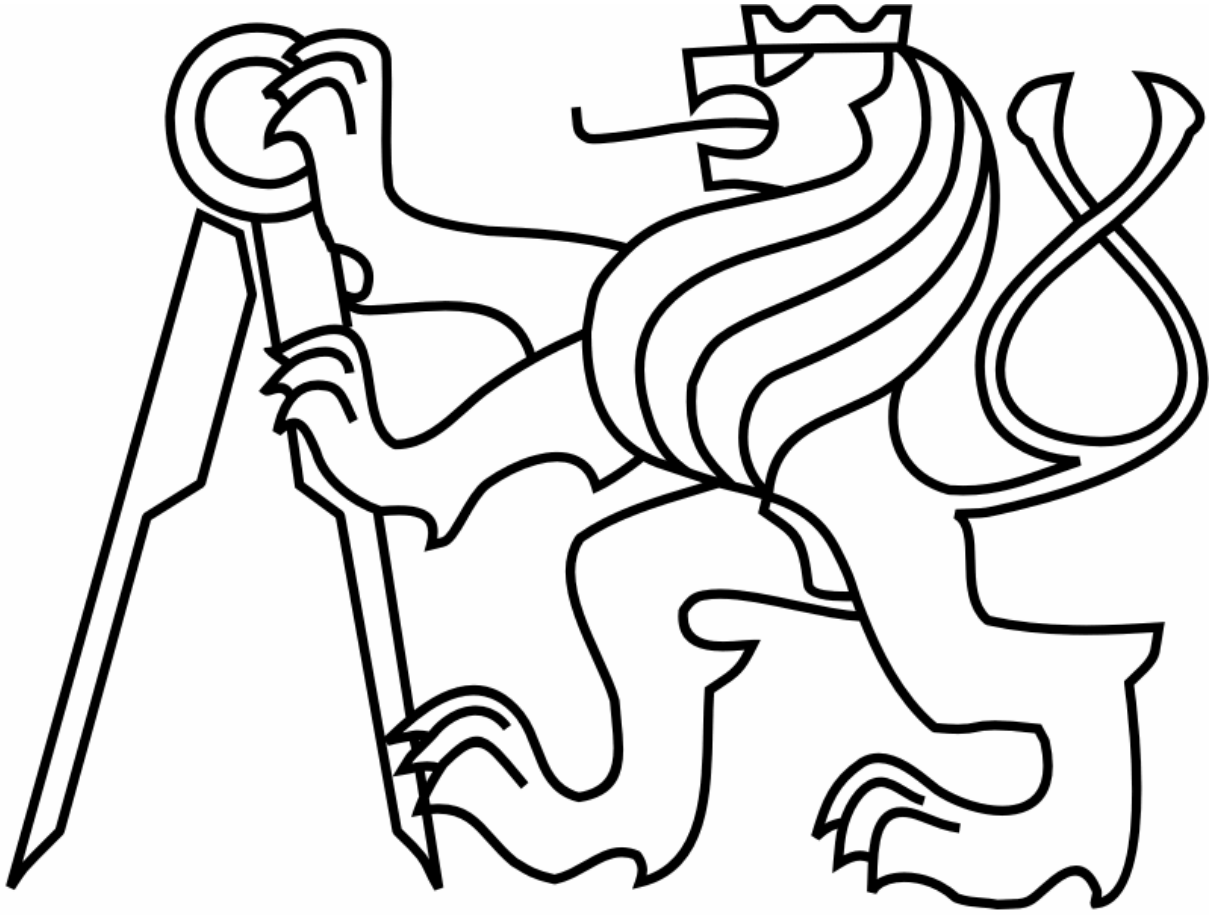
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ
FAKULTA ELEKTROTECHNICKÁ



DIPLOMOVÁ PRÁCE

2008

Metody průběžného zpracování obrazu a jejich implementace do signálového procesoru Blackfin ADSP-BF532

Vedoucí práce:
Ing. Jan Fischer CSc.

Diplomant:
Tomáš Pavlíček

Anotace

Metody průběžného zpracování obrazu a jejich implementace do signálového procesoru Blackfin ADSP-BF532. Zpracování obrazu po jednotlivých řádcích pro určování polohy a rozměru objektu bez pomoci externí paměti a se zaručenou dobou odezvy systému. Metoda je vhodná pro spojité snímání „nekonečného“ pásového obrazu.

The methods of continuous image proccesing and their implementation to signal proccesor Blackfin ADPS-BF532. The image proccesing round odd lines for determination a position and a measurement of object without an external memory and with guaranteed time of a system. The method is available for a continuous scanning of „infinite“ a full-track image.

Čestné prohlášení

Prohlašuji, že jsem zadanou diplomovou práci zpracoval samostatně a používal jsem literaturu v této diplomové práci uvedenou. Dále prohlašuji, že nemám námitek pro půjčování nebo zveřejňování mé diplomové práce nebo jejich částí a využití výsledků této diplomové práce ČVUT – fakultou elektrotechnickou se souhlasem katedry měření.

Datum: ……. Podpis diplomanta: …….

Poděkování

Děkuji všem pracovníkům katedry měření, především Ing. Janu Fischerovi, Csc. za vedení a odborné konzultace a korekci tohoto textu. Také bych rád poděkoval mé rodině za velkou podporu a trpělivost během celého mého dlouhého studia.

OBSAH

SEZNAM OBRÁZKŮ:

SEZNAM TABULEK:

1 ÚVOD

1.1 Úvod do problematiky

V dnešní době se ve výrobě používá stále více obrazové kontroly jako jednoho z mnoha způsobů bezdotykového měření. Hlavní výhodou bezdotykového měření je, že minimálně ovlivňuje měřenou soustavu. Jeden z oboru bezdotykového měření se nazývá videometrie. Videometrie se mimo jiné také zabývá měřením ifnormace získané z obrazového senzoru. Použité senzory jsou CCD či CMOS, jednořádkové či plošné. Plošné (2D) senzory mají definované rozlišení v obou snimací směrech, naproti tomu jednořádkové senzory pouze v jednom snímacím směru. Ten druhý není definován. Chceme-li jednořádkový senzor použít ke snímaní určité scény (plochy), rozlišení druhého snímacího směru záleží pouze na snímacím kroku. Takže lze ve výsledku pomocí jednořádkového snezoru získat lepší snímek s větším rozlišením, než z plošného snímače, avšak za předpokladu, že se pohybuje jen jedna část soustavy, buď jednořádkový snímač, či snímaná scéna.

Zpracovávat takto získaná data lze mnoha způsoby, jako například signálovým procesorem či prgogramovatelným polem (CPLD, FPGA). Způsob zpracování záleží na tom, jak k pořízeným datům přistupujeme. Přistupovat lze jednořádkově či víceřádkově – plošně. Výhodou jednořádkového přístupu je podstatně menší potřeba úložného prostoru pro nasnímaná data a možnost zpracovávat data průběžně, v reálném čase oproti plošnému přístupu. Nevýhodou jednořádkového přístupu je omezení použitelných metod zpracování obrazu oproti plošnému přístupu, u kterého lze jen težko či vůbec data zpracovávat v reálném čase v dostatečném rozlišení.

Z pospané bilance jednořádkového a plošného přístupu plyne použitelnost pro daný aplikační problém. Či-li snahou této diplomové práce bude nalezení metod, které umožní průběžné zpracování obrázků pomocí jednořádkového přístupu k nasnímaným datům. Protože průběžným zpracováním obrázků nelze docílit takové výpočetní kvality, jako je klasické řešení, kdy se provádějí obrazové metody nad souhrnem celých obrázků, bude se muset najít kompromis mezi efektivitou, rychlostí a kvalitou zpracování. Je tedy jasné, že tyto metody najdou uplatnění především v průmyslu k bezdotykovým měřením a kontrolám, hlavně na dopravnících a pásech. Tedy kde je hlavním kritériem rychlost zpracování obrazu a malá operační paměť, je výhodné použít metody s jednořádkovým přístupem s vědomím, že nelze použít robustní metody zpracování obrazu.

Výsledkem této diplomové práce bude snímací systém, který bude k pořízeným datům přistupovat jednořádkově a implementované metody budou schopny určit polohu a plochu snímaných objektů. Jako signálový procesor je použit procesor BF532 od firmy Analog Devices, který zpracovává nasnímaná data a řídí přenos mezi PC a snímací deskou. Dále je v CPLD obvodu pomocí VHDL jazyku naprogramován generátor synchronizačních signálů, který generuje potřebné signály určené pro CCD snímač, AD převodník a PPI rozhraní procesoru BF532. Dále snímací deska obsahuje obvod pro přenos dat přes USB rozhraní s obovdem Cypress pracujícím v režimu high speed.

Následující kapitoly budou postupně popisovat řešní této diplomové práce. Hned ve druhé kapitole se budu zabývat jednotlivými metodami detekce a nalezení objektů. Bude zde popsána jejich funkce a aplikační vhodnsot pro dané řešení problému. Dále také omezení a paměťové či výpočetní nároky. Nebudou samozřejmě chybět vývojové diagramy, které představu fungování navržené a implemetované metody lépe objasní.

Ve třetí kapitole popíší realizovanou desku z pohledu postupu videosignálu, čili od senzoru, přes AD převodník, CPLD až do procesoru, a konečně poslání výsledků přes USB do PC. Bude zde popis jak hardwareového tak i programového řešení návrhu s doporučenými nastaveními.

Ve čtvrté kapitole shrnu naměřené výsledky a zhodnotím použité metody detekce a hledání objektů. Také uvedu garantovanou dobu trvání a odezvy použité metody potažmo sytému. V závěru této kapitoly ukáži výsledky v podobě souřadnic nalezených objektů a obrázek nasnímaného objektu.

V závěru se zamyslím nad celkovou použitelností a vhodností mnou navrženého řešení. Také naznačím případné alternativami řešení stejného problému.

V příloze si lze potom prohlédnout schémata a plošné spoje navržených desek, popřípadě jejich fotografie. Také popíší ovládání navržené aplikace, která sprostředkovává komunikaci uživatele a snímací desky přes rozhraní USB.

2 TEORETICKÝ ROZBOR

V této kapitole budou jednotlivě uvedeny metody pro předzpracování a zpracování obrazu a dále jejich popis, použití a vhodnost pro daný aplikační problém. Předzpracováním je míněna především filtrace pořízeného snímku. Filtrace by měla být obecně zahrnuta v předzpracování nejen obrazové informace. Dále budou představeny hranové detekční metody, které jsou v předzpracování použity.

Poté popíši mnou navržené metody hledání objektů. Určité metody umožnují zpracování po jednotlivých řádcích, například výpočet plochy objektu. Nicméně pro správnou iterpretaci objektu, který je zpracováván po jednotlivých řádcích, je nutné zajistit souvislost nasnímaného objektu. Aby se zajistilo, že nasnímaný řádek, respektive hodnoty jasu, patří objektu, je nutné tedy nejdříve zajistit souvislost dat, či-li zjistit relaci sousedtství. Níže popsané postupy ukáží, jak zajistit souvislot, identifikaci a správnou interpretaci objektu, nad jehož naměřenými hodnotami jasu budou prováděny metody zpracování obrazu, jako například plocha a určení težiště atd.

2.1 Metody pro předzpracování obrazu

2.1.1 Filtrace šumu

V této kapitole se budu zabývat filtrací obrázku, respektive řádku obrázku. Je to celkem velmi důležítá a i časově náročná výpočetní operace. V této diplomové práci jsou implementovány dvě funkce filtrace obrazového signálu. Jednotlivé filtrace budou popsány v následujících kapitolách.

2.1.1.1 Mediánová filtrace

Tato filtrace používá výběrový kvantil, medián. Je definován jako hodnota M, pro kterou je pravděpodobnost sledovaného jevu rovna jedné polovině.

$$P\left[X \le x_{0,5} \right] = 0,5 \qquad \textbf{(1)}$$

Pro obrazovou funkci je výpočet mediánu jednoduchý. Stačí uspořádat hodnoty jasu od nejmenší k největší v lokálním okolí a medián je hodnota jasu, která je uprostřed tohoto uspořádání. Pro snaží implementaci je vhodné volit masky (lokální okolí) s lichým počtem

bodů. Jelikož mohu zpracovávat pouze jeden řádek na jednou, a také pro rychlejší filtraci, používám masku 3x1, viz Obr. 2-1.

Obr. 2-1 a) Maska 3x1 pro určení medianu, b) Maska 1x3 pro určení medianu

Tento druh filtrace je vhoný pro impulzní šum. Nevýhodou je, že porušuje tenké čáry a ostré rohy v obraze, ale vzhledem k malé velikosti použité masky není tento neduh tak výrazný.

Mnohem výhodnější by bylo, kdyby mediánová filtrace se prováděla řádkově místo mnou používaného způsobu, sloupcově. V důsledku co možná nejmenších paměťových nároků jsem byl však nucen zvolit méně výhodný způsob sloupcové filtrace, která, jak již bylo napsáno, porušuje tenké čáry a ostré rohy v obraze. Řádkový medián tímto problémem netrpý, jelikož se nasnímá stejná scéna několikrát a vybere se medián z několika řádků stejného sloupce.

Implementované funkci s název Median3 se předávají následující parametry: pointer na vstupní pole a výstupní pole nasnímaného řádku, index řádku, pointer na pomocné pole, do kterého se pouze ukládají mezivýsledky a konečně velikost vstupního resp. výstupního pole *size*.

```
void Median3(unsigned char *in, unsigned char *out, unsigned short R, unsigned short *az, int size);
```

2.1.1.2 Filtrace průměrováním

Jedná se o velmi jednoduchou a přitom účinnou formu filtrace, především pro signály zašuměné Gausovským šumem, mezi které lze počítat i náboje z CCD snímače generované teplem krystalové mřížky senzoru. Odhad skutečné hodnoty jasu lze počítat pomocí aritmetického průměru

$$\overline{X} = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad (2)$$

Kde N je počet hodnot zůčastněných průměrování.

V této diplomové práci je tato filtrace průměrováním realizována, tak že se průměrují pořízené řádky mezi sebou – meziřádková filtrace, protože kdyby se průměrovaly hodnoty

jasu v jednom řádku (sloupcově), došlo by k rozmazání hran, což je nepřípustné. Či-li jedno místo zkoumané scény je nasnímáno minimálně dvakrát. Hodnota jasu každého pixelu je sčítána s hodnotou jasu z předešlého řádku ze stejného sloupce – sčítaní řádků. Výsledek je uložen do pomocného pole *Average*. Výslednou sumu každého pixelu vydělím počtem snímaných řádků.

S rostoucím počtem průměrů roste i doba pro další zpracování, což je logické. Proto je tento druh lineární filtrace vhodný pouze pro situace, kdy potřebujeme opravdu přesné výsledky měření objektu.

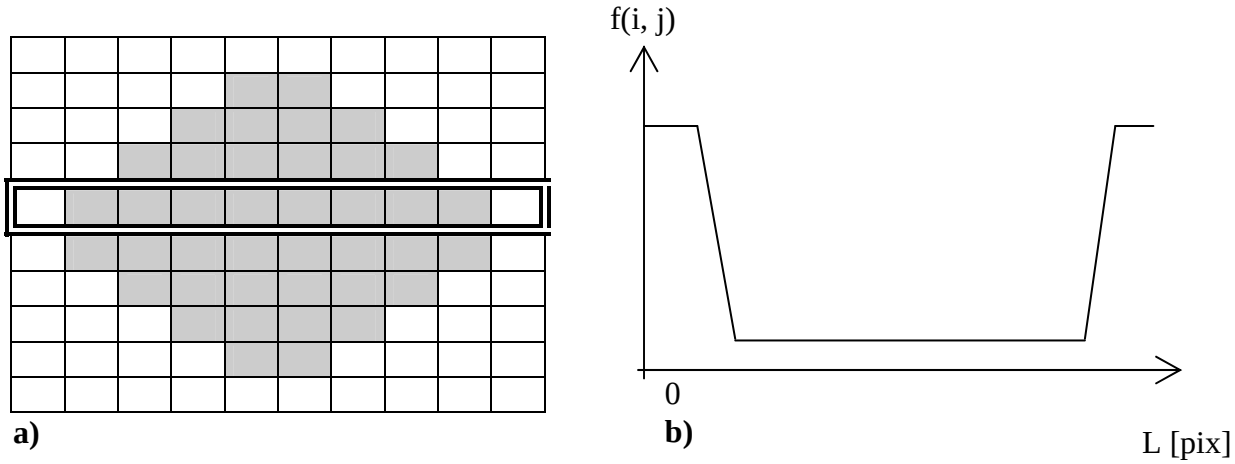
void AverageRow(unsigned char *Out, unsigned short *Average, unsigned char R, int size, int meanrows);

2.1.2 Detekce hran

V této kapitole popíši nejznámější metody pro detekci hran jako je hledání hran na základě 1. diference (derivace), 2. diference (derivace) a komparační metoda (threshold).

Než přistoupím k jednotlivým metodám, tak nejdříve nastíním, co to vlasntě hrana je a jak ji můžeme definovat.

Hrana ve své podstatě je oblast, kde se hodnota obrazové funkce f(i, j) mění náhle, či-li je patrné, rozhraní mezi dvěma oblastmi. Lze to také vysvětlit tak, že například zkoumaná scéna je světlejší a hledaný objekt je tmavší, či-li náhlý přechod ze světlé oblasti do tmavé, či naopak, a tím může být pokládán za hranu objektu. Na následujím obrázku si to lze názorně představit. Je zde znázorněn přůřez řádkem zkoumaného objektu a jemu odpovýdající jasový profil daný obrazovou funkcí. Fyzikální rozměr funkce zde neuvádím, jelikož jednotka může být prezentována či zakódována do digitální podoby. Jasovou hloubkou bývá vetšinou jeden bajt (8bitů), či-li rozměr v intervalu 0 až 255.



Obr. 2-2 a) zkoumaný objekt, b) jeho řádkový jasový profil

Na Obr. 2-2 b) je patrné, že ve skutečnosti přechod ze světlé do tmavší části oblasti, respektive změna obrazové (jasové) funkce není až tak náhlá (v ideální případě skoková). Z tohoto důvodu se používají matematické nástroje, které se snaží najít „skutečnou“ hranu objektu ze snímané scény. Jedním takovým nástrojem jsou parciální derivace, kde velikost změny obrazové funkce udává její gradient, respektive modul gradientu.

2.1.2.1 Detekce hran pomocí 1. diference

Jak plyne z názvu kapitoly, tato metoda využívá k detekci hrany nalezení maxima první derivace, respektive modulu gradientu obrazové funkce f(x, y), dle vzorce:

$$\left|\nabla f\left(x,y\right)\right| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \qquad (3)$$

V digitalizovné podobě signálu se 1. derivace nahrazuje 1. diferencí pro 2D, dle vzorce:

$$\begin{aligned} \Delta f_i\left(i,j\right) &= f\left(i,j\right) - f\left(i-n,j\right) \\ \Delta f_j\left(i,j\right) &= f\left(i,j\right) - f\left(i,j-n\right) \end{aligned} \qquad (4)$$

Pro 1D signál:

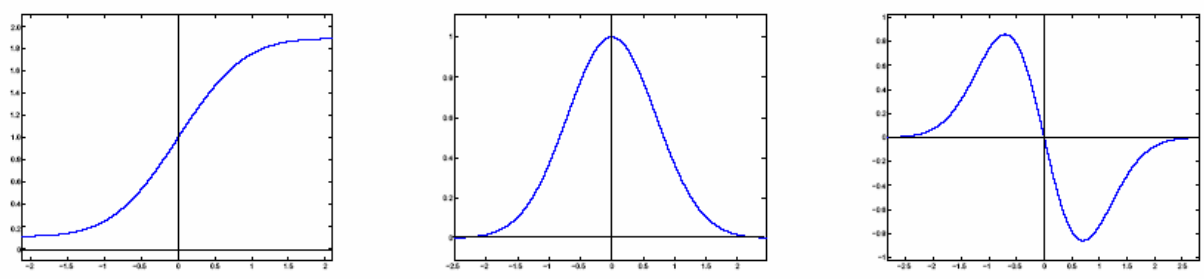
$$\Delta f_i\left(i\right) = f\left(i\right) - f\left(i-n\right) \qquad (5)$$

Kde *f(i, j)*, resp. *f(i)* představuje jas pixelu na pozici [i,j], resp. [i] a *n* je celé malé číslo, obvykle 1. První derivaci lze také aproximovat konvolucí masky operátoru a obrazové funkce, kde $f'(i)$ je 1. derivace diskretní (obrazové) funkce.

$$f'\left(i\right) = \left[-1,0,1\right] * f\left(i\right) \qquad (6)$$

Často je používána konvoluční maska, která aproximuje zmiňovaný gradient, jako např. operátor Prewittové, viz rovnice (6).

Na Obr. 2-3 si lze ukázat příklad signálu (např. jasový profil řádku) a jeho 1. a 2. diference:



a) **b)** **c)**

Obr. 2-3 Jasový profil 1D obrazové funkce: a) hrana obrazové funkce, b) první derivace - hrana je v lokalním maximu, c) druhá derivace - hrana je nule

Tato detekce hran je velmi často používaná. Její nevýhodou je však už z principu závislost modulu gradientu, respektive hledaného maxima, na strmosti hrany. Či-li bude-li přechod obrazové funkce (jasové funkce) více pozvolný, bude i maximum menší. Proto bývá i stanovení velikosti maxima lokálního okolí, které odpovídá hledané hraně, často méně určité a jasné.

Tento způsob detekce jsem implementoval v podobě následující funkce:

```
void FirstDiff(unsigned char *In, char *Out, unsigned short R,  int size);
```

Funkce obsahuje kovoluční matici podle Prewittové [-1, 0, 1]. Jak již bylo uveden, důvodem této masky je snaha docílit co nejrychlejší detekce hrany. Proto i počet členů matice je minimální.

2.1.2.2 Detekce hran pomocí 2. diference

Nevýhodu stanovení velikosti maxima (modulu gradientu) 1. derivace odstraňuje 2. derivace, protože ta stanovuje polohu hrany jako místo, kde je 2. derivace nulová. Proto je nalezení místa, kde nabýva 2. derivace nuly snažší, než určení velikosti maxima 1. derivace.

Opět, pokud pracujeme s digitalizovaným signálem je 2. derivace nahrazena 2. diferencí dle vzorce:

$$\Delta^2 f_i(i) = \Delta f(i) - \Delta f(i-n) \qquad (7)$$

Kde $\Delta f(i)$ je 1. diference funkce $f(i)$, n je celé malé číslo, obvykle 1.

Také 2. derivaci diskrétního signálu (obrazové funkce) lze aproximovat konvolucí obrazového operátoru a obrazové funkce, kde operátor aproximující druhou derivaci je tvořen složením konvolucí, viz:

$$\frac{\partial^2}{\partial i^2} = [-1,1] * [-1,1] = [1,-2,1] \qquad (8)$$

$$f''(i) = [1, -2, 1] * f(i) \quad (9)$$

Výsledný průběh 2. diference je patrný z Obr. 2-3.

Maska Laplacianu: [2, -4, 2]

Nevýhodou 2. diference je větší citlivost na šum, proto při jejím použití by měla nejdříve předcházet filtrace signálu. Ale filtrace by měla být obecně zahrnuta v předzpracování sígnálu. Tuto metodu jsem implementoval do procesoru a vyzkoušel její detekční schopnosti. Ale jak jsem uvedl, tato hranová detkční metoda je velmi citlivá na šum a také občasné dvojité detekce stejné hrany. I když doba trvání této metody byla srovnatelná s dobrou trvání detekce založené na první derivace (aproximaci první derivace pomocí konvoluční masky, abych byl přesnější), tak víše popsané argumenty ji znevýhoňují pro „real-time“ zpracování.

