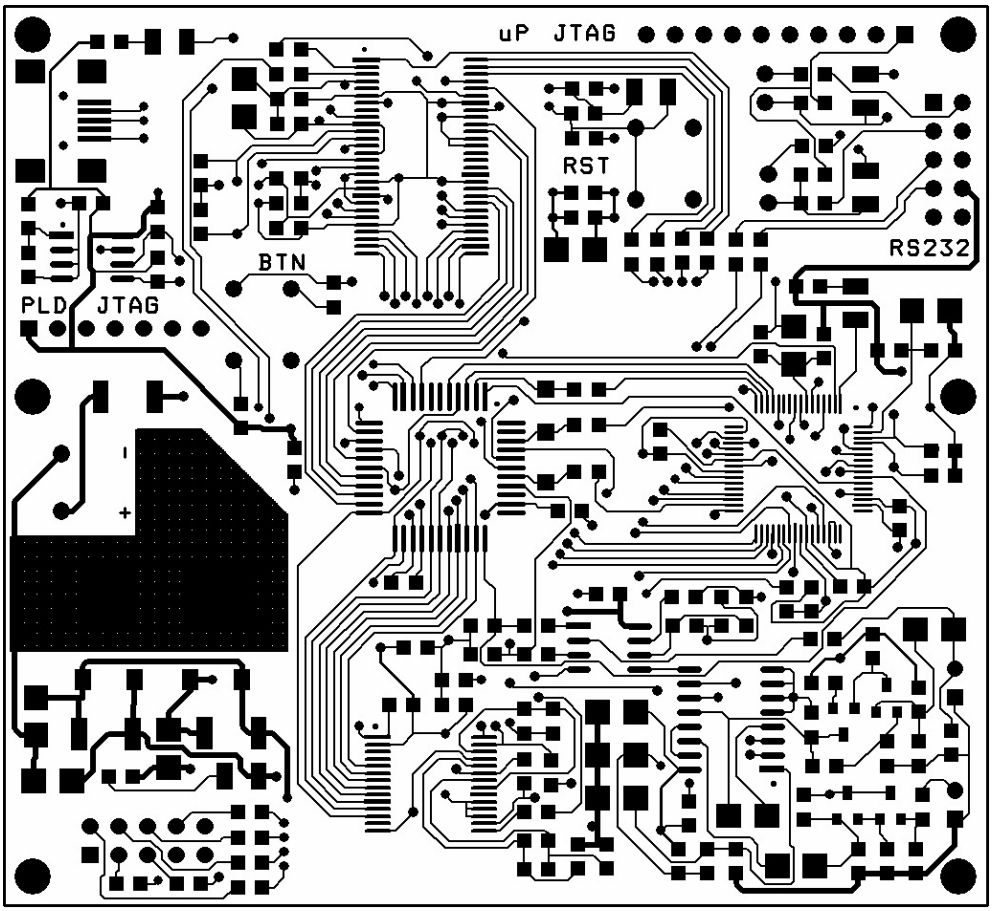
Příloha 6: Schéma – Videoregulátor, zdroj napájení