2.1.2.3 Detekce hran pomocí LoG

Jedná se o detekční metodu hran, která poskytuje výhody druhé derivace a zároveň filtruje obrazový signál, protože jak již bylo řečeno, druhá derivace je více citlivější na šum. Tato metoda detekce hran je známa pod pojmem (angl. Laplacian of Gaussian).

Gaussovské rozložení pro 2D prostor je definováno

$$G(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (10)$$

kde *x*, *y* jsou souřadnice v obraze a σ je středněkvadratická odchylka, která udává, na jak velkém okolí filtr operuje.

Filtraci Gaussiánem obrazové funkce získáme konvolucí $G(x, y, \sigma) * f(x, y)$.

Pro odhad druhé derivace vyfiltrované obrazové funkce lze použít Laplacián ∇^2, který je definován:

$$L = \nabla^2 = \nabla * \nabla = \frac{\partial^2 f(x, y)}{\partial x^2} + \frac{\partial^2 f(x, y)}{\partial y^2}, \quad (11)$$

čímž získáme rovnici $\nabla^2 \left[G(x, y, \sigma) * f(x, y) \right]$. Díky linearitě popsaných operací lze zaměnit pořadí derivace a konvoluce $\nabla^2 (G * f) = (\nabla^2 G) * f$. Operace $(\nabla^2 G)$ lze uspořadat do jediné konvoluční masky, známé také jako LoG operátor.

Nebudu zde popisovat jak se k tomuto operátoru dojde, pouze shrnu dohledané výsledky. Více lze najít např. v [1].

Operátor LoG (Laplacián Gaussiánu) je definován:

$$H(x,y) = c\left(\frac{x^2 + y^2 - \sigma^2}{\sigma^4}\right)e^{-\frac{x^2+y^2}{2\sigma^2}}, \qquad (12)$$

kde normalizační konstanta *c* zajišťuje nulový součet všech koeficientů v masce.

Konvolucí této masky a obrazové funkce lze získat hrany v obraze. Změnou hodnoty koeficientu σ^2 lze více či méně filtrovat obraz, ale zároveň se tím i ovlivňuje detekce hran, či-li pro větší hodntou σ^2 bude obraz krásně vyfiltrován, ale zároveň budou potlačeny i zajímavé a významné hrany v obraze.

Pro detekci hran v řádku je po konvoluci LoG operátoru a obrazové funkce nutné testovat pixely dle následující masky:

-	+	

	+	-

+	0	-

-	0	+

Obr. 2-4 Maska pro detekci hran

Tato detekce podává celkem slušné výsledky, avšak díky použití Goussovské fltrace rozostřuje hrany. Tuto detekci jsem rovněž implemetoval do procesoru, avšak po zjištění jsem tuto metodu v důsledku velkých dob trvání nutných k nalezení hran zavrhnul, pouze ji zde uvádím jako jednu z možností.

2.1.2.4 Komparační metoda – threshold

Jedná se o velmi jednoduchou a výpočetně nenáročnou (rychlou) metodu nalezení objektu v obraze. Porovnává se obrazová funkce (signál ze snímače) s komparační úrovní *k*.

$$h(x,y) = \begin{cases} 1 \; pro \; f(x,y) \le k \\ 0 \; pro \; jinak \end{cases} \qquad (13)$$

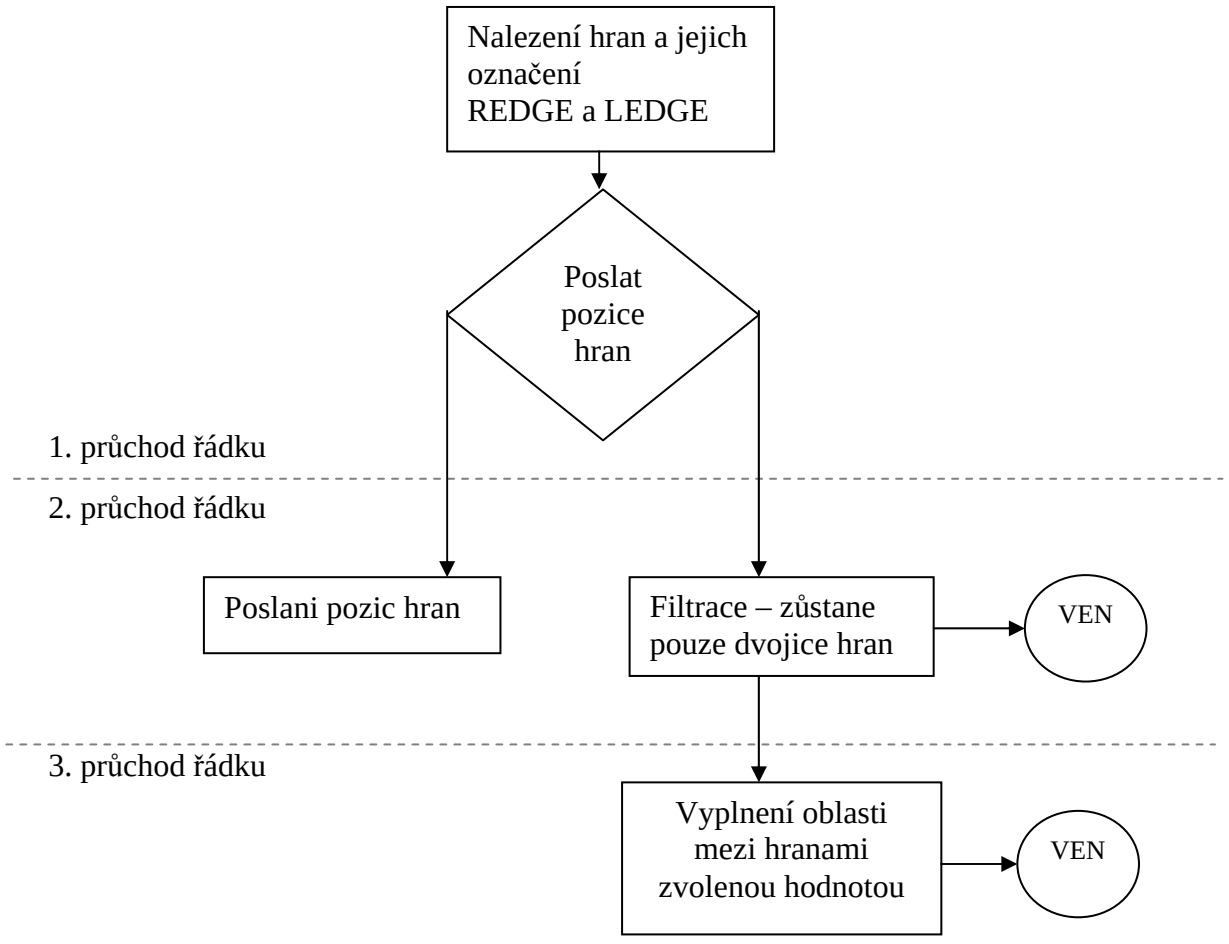
Kde *k* je komparační úroveň a *f(x,y)* obrazová funkce, *h(x,y)* je nová obrazová funkce, která nabýva hodnot 1 pro hodnoty jasu, které jsou menší než *k* (jedná se zřejmě o objekty) a nula pro větší než *k* (pozadí zkoumané scény). Nevýhodou této metody je závislost na nerovnoměrném osvitu zkoumané scény. Či-li bude-li část scény méně osvětlena, tak hodnoty jasu pozadí mohou býti brány jako by se jendalo o objekty a naopak, čímž může dojít k tomu, že se touto metodou mohou data znehodnotit. Proto pro přesné určení polohy hrany se tato metoda nepoužívá a používá se jedna z výše popsaných metod založených na derivacích obrazové funkce. Shrnutí výsledků navržených detekčních metod uvádím v kapitole 4.

void Threshold(unsigned char *In, unsigned char *Out, unsigned short R, unsigned char level, unsigned char output, int size);

2.1.2.5 Nalezení oblasti - postdetekce

void FindArea(char *In, unsigned char *Out, unsigned short R, unsigned char level, unsigned char output, int size);

Tato funkce označí hrany podle masky zobrazené na obrázku Obr. 2-4, které byly detekovány výše popsanými metodami detekce hran. Hranu označí definovanou hodnotou (REDGE nebo LEDGE). Následně prochází řádek a nuluje hodnoty pixelů, které nejsou rovny REDGE nebo LEDGE, čili eliminuje falešné hrany a šum podle zvoleného prahu *level*. Poté eliminuje hrany, které nejsou v páru, protože každý objekt má dvě hrany (dvojice hran), „levou a pravou hranu“. Poté je oblast mezi těmito dvěma hranami vyplněna hodnotou danou v parametru funkce (*output*). Vyplnění je určeno především pro metodu „Labeling“ a hodnota výplně je rovna „1“. Tato funkce, je také schopna zasílat pozice hran, které našla.

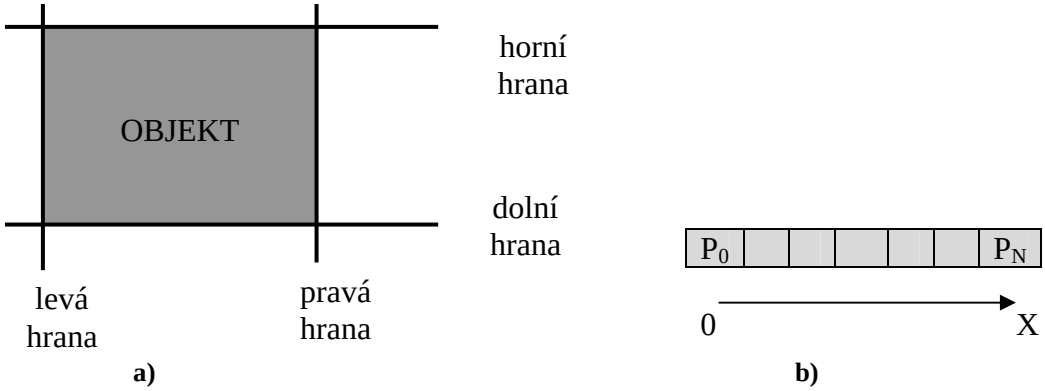


Obr. 2-5 Vývojový diagram funkce, která hledá hrany - oblasti

2.1.2.6 Dodatek k detekci hran

Je jasné, že z výše popsaných metod detekce hran, které operují pouze nad jediným řádkem, lze detekovat hrany kolmé ke směru pohybu CCD snímače, tedy levou a pravou hranu, viz Obr. 2-6. Hrany ve směru pohybu CCD snímače budou vzhledem k jednořádkovému přístupu nedekovatelné.

Metody detekce hran založené na druhé derivaci, zejména metoda s LoG operátorem, jsou více univerzálnější a poskytují dobré výsledky i z méně vydařených záznamů zkoumané scény (více zašuměné nebo málo osvětlená scéna). Jsou však více výpočetně (časově) náročnější. Naproti tomu konmparační metoda a metody heldající hrany na základě první a druhé derivace jsou výpočetně rychlejší, ale za předpokladu lepších snímacích podmínek, jako např. dobrý kontrast objekt-pozadí, konstatní osvětlení napříč celou zkoumanou scénou, nízká hodnota přídavného šumu, způsobená zejména již zmíněným tepleným šumem „za tmy“ generující „temné“ náboje, které nikterak nesouvisí se zkoumanou scénou aj. Je proto vhoné vědět co a za jakých snímacích podmínek budeme pořizovat řádkové snímky zkoumané scény a k tomu zvolit odpovídají metody hledání hran, aby se dospělo ke kompromisu rychlost – kvalita detekce.



Obr. 2-6 a) Objekt a jeho specifikace hran, b) indexace sloupců – souřadnice X

Pro spravnou interpretaci nalezeného objketu určeným podle zkoumaných oblastí, jsem si definoval levou, pravou, horní a dolní hranu oblasti a ve výsledku také objektu. Jak bylo uvedeno, hrana je definována náhlou změnou obrazové funkce. Pro snažší orientaci ve snímaném řádku jsem si zvolil počátek souřadnic tak, že první přijatý pixel z CCD snímače má nejmenší hodnotu souřadnice X a poslední přijatý pixel má hodnotu souřadnice X největší. Takže první hrana objektu (oblasti) nacházející se na souřadnici s menší hodnotou X souřadnice jsem nazval levou hranou a poslední hranu objektu (oblasti) nacházející se na vyšší souřadnici X jsem nazval pravou hranou.

Pakliže se objekt skládá z více řádku, tak první řádek na kterém byl objekt detekován jsem definoval jako horní hranu a poslední detekovaný řádek objektu jako dolní hranu, i když definice horní a dolní hrany vzhledem k neschopnosti detekce jednořádkového senzoru kolmého ke směru pohybu objektu je zavádějící, definicí těchto termínů si pouze ulehčuji orientaci v celkovém problému.

2.2 Metody zpracování obrazu - objektu

V následující kapitole vysvětlím základní metody pro zjištění plochy a těžiště objektu. Dále představím dvě metody, které jsem navrhnul pro nalezení objektů, nad kterými mohou být tyto metody vykonány. Tyto metody pro nalezení objetků se liší jak svou použitelností pro složitější kontury hledaných objektů, tak i paměťovými a výpočetními nároky. Více už však v následujících kapitolách.

2.2.1 Výpočet plochy objektu

Tato metoda patří mezi ty nejjednodušší. Pro určení plochy objektu stačí pouze zpočítat počet obrazových bodů (pixelů) patřící k danému objektu.
Plocha objektu se spočítá:

$$S = \sum_{j \in O} \sum_{i \in O} f(i, j) \qquad \textbf{(14)}$$

kde *i, j* jsou souřadnice bodu, a *f(i,j)* je obrazová funkce (v našem případě představuje bod objektu).

Pro výpočet plochy objektu v *mm* je ještě nutné plochu *S* vynásobit konstantou *K* [mm/pix]. Tuto konstantu získáme kalibrací senzoru (zaostření objektivu a změření plochy referenčního objektu).

2.2.2 Určení těžiště objektu

Další ze základních metod zpracování obrazové informace je určení polohy těžiště nasnímaného objektu. Souřadnice těžiště X_T, Y_T objektu se získají z následujících rovnic:

$$X_T = \frac{\sum_{i \in objekt} i.f(i,j)}{\sum_{i,j \in objekt} f(i,j)} \quad \textbf{(15)}$$

$$Y_T = \frac{\sum_{j \in objekt} j.f(i,j)}{\sum_{i,j \in objekt} f(i,j)} \quad \textbf{(16)}$$

Kde *f(i, j)* představuje obrazovou funcki objektu a prvky *i, j* představují souřadnice pixelů v horizontálním směru x, resp. ve vertikálním směru y.

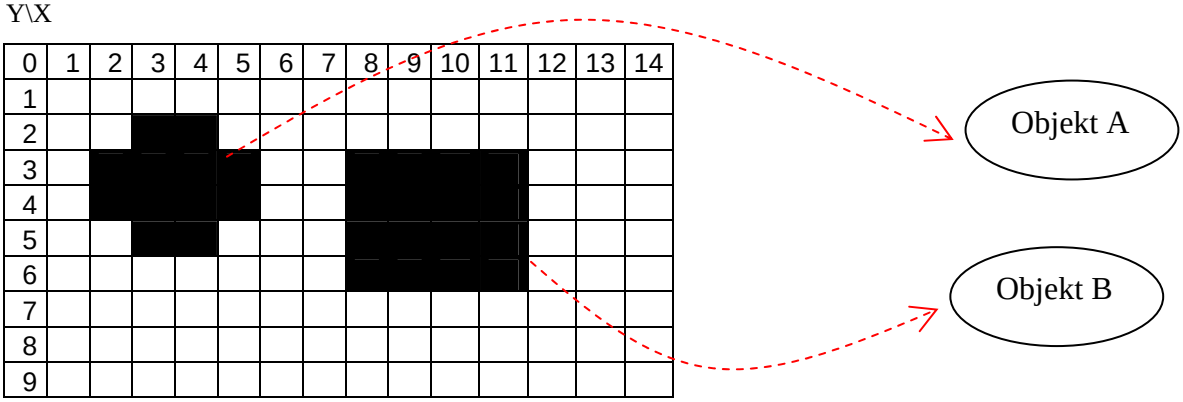
2.2.3 Metoda „překrívajících se pruhů“

Následující metoda je určená pro hledání objektů jejichž kontura není příliš složitá. Míní se tím to, že pro jeden objekt nesmí být v jednom řádku detekováno více hran než maximální počet dvě, a to sestupná a náběžná hrana („levá“ a „pravá“). Tato metoda vznikla inspirací hledání objektů z práce [9].

Takže objekty například ve tvaru X, V, O aj. nejsou touto metodou možné detekovat, protože by došlo k nesprávné interpretaci nalezených souřadnic. Tuto značnou nevýhodu vylepšuje fakt, že je tato metoda méně náročná na operační paměť a výpočetní výkon oproti metodě, která bude pospána v následující podkapitole. V paměti je tedy třeba uchovávat pouze právě nasnímaný řádek a dále prostor pro data obsahující informace o již nalezených objektech, které jsou však stále detekovány, či-li nacházejí se stále pod CCD snímačem.

Dále musím ještě uvést, proč se tato metoda nazývá právě „překrývání pruhů“. Je to z toho důvodu, že pixely v právě zkoumaném řádku patřící k objektu, vytvářejí pruh – výseč nalezeného objektu. A také z toho důvodu, že se na daný pruh nahlíží jako na samostatný objekt, který je ohraničen body, které reprezentují detekované hrany. Nepočítá se tedy přímo z jednotlivými pixely mezi detekovanými hranami, ale pouze s okraji těchto „pruhů“. To je tedy z hlavních důvodů, proč tato metoda není schopná nalézat a zpracovávat objekty se složitějšími tvary. Naproti tomu metoda v kapitole 2.2.4 pracuje s jednotlivými pixely.

Na následujícím obrázku si vysvětlíme princip této metody a výpočet základních atributů jako je poloha těžiště a plocha objektu.



Obr. 2-7 Hedání objektů

Ve 2. řádku Obr. 2-7 je detekována první hrana objektu A. Záleží pouze na použité metodě předzpracování řádku, jaká detekce bude použita. Jestli se bude detekovat pomocí první diference či druhé. Či-li zde nebudu popisovat konkrétní podobu rozpoznání hrany, ale budu již pracovat s předzpracovaným řádkem a pouze s tím, že tam ta hrana již je.

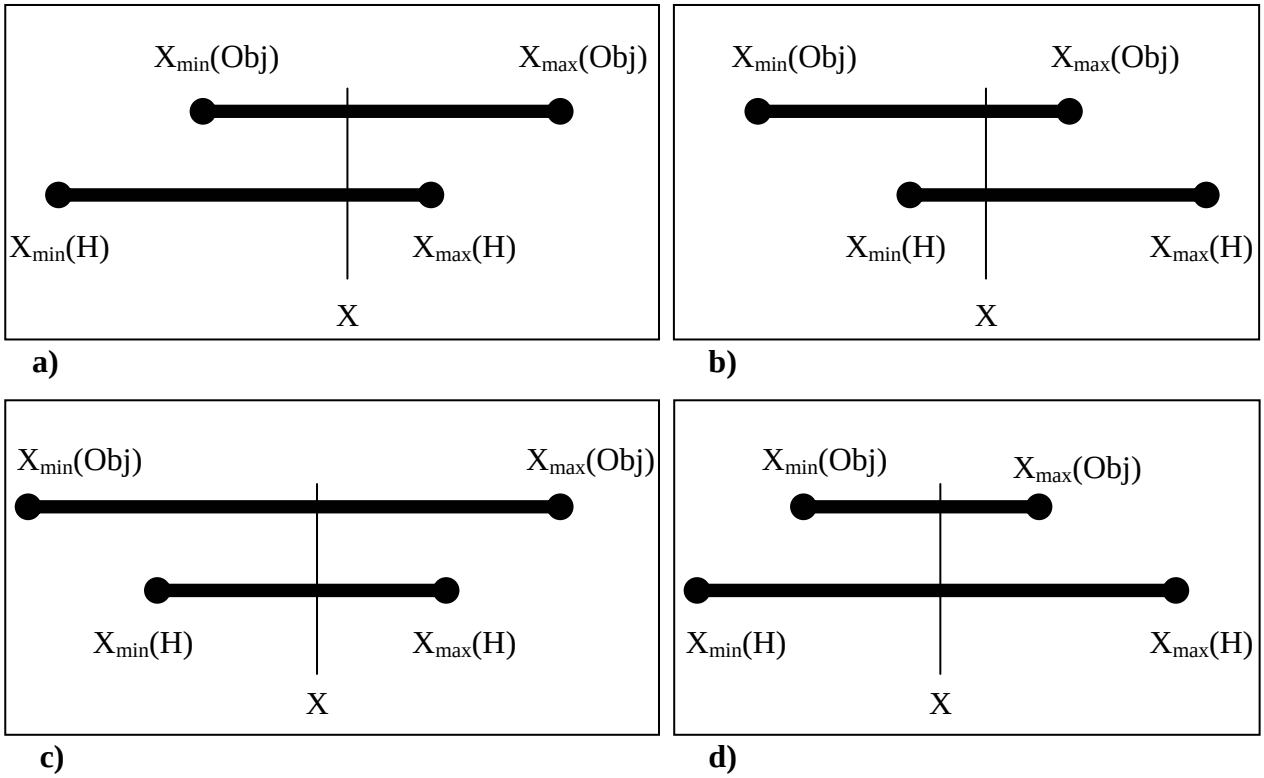
Takže po nalezení pixelů ve druhém řádku, které zřejmě patří k hledanému objektu je do pracovní paměti uložena souřadnice minimální a maximální hodnoty X souřadnice. Tyto souřadnice jsou porovnávány s hodnotami minima a maxima souřadnice X s již nalezenými, ale zatím s neukončenými objekty. Neukončeností se tím zde myslí, jak již bylo uvedeno, že momentálně zpracovávaný objekt je zatím stále detekován pod kamerou (pod CCD snímačem) a spodní hrana objektu, pohybuje-li se pás od spoda na horu, zatím neprošel pod kamerou. Nenalezla-li se shoda mezi hodnotami uloženými v databázi a právě nalezenými hodnotami minima a maxima X souřadnice, je to vyhodnoceno jako nalezení nového objektu.

Je-li nalezena shoda, jsou data uložena a přidána do paměti určené pro měřený objekt, jinak jsou data uložena na nové místo pro nový objekt.

Pro upřesněnní musím vysvětlit, co je míněno minimální a maximální X souřadnicí. Jak již bylo uvedeno, oblast (výseč objketu = „pruh“) je mnou definována podle nalezení levé a pravé hrany. Levá hrana je tedy nejmenší X souřadnice oblasti (ve výsledku celého objketu) a pravá hrana je maximální X souřadnice oblasti.

Nyní se pokračuje ve zkoumání řádku a je-li opět nalezen pixel odpovídající první hraně (snímame-li od leva do prava, od shora dolů), operace se opakuje, čili opět se hledá shoda.

Jak už plyne z názvu této metody „překrývání pruhů“, je shoda možná celkem ve čtyřech možných případech, viz následující Obr. 2-8.



Obr. 2-8 Možné případy překrývání průhů

Na Obr. 2-8 a) je znázorněna jedna ze čtyř možností, jak se mohou „pruhy“ překrývat patřící ke stejnému objektu. Body $X_{min}(Obj)$, $X_{max}(Obj)$ odpovídají minimální a maximální souřadnici X již nalezeného objektu (uložené v databázi). Body $X_{min}(H)$, $X_{max}(H)$ odpovídají minimální a maximální souřadnici X právě nalezeného objektu, které budou porovnávány s již nalezenými souřadnicemi X objektů. Či-li oba „pruhy“ – řádky mají společné pixely tehdy, když splňují podmínku

$$\left\{X_{\min}\left(Obj\right) \le X_{\max}\left(H\right)\right\} \wedge \left\{X_{\min}\left(Obj\right) \ge X_{\min}\left(H\right)\right\}, \qquad (17)$$

To znamená, že na právě zkoumaném řádku se našli pixely (oblast), které patří k již nalezenému objektu. Místo toho, aby se pro každý pixel zkoumalo, zda patří či nepatří nějakému objektu, zkoumá tato metoda relaci sousedství podle krajních bodů právě z výše nastítěných možností dle Obr. 2-8. Toto zjednodušení zvyšuje a zefektivňuje rychlost hledání objektů popisované metody. Obdobně lze vyhodnotit ekvivalentní možnosti překrývání „pruhů“.

Z Obr. 2-8 b) lze vyhodnotit podmínku

$$\left\{X_{\max}\left(Obj\right) \le X_{\max}\left(H\right)\right\} \wedge \left\{X_{\max}\left(Obj\right) \ge X_{\min}\left(H\right)\right\} \qquad (18)$$

Z Obr. 2-8 c) podmínku

$$\left\{X_{\min}(Obj) \le X_{\min}(H)\right\} \wedge \left\{X_{\max}(Obj) \ge X_{\max}(H)\right\} \quad \textbf{(19)}$$

Z Obr. 2-8 d) podmínku

$$\left\{X_{\min}(Obj) \ge X_{\min}(H)\right\} \wedge \left\{X_{\max}(Obj) \le X_{\max}(H)\right\} \quad \textbf{(20)}$$

Z právě nalezeného „pruhu“ lze také získat potřebné informace pro pozdější výpočet težiště. Plocha pruhu je vypočtena okamžitě a přičtena k ploše odpovýdajícímu objektu. Výpočet plochy není nikterak složitý. Pouze se odečtou krajní souřadnice pruhu, dle rovnice (8).

$$S = X_{\max}(Obj) - \left(X_{\min}(Obj) - 1\right) \quad \textbf{(21)}$$

Odečet minus jedné je pouze pro získání správné hodnoty plochy z důvodu toho, že odečítáme souřadnice, na kterých jsou ještě pixely daného objektu.Takže například plocha z řádku tři na Obr. 2-1 je podle výše pospaného vzorce

$$S = X_{\max}(Obj) - X_{\min}(Obj) + 1 = 5 - 2 + 1 = 4\left[pixel\right] \quad \textbf{(22)}$$

Poté je nutné zajistit hodnoty pro výpočet těžište dle vzorců (15) a (16). Získání sumy v čitateli rovnice obou souřadnic přímým přístupem, či-li součet indexů daných pixelů objektu, by bylo časově náročnější a znehodnotilo by to výše popsaný přístup této metody. Lze to tedy obejít z využitím aritmetické posloupnosti. Pokud potřebuju například sečíst souřadnice z řádku 3 objektu A, tj.

Σ = 5 + 4 + 3 +2 = 14, je nutné provést čtyřikrát součet. Pokud k tomu ale budu přistupovat z pohledu aritmetické posloupnosti, zjednoduším si početní operaci jednoduchými vzorci (23), (24).

Pro získání sumy v čitateli souřadnice X těžište objektu.

$$S_X = \left(X_{\max}(Obj).\frac{X_{\max}(Obj)+1}{2}\right) - \left(X_{\min}(Obj).\frac{X_{\min}(Obj)-1}{2}\right) \quad \textbf{(23)}$$

Pro získání sumy v čitateli souřadnice Y těžište objektu.

$$S_y = \left(X_{\max}(Obj) - X_{\min}(Obj)\right).Y \quad \textbf{(24)}$$

Prvek Y v rovnici (24) značí momentální délku objektu ve směru pohybu objektu, či-li počet řádků daného objektu. Tato délka je zvyšována na začátku následujícího cyklu zkoumaného řádku. Jelikož jsou tyto metody navrhovány tak, aby byly použitelné i například

pro zpracování dat z pásového dopravníku, tak globální index počtu řádků nepřichází v úvahu, protože by musel být teoreticky nekonečný. Proto se indexuje počet řádků pro daný objekt zvlášť a udává tak jeho délku.

Lze namítnou, že násobení a dělení je také časově náročnější, ale jelikož se bude počítat pomocí signálového procesoru, který má dokonce dvě násobičky, není to vůbec žadný problém. A další výhoda je ta, že pro větší úseky „pruhů“ například 100 pixelů, by bylo nutné minimálně sto krát sčítat a násobit souřadnice pixelů daného objektu. Což je tedy daleko časově náročnější, než výše posaný přístup zjištění dat pro výpočet težiště.

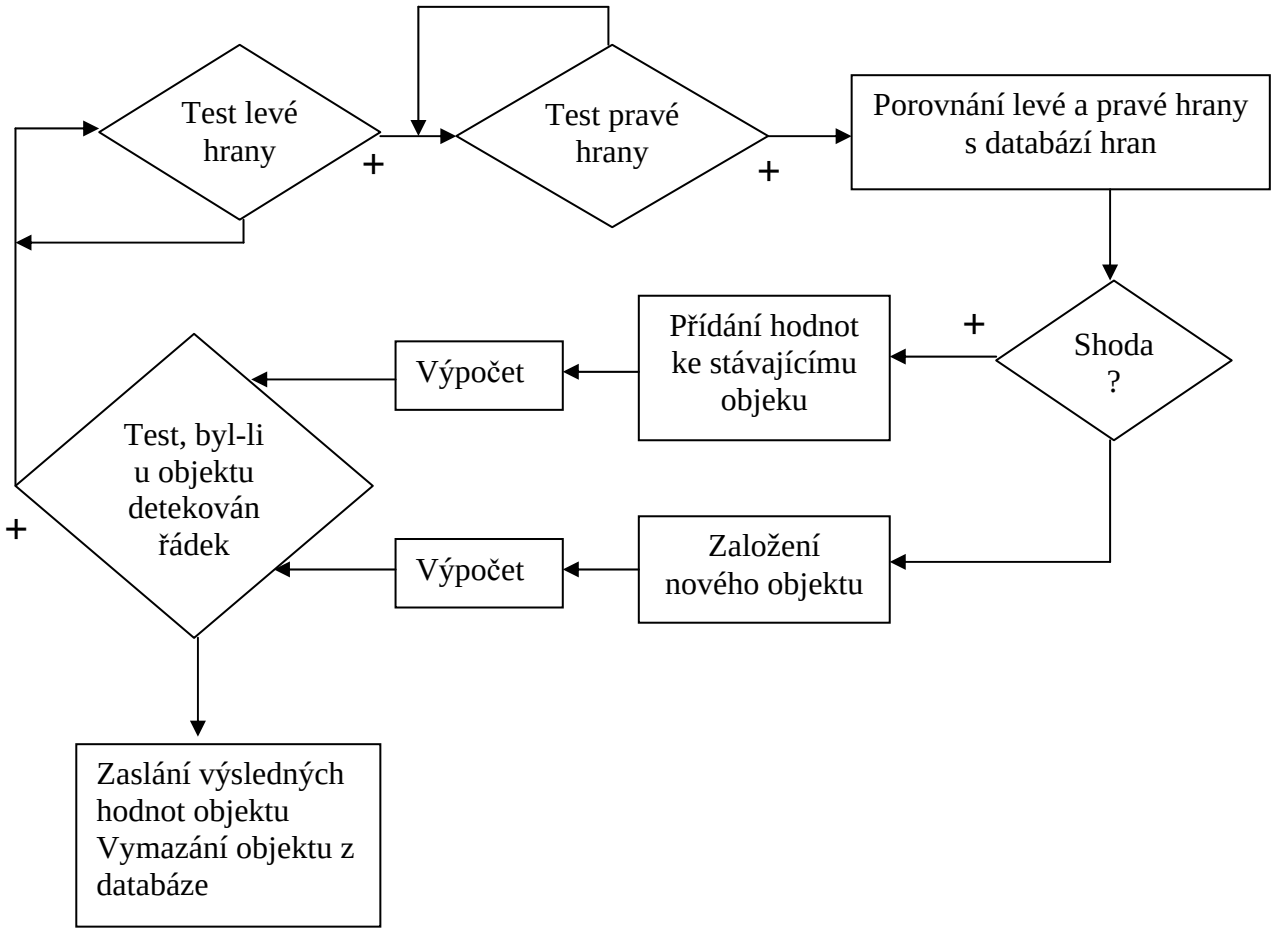
Takto se bude postupovat pro každý nasnímaný řádek. Pokud v následujícím řádku není zjišten „pruh“ zpracovávaného objektu, znamená to, že objekt se již nenalézá pod kamerou, lze tedy ukončit jeho zracovávání a uvolnit jemu přidělenou paměť. Dále se vypočte težiště z průběžně získaných dat, jako je plocha a sumy v čitely z rovnic (15), (16) a pošle se výsledek přes požadované rozhraní USB, UART nebo binární výstupy. Aby se zabránilo „rozházení“ dat příslušící zkoumaným objektům přes celou přidělenou pracovní paměť a byl využit vždy začátek pracovní paměti, tak na právě uvolněné paměťové místo zkončeného objektu, jsou uloženy data objektu, který byl nalezen jako poslední. Tato operace má také výhodu v tom, že se projíždí vždy ten úsek paměti, ve kterém jsou skutečně data zkoumaných objketů a nemusí se projíždět úplně celá paměť.
Pospaný algoritmus je implementován v následující funkci:

```
void EdgeLabeling(unsigned char *in, unsigned short R, unsigned char *label,
unsigned short FiltrS, unsigned short *min_x, unsigned short *max_x, unsigned
short *y, unsigned int *area, unsigned int *Sx, unsigned int *Sy, bool *loc, int size);
```

Pointer *in odkazuje na uložený a předzpracovaný řádek z CCD snímače. Index R ukazuje na řádek, který se bude zpracovávat. Pointer *label obsahuje indexy zpracovávaných objektů.
Pointer *FiltrS odkazuje na hodnotu plochy, která udává, jedná-li se o šum či nečistotu a nebo o hledaný objekt. Stručně řečeno svou hodnotou prahuje. Parametr size svou hodnotou vymezuje velikost pole řádku. Defaultně je nastaven na 2048, či-li počet pixelů senzoru. Ostatní parametry již odkazují na pole se zpracovávanými daty objektů.
Pro bližší rozbor funkce doporučuji si prohlédnou zdrojový kód.

Výše pospaná metoda tedy přesně splňuje zadání řádkového přístupu ke zpracovávaným datům. Takže základní parametry jako je težiště a plocha jsou k dispozici na

konci každého řádku. Naproti tomu je zde již zmiňovaná silná nevýhoda, která limituje použití této metody pro zkoumané objekty složitějších tvarů, či-li je na zvážení, jaké objekty se budou zpracovávat.



Obr. 2-9 Vývojový diagram metody překrívajících se pruhů

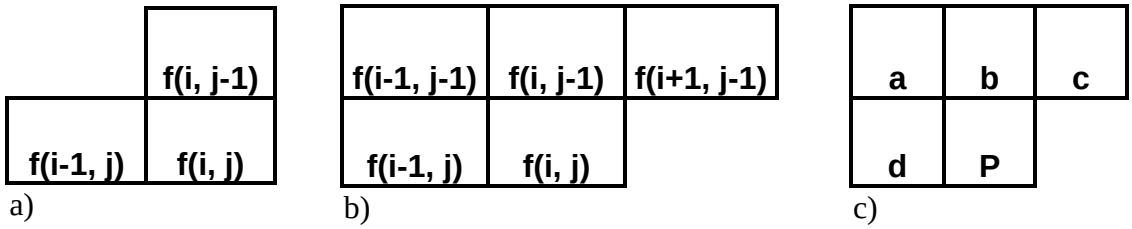
2.2.4 Metoda „Labeling“

Tato metoda je oproti předešlé metodě hledání objektů v obraze výpočetně robustnější a je tedy vhodná pro hledání objektů, jejichž tvary mají složitější charakter. V podstatě je možné touto metodou nalézt objekty jakýchkoliv tvarů. To mu však odpovídá výraznější nárůst nároků na paměťový prostor a výpočetní výkon. Je nutné mít v paměťi uložené dva řádky, které zajistí, že pixely z obou řádků mají mezi sebou relaci sousedství. Také mít v paměti uloženou tabulku ekvivaletních pixelů (vysvětlím dále) a paměťový prostor pro zpracovávané objekty. Dále je nutné stejný řádek prozkoumat dvakrát za sebou, oproti metodě „překrývajících se pruhů“, která potřebovala řádek prozkoumat pouze jedenkrát. Mimo

to, tato metoda přistupuje ke každému pixelu zvlášť a nahlíží na něj jako na samostatou oblast. Proto lze hledat v obraze objekty nejrůznějších tvarů.

Popisovaná metoda je známa pod pojmy jako „colouring“ či „labeling“ a používá se spíše pro počítače s velkým paměťovým prostorem (desítky až stovky MB) a pracuje nad již kompletním obrázkem uloženým v paměti (2D prostor). Proto jsem tuto metodu upravil tak, aby potřebovala především minimum paměťového prostoru a přistupovala k datům „řádkově“.

Základem je maska pro regionální identifikaci. Maska může být pro 4-sousedství nebo pro 8-sousedství, viz obrázek Obr. 2-10.

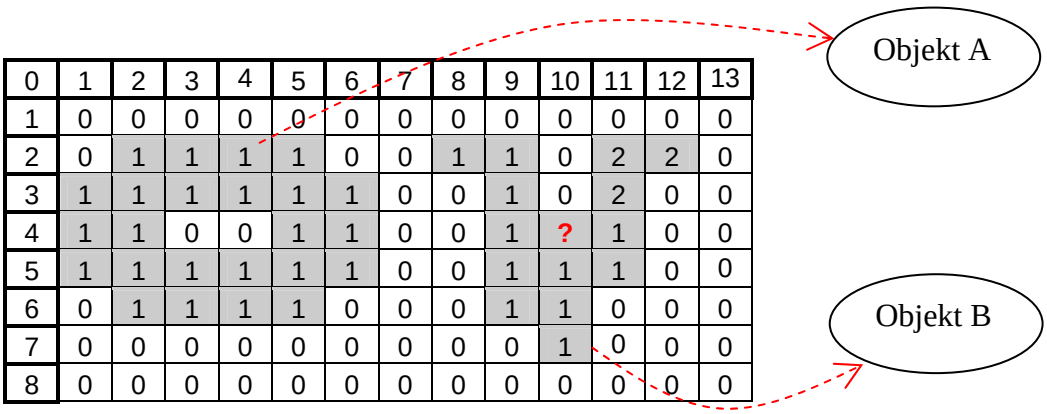


Obr. 2-10 Maska pro regionální identifikaci: a) 4 - sousedství, b) 8 – sousedství, c) 8 - sousedství

V okolí pixelu f(i, j) se zkoumají okolní pixely podle masky, zda-li nepatří pixel f(i, j) k nim a tím pádem ke zkoumané oblasti - objektu.

Opět zde již neřeším otázku, jak rozhodnout, zda daný pixel je objekt, či pozadí. To je již vyřešeno v předzpracování řádku. Jenom pro úplnost dodám, že hodnota pixelu označující, že se jedná o oblast (objekt), je různá od nuly (1 a výše). Hodnota pixelu označující pozadí zkoumané scény, je nula.

Na následujícím obrázku, který představuje příklad zkoumané scény, si vysvětlíme princip této metody.



0	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	1	1	1	1	0	0	1	1	0	2	2	0
3	1	1	1	1	1	1	0	0	1	0	2	0	0
4	1	1	0	0	1	1	0	0	1	?	1	0	0
5	1	1	1	1	1	1	0	0	1	1	1	0	0
6	0	1	1	1	1	0	0	0	1	1	0	0	0
7	0	0	0	0	0	0	0	0	0	1	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0

Obr. 2-11 Příklad zkoumané scény

Řádek po řádku se zkoumájí relace sousedství mezi jednotlivými pixely dle masky např. pro osmisousedství, viz Obr. 2-10 c). Hodnota zkoumaného pixelu *P* je porovnávána s hodnotami pixelů *a, b, c, d*. Je-li nalezena shoda mezi všemi zkoumanými pixely dle matice, je na pozici pixelu *P* uložena hodnota jeho sousedů, pakliže jsou tyto hodnoty ruzné od nuly. Liší-li se alespoň jeden ze sousedů *a, b, c, d* svou hodnotou, je na pozici *P* uložena nejmenší hodnota pixelů z pozic *a, b, c, d*. Dále je tato neshoda – kolize uložena na patřičné místo do ekvivalentní tabulky. Ekvivaletní tabulka proto, protože jednomu pixelu *P* odpovídá (je ekvivalentní) více hodnot dané jeho sousedy *a, b, c, d*, čemuž lze také rozumět tak, že doposud byla oblast chápána a zpracovávána jakoby se jednalo o dvě ruzné oblasti v obraze. Příkladem může být bod [10;4] na Obr. 2-11 označený „**?“**. Zde dochází v podstatě ke spojení oblastí, které patří k jednomu objektu B.

V literaturách zabývající se tímto problémem nezáleží, jaká z hodnot (v našem případě 1 a 2) pixelů je uložena do ekvivalentní tabulky a na místo pixelu *P*. Pouze je poznamenána kolize mezi těmito hodnotami pixelů. Z důvodu řádkového pohledu, minimálních paměťových nároků a co nejmenšího počtu průchodů stejného řádku, jsem zvolil následující postup poznamenání kolizí mezi hodnotami pixelů do ekvivalentní tabulky. Abych eliminoval rozsah paměťového prostoru obsahující data zkoumaných objektů, respektive jejich prohledávání v tomto prostoru, vždy se snažím číslovat objekty od nejmenších hodnot, či-li 2 a více. Míní se tím to, aby nebyly například zpracováváni pouze 3 objekty a každý se nenacházel v jiné části paměťového prostoru a měli libovolnou hodnotu pixelu (např. 5, 98, 178), která zároveň identifikuje oblast podle této hodnoty pixelu. Bylo by potom nutné prozkoumávat paměťový prostor od 2 do 178, když by přitom stačilo prohledávat od 2 do 4. Proto vždy vybírám hodnotu pixelu pro pixel *P* jako minimum z hodnot jeho sousedů. Tím také jednoznačně určím, která z hodnot pixelů jednoznačně identifikuje oblast – objekt. Bude-li v oblasti více hodnot pixelů, všechny se budou podle ekvivalentní tabulky odkazovat na tu jejich nejmenší hodnotu (mnou nazvanou jako „root“ - kořen). Tímto způsobem také získám kontrolu nad pixely, které byly vlivem tvarových poruch či složitosti snímaného objektu, nezpracovatelné minulou metodou. Dále je pomocí tohoto minima – „rootu“ (hodnoty pixelu) možno přiřadit paměťový prostor obsahující informace o právě zkoumané oblasti, potažmo objektu, jinak také řečeno, že hodnota pixelu „rootu“ zároveň určuje pozici v paměti přidělenou pro objekt.

Schválně zde rozděluji termíny oblast a objekt, protože objekt se může skládat z více oblastí, jelikož k tomu přistupuji řádkově a s co nejmenším počtem opakování průchodů

řádku, protože se ještě neví, že zatím dvě naprosto nezávislé oblasti budou v další řádcích spojeny, protože patří k jednomu objektu.

Po prvním průzkumu řádku a zjištění, které pixely jsou ekvivalentní s jinými a které ne, následuje další průchod stejného řádku, který ekvivalentní pixely přepíše („přebarvý“) na minimum („root“). Zároveň se již získávají informace pro zjištění plochy a data pro výpočet těžiště. Paměťový prostor vyhrazený ekvivalentním pixelům se uvolní pro budoucí použití.

Nyní uvedu názorný příklad, který výše popsanou metodu přiblíží a lépe objasní. Na Obr. 2-12 je objekt, který by byl minulou metodou nezpracovatelný, či-li v jenom řádku více hran, které patří ke stejnému objektu. Může se třeba jednat o transformátorový plech a chceme zjistit jeho plochu a polohu těžiště.

0	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	0	0	**1**	0	**1**	**1**	0	**1**	0	0
3	0	0	**1**	0	**1**	**1**	0	**1**	0	0
4	0	0	**1**	0	**1**	**1**	0	**1**	0	0
5	0	0	**1**	**1**	**1**	**1**	**1**	**1**	0	0
6	0	0	0	0	0	0	0	0	0	0

Obr. 2-12 Příklad zkoumaného objektu s více hranami v jednom řádku

V mém programu dávám krajní hodnoty pixelů (první a poslední sloupec) nulové a rovněž první řádek, protože stejně se jedná o místa, kde by se hrana objektu těžko získávala a zároveň to velmi zjednoduší návrh programu. Takže se prozkoumává až druhý řádek a dorazí se do bodu [3;2], viz Obr. 2-13.

0	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	0	0	**2**	0	**1**	**1**	0	**1**	0	0
3	0	0	**1**	0	**1**	**1**	0	**1**	0	0
4	0	0	**1**	0	**1**	**1**	0	**1**	0	0
5	0	0	**1**	**1**	**1**	**1**	**1**	**1**	0	0
6	0	0	0	0	0	0	0	0	0	0

Obr. 2-13 Nález prvního pixelu objektu

Jelikož sousedé bodu [3;2] mají nulovou hodnotu, tak na tuto pozici uložím hodnotu pixelu 2 a zvýším index hodnot pro „barvení“. Je to z toho důvodu, protože hodnotě 1 odpovídá hodnota ještě neprozkoumané oblasti. Tato drobná úprava opět zjednoduší návrh

programu. Hodnotu 2 zároveň uložím do ekvivalentní tabulky na pozici 2. Jelikož na pozici ekvivalentní tabulky 2 je rovněž hodnota 2 (odkazuje sama na sebe), tím si takto definuji, že se jedná o „root“. Pokračuji v prozkoumávání řádku a dojdu do bodu [5;2], viz Obr. 2-14.

0	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	0	0	**2**	0	**3**	**1**	0	**1**	0	0
3	0	0	**1**	0	**1**	**1**	0	**1**	0	0
4	0	0	**1**	0	**1**	**1**	0	**1**	0	0
5	0	0	**1**	**1**	**1**	**1**	**1**	**1**	0	0
6	0	0	0	0	0	0	0	0	0	0

Obr. 2-14 Nález nového pixelu jiné oblasti

Zde opět najdu v sousedství nulovou hodnotu, tak na pozici [5;2] uložím hodnotu 3, zvýším index „barvení“ a opět provedu zápis hodnoty 3 do ekvivalentní tabulky na pozici 3. Takto pokračuju až na konec řádku. Nýní provedu zjištění v ekvivalentní tabulce, neodkazuje-li se nějaká hodnota jinam, než sama na sebe. Pak-li že ano, provedu „přebarvení“ na hodnotu, která odkazuje na „root“. Pozice v ekvivalentní tabulce označovaná jako „root“ obsahuje již hodnotu pixelu, která definuje oblast (v konečné fázi objekt). Poté provedu „přebarvení“ řádku s následným použitím metod pro určení plochy a těžiště.

Nyní přejdu na další řádek opět provádím tu samou, výše popsanou operaci, viz a) Obr. 2-15.

0	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	0	0	**2**	0	**3**	**3**	0	**4**	0	0
3	0	0	**2**	0	**3**	**3**	0	**4**	0	0
4	0	0	**1**	0	**1**	**1**	0	**1**	0	0
5	0	0	**1**	**1**	**1**	**1**	**1**	**1**	0	0
6	0	0	0	0	0	0	0	0	0	0

a)

0	1	2	3	4	5
0	0	2	3	4	0

b)

c)

Obr. 2-15 Průzkum 3 řádku: a) zkoumaná scéna, b) ekvivalentní tabulka, c) schématicky naznačené „rooty“ – oblasti.