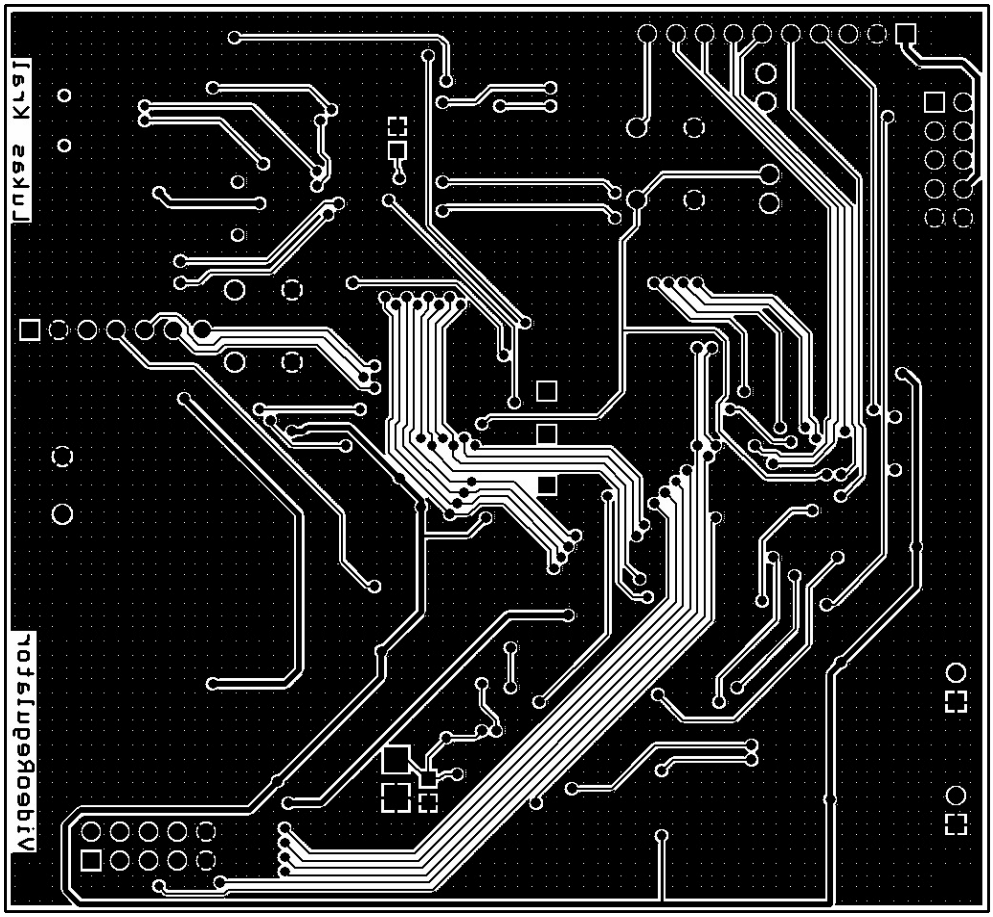
Příloha 7: Schéma - Modul pro řízení motorů, část 1



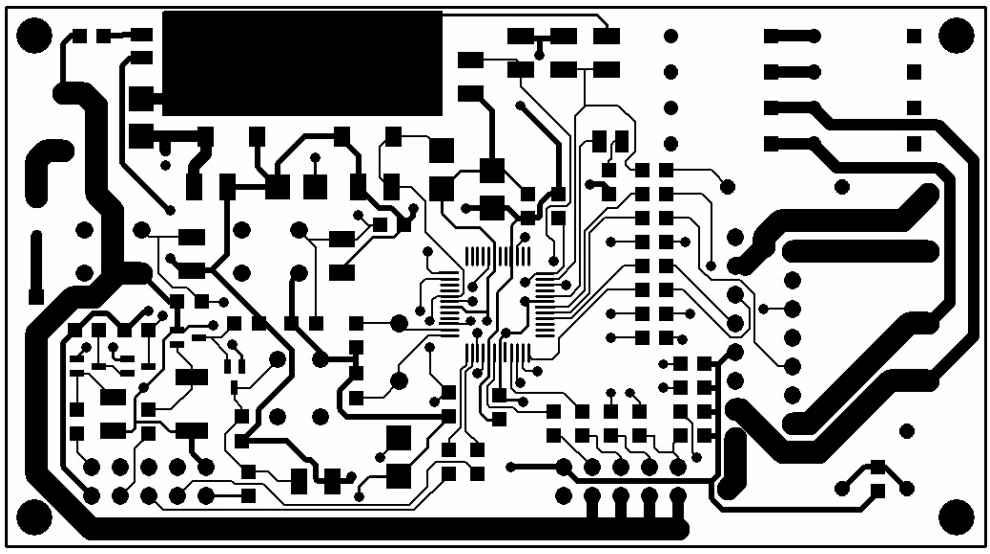
Příloha 8: Schéma - Modul pro řízení motorů, část 2