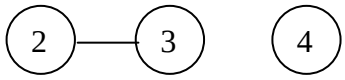
Po zkončení druhého průchodu třetím řádkem je obsah ekvivalentní tabulky na obrázku Obr. 2-15 b). Byly nalezeny zatím tři oblasti 2,3,4. Z nichž oblast 2 má zatím plochu dva pixely, oblast 3 má plochu čtyři pixely a oblast 4 má plochu dva pixely.
Pokračuju průzkumem dále až narazím na první kolizi v bodě [4;5], viz Obr. 2-16.

0	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	0	0	**2**	0	**3**	**3**	0	**4**	0	0
3	0	0	**2**	0	**3**	**3**	0	**4**	0	0
4	0	0	**2**	0	**3**	**3**	0	**4**	0	0
5	0	0	**2**	**2**	**1**	**1**	**1**	**1**	0	0
6	0	0	0	0	0	0	0	0	0	0

a)

0	1	2	3	4	5
0	0	2	2	4	0

b)



c)

Obr. 2-16 Detekce první kolize - ekvivalence mezi 2 a 3: a) zkoumaná scéna, b) ekvivalentní tabulka, c) schématicky naznačené „rooty“

V tomto bodě se významně projevý má úprava dosadit na pozici [4;5] nenulové minimum ze sousedů (vyznačeno žlutě). Tuto kolizi dosadím do ekvivalentní tabulky. Na pozici 3 dosadím minimum 2, tedy „root“. Výsledek této operace je znázorněn na Obr. 2-16.

V bodě [4;7] detekuji opět kolizi a na pozici 4 v ekvivalentní tabulce uložím také hodnotu 2.

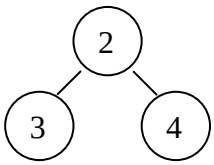
Při druhém průzkumu stejného řádku (tohoto řádku, číslo 5) již zjišťuji informace (pro výpočet plochy a těžiště) pro oblast 2 a dosavadní zjištěná data oblasti 3 přičtu k oblasti 2 a v ekvivalentní tabulce provedu konečnou úpravu zápisem nuly, čímž tak uvolňuji místo pro další možné oblasti, které mohou nést toto číslo – hodnotu pixelu. Totéž provedu i pro oblast 4, jejíž hodnoty opět přičtu k hodnotám oblasti 2 a zapíšu nulu do ekvivalentni tabulky na pozici 4. Výsledek prohledané scény je na následujícím obrázku Obr. 2-17.

0	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	0	0	**2**	0	**3**	**3**	0	**4**	0	0
3	0	0	**2**	0	**3**	**3**	0	**4**	0	0
4	0	0	**2**	0	**3**	**3**	0	**4**	0	0
5	0	0	**2**	**2**	**2**	**2**	**2**	**2**	0	0
6	0	0	0	0	0	0	0	0	0	0

a)

0	1	2	3	4	5
0	0	2	0	0	0

b)



c)

Obr. 2-17 Výsledek po průzkumu scény: a) zkoumaná scéna, b) ekvivalentní tabulka, c) schématicky naznačené „rooty“

Výsledkem této operace je nalezení jednoho objektu s číslem 2 a vypočtená plocha 18 pixelů s těžištěm $X_T = 5,5$ a $Y_T = 3,7$. Data jsou poslána nadřízenému zařízení (např. PC) přes jedno z možných rozhraní (např. UART, USB) po zkončení druhého průchodu 6 řádku. Po odeslání dat je i na pozici 2 ekvivalentní tabulky zapsána nula a vymazána paměť dat oblasti

2, čímž se metoda dostala do stavu tesně po první inicializaci a je schopna pracovat se svou celou přidělenou pamětí.

Tímto způsobem je možno hledat a zpracovávat nasnímané objekty, které mají až netypické tvary, jako na Obr. 2-18.

0	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	**1**	0
3	0	0	0	0	0	0	0	0	0	**1**	0	**1**	0
4	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	**1**	**1**	**1**	0
5	0	**1**	0	0	0	0	0	**1**	0	**1**	**1**	0	0
6	0	**1**	0	**1**	**1**	**1**	**1**	**1**	0	**1**	**1**	0	0
7	0	**1**	0	0	0	0	0	0	0	**1**	**1**	0	0
8	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0

Obr. 2-18 Netypický tvar objektu

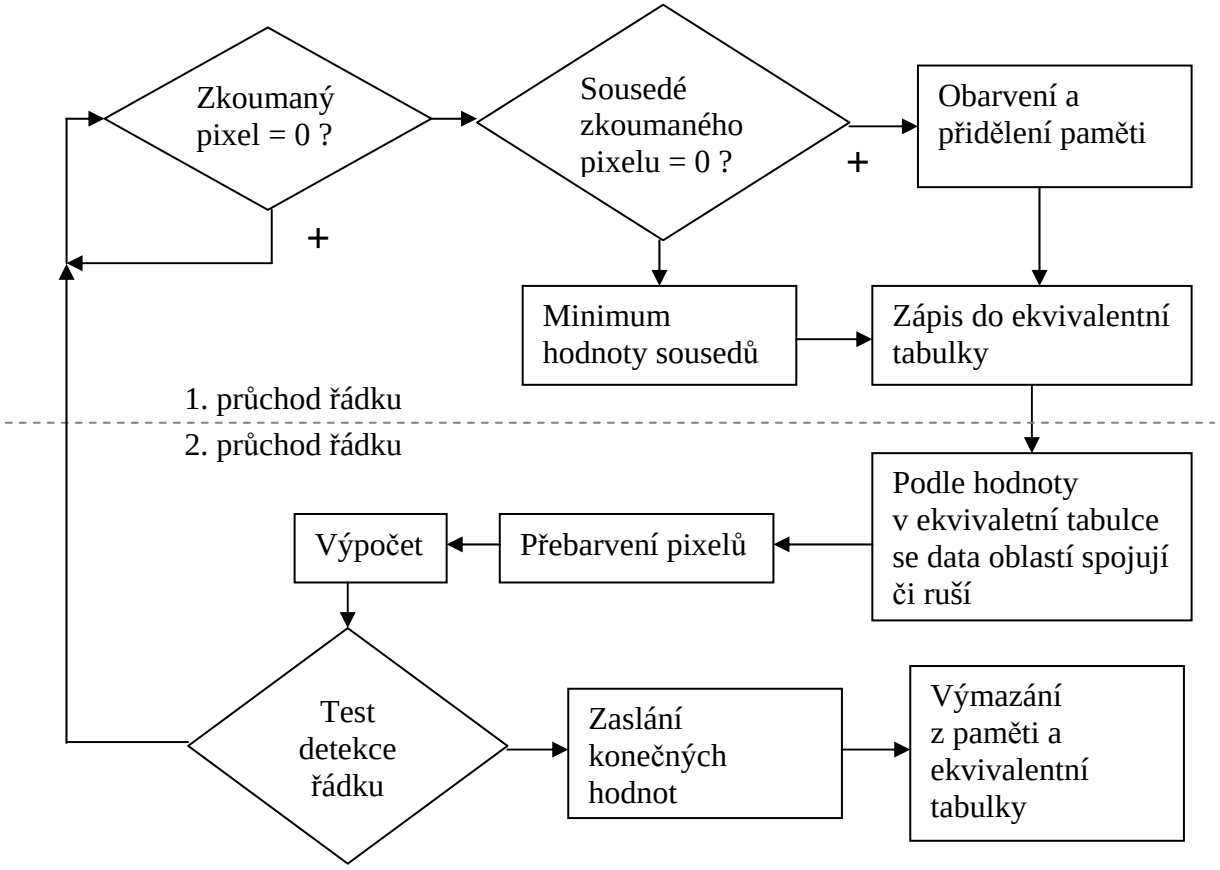
Výše pospaný algoritmus je implementován v následující funkci:

```
void Labeling(unsigned char *in, unsigned char *out, unsigned short R, unsigned
short L, unsigned short *az, unsigned char *Table, unsigned char *label, unsigned
char *min, unsigned short *min_x, unsigned short *max_x, unsigned short *y,
unsigned int *Sx, unsigned int *Sy, unsigned int *area, bool *loc, unsigned short
FiltrS, int size);
```

V paměti procesoru je definována paměť pro globalní proměnou, se kterou funkce pracuje pomocí pointerů. Všechny významné parametry si funkce takto předává, ušetří to více paměťového místa a času při spouštění funkce. Parametry *in a *out jsou pointery které odkazují na paměťové místo, kde je uložen předzpracovaný řádek. Pokud se čte a zapisuje ze stejného místa, je pointer *in a *ou) stejný. Dále parametr *Table odkazuje na ekvivalentní tabulky. Vzhledem k tomu, že je definována jako unsigned char, lze zatím detekovat maximálně až 254 oblastí potažmo objektů. Hodnota 254 z důvodu toho, že honotu nulu a jedničku nelze brát jako detekovanou oblast. Nulu jsem zvolil jako hodnotu určující pozadí scény a jedničku jako hodnotu oblasti, která nebyla ještě „barvena“, pouze byla předzpracována (např. detekrorem hran na základě druhé derivace). Pokud by se chtělo pracovat s více oblastmi, lze to přetypovat na unsigned short, pokud bude dostatek volné paměti. Parametr *label určuje hodnotu budoucího pixelu, pokud bude rozhodnuto, že se jedná o novou oblast. Ostatní parametry se odkazují na paměťové místo přidělené pro data

nalezených a zpracovávaných objektů, respektive zatím o nich zjištěných hodnot a dále pomocné paměťové registry .

Následující vývojový diagram opět přiblíží výše popisovanou metodu „Labeling“.



Obr. 2-19 Vývojový diagram metody "Labeling"

2.2.5 Shrnutí faktů k představeným metodám hledání objektů

Musím zde uvést, proč není u metody „Labeling“ použito filtrace průměrováním a proč u metody „překrývání pruhů“ není zase použita detekce hran pomocí metody Threshold.

Metoda „Labeling“ nepoužívá fitraci průměrováním z paměťových důvodů. Abych docílil co nejrychlejšího zpracování obrazu, provádím zpracování předešlého řádku v době načítání nového řádku. Proto mám vyhrazenou vyrovnávací paměť podle následujícího obrázku:

PŘEDEŠLÝ ŘÁDEK	N-1	1
ZPRACOVÄVANÝ ŘÁDEK	N	2
NOVÝ ŘÁDEK	N+1	3

Vyrovnávací paměť má tedy celkem tři položky, Nový řádek určen pro ukládání nového řádku přes PPI pomocí DMA, Zpracovávaný řádek (nad kterým jsou momentálně vykonávány metody) a Předešlý řádek určený především pro metody „Labeling“, aby byla zajištěna správná návaznost a relace sousedství mezi pixely (oblastmi). Každá položka cyklicky rotuje, čili po naplnění novým řádkem na pozici 3, se změní index pro pozici ukládání na pozici 1, zpracovávaný řádek změní svou pozici z 2 na pozici 3 a předešlí řádek změní svou pozici z 1 na pozici 2 atd.

Takže u metody „Labeling“není pro půměrování dostatek paměťového místa. I kdyby se požadované paměťové místo vyhradilo, tak softwarová implementace by byla o dost náročnější (zajištění správné návaznosti – relace sousedství) a celkově by to prodloužilo operační dobu, protože by to vyžadovalo více datových přesunů.

Metodu „překrývajících se pruhů“ nelze použít v kombinaci s hranovou detekční metodou Threshold z důvodu toho, že tato metoda neposkytuje informace ve správném formátu o místě levé a pravé hrany, která je pro metodu „překrývajících se pruhů“ velmi důležitá. Za to však elegantním a velmi rychlým způsobem řeší nalezení a vyplnění oblasti pro metodu „Labeling“.

3 SNÍMACÍ SYSTÉM - DESKA

Snímací systém obrazového signálu je schopen zpracovávat data z CCD senzoru na jejichž základě procesor vyhodnotí jejich obsah, či jenom zprostředkuje přenos video informace přes USB do PC, či-li pracuje v nejjednodušším módu – scanner.
Způsob módu činnosti desky je možno konfigurovat na PC přes USB.

Na *Obr. 3-1* je vidět blokové schéma celého zařízení. Pro základní představu funkčnosti desky shrnu a stručně popíši vlastnosti jednotlivých komponent. Detailnější rozbor duležitých prvků bude popsán a rozebrán v následujích subkapitolách.

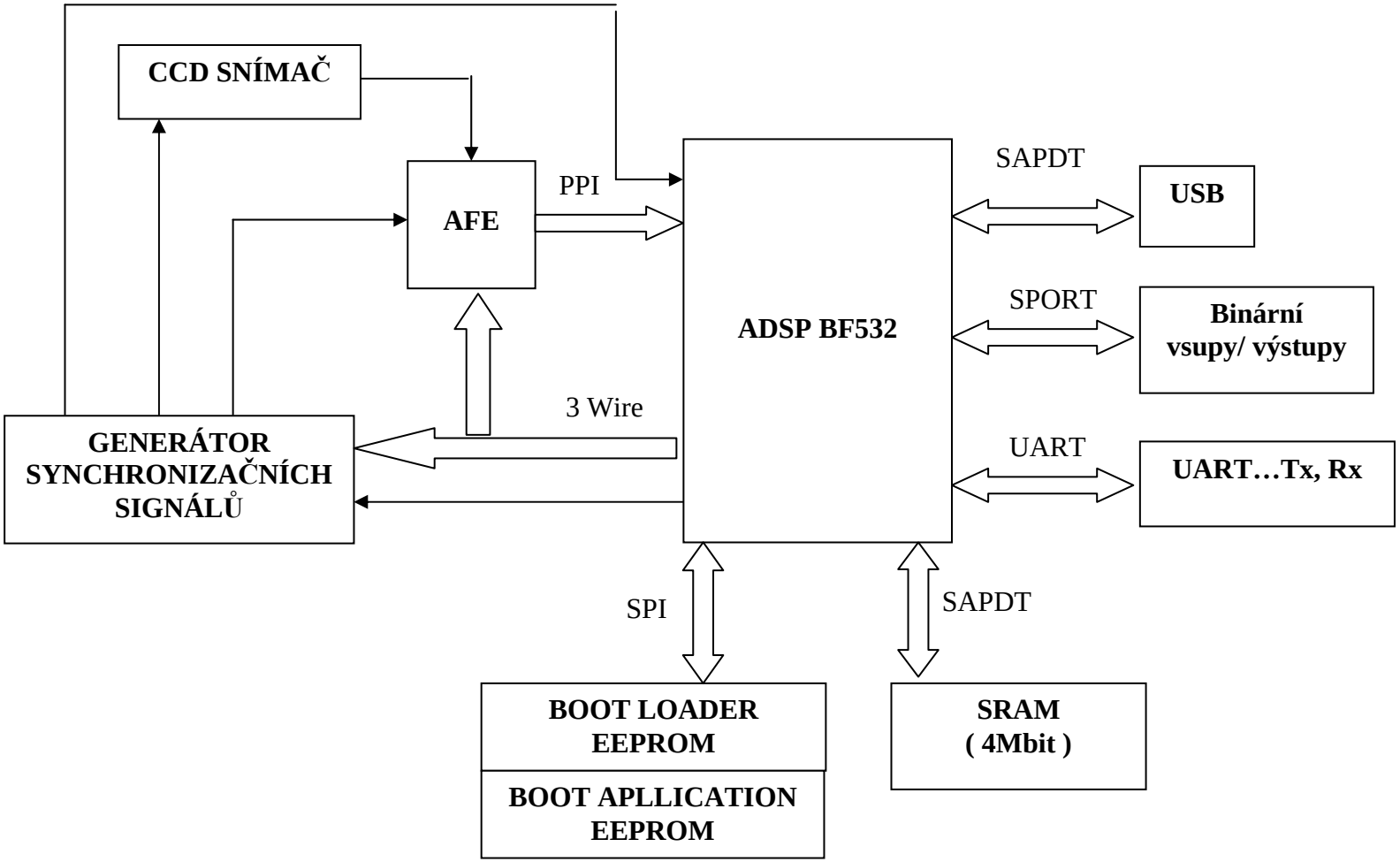
K desce lze tedy připojit CCD jednořádkové senzory o maximální počtu snímacích bodů – pixelů 16368, a to jak jednobarevné, tak i tří barevné. Konektor, ale ostatně celá deska je k tomu uzpůsobena. Lze ji jednoduše nastavit přenastavením binárních vstupů pro tříbarevné snímaní v obvodech generace syn. pulzů, AFE a samotném procesoru. Defaultní nastavení desky je však pro jednobarevné snímaní. Konektor ale i GSS je navržen tak, aby se dalo k desce připojit více typů CCD senzorů a to nejen již zmiňované velikosti počtů pixelů, ale i ve způsobu generace a řízení pulzů, protože ne každý CCD senzor obsahuje vnitřní logiku pro generaci hodin pro vysouvání náboje na výstupní detektor CCD senzoru. Bližší popis možných typů pro připojení CCD a rozložení pinů konektoru bude popsáno v 3.2.

Zmiňovanou generaci pulzů zajišťuje GSS, naprogramovaný v CPLD. Ten je ovládán a programován samotným procesorem. GSS generuje i pulzy pro AFE a pro PPI rozhraní procesoru. Či-li digitalizovaný videosignál z jednořádkového CCD snímače se uloží přes PPI autamaticky do paměti pomocí DMA. Pamět může být použita jak interní paměť procesoru L1, tak i externí paměť SRAM. Uložená data mohou být buď zpracována výše popsanými metodami a jejich výsledky poslány přes UART, USB či přes binární výstupy. Nebo se pošlou uložená data přes USB do PC a deska bude fungovat jako scanner.

3.1 Základní vlastnosti a charakteristika snímací desky

- Dvojité napájení 5V, 12V, typický odběr až 320mA
- Vyvedené napěťové piny pro 3,3V, 5V, 9V a zemnící piny
- Signálový procesor od Analog Devices BF 532 pracující na hodinové frekvenci jádra 396MHz, frekvence jeho periferií 79MHz
- Nastavitelný 14 bitový AD video – převodník AD9822 (využito pouze horních 8 bitů)
- Externí 4Mbitová SRAM
- Komunikace přes USB, UART
- Osm binárních vstupů, osm binárních výstupů
- Dva rychlé binární vstupy přivedené přímo na binární vstupy procesoru s možností tak generovat přerušení v procesoru
- Možnost naprogramování SPI EEPROM 32kB přímo pomocí vyvedeného portu
- Generátor synchronizačních signálů je realizován obvodem CPLD, který je možno kdykoliv přeprogramovat přes vyvedených JTAG port. Pro senzory je generována pevná pixel frekvence 3MHz

V následujících subkapitolách popíši jednotlivě konstrukci, složení a základní charakteristiky snímací desky obrazového signálu.



Obr. 3-1 Principiální schéma snímací desky

3.2 Řádkový CCD snímač a konektor pro jeho připojení

Jak již vyplývá ze zadání, tak deska byla navržena pro zpracovaání videosignálu z jednořádkového CCD snímače. Jednořádkový CCD snímač obsahuje v jedné řadě fotocitlivé buňky (pixely), jejichž počet může být typicky v intervalu od 256 až do 20 tisíc a více. Rozměry fotociltivých elemtnů (pixelů) se pohybuje od jednotek až do deseti a více mikrometrů. K jejich řízení stačí zpravidla dva signály. Jeden řídící signál vysouvá naakumulované náboje ven ze snímače a určuje tak pixelovou frekvenci. Druhý signál přesouvá naakumulované náboje z pixelů do CCD posuvného registru a jeho perioda tak vymezuje tzv. integrační dobu. Ne každý ze snímačů má v sobě zabudouvanou potřebnou logiku pro řízení a je tedy nutné k němu přivést více jak dva zmiňované signály. Typicky to bývá resetovací signál, který obnovuje napěťovou uroveň na výstupní plovoucí dioodě detektoru snímače, a tím ji tak připravuje pro další převod náboje z CCD registru do detektoru (převodník náboj / napětí). Dále to bývají hodinové signály pro přímé řízení hradel CCD registru pro posun nábojů.

Navržená deska byla odzkoušena na černobílém řádkovém CCD snímači Sony ILX503A a ILX551A. Jelikož deska je schopná zpracovávat videosignál z mnoha řádkových CCD snímačů, byl ILX503A, respektive ILX551A, vybrán jako demonstrativní, jehož základní parametry jsou:

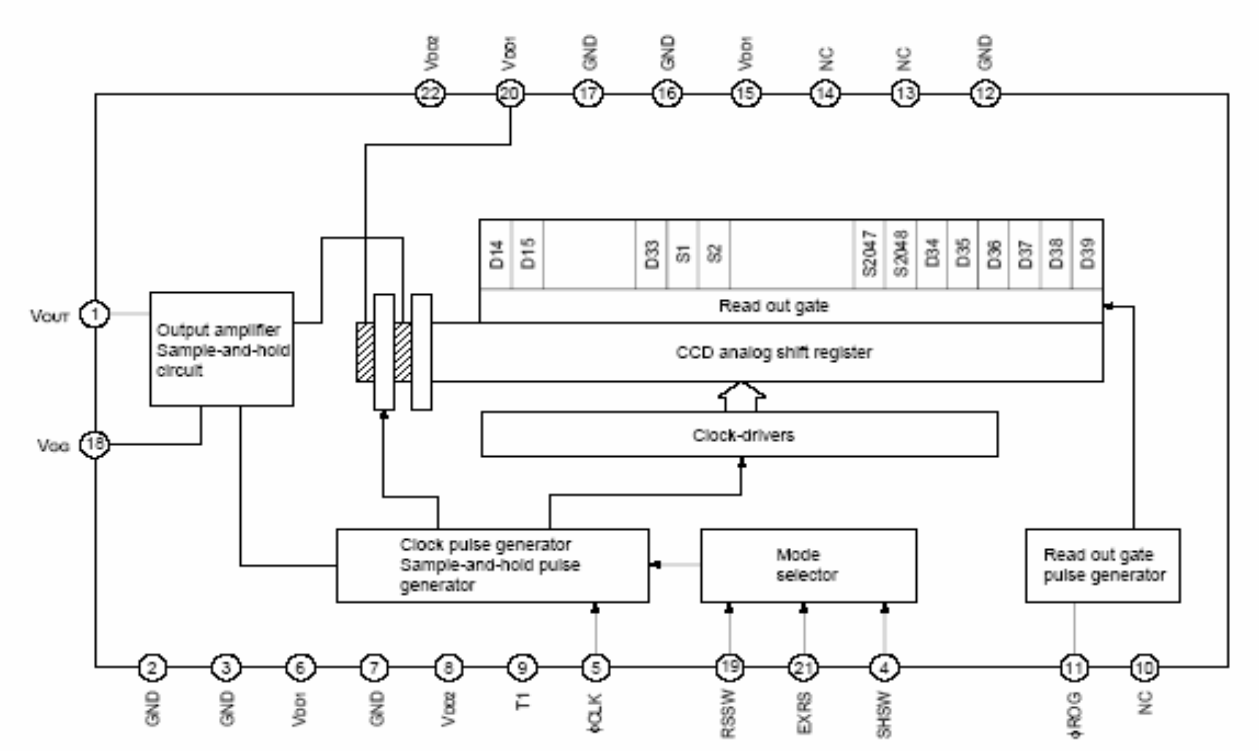
- celkem 2087 pixelů, 2048 efektivních (aktivních) signálových pixelů
- rozměr pixelu 14x14 μm
- maximální frekvecne vyčítání 5MHz
- ultra nízké zpoždění, řízení signály CMOS logické úrovně

Sony ILX503A má tedy z celkového počtu 2087 pouze 2048 efektivních pixelů. Zbyvají pixely mají tento význam a pořadí:

- 13 „dummy“ pixelů
- 18 zakrytých pixelů – jejichž video úroveň odpovídá oticky černé
- 2 „dummy“ pixely
- 2048 efektivních pixelů
- 6 „dummy“ pixelů

Výrobcem video úroveň „dummy“ není specifikovaná. Senzor ILX551A má stejné parametry, avšak není potřeba k němu přivádět externí resetovací signál EXRS.