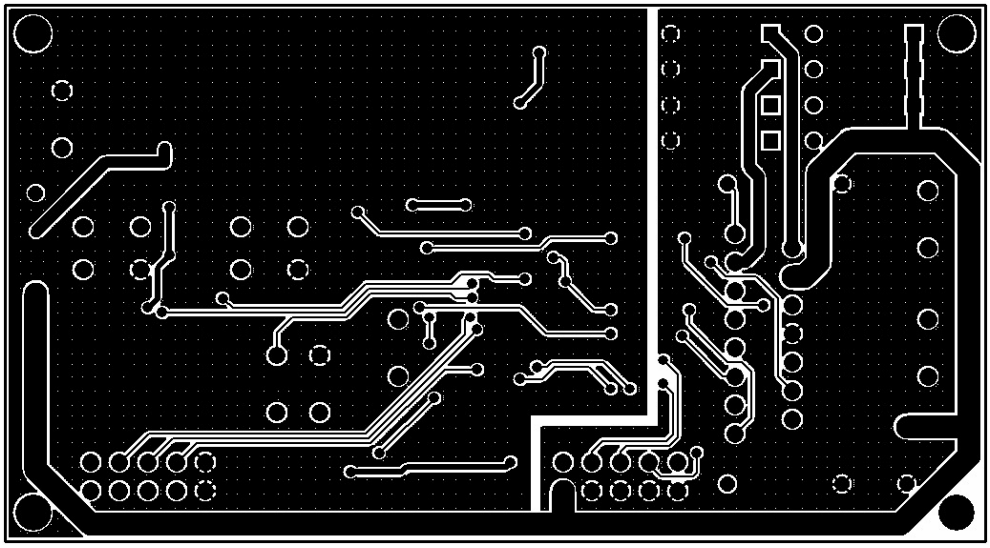
Příloha 9: Deska plošného spoje – Videoregulátor, TOP



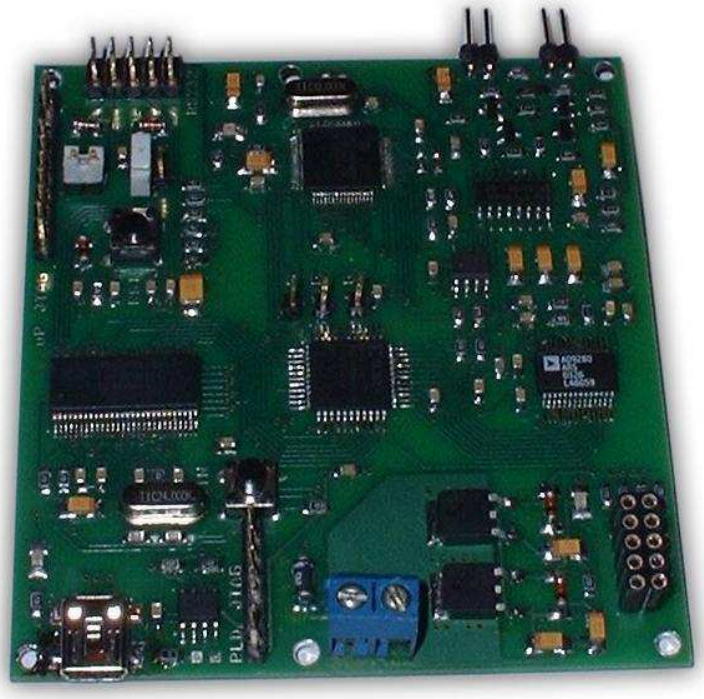
Příloha 10: Deska plošného spoje – Videoregulátor, BOTTOM