Blokové schéma tohoto CCD senzoru je na *Obr. 3-2*



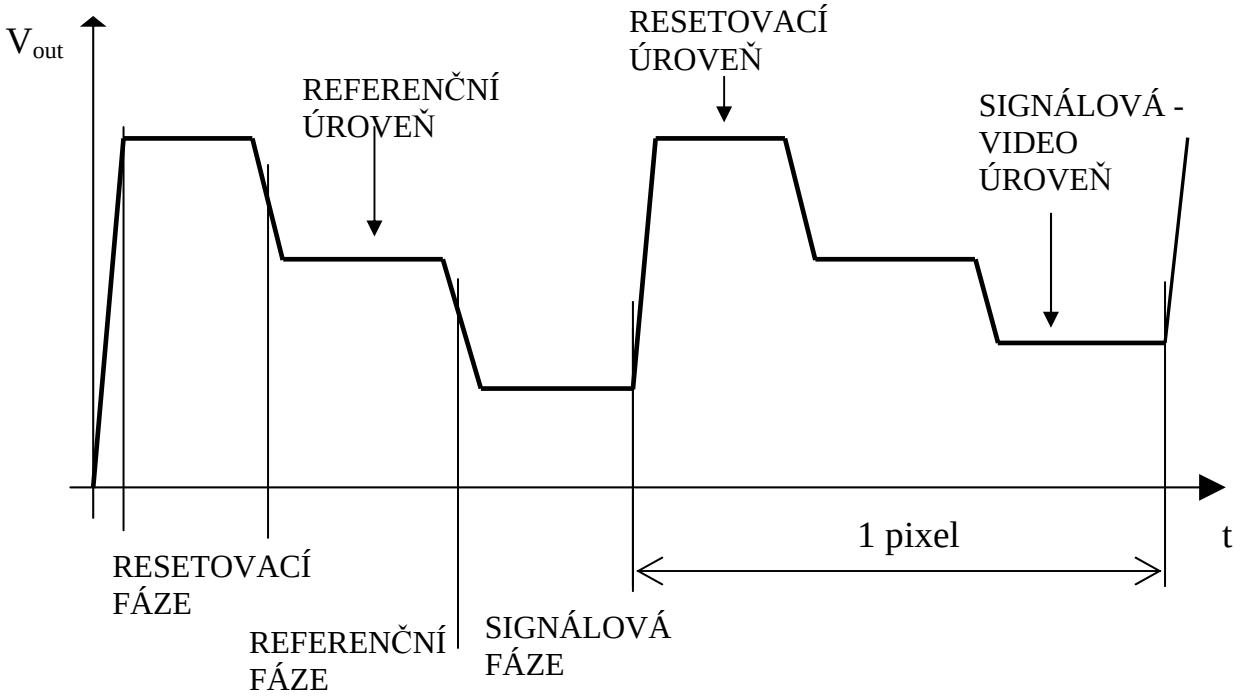
Obr. 3-2 Blokové schéma ILX503A

Význam jednotlivých řídících a synchronizačních signálů CCD snímače je následující:

- Φ_{ROG} (GROG) způsobí přesun naakumulovaného náboje z fotocitlivých elementů senzoru do posuvného registru. Také dobou periody tohoto signálu se řídí integrační doba akumulace náboje na fotoelementech.
- Φ_{CLK} (GCLK) řídí vysouvání náboje jednotlivých pixelů v posuvném registru ven na interní detektor (převodník náboj / napětí) CCD snímače
- Φ_{RST} (EXRS) resetuje interní nábojový detektor po každém přenosu pixelu. U senzoru ILX503A lze nastavit jak interní generování resetovacího signálu, tak přivedení externího resetovacího signálu. Senzor ILX551A tento vstup nemá.

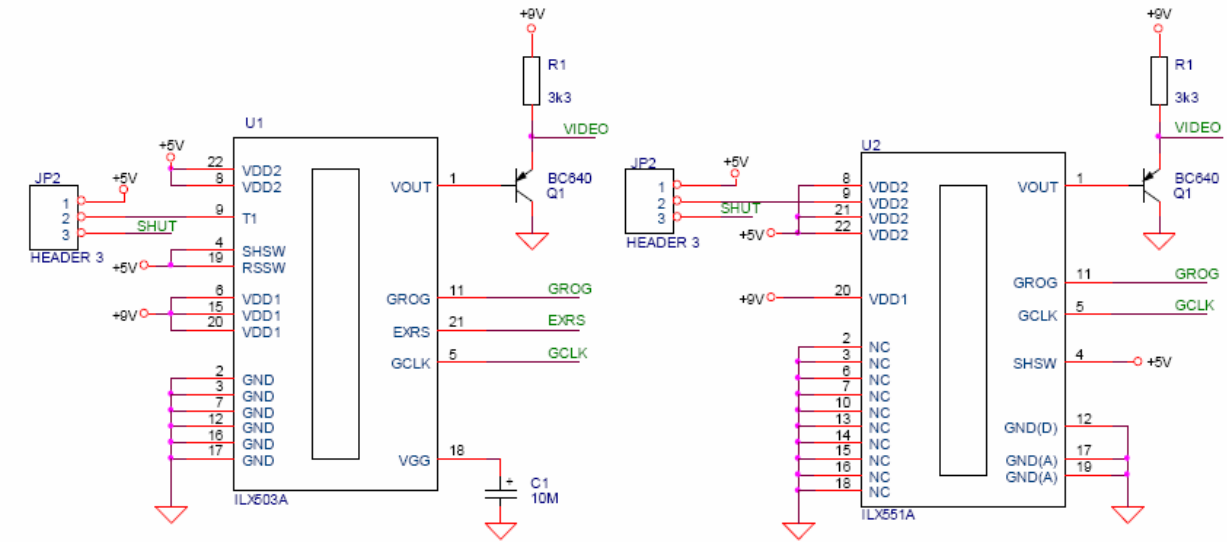
Bližší popis časování a další funkce celého obvodu viz příloha [4].

Na *Obr. 3-3* lze vidět typický průběh video-signálu na výstupu detektoru. Použitý CCD senzor má v sobě zabudovaný dvojitý korelovaný vzorkovač, či-li na výstupu senzoru je možno tedy mít signál viz *Obr. 3-3*, nebo signál, kterým má pouze pulzy, jejichž amplituda odpovídá přímo naakumulovanému náboji.



Obr. 3-3 Výstupní signál z řádkového CCD snímače

V rámci zpětné kompatibility a využití již navržených a hotových desek s CCD senzory jsem tedy převzal schémata z práce [7], což zvyšuje univerzálnost a variabilitu mé snímací desky, protože přes dohodnuté zapojení konetktoru, který bude popsán níže, lze připojit více typů CCD snímačů, které mají své vlastní desky plošných spojů.

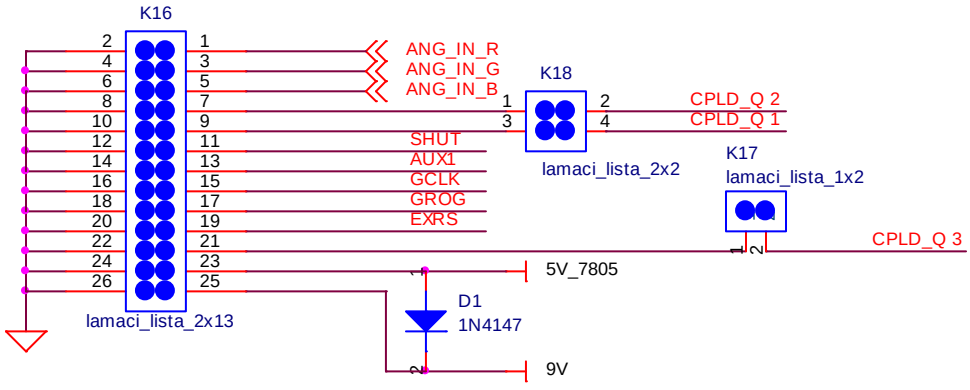


Obr. 3-4 Zapojení řádkových CCD snímačů ILX503/ILX703A a ILX551A/ILX751A

Volba mezi ILX503A a ILX703A (resp. ILX551A a ILX751A) se provádí konfigurační propojkou JP2 (obr. 30). Tato propojka připojuje na vývod 9 snímače buď +5V

pro ILX503A a ILX551A[1], nebo signál SHUT pro snímače ILX703A a ILX751A[2]. Snímače jsou na plošných spojích zapojeny [1]podle výrobcem doporučeného zapojení.

K této desce lze připojit i jiné typy CCD senzoru, než jenom ILX503A. Jak již bylo řečeno, mohou to být i tří barevné senzory, senzory s elektronickou závěrkou, nebo senzory, které nemají interní generátor hodin, které ovládají přenos náboje mezi jednotlivými elektrodami CCD registru, jako například senzor ILX 553A [8]. Připojení senzoru senzoru je přes standardní header konektor, viz *Obr. 3-5*.



Obr. 3-5 Schéma zapojení konektoru

Zapojení konektoru bylo sice rovněž převzato z [7], ale bylo upraveno tak, že původní země jsou nyní nahrazeny dalšími signály (CPLD_Q 1, CPLD_Q 2, CPLD_Q 3). Tyto nově přidané signálové vodiče mají v cestě vložen jumper, který řeší případné kolize – zkrat, s předchozími verzemi. Či-li je zde opět dodržena zpětná kompatibilita.

Význam jednotlivých signálů je následující:

- ANG_IN_R, ANG_IN_G, ANG_IN_B jsou výstupy z detektoru – sledovače CCD senzoru a vedou přímo na analogový vstup AFE.
- CPLD_Q 1, CPLD_Q 2, CPLD_Q 3 nemají pevně daný význam, pouze rozšiřují možnosti pro připojení více typů senzorů.
- SHUT řídící signál elektronické závěrky, AUX1
- GCLK, GROG, EXRS, viz význam jednotlivých řídících a synchronizačních signálů CCD snímače
- 5V_7805, 9V je napájení CCD senzoru

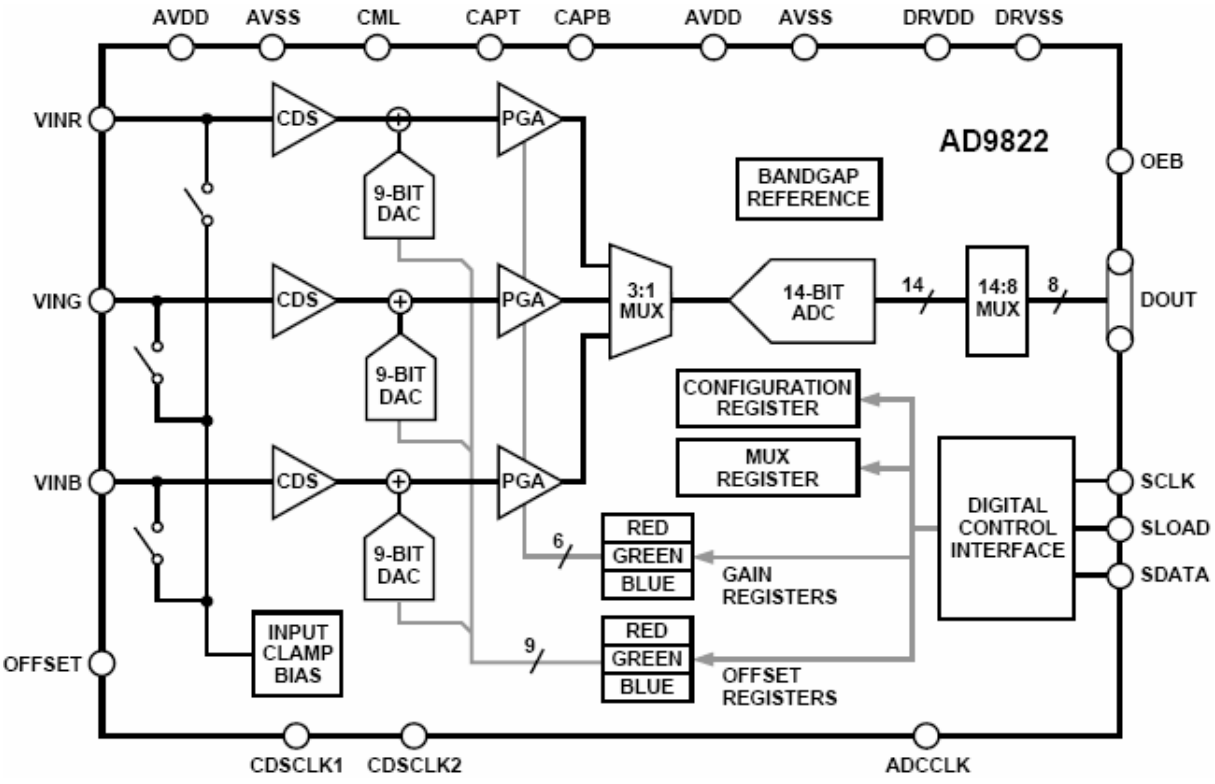
[1] Pokud nechceme u snímačů ILX703A a ILX751A ovládat elektronickou závěrku, lze jejich vývod 9 také propojkou připojit na +5V.
[2] Signál *SHUT* nesmí být připojen ke snímačů m ILX503A a ILX551A! Při *SHUT*=0V by mohlo dojít k poškození snímače.

3.3 CCD/CIS signálový procesor – AFE

Tento obvod je určen pro zpracování signálů z řádkového snímače. Jedná se v podstatě 14-bitový AD převodník. Lze jím digitalizovat signál jak z černobílích CCD, tak i z barevných CCD. Pro každou barevnou složku má tedy vlastní vstup (R,G,B).
Každý ze tří kanálů obsahuje:

- Vstupní upínací obvod pro korekci vstupního signálu na napěťový rozsah AD (3V nebo 4V)
- Obvod pro dvojí korelované vzorkování (CDS…Correlated Double Sampler)
- Obvod pro korekci offetu signálu (9 bitový DA převodník)
- Obvod s programovatelným zesílením signálu (PGA…Programmable Gain Amplifier)

Všechny tři kanály jsou před samotným 14-bitovým AD převodníkem multilexovány. Blokové schéma tohoto obvodu je na Obr. 3-6.



Obr. 3-6 Funkční blokové schéma AD9822

Přehled základních charakteristik a vlastností:

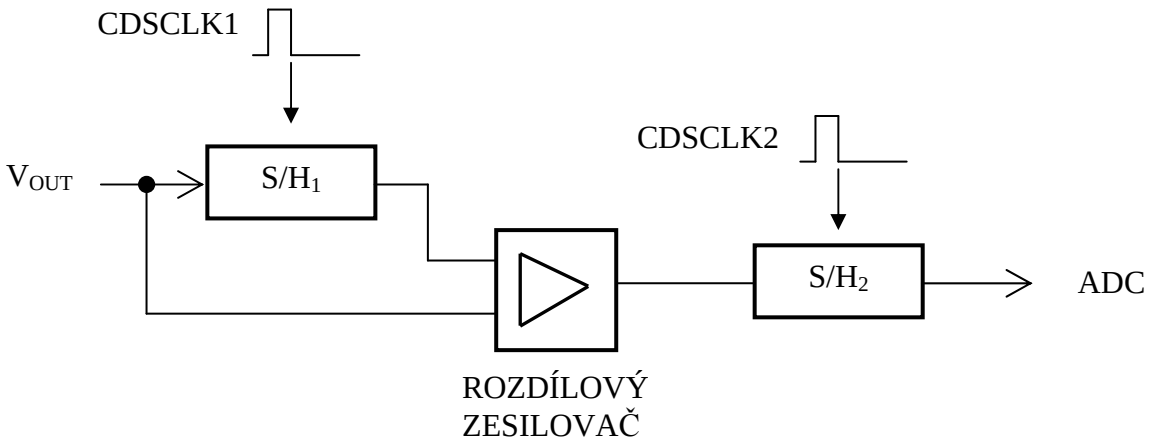
- Možnost digitalizace v jednokanálovém nebo tříkanálovém módu, kdy v jednokanálovém módu si lze vybrat ze tří libovolných vstupů (R, G, B)
- Pro jednokanálový mód je rychlost převodu 12.5 MSPS

- Pro tříkanálový mód je rychlost převodu 15 MSPS
- Individuální nastavení offsetu v rozsahu +-350mV po 512 krocích pro vstupy R,G, B
- Individuální nastavení zesílení v rozmezí 1x až 5,7x po 64 krocích pro vstupy R, G, B
- Zpoždění mezi okamžikem vzorkování signálu a okamžikem jeho digtalní podoby na výstupu AD jsou pevně dané 3 cykly signálu ADCCLK
- Vstupy / výstupy 3V/5V kompatibilní; CMOS
- 3-wire komunikace
- Volba módu vzorkování pro jednokanálový či tříkanálový mód
    - SHA – obyčejné vzorkování na spádovou hranu signálu CDSCLK2, kdy na vstupu ADC je signál, jehož hodnota napětí pulzu odpovídá naakumulovanému náboji na pixelu
    - CDS – využívá se na vstupu obvodu pro dvojí korelované vzorkování, popis viz níže a na *Obr. 3-7*.

3.3.1 Obvod dvojitého korelovaného vzorkovače

Průběh z *Obr. 3-3* představuje typický průběh signálu z CCD senzoru (vypnutý S/H režim, povolený CDS režim), určený pro obvod dvojítého korelovaného vzorkovače. Tento obvod je určen pro korekci vlivu resetovacího šumu.

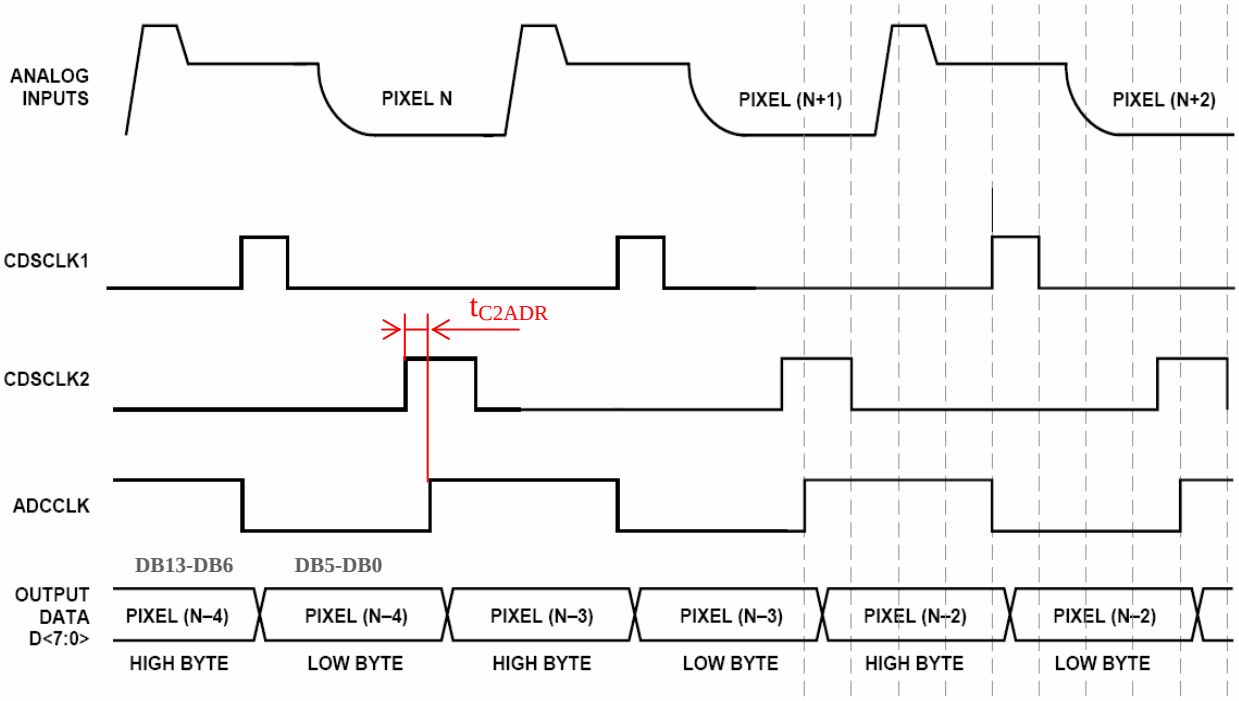
Vychází se ze skutečnosti, že stejnou aditivní chybou způsobenou resetovacím šumem je zatíženo jak referenční napětí, tak i signálové napětí. Výsledný užitečný signál je tedy dán rozdílem signálového napětí a referenčního napětí. Na *Obr. 3-7* je principíalní blokové schéma tohoto obvodu.



Obr. 3-7 Blokové schéma dvojitého korelovaného vzorkovače

Vzorkovací jednotka S/H_1 vzorkuje referenční úroveň video signálu. To se dejě v AFE na spádovou hranu pulzu signálu CDSCLK1. Vzorkovací jednotka S/H_2 vzorkuje signál na výstupu rozdílového zesilovače, kde na jeden jeho vstup je přivedena referenční úroveň ovzorkovaná obvodem S/H_1 a na druhý vstup je přivedena signálová úroveň. Tento rozdíl je tedy ovzorkován S/H_2 na spádovou hranu pulzu signálu CDSCLK2 v době tzv. signálové fáze. Na výstupu tohoto obvodu je již signál, jehož amplituda (velikost) odpovídá nábojovému svazku z CCD, či-li čím větší osvit CCD, tím vetší pulz.

Na Obr. 3-8 je vidět přehled jednotlivých důležitých signálů a fází pro AFE a jeho CDS. Signály CDSCLK1, CDSCLK2 a ADCCLK generuje jednotka GSP realizovaná CPLD, která bude popsána v následující kapitole. Chtěl bych jenom podotknout, že poloha signálů CDSCLK1 resp. CDSCLK2 je vůči sestupné resp. náběžné hraně signálů ADCCLK pro jednokanálový resp. tříkanálový režim jiná. U jednokanálového režimu se CDSCLK1, CDSCLK2 generuje po sestupné respektive náběžné hraně ADCCLK, kdežto u tříkanálového režimu před výskytem hrany ADCCLK.

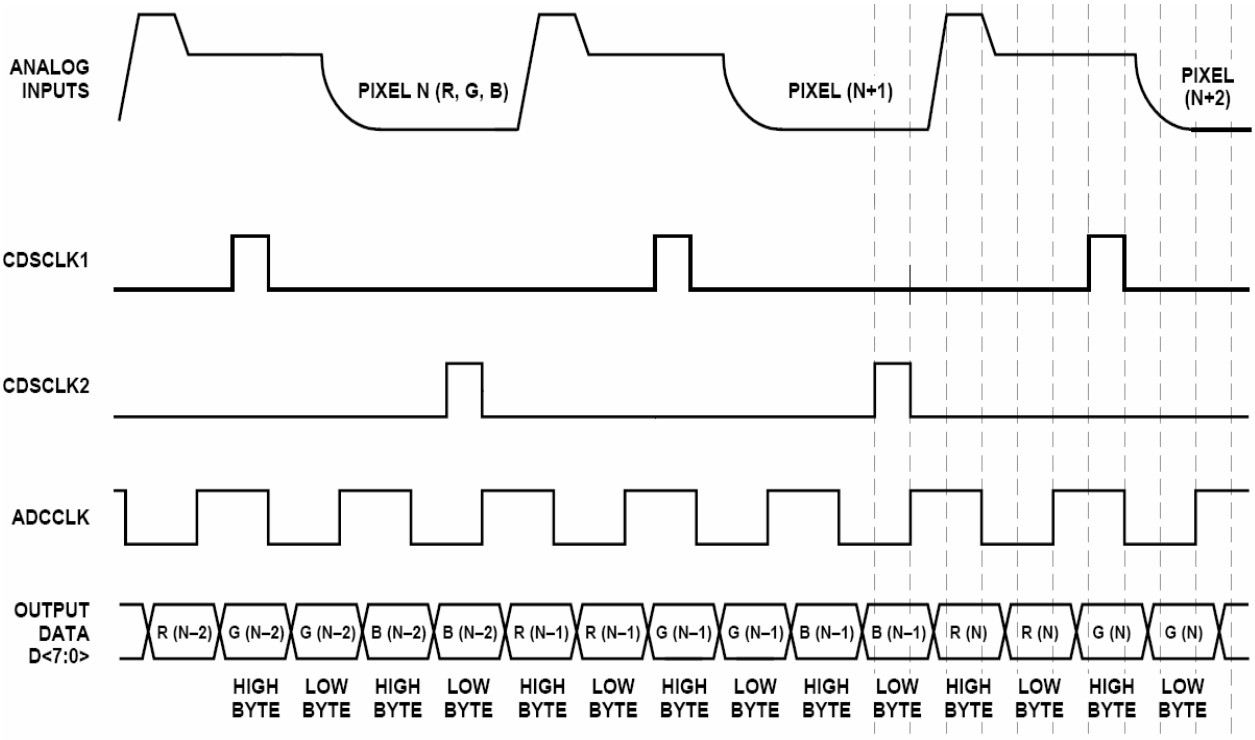


Obr. 3-8 Průběh časování jednokanálového CDS módu

U obou režimů jsou data na výstupu AFE řízena frekvencí ADCCLK. Horní bajt (High byte) pixelu je na výstupu při ADCCLK v logické 1, dolní bajt (Low byte) při logické nule. Formát takto přenášených dat je poněkud odlišný, než jsme zvyklí třeba v procesorech. Ve

standardním 16 bitovém formátu je 14 bitová informace uložena tak, že v dolním bajtu je dolních 8 bitů a v hodním bajtu je zbylých 6 bitů ze 14 bitů. Avšak u AFE je tomu trochu jinak z ryze praktického důvodu. Pro případ, že bychom chtěli pracovat pouze s horními 8 bity videoinformace, museli bychom provádět navíc logické operace jako rotace či posuny (pro standardní formát ukládání dat). AFE toto řeší elegantním způsobem. V horním bajtu není dle standardního fomátu 6 bitů, jak by se mohlo předpokládat, ale 8 bitů a v dolním bajtu je zbývajících dolních 6 bitů, či-li pro práci s horními 8 bity stačí vždy dolní bajt ignorovat (neukládat). High byte = DB13 ÷ DB6, Low byte = DB5 ÷ DB0.

Dále zde musím zdůraznit jeden důležitý fakt. V Obr. 3-8 je červeně znázorněn čas t_{C2ADR} , který v datasheetu má minimální hodnotu 0ns. Dále je tam také uvedeno, že tento čas by měl být pro jistotu větší než nula. Toto důrazně doporučuji, jinak nastanou velké problémy s převodem videosignálu, jestli vůbec k nějakému převodu dojde. Či-li předstih náběžné hrany signálu CDSCLK2 před náběžnou hranou signálu ADCCLK by měl být větší než nula. Stačí několik jednotek až desitek ns. V mém případě mám raději kolem 50ns, ale stačilo by i méně.



Obr. 3-9 Průběh časování tříkanálového CDS módu

3.3.2 Registry signálového procesoru – AFE

V následující kapitole popíši duležité registry, které je nutné brát v potaz pro oživení a spravnou funkčnost.

Nakonfigurovat AFE lze zápisem do jeho vnitřních registrů přes (3 wire) 3W rozhraní. AFE obsahuje celkem osm registrů, jejichž datová šíře je 9 bitů. Seznam registru je v následující Tab. 1.

Název registru	Adresa registru			Funkce registru
	A2	A1	A0	
Configuration	0	0	0	Základní konfigurace AFE
MUX	0	0	1	Nastavení kanálového multiplexeru
Red PGA	0	1	0	Nastavení zesílení "červeného" kanálu
Green PGA	0	1	1	Nastavení zesílení "zeleného" kanálu
Blue PGA	1	0	0	Nastavení zesílení "modrého" kanálu
Red offset	1	0	1	Nastavení offsetu "červeného" kanálu
Green offset	1	1	0	Nastavení offsetu "zeleného" kanálu
Blue offset	1	1	1	Nastavení offsetu "modrého" kanálu

Tab. 1 Seznam registrů AFE AD9822

První z registrů se nazývá Configuration registr, který obsahuje bity pro základní nastavení AFE. Jeho detailní popis je v Tab. 2 .Důležité je nastavit bity D8, D7, D1, D0 na logickou nulu. Bit D6 řídí interní zdroj referenčního napětí. Bit D5 vybírá mód činnosti AFE, tj. 3-kanálový mód (nastavením „1“) nebo jednokanálový mód („0“). Bit D4 zapíná či vypíná (1/0) obvod dvojitého korelovaného vzorkovače. Pakliže je vypnut, je v módu SHA a AFE se chová jako obyčejný AD převodník. Bit D3 zapíná vstupní upínací ovbod, který určuje, na jaké napětí bude vstupní signál upnut. AD9822 pomocí externího kondenzátoru odfiltruje stejnosměrnou složku z video signálu CCD senzoru a upne signál k hodnotě 3V nebo 4V. Nastavením bitu D2 uvedeme obvod do úsporného režimu.

Šedě zobrazené řádky představují výchozí a mnou vyzkoušené nastavení registrů AFE.

D8	D7	D6	D5	D4	D3	D2	D1	D0
		Interní Vref	**Počet kanálů**	**Funkčnost CDS**	**Upínací úroveň**	**Úsporný režim**		
0	0	1 = povolena	1 = 3 kanály	1 = CDS	1 = 4V	1 = zapnut	0	0
0	0	0 = zakázána	0 = 1 kanál	0 = SHA	0 = 3V	0 = vypnut		
0	0	1	0	1	1	0	0	0

Tab. 2 Nastavení Configuration registru

PGA registr nastavuje zesílení videosignálu pro každý kanál. Bity D8, D7, D6 je nutné nastavit do nuly. Zbyvající bity D5 až D0 nastavují požadované zesílení, od minimálního zesílení 1 (0 dB) až po maxímální zesílení 5,7 (15,1 dB). Celkem tedy možno 64 kombinací zesílení.

D8	D7	D6	D5	D4	D3	D2	D1	D0	Zesílení (V/V)	Zesílení (dB)
			MSB					**LSB**		
0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	1	1	1	1	1	5,7	15,1

Tab. 3 Nastavení PGA registru

$Zesílení = \frac{5,7}{1+4,7\left(\frac{63-G}{63}\right)}$. G je obsah registru PGA, respektive hodnota v registru PGA daná bity D5 až D0. Maximální zvětšení moc nedoporučuji, jelikož pak je již signál dosti zašuměn. Raději ať se nastavý delší integrační doba.

Offset registr nastavuje offset pro každý kanál v rozmezí -350 mV až +350 mV, dané bity D8 až D0. Celkem tedy možno 512 kombinací zesílení. Polarita (znaménko) zeílení je daná bitem D8. Offsetu lze využít pro případ, že AFE na vstupu není schopno v důsledku velikosti videosignálu dále pracovat (velký osvit senzoru), tak nastavením offsetu do záporných hodnot jakoby videosignál z CCD „stáhneme“ do urovně přijatelné pro zpracování.

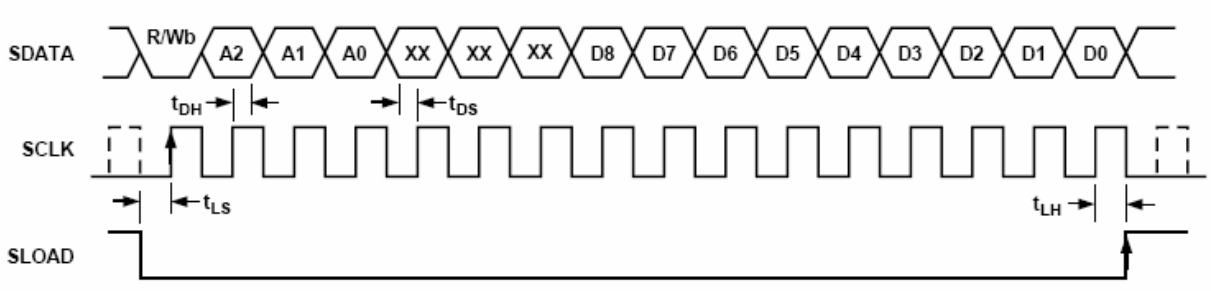
D8	D7	D6	D5	D4	D3	D2	D1	D0	Offset (mV)
MSB								**LSB**	
0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	1	-350
1	1	1	1	1	1	1	1	1	+350

Tab. 4 Nastavení offset registru

Zápis do registrů přes 3W rozhraní AFE realizuje SPI periférie procesoru BF532, jejíž nastavení je popsáno v 3.5.6.

Časování pro provedení seriového zápisu je na následujícím Obr. 3-10.

Bližší popis obvodu AD9822 v [3]



Obr. 3-10 Časování seriového zápisu

3.4 Generátor synchronizačních signálů - CPLD

Důvod proč pro generaci synchronizačních pulzů bylo vybráno právě CPLD, je především v univerzálnosti celé snímací desky. Protože CPLD jde snadno přeprogramovat pomocí JTAG rozhraní, takže pro netypycké časovaní jiného CCD senzoru není třeba složitějších úprav ať už software tak i hardware. Také by procesor BF532 svým počtem pinů z daleka nevyhovoval a hlavně by mu to neúměrně odebíralo jeho „drahý“ výpočetní výkon, který by se mohl věnovat jeho primární funcki, zpracování obrazové informace. Či-li výhody jsou zcela zjevné. Pro mou práci byl vybrán cenově dostupný obvod Xilinx XC9536XL.

Jeho základní charakteristiky a vlastnosti jsou následující:

- Zpoždění 5ns pin-to-pin
- Systémová frekvence až do 178MHz
- 34 uživatelem používaných-definovaných vstupů/výstupů
- 3,3V napájení; 2,5V / 3,3 / 5V napěťově tolerantní vstupy/ výstupy; CMOS technologie
- Full IEEE Standard 1149.1 boundary scan (JTAG)
- 10 000 programovacích/mazacích cyklů
- 72 makrobuněk

3.4.1 Princip a pořadí generace signálů

V této kapitole popíši postup a zduvodnění pro generaci synchronizačních signálů, které řídí přenos videoinformace de facto od pixelu až do procesoru.

Pro vyčtení celého řádku CCD je nutné dodržet následující postup a pořadí, který se neustále opakuje s dalším vyčítaním z CCD snímače. Na Obr. 3-12 je vidět principiální blokové schéma GSP.

3.4.1.1 Předdělička - generace hlavních signálů

Základní systémová frekvecne je odvozena od krystalu procesoru, tedy 24MHz. Tento hodinový signál je připojen přímo do 3 bitové děličky modulo 8, která vymezuje frekvenci pro CCD senzor, AFE a PPI rozhraní procesoru. V podstatě genereju přímo veškeré signály, jako CDSCLK1, CDSCLK2, GCLK, ADCCLK, PPI_CLK, EXRST. Volbou logické úrovně vstupního pinu CPLD_IN_1 se zvolí tří nebo jedno kanálový režim a podle toho fáze, frekvence a šířka těchto signálu v závisloti na módu (viz [3], Obr. 3-8, Obr. 3-9).

Zobrazení části VHDL kódu reprezentující předděličku modulo 8:

```
div_counter: process (IN_CLK, RESET)
        begin
                if (RESET = '1') then CLKD3 <= "000";
                elsif (IN_CLK'event and IN_CLK = '1') then
                        CLKD3 <= CLKD3 + 1;
                end if;
        end process;
```

Z [3] je patrné, že časování v 3-kanálovém módu je třeba třikrát signál GCLK zpomalit tak, aby během jedné periody signálu GCLK proběhly tři periody signálu ADCCLK. Z tohoto důvodu je nutné implementovat děličku modulo 3.

Generace hodinového signálu určeného pro GCLK v 3 kanálovém módu – dělička modulo 3:

```
three_mode_A : process (RESET,CLKD3(1), REG_NUMBER_IN, MOD3(1 downto 0))
begin
  if (RESET ='1') or (REG_NUMBER_IN = '1') or (MOD3 = "11") then MOD3 <= "00";
  elsif (CLKD3(1)= '0' and CLKD3(1)'event) then MOD3 <= MOD3 + 1;
  end if;
end process three_mode_A;
----------------------
modulo : process (MOD3, RESET, REG_NUMBER_IN)
begin
        if (RESET = '1') or (REG_NUMBER_IN = '1') then CLK_EDGE_A <= '0';
        elsif (MOD3 = "11") then      CLK_EDGE_A <= not CLK_EDGE_A;
        end if;
end process modulo;
```

Z uvedené ukázky kódu je patrné, že při hodnotě čítače rovno třem se automaticky čítač resetuje a invertuje se logická hodnota signálu GCLK (v ukázce reprezentovaná signálem CLK_EDGE_A).

Logika předěličky, která v závislosti na logické úrovni signálu REG_NUMBER_IN (CPLD_IN 1) generuje signály, které pokračují dále do Řídící logiky (stavový automat) ve zvoleném módu (3/1 kanálový mód). Pokud je logická úroveň REG_NUMBER_IN (CPLD_IN 1) v jedničce, generuje se signál přímo z předděličky modulo 8. Pokud je tato úroveň nulová, je signál ještě dělen v děličce modulo 3, takže se pixel frekvence třikrát zmenší 24MHz / 8 = 3 MHz / 3 = 1MHz. Čili je to nakombinováno tak, aby během jednoho pulzu GCLK proběhly tři pulzy ADCCLK, takže GCLK pro jednokanálový režim trvá osm tatků a pro tříkanálový režim trvá dvacet čtyři taktů vstupních hodin IN_CLK (24MHz).

Ukázka kódu logiky, která v závislosti na přivedené logické úrovni signálu REG_NUMBER_IN, řídí přenos hodin GCLK (CLK_EDGE_1) do Řídící logiky.

```
log_counter : process (CLKD3(2 downto 0), RESET, IN_CLK, REG_NUMBER_IN, CLK_EDGE_A)
        begin
                if (RESET = '1') then
                        CLK_EDGE_1 <= '0';
                        AFE_PPI <= '0';
                        AFE_CLK <= '0';
                        RST_1_OUT <= '0';
                elsif (IN_CLK'event and IN_CLK = '1') then
                        AFE_CLK <= CLKD3(2);
                        AFE_PPI <= CLKD3(1);
                if (REG_NUMBER_IN = '1' ) then
                                CLK_EDGE_1 <= not CLKD3(2);
                                RST_1_OUT <= not((not CLKD3(1)) and (not CLKD3(2)));
                        elsif  (REG_NUMBER_IN = '0' ) then
                                CLK_EDGE_1 <= CLK_EDGE_A;
                                RST_1_OUT <= not((not CLK_EDGE_A) and (not MOD3(1)));
                        end if;
                end if;
        end process log_counter;
```

V závislosti na zvoleném módu jsou generovány příslušné CDS signály, protože fáze mezi ADCCLK a CDS1, CDS2 je v každém módu jiná (viz. Obr. 3-8, Obr. 3-9 nebo [3]). Aby se efektivně šetřilo s vnitřními buňkami CPLD, využívám již generované signály a jejich vzájemnou kombinací tak generuji CDS signály, viz následující ukázka kódu pro jednokánálový mód.

```
CDS1_1 <= not CLKD3(2) and not CLKD3(1) and CLKD3(0) and not RESET_CDS and
REG_NUMBER_IN; -- 1 channel
CDS2_1 <= AFE_PPI and not RESET_CDS and not AFE_CLK and not CLKD3(0) and
REG_NUMBER_IN; -- 1 channel
CDS2_12 <= not AFE_PPI and not RESET_CDS and AFE_CLK and CLKD3(0) and
REG_NUMBER_IN; -- 1 channel
```

Generace signálu CDS1, který v důsledku využití signálu z děličky modulo 3, je nutný synchronizovat na hranu vstupních 24MHz hodin IN_CLK. Obdobně tak signál CDS2.

```
gen_CDS: process(IN_CLK)
begin
  if (IN_CLK'event and IN_CLK = '1') then
        CDS1_2 <= MOD3(1) and not MOD3(0) and CLKD3(2) and CLKD3(1) and CLKD3(0) and
not REG_NUMBER_IN and not RESET_CDS;
        CDS2_2 <= MOD3(1) and not MOD3(0) and not CLKD3(2) and CLKD3(1) and CLKD3(0)
and not RESET_CDS and not REG_NUMBER_IN;
  end if;
end process;
```

Je-li použit jednokanálový mód, tak CDS signály určené pro tříkanálový mód jsou nulovány a naopak. Proto lze ve výsledku oba signály logicky sečíst a na výstupu bude vždy ten tvar a časování signálu, který odpovídá svému zvolenému módu, viz následující ukázka VHDL kódu.

```
CDS1 <= CDS1_1 or CDS1_2;
CDS2 <= CDS2_1 or CDS2_2 or CDS2_12;
```

3.4.1.2 Řídící logika – hlavní čítač

Řídící logika se skládá z hlavního 14 bitového čítače. Na základě porovnání jeho hodnoty a pevně definované hodnoty se provede příslušná operace (uvedení příslušných signálu do požadovaných logických úrovní). V podstatě se jedná o stavový automat. Přehled duležitých stavů je shrnut v Tab. 5.

Pracovní frekvence řidící logiky je závislá na frekvenci signálu ADCCLK a je tedy pevně stanovena na 3MHz. Řídící logika generátoru synchronizačních signálů (dále jen GSS) sama generuje pouze dva signály GROG a PPI_FS1 a sprostřetkovává přenos signálu SHUT (elektronická závěrka), který je generován procesorem.

Čítač	GROG	GCLK	EXRST	ADCCLK	CDSCLK1	CDSCLK2	PPI_CLK	PPI_FS1
RESET	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**0**
0	1	x	x	1	0	0	0	0
1	1	1	x	1	0	0	0	0
3	0	1	x	1	0	0	0	0
33	1	1	x	1	0	0	0	0
35	1	x	x	1	0	0	0	0
68	1	x	x	x	x	x	x	1
16368	1	x	x	x	x	x	x	1
16369	1	x	x	1	0	0	0	0
16376	1	x	x	1	0	0	0	0

Tab. 5 Popis pořadí generovaní synchrnonizačních signálů

Při hodnotě čítače 0 jsou aktivní signály GCLK a EXRST (označené x). Do této hodnoty se také přechází, byla–li překročena hodnota čítače 16376. Při překročení hodnoty 16376 se čítání zastaví a čeká se na příchod sestupné hrnay signálu CPLD3 (časovač TMR2 z BF532), který svou periodou vymezuje integrační dobu. Hodnota čítače 16369 představuje maximální hodnotu počtu pixelů použitelného senzoru. Pokud přijde sestupná hrana signálu CPLD3 dříve než je 16369, respektive 16376, čitač přejde automaticky do stavu 0. Po sestupné hraně a patřičné synchronizaci se opět povolí čítání a celý cyklus se opakuje.

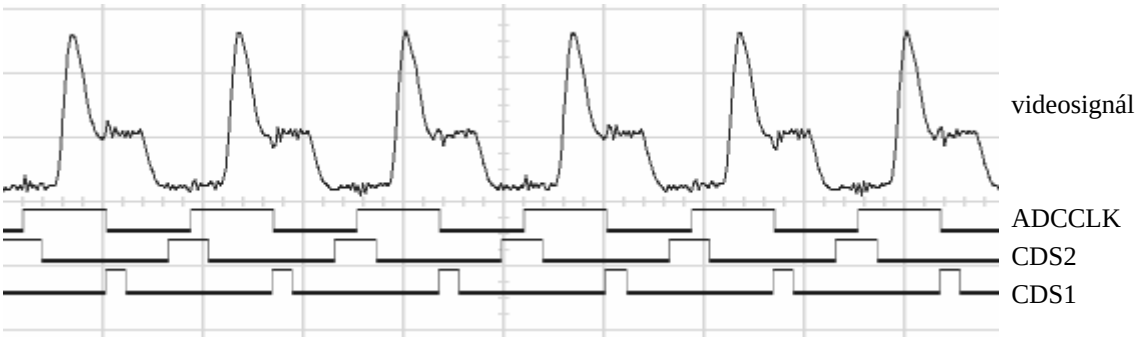
Signál GROG je v nule pouze tehdy, když se přesouvají náboje z fotocitlivých do posuvného CCD registru a to tehdy, když je tento registr v „ klidu“, či-li GCLK je také v jedničce a neaktivní. Délka pulzu GROG je od hodnoty čítače 3 do hodnoty 32 včetně, tedy 30 pulzů, což činí při pixel frekvenci 3MHz celkem 10µs. V datasheetu je udávána typická hodnota 1µs, ale je nutné pro zaručený přesun všech naakumulovaných nábojů tuto dobu přeci jenom zvýšit, čímž se zároveň zvýší citlivost a na celkovou rychlost vyčítání to nemá prakticky žádný vliv.

Signály pro PPI rozhraní a pro AFE jsou generovány až od hodnoty čítače 68, protože až od této hodnoty je na výstupu detektoru CCD snímače aktivní videoinformace. Význam jednotlivých pixelů je uveden v kapitole 3.2. Konec generace signálu pro PPI a AFE je od hodnoty 16369 včetně. Proč až tak vysoká hodnota, když pro můj senzor stačí hodnota 2158 = 2048+ 33 + 6 + 3 + 35 (aktivní pixely, první černé a dummy pixely, poslední dummy pixely, tři takty navíc z důvodu zpoždění tří cyklů AFE, posunutí začátku generace signálů)? Důvod je jasný a to použitelnost pro senzory s různým počtem pixelů. Daní pro toto rozšíření byl problém s přenosem korektního řádku. Vysvětlení a vyřešení tohoto problému lze nalézt v kapitole 3.5.3.

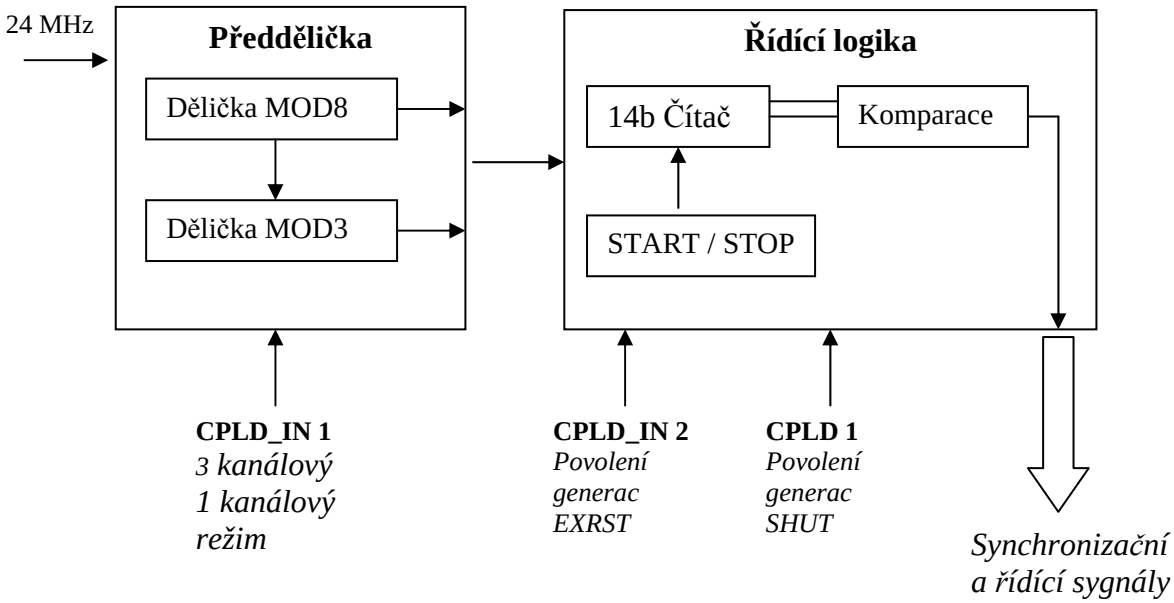
Hodnotou 2158 čítače a pixelovou frekvencí je vymezena minimální integrační doba pro můj senzor, či-li doba akumulace náboje ve fotocitlivých elementech. Ta je dáná následující rovnicí:

$$T_{INT} = \frac{\mathrm{COUNT}}{f_{pixel}} = \frac{2158}{3.10^6} = \underline{719{,}3\,\mu s} \qquad \textbf{(25)}$$

Kde COUNT je minimální možná hodnota čítače odpovýdající minimálnímu počtu cyklů GCLK pro vypsání jednoho řádku z CCD snímače. Nižší hodnota je přípustná pouze pro CCD snímače s nižším počtem pixelů, či-li počet pixelů vymezuje tuto hodnotu. Dále f_{pixel} je pixelová frekvence. Ta je pevně dána na 3 MHz.