Příloha 11: Deska plošného spoje - Modul pro řízení motorů, TOP



Příloha 12: Deska plošného spoje - Modul pro řízení motorů, BOTTOM



Příloha 13: Fotodokumentace modulu videoregulátoru

Příloha 14: Fotodokumentace modulu pro ovládání motorů

```
void grZobraz( unsigned char linenr )
{
      char i,j;                              // citace
      char px4;                              // prave zpracovavany byte
      unsigned int ukaz = linenr*64;         // ukazatel do graficke pameti
      unsigned long FIO1MASKbck = FIO1MASK; // zalohovani registru FIO1MASK

      FIO1MASK = ~( 3 << 22 );       // nastaveni masky pro zapis na branu
      for (i = 0; i < 10; i++) ;     // cekani na aktivni oblast videosignalu

      for (i = 0; i < 64; i++)       // 64x4 zobrazitelne pixely na radku
      {
            px4 = grMem[ukaz];       // promenna obsahuje hodnotu 4 sousednich px

            FIO1PIN = px4 << 22;   // nastaveni ridicich signalu podle g.pameti
            __asm("nop;nop;nop;"); // cekani
            FIO1PIN = px4 << 20;   // nastaveni ridicich signalu podle g.pameti
            __asm("nop;nop;nop;"); // cekani
            FIO1PIN = px4 << 18;   // nastaveni ridicich signalu podle g.pameti
            __asm("nop;nop;nop;"); // cekani
            FIO1PIN = px4 << 16;   // nastaveni ridicich signalu podle g.pameti

            ukaz++;
      }
      FIO1PIN = 0;                   // prepnout na videosignal
      FIO1MASK = FIO1MASKbck;        // obnoveni puvodni hodnoty registru
}
```

Příloha 15: Funkce pro vkládání grafiky do videosignálu

```
void setStMotorSpeed(char i)
{
	unsigned int r = abs(motor[i].rychlost)*motor[i].rychlostnikonstanta;
	signed char znamenko;
	if (r>getMaxPwmRate())
	{
		r=getMaxPwmRate();
	}
	if (motor[i].rychlost>0) {
		znamenko = 1;
	} else if (motor[i].rychlost==0) {
		znamenko = 0;
	} else {
		znamenko = -1;
	}

	// vystup bude na piny podle definovaneho portu
	switch (motor[i].port)
	{
		case 0:
			if (znamenko>0) {
				IOSET0 = 1<<6; // zapne p0.6
				IOCLR0 = 1<<7; // vypne p0.7
			} else if (znamenko<0) {
				IOCLR0 = 1<<6; // vypne p0.6
				IOSET0 = 1<<7; // zapne p0.7
			} else {
				IOCLR0 = 1<<6; // vypne p0.6
				IOCLR0 = 1<<7; // vypne p0.7
			}
			set_pwm2(getMaxPwmRate()-r);
			break;
		case 1:
			if (znamenko>0) {
				IOSET0 = 1<<8; // zapne p0.8
				IOCLR0 = 1<<9; // vypne p0.9
			} else if (znamenko<0) {
				IOCLR0 = 1<<8; // vypne p0.8
				IOSET0 = 1<<9; // zapne p0.9
			} else {
				IOCLR0 = 1<<8; // vypne p0.8
				IOCLR0 = 1<<9; // vypne p0.9
			}
			set_pwm3(getMaxPwmRate()-r);
			break;
		default:
			break;
	}
}
```

Příloha 16: Funkce pro ovládání stejnosměrného motoru

```
char kroktable[3][8] = {
      { 0x08, 0x0A, 0x02, 0x06, 0x04, 0x05, 0x01, 0x09 },
      // s polovicnim krokem
      { 0x08, 0x02, 0x04, 0x01, 0x08, 0x02, 0x04, 0x01 }
      // s plnym krokem, jednofazove
      { 0x0A, 0x0A, 0x06, 0x06, 0x05, 0x05, 0x09, 0x09 }
      // s plnym krokem, dvojfazove
                        };

void setKrMotorSpeed(char i)
{
      signed int plus;

      plus = motor[i].rychlost * motor[i].rychlostnikonstanta;
      if (plus > 10000)
      {
            plus = 10000;  // „plus“ nesmi byt vetsi nez delka stradace
      }
      if (plus < -10000)
      {
            plus = -10000; // „plus“ nesmi byt vetsi nez delka stradace
      }
      krokprescalor[i] += plus;


      // bude krok na dalsi polohu
      while (krokprescalor[i] > 10000) {
            krokprescalor[i] -= 10000;
            krok[i] += 1;
            krok[i] &= 0x07;
            krokpwm[i] = 0;
      }

      // bude krok na predchozi polohu
      while (krokprescalor[i] < 0) {
            krokprescalor[i] += 10000;
            krok[i] -= 1;
            krok[i] &= 0x07;
            krokpwm[i] = 0;
      }

      // vystup bude na piny podle definovaneho portu
      switch (motor[i].port)
      {
            case 0:
                  // vystupy p0.6 - p0.9
                  IOSET0 = (  kroktable[motor[i].zpr][krok[i]] & 0x0F ) << 6;
                  IOCLR0 = ( ~kroktable[motor[i].zpr][krok[i]] & 0x0F ) << 6;
                  set_pwm2( krokpwm[i] );
                  set_pwm3( krokpwm[i] );
                  break;
            default:
                  break;
      }

      if (krokpwm[i] > getMaxPwmRate())
      {
            krokpwm[i] = getMaxPwmRate();
      }
      else
      {
            krokpwm[i] += 60;      // rychlost zmensovani PWM rate
      }

}
```

Příloha 17: Funkce pro ovládání krokového motoru

```
entity EdgeDetector is
    Port ( DataIn : in std_logic_vector(7 downto 0);
           clk : in std_logic;
           FallEdge : out std_logic;
           RiseEdge : out std_logic);
end EdgeDetector;

architecture Behavioral of EdgeDetector is

signal lastDataIn1: STD_LOGIC_VECTOR (7 downto 0); -- registry pro ulozeni
signal lastDataIn2: STD_LOGIC_VECTOR (7 downto 0); --   starsich hodnot jasu

signal rozdil: STD_LOGIC_VECTOR (7 downto 0);      -- registr pro ulozeni
                                                   --   diference 2 hodnot

begin

process (clk, DataIn)
begin
   if clk='1' and clk'event then                   -- pri kazde nabezne hrane
                                                   --   hodin
             rozdil <= DataIn - lastDataIn2;       -- vypocet rozdilu jasu

             lastDataIn2 <= lastDataIn1;           -- uchovani 2 minulych hodnot
             lastDataIn1 <= DataIn;

             if ((rozdil>20) and (rozdil<=127)) then
                                                   -- nalezena nabezna hrana
                    riseEdge <= '1';
             else
                    riseEdge <= '0';
             end if;

             if ((rozdil>127) and (rozdil<235)) then
                                                   -- nalezena spadova hrana
                    fallEdge <= '1';
             else
                    fallEdge <= '0';
             end if;
   end if;
end process;

end Behavioral;
```

Příloha 18: Modul přírůstkového hranového detektoru

```
auto_1: process (clk, reset)
begin

      if reset = '0' then
             auto1 <= S2;
             pktend <= '1';
      elsif rising_edge(clk) then
      case auto1 is
      when s1 => -- signály jsou nastavené do neaktivní úrovně, čeká na start=0
             pktend <= '1';
             if start = '0' then
                    auto1 <= s2;
             else
                    auto1 <= s1;
             end if;

      when s2 => -- čeká na konec probíhajícího snímku
             if vsync = '1' then
                    auto1 <= s2;
             else
                    auto1 <= s3;
             end if;

      when s3 => -- čeká na začátek platného snímku
             if (vsync = '1') and (hsync = '1') then
                    auto1 <= s4;
             else
                    auto1 <= s3;
             end if;
      when s4 => -- při platných datech hradluje pclk na slwr
             if (vsync = '0') or (fflag = '0') then
                    auto1 <= s1;
                    pktend <= '0';
             end if;
      when others =>
             auto1 <= s1;
      end case;
      end if;
end process;

-- vystup hodin
mux1: process (auto1, hsync)
begin
      case auto1 is  -- Pokud je automat ve stavu s4, na vystup je pusten
                     -- hodinovy signal, jinak je tam stále log. 1

             when s4 =>
                    fifowr  <= pclk;

             when others =>
                    fifowr  <= '1';
      end case;
end process;
```

Příloha 19: Modul pro řízení zápisu dat do paměti FIFO