Obr. 3-11 Zobrazení signálů CDS1, CDS2, ADCCLK a videosignálu na osciloskopu



Obr. 3-12 Zjednodušené principiélní blokové schéma generátoru synchronizačních signálů

Následují ukázky VHDL kódu pro přiblížení této problematiky.

Process *main_counter* realizuje 14 bitový čítač, který na základě signálů *STOP_COUNT* a *STOP_DISABLE* zastavuje (resetuje) či spouští čítání.

```
main_counter : process (RESET, ADCCLK_EDGE, STOP_COUNT, STOP_DISABLE)
begin
        if (RESET = '1') or (STOP_COUNT = '1') or (STOP_DISABLE = '1') then COUNT <=
"00000000000000";
        elsif (ADCCLK_EDGE = '0' and ADCCLK_EDGE'event) then
                COUNT <= COUNT + 1;
        end if;
end process main_counter;
```

Zastavení či znovu spuštění čítání je velmi důležité z několika důvodů. Především 14b velikost čítače a pevně dané 3MHz frekvence limituje velikost integrační doby na maximálně

$\frac{16376}{3\text{MHz}} = 5{,}459\ ms$, což je velmi omezující. Proto pro delší integrační doby je nutné čítání zastavit, aby se nespustil znovu cyklus vyčítání. A naopak, pokud potřebujeme vyčítat dříve, čili použijeme menší integrační dobu (méně než zmíněných 5,459 ms), je nutné vyčítání regulérně a synchronně zastavit (*STOP_DISABLE=1*) a spustit jej znovu. Start nového vyčítání je řízen na spádovou hranu signálu CPLD3 (časovač TMR2 procesoru BF532), který tím určuje i integrační dobu. Process *over_count* řeší problém při delších integračních dobách, kdy po překročení hodnoty 16376 čítání zastaví logickou úrovní *STOP_COUNT = 1*.

```
over_count : process (COUNT, RESET, STOP_DISABLE)
begin
        if (RESET = '1')  or (STOP_DISABLE = '1') then STOP_COUNT_1 <= '0';
                                        --          16376
        elsif COUNT >= "11111111111000" then STOP_COUNT_1 <= '1';
        end if;
end process over_count;
```

Pokud příjde spádová hrana signálu CPLD3 a synchronizuje se na náběžnou hranu signálu ADCCLK, aktivuje se signál *STOP_DISABLE* , který časovač uvede do výchozího stavu a zruší se indikace zastavení čítání (pokud byla aktivována). Tímto je zaručeno regulérní znovu spuštění čítání 14b čítače.

Další procesy-funkce uvádět nebudu, jelikož pouze reflektují stavy uvedené v Tab. 5.

3.4.2 Popis pinů a ovládání GSS

V této kapitole popíši jednoduché ovládání GSS a význam jednotlivých pinů – vstupně / výstupních signálů.

Na Obr. 3-13 je schéma zapojení CPLD ve funkci GSS. Signály CPLD1, CPLD2, CPLD3, CPLD4 vedou přímo od procesoru, kde signály CPLD2 a CPLD3 jsou od časovačů procesoru TM0, TMR2 a signály CPLD1 a CPLD4 od PF11, PF10 procesoru. Signál CPLD2 je použit jako externí signál pro vymezení integrační doby CCD snímače. Signál CPLD3 je použit ro generaci pulzu elektronické závěrky. Signál CPLD1 je použit pro povolení/zakázání (L/H) generace signálu EXRST. Zbývající signál CPLD4 nemá zatím využití. Je zde pro budoucí eventuální použití.

Signál IN_CLK je použit jako vstupní hodinový signál vedoucí přes oddělovací invertor přímo od krystalu procesoru, takže frekvence vstupních hodin je 24MHz.

Signály ADCCLK, CDS1, CDS2 jsou signály pro AFE, které řídí převod - digitalizaci, blíže v kapitole 3.3. Frekvence signálu ADCCLK je 3MHz. Naproti tomu šířka

pulzů signálů CDS1 a CDS2 je odvozena od vstupních hodin IN_CLK, tedy 42 ns a 84ns. Výrobcem požadovaná minimální šířka pulzů CDS1 a CDS2 obvodu AFE je 10 ns, což je splněno.

Signály PPI_CLK, PPI_FS1 jsou řídící signály pro rozhraní PPI procesoru, kde na každou náběžnou hranu signálu PPI_CLK je uskutečněn transport dat z AFE přes PPI pomocí DMA do paměti procesoru. Frekvence signálu PPI_CLK je stejná jako frekvence signálu ADCCLK. Vyšší bajt pixelu je na výstupu AFE při ADCCLK v logické „1“, zatímco nižší bajt (6bitů) je v logické nule. Jak již bylo vysvětleno v kapitole 3.3.1, přenáší se pouze horní bajt, je tedy frekvence PPI_CLK stejná s ADCCLK. V opačném případě by byla frekvence PPI_CLK dvojnásobná pro přenos obou bajtů patřících k danému pixelu.

Délka signálu PPI_FS1 vymezuje délku – potvrzuje platnost přenášených dat z AFE do procesoru.

Signály SLOAD_CPLD, SCLK, SDATA jsou signály pro vnitřní posuvný registr, který z důvodu nedostatku potřebného množství makrobuňek není zatím implementován. SLOAD_CPLD je tedy použit jako celkový reset CPLD.

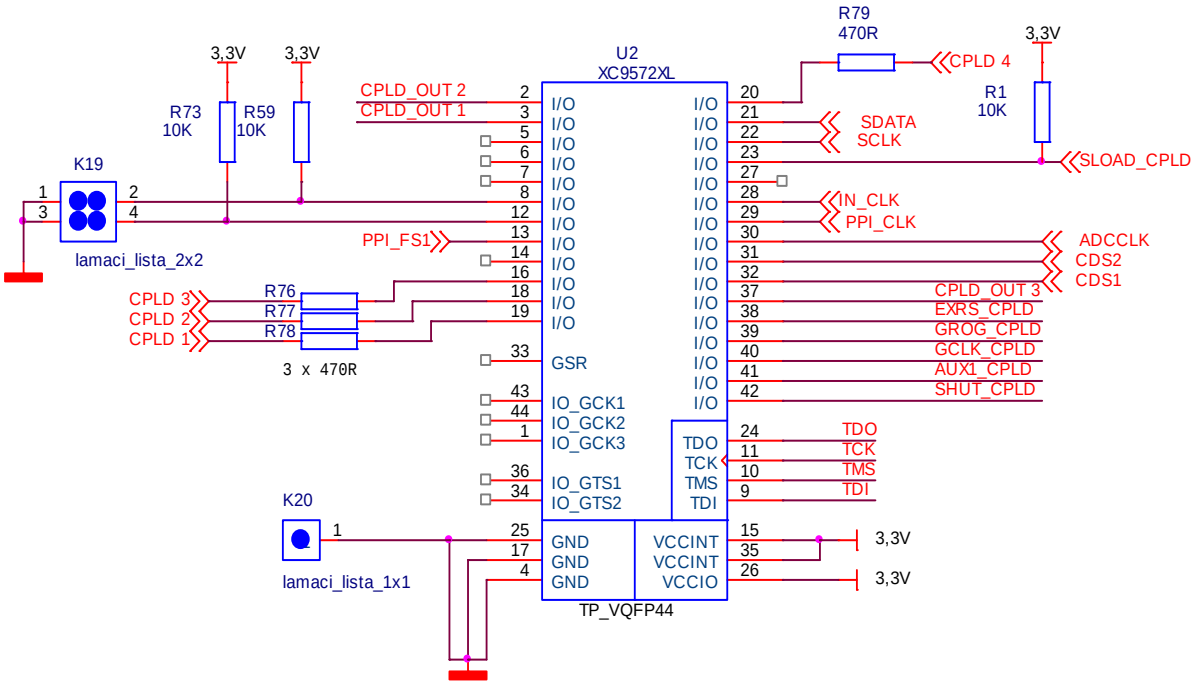
Signály na pinech 8 a 12 CPLD (lamací lišta) jsou přidavné konfigurační signály, které rozšiřují konfigurovatelnost CPLD. CPLD_IN1 pro volbu módu, 3 kanálový / 1 kanálový mód, (L/H). CPLD_IN2 pro povolení/zakázání generace signálu SHUT, či-li elektronické závěrky (L/H).

Signály JTAG rohraní TDO, TCK, TMS, TDI umožňují ISP naprogramování CPLD. Tyto piny jsou vyvedeny na lamací lištu umožnující připojení programovacího obvodu.

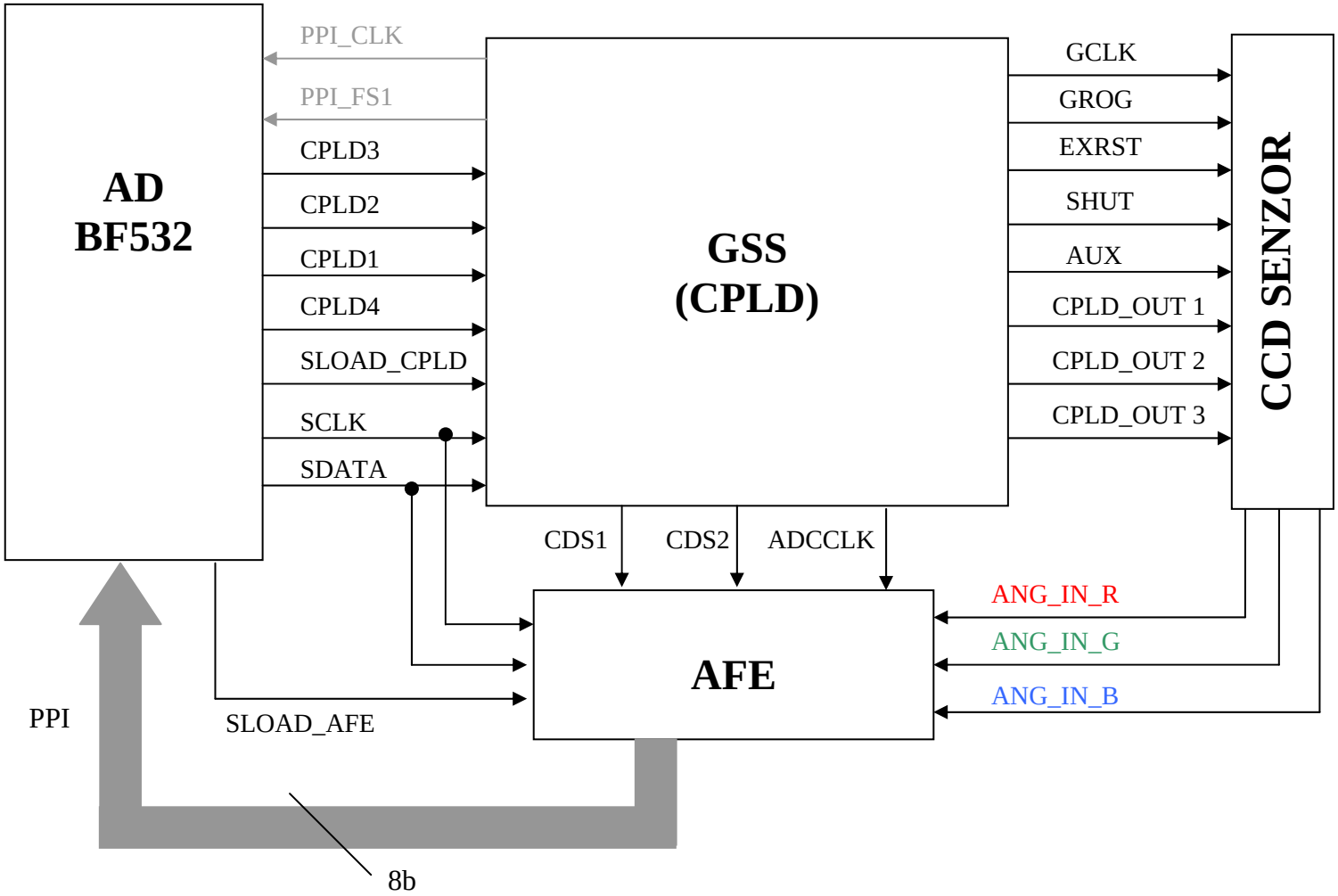
Zbývající signály CPLD_OUT1, CPLD_OUT2, CPLD_OUT3, EXRS_CPLD, GROG_CPLD, GCLK_CPLD, AUX1_CPLD, SHUT_CPLD jsou popsány výše a bylo o nich zde již několikrát hovořeno.

	Význam signálu	Povolení	Zakázání
CPLD_IN 1	3 kanály / 1 kanál	0	1
CPLD_IN 2	Generace EXRST	0	1
CPLD 1	Generace SHUT	0	1
SLOAD_CPLD	Reset CPLD	0	1

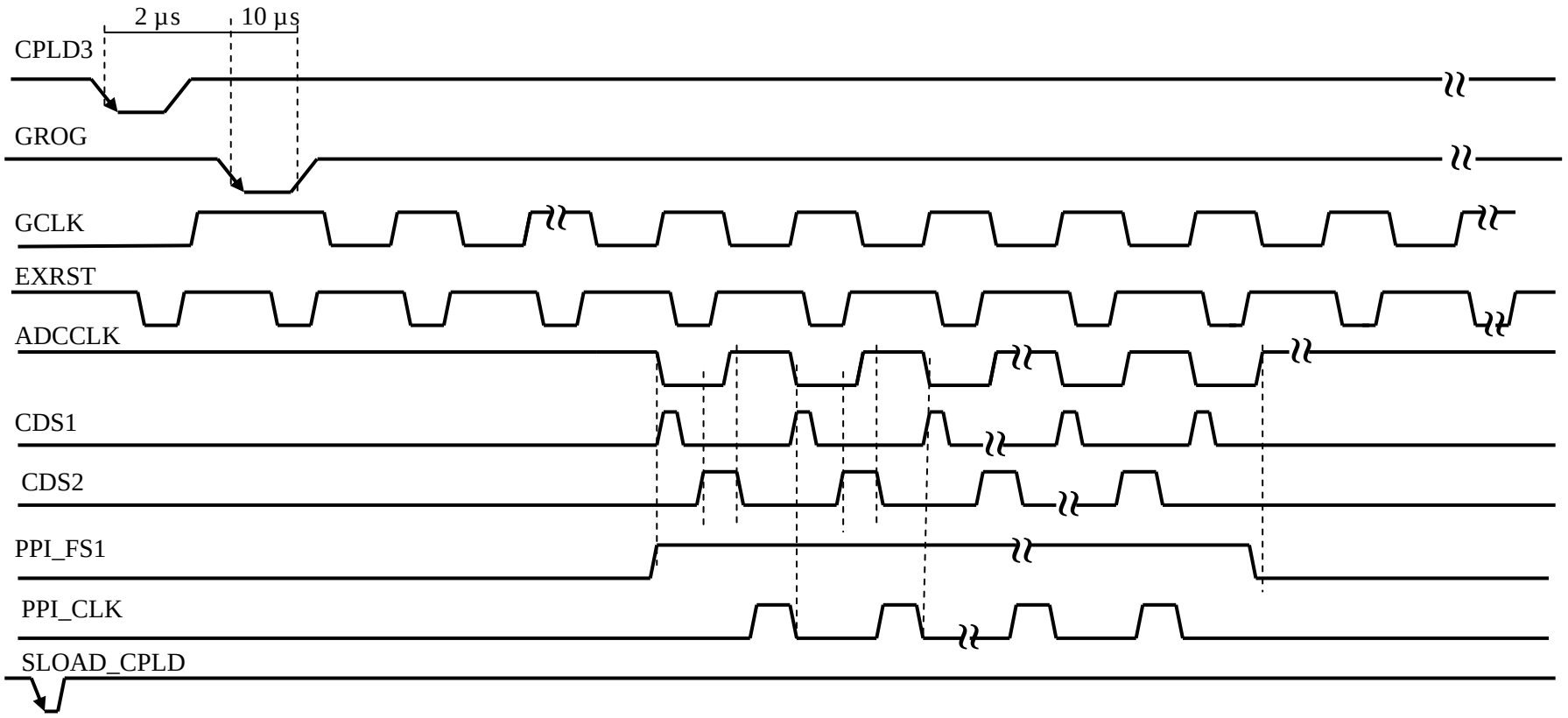
Tab. 6 Nastavení GSS



Obr. 3-13 Schéma zapojení CPLD jako generátoru synchronizačních signálů-GSS



Obr. 3-14 Blokové schéma propojení CPLD



Obr. 3-15 Časový diagram generace významných signálů a komunikace CPLD

3.5 ADSP - BF 532

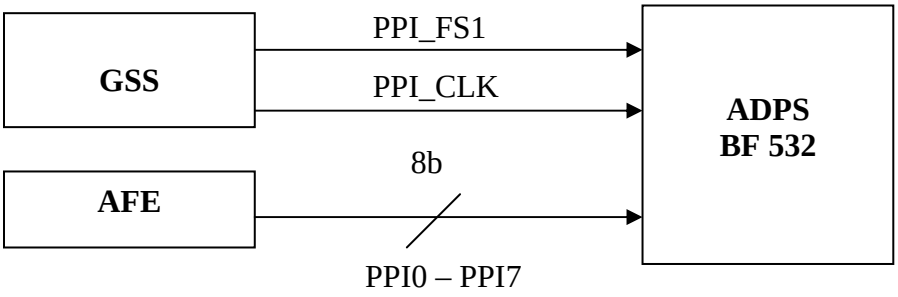
V této kapitole popíši hlavní rysy tohoto procesoru. Jedná se o signálový RISC procesor od firmy Analog Devices z rodiny procesorů Blackfin. Je schopen pracovat až do kmitočtu 400MHz pomocí vnitřní násobičky kmitočtu a obvodu PLL.

- Dvě 40 bitové ALU, čtyři 8 bitové video ALU, 40 bitový posuvný registr, dvě 16 bitové násobičky
- Obsahuje až 32KB programové paměti a až 32KB datové paměti.
- Flexibilní paměťové bootování přes SPI rozhraní a externí paměť
- PPI rozhraní podporující ITU-R 656 video datový formát
- Dva duální kanály plně duplexní synchronní seriové porty
- UART, SPI
- 12 kanálový DMA řadič
- tři čítače/časovače podporující generaci PWM
- Real-time hodiny, watchdog timer, PLL obvod až pro 63 násobné zvetšení frekvence krystalu pro jádro procesoru
- Podpuruje SDRAM, SRAM, FLASH, ROM
- JTAG/ debug rozhraní
- 16 programovatelných pinů

Detailnější popis je v [2].

3.5.1 PARALLEL PERIPHEAL INTERFACE - PPI

Jená se o poloduplexní obousměrné rozhoraní až do 16 bitů (PPI0-PPI15). Třináct bitů je sdíleno s programobvatelnými piny (PF – programmable flag). Jelikož AFE má velikost praralelního rozhraní osm bitů, je na snímací desce využito rovněž osm bitů (PPI0-PPI7). Dále je použit synchronizační bit přenosu PPI_FS1 a hodinový signál PPI_CLK. Či-li pro přenos dat z AFE je potřeba celkem 10 vodičů.



Obr. 3-16 Blokové schéma přenosu přes PPI

Pro správný přenos je nutné nastavit následující registry:

- **PPI_CONTROL Register –** konfigurační registr, který nastavuje citlivost na hrany (náběžná / spádová), datovou šířku, polaritu přenosu, mód činnosti atd. (0x000D).
  *pPPI_CONTROL = NON_ITUR656 | POLC | PORT_EN;
- **PPI_DELAY Register –** hodnota uležená v tomto registru určuje, za kolik cyklů PPI_CLK se po vyskytnutí přenosové podmínky dané signálem PPI_FS1 budou přenášet data (čtení/zápis). Z důvodu tří cyklového zpoždění dat na datovém výstupu AFE se do toho registru uloží hodnota 6, protože data na datovém výstupu AFE jsou na každou hranu signálu ADCCLK pro přenos 2B. Jinak se do tohoto registru uloží 3, pro přenos pouze 1B (horních 8 bitů).
  *pPPI_DELAY = 3;
- **PPI_COUNT Register** – hodnota v tomto registru určuje počet vzorků – přenosů, takže např. chceme-li přenést jeden řádek z CCD, který má 2048 pixelů, do tohoto registrů se uloží hodnota 2047.
  *pPPI_COUNT = 2047;

Pin PPI_CLK procesoru může mít na svém vstupu frekvenci až do SCLK / 2. Jelikož SCLK se nastavuje v obvodu obsahující PPL, který má základní frekvenci odvozenou od 24 MHz krystalu, je možno na pinu PPI_CLK připojit frekvenci až 39MHz.
PPI je efektivní pouze v součinosti s obvodem DMA, který stručně popíši v kapitole 3.5.2.

3.5.2 DIRECT MEMORY ACCESS – DMA

Procesor používá DMA pro přenos dat do paměťového prostoru nebo mezi paměťovým prostorem a periférií. Procesor může specifikovat operaci datového přenosu a vrátit se k normální činosti (jiné zpracování), zatímco plně integrovaný řadič DMA přenáší

data nezávisle na aktivitě procesoru – procesor nezatěžuje. Systém obsahuje šest DMA schopných periférií včetně paměti.

DMA řadič může vykonat několik typů datových přenosů:

- mezi pamětí a pamětí (MDMA)
- mezi pamětí a SPI
- mezi pamětí a SPORT
- mezi pamětí a UART port
- mezi pamětí a PPI

Nyní popíši mnou nastavenou konfiguraci registrů pro správný přenos:

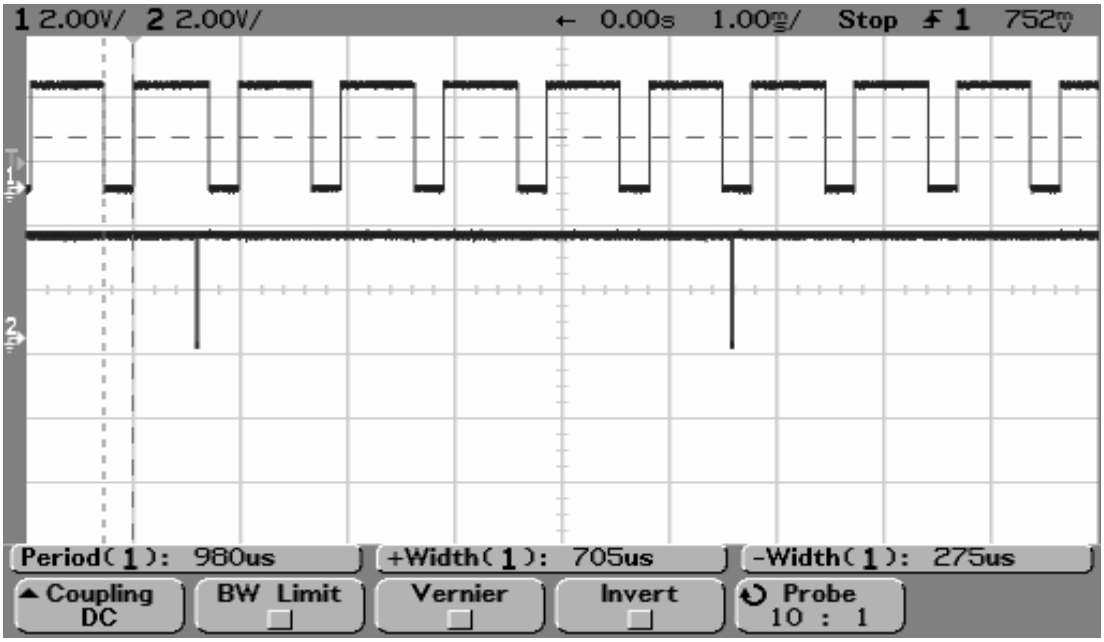
- **DMA0_START_ADDR** = *adresa*; - obsahuje počáteční (cílovou) adresu, odkud se začnou data přenášet.
- **DMA0_X_COUNT** = *2048*; – pro 1D přenos specifukuje počet přenášených elementů (v našem případě pixelů na řádek), pro 2D přenos specifikuje počet řádků.
- **DMA0_X_MODIFY** = 0x1; – jedná se o ingrement registr, který podle daného obsahu inkrementuje adresu po každém přenosu, či-li pro bajtové přenosy bude obsahovat 0x1, pro 16b přenosy bude obsahovat 0x2.
- **DMA0_PERIPHERAL_MAP** = 0x0; podle své hodnoty určuje přenosový kanál, či-li o jakou periferii se bude jednat (PPI, SPORT0 RX, SPORT0 TX, SPORT1 RX, SPORT1 TX, SPI, UART RX, UART TX) či jestli se bude jednat o periferijní přenos nebo o přenos paměťový (uvnitř paměti).
- **DMA0_CONFIG** = DMAEN | DI_EN | WNR | WDSIZE_8 | RESTART; (0x00A3)

– konfigurační registr pro nastavení DMA parametrů a operačních módů, polaritu přenosu, datovou šíři atd.

3.5.3 Dodatek k nastavení PPI, DMA a správnému přenosu jednoho řádku

Jak již bylo v kapitole 3.4.1 řečeno, neúměrnou délkou signálu PPI_FS1 oproti potřebné délce danou počtem aktivních pixelů, docházelo k nesprávnému přenosu digitalizovaných dat řádku, což způsobovalo jeho „útíkání“. Jelikož v procesoru zapínám periférie PPI a DMA vždy podle potřeby, tedy asynchronně, a v jakékoliv fázi vyčítání řádku a podle milného předpokladu vyčteného z datasheetu, že dané množství dat (pixelů) se přes PPI-DMA načte a pak se čeká na další hranu signálu PPI_FS1. Domnívám se, že po načtení daného množství dat (pixelů), kdy jsem obě periferie vypnul, nedošlo asi k vypnutí indikace

hrany synchronizačního signálu PPI_FS1. Jelikož přenos 2048 B při frekvenci 3MHz trval přibližně 700 µs a integrační doba trvala podstatně déle, například 15ms, docházelo při opětovném zapnutí PPI a DMA k milné interpretaci celkového stavu (např. logické úrovně signálu PPI_FS1), který byl v podstatě po celou dobu trvání akumulace náboje v CCD v aktivní úrovni. Na následujícím obrázku Obr. 3-17 si lze clekový stav přiblížit.

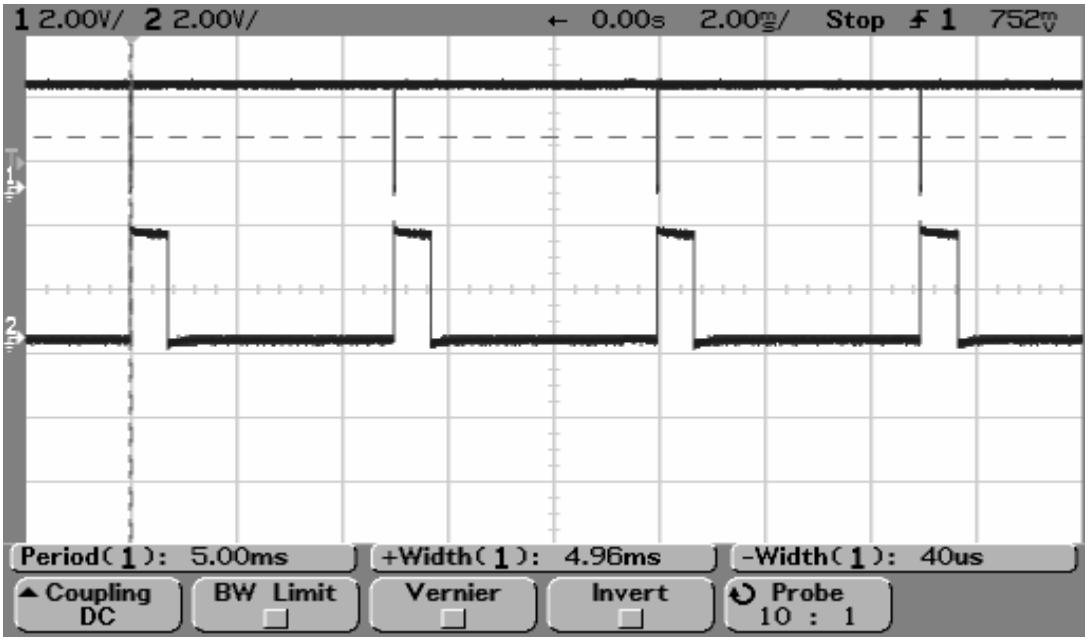


Obr. 3-17 Indikace milného přenosu dat řádku

Pro zobrazení přenosu dat z AFE přes PPI-DMA jsem si vždy nahodil binární výstup (indikační bit) procesoru do logické jedničky a po zkončení přenosu do logické nuly. Z obrázku Obr. 3-17 je patrné, že se jediný řádek navzorkoval přibližně pětkrát, přičemž ani jeden přenos nebyl správný, respektive data neodpovídala nasnímané scéně. Začátek dalšího řádku je indikován krátkým poklesem PPI_FS1 do nuly. Tímto opakovaným přenos během jediného vyčítání řádku docházelo k již zmíněnému „utíkání“ řádku. Občas se přenos trefil do správné fáze vyčítání řádku a řádek se zobrazil na monitoru ve více méně správné podobě, tedy abych byl přesnější uvedu malý příklad.

Snímal jsem předlohu v podobě „zebry“, sled bílých a černých pruhů stejné šíře, jenom v prostřed byl jeden proužek širší. Snímanou scénu jsem si zobrazoval na monitoru v podobě jasového profilu řádku. Či-li „utíkáním“ řádku jsem mínil, že snímaná zebra se nezobrazovala stále ve stejném tvaru jasového profilu, ale pohybovala se podle toho, jak jsem se trefil startem přenosu (AFE – PPI - DMA). Takže např. širší prostřední pruh se zobrazil na kraji, nebo se zebra nezobrazila vůbec, jelikož se přenášeli již prázdné černé pixely, které se neustále vypisují z CCD senzoru.

Bylo tedy nutné řádek jaksi „chytit“. To jsem realizoval tak, že jsem přenos spustil vždy a pouze na konci periody integrační doby (časovač TMR2), čímž jsem zajistil synchronizaci, která se zřejmě po každém vypnutí periferie PPI ztratila. Obdobně jsem si tento přenos zobrazil na osciloskopu pomocí indikačního binárního výstupu procesoru, viz následující obrázek.



Obr. 3-18 Indikace správného přenosu dat řádku

3.5.4 TIMERS

Procesor obsahuje 3 identické obecně použitelné 32b čítače/časovače. Každý timer má na čipu vyhrazený jeden obousměrný pin.

Každý čítač / časovač může být individuálně nastaven v jendom ze tří módů:

- režim generace pulsně šířkové modulace PWM_OUT
- režim měření šířky pulzu WDTH_CAP
- režim zachycení externí události EXT_CLK

TIMER_PERIOD a TIMER_WIDTH:

32 bitové registry, které svou hodnotou určují velikost periody, respektive velikost (šířku) pulsu zvolené aktivní úrovně (nastaveno v registru TIMER_CONFIG). Výhodou těchto registrů, že je lze měnit za běhu (tzv. on-the-fly).

TIMERx_PERIOD = f_{SCLK} / f_{out} , kde f_{SCLK} je hodinový kmitočet procesoru a f_{out} je výstupní frekvence.

Činitěl plnění = (TIMERx_WIDTH / TIMERx_PERIOD) .100 %

Registr může mít hodnotu od nuly až do hodnoty dané v registru TIMERx_PERIOD – 1, či-li tím je míněno, že nelze nastavit aktivní úroveň po celou periodu.

Časovač TMR2 (pin č. 77) je použit pro určování doby integrace. Pro zvolenou frekvenci systémovou procesoru SCLK 79 MHz je možno nastavit integrační dobu maximálně:

$T_{INT} = \frac{2^{32} - 1}{79MHz} = 54{,}37s$. Samozřejmě pokud se nastaví SCLK menší, lze tuto dobu ještě prodloužit, avšak zpomalí se přístup k okolním periferiím.

Časovač TMR0 je zvolen pro řízení elektronické závěrky na senzoru.

(*pTIMER2_CONFIG = IRQ_ENA | PWM_OUT | PERIOD_CNT | EMU_RUN;)

(*pTIMER0_CONFIG = PULSE_HI | PWM_OUT | PERIOD_CNT | EMU_RUN;)

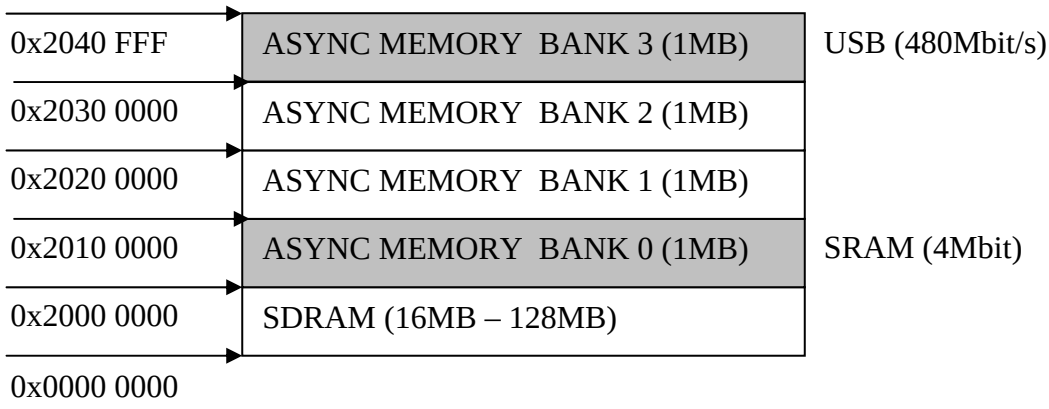
Následují část kódu ukazuje, jak jsem řešil synchronní spouštění přenosu, abych „chytil“ řádek. Po každém skončení periody (integrační doby) časovače se spustilo přerušení. V tomto přerušení jsem testoval proměnnou *StartCCD*, která na základě své hodnoty („1“) spouštěla přenos AFE-PPI-DMA vždy po výskytu hrany signálu PPI_FS1.

```
EX_INTERRUPT_HANDLER(Timer2_ISR)
{
        *pTIMER_STATUS = TIMIL2;                    // zrušení indikace potřeby obsluhy
        ssync();
        if (StartCCD == true) {
                *pDMA0_START_ADDR = CCD[K];         // cilová adresa
                ssync();
                *pDMA0_CONFIG |= DMAEN;             //enable DMA
                *pPPI_CONTROL |= PORT_EN;           //enable PPI
        }
}
```

3.5.5 EXTERNAL BUS INTERFACE UNIT - EBIU

Procesor podporuje asynchronní rozhraní jako je SRAM, ROM, FIFO, flash paměť, a ASIC/FPGA. EBIU obsluhuje požadavky od jádra procesoru a od DMA řadiče. EBIU je řízen systémovými hodinami (SCLK).

Na následujícím obrázku si ukážeme paměťový prostor, který je tato jednoka schopna adresovat:



Obr. 3-19 Exerní adresový prostor

Od adresy 0x2000 0000 je mapována část paměťového prostoru, kterou využívám k adresaci externí SRAM a k adresaci USB řadiče paměťový prostor od adresy 0x2030 0000. Jedná se o ASYNC MEMORY BANK 0 a ASYNC MEMORY BANK 3, které jsou vybírány piny („chip select“ vodiči) ASM0 a ASM3. Každá ze čtyř bank může být nezávisle nastavena, respektive délka přístupu k nim. To je velmi výhodné, protože rychlost přístupu k USB FIFO je jinak limitována než rychlost přístupu k SRAM. Nastavení se týká signálu chip select (ASMx), zápis (AWE), čtení (ARE), povolení výstupu (AOE), připravenost paměti (ARDY). Pro jednotlivé signály lze (pro každý BANK) nastavit délku tvrvání respektive délku předstihu či „držení“ (po deaktivaci jiného signálu). Délka je uváděna v počtech půlzů hodin periferie SCLK. Takže pro USB je délka zápisu a čtení 4 / 79MHz = 50,6 ns a doba mezi jednotlívými zápisi (čteními) je 7/79MHz = 88,6 ns. Tedy celková doba zápisu (čtení) 16b je 139 ns. Blíže k popisu jednotlivých registrů lze EBIU, viz v [2].

Nastavení EBIU pro ASYNC MEMORY BANK 3, do kterého je namapováno USB-Cypress je následující: *pEBIU_AMBCTL1 = 0x44C2FFC2; takže jeden zápisový cyklus do FIFO trvá 120ns, při SCLK = 79MHz (hodiny periferie).

Nastavení EBIU pro ASYNC MEMORY BANK 0, do kterého je namapována SRAM je následující*pEBIU_AMBCTL0 = 0xFFC233C2;

3.5.6 SPI rozhraní

Procesor má v sobě integrovánou periferii SPI. Dále je SPI rozhraní schopno vybírat až dalších sedm externích zařízení přes vyhrazené piny.

SPI rozhraní používám pro nastavení registrů v AFE. I když AFE vlastní pouze 3-wire rozhraní, lze vhodným nastavením registrů SPI uspěšně navázat komunikaci. Bohužel se

nebude moci z AFE registrů číst, což však vůbec nevadí, jelikož stejně „vím“, co je v nich obsaženo, protože při počáteční inicializaci do všech zapisuji mnou požadované hodnoty, které jsou zálohovány v PC.

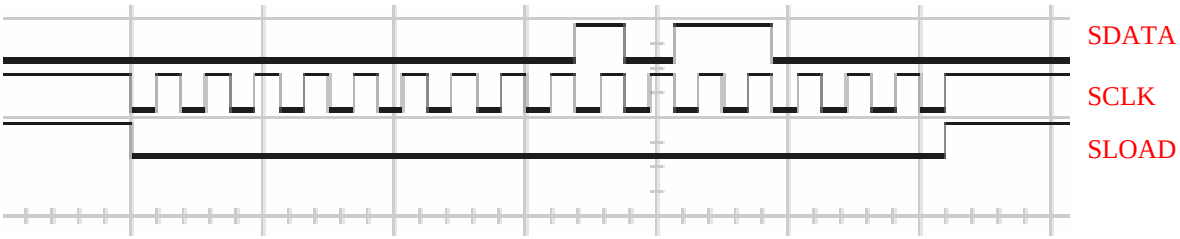
Dále je SPI využito při zavádění mého programu (po každém resetu desky), který je nahrán do 32KB externí EEPROM paměti.

Také je SPI připojeno na vybrané piny CPLD pro eventuální použití. Avšak z důvodů zmíněných dříve (nedostatek makrobuňek v CPLD) je komunikace mezi CPLD a BF532 přes SPI nefunkční.

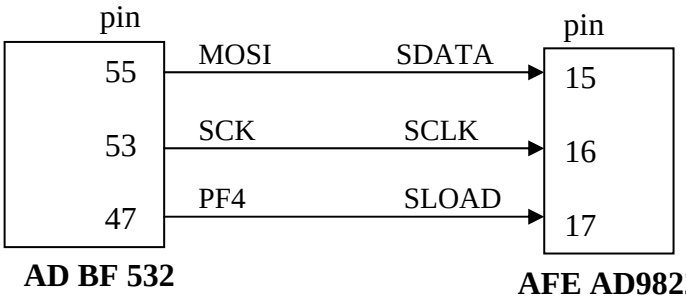
Pro úspěšné nastavení vnitřních registrů AFE je nutné nastavit SPI registry následovně:

***pSPI_BAUD = SCLK/(24);** - „baudová“ rychlost je tedy pro SCLK = 79MHz přibližně 1,6MHz.

***pSPI_CTL = 1<<0 | SIZE | SZ | MSTR | SPE;** - posílá nejdříve MSB, velikost přenášeného packetu dat je 16 bitů, hodinový pulz je platný až v prostřed datového pulzu, aktivní úroveň hodin je v logické jedničce, nastavuje SPI rozhraní procesoru jako master.



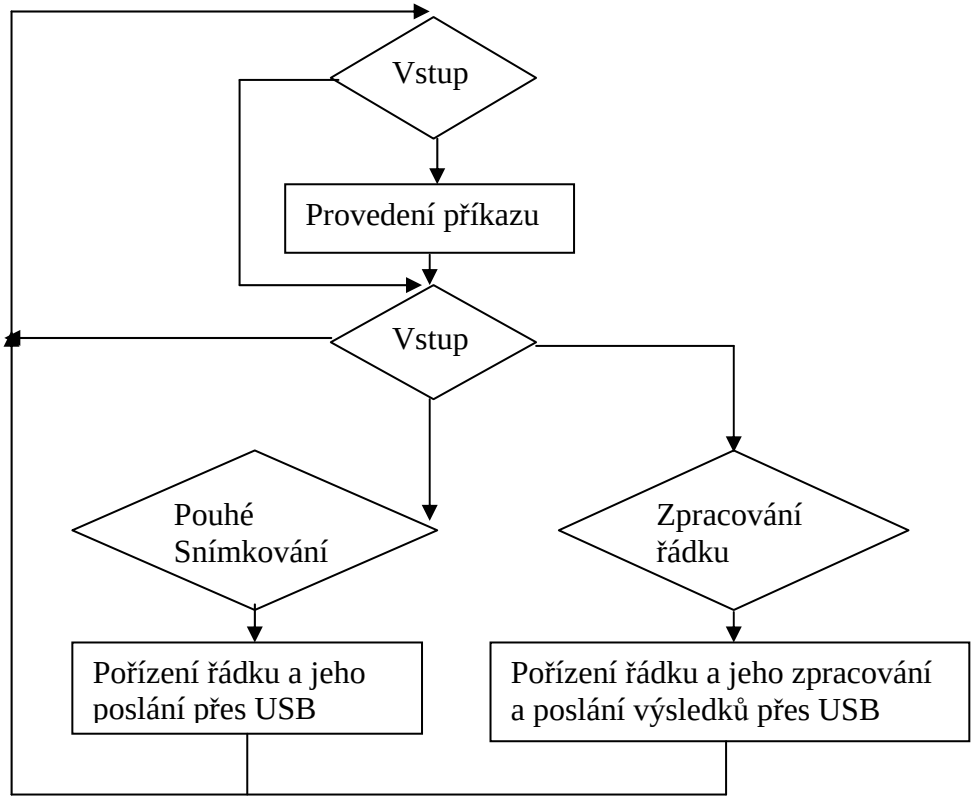
Obr. 3-20 Spráné časování přenosu dat (0x0058h) – zápis do configuračního registru



Obr. 3-21 Blokové schéma zapojení komunikace mezi procesorem BF532 přes SPI rozhraní a AFE

3.6 Stručný popis chování programu v procesoru BF 532

Po zapnutí či restartu se v procesoru provedou všechny nezbytné inicializace všech využívaných registrů a periférií. Potom již procesor čeká na spuštění přenosu řádku přes PPI-DMA-MEMORY L1. Spouštění a nastavovení snímání či zpracování se děje přes USB rozhraní. Pokud PC (moje aplikace) pošle příkazový rámec, viz kapitola 3.7.2, vyvolá se přerušení. Toto přerušení deaktivuje další vyvolání tohoto přerušení a nastavý proměnnou, která podle svého obsahu povoluje či zakazuje přístup do cyklu, ve kterém se provádějí jednotlivé nastavující příkazy. Pokud tedy přes USB přišel příkaz, vstoupí se do cyklu. V cyklu se přečte celý rámec (12B). Podle hodnoty *command* (viz kapitola 3.7.2) se provede příkaz a skočí se na konec cyklu, kde se opětovně povolý přerušení (povolující indikaci dat ve vystupní FIFO paměti EP4 z pohledu Host) a deaktivuje se příznak pro opakování příkazového cyklu. Pokud je tedy ještě ve FIFO EP4 další rámec, celá procedura se opakuje. Pokud ne, pokračuje se dále v programu. Dále v programu je již samotný algoritmus pořizování řádku a jeho zpracování. Tento algoritmus je však nutno zpřístupnit zapisem hodnoty do proměnné, do které může být zapsáno pouze v příkazovém cyklu. Jinak se tento algoritmus přeskočí a čeká se na další pokyny od PC (protože nic dalšího se v procesoru nezpracovává). Na následujícím diagramu si to lze vše přiblížit.



Obr. 3-22 Vývojový diagram chodu hlavního programu v procesoru BF532

Posílání příkazů je rovněž kontrolováno přerušením. Pokud je vstupní FIFO EP2 plná, je to indikováno signálm FLAGC a opět vyvoláno přerušení. Toto přerušení trvá tak dlouho, dokud není FIFO opět prázdná.

Výběr jednotlivých metod se děje porovnáváním jednotlivých bitů proměnné typu *unsigned shot podmínka* , protože jednotlivému bitu této proměnné je přiřazena určitá metoda a nastavující vlastnost metody.

Pro bližší pochopení implementace odkazuji na zdrojové kódy pro procesor BF532

3.7 USB

Pro komunikaci mezi snímací deskou a PC (uživatelem) je použito USB rozhraní. Obvod, který tuto komunikaci zprostředkovává je CY7C68013A (dále jen EZ-USB), viz [5]. Důvod, proč bylo použito USB rozhrahí je samozřejmě především v jeho přenosové rychlosti dat a taky z toho důvodu, že USB port je v dnešní době na každém PC či Notebooku, oproti standardnímu rozhraní jako je RS-232 (COM), nebo paralelní port (LPT).

Základní vlastnosti a charakteristiky EZ-USB jsou následující:

- specifikace USB2.0, či-li podporuje rychlosti Low-speed (1,5Mbit/s), Full-speed (12Mbit/s) a High-speed(480Mbit/s)
- čtyři programovatelné BULK / INTERRUPT / ISOCHRONOUS endpoints
- čtyři integrované 8- nebo 16- bitové FIFO, které mohou být dvojnásobně, trojnásobně až čtyřnásobně bufferované
- obsahuje řadič sběrnice USB (SIE)
- vnitřní mikroprocesor řady 8051 s integrovanou 16KB code / data RAM
- integrovaný I^2C kontroler, na který může být připojena I^2C EEPROM obsahující základní nastavení celého obvodu USB (firmware)
- pokud není k dispozici žádný software pro nastavení obvodu, EZ-USB se připojí jako USB device v základní konfiguraci, kde se o USB příkazy stará hardware
- napájení 3,3V s 5V tolerantními vstupy
- 16 bitovou datovou sběrnici s řídící sběrnici, která může pracovat ve dvou režimech
    - synchronní, který je pro přenos dat rychlejš oproti asynchronnímu režimu
    - asynchronní
- „flag“ piny, které informují o stavech FIFO paměti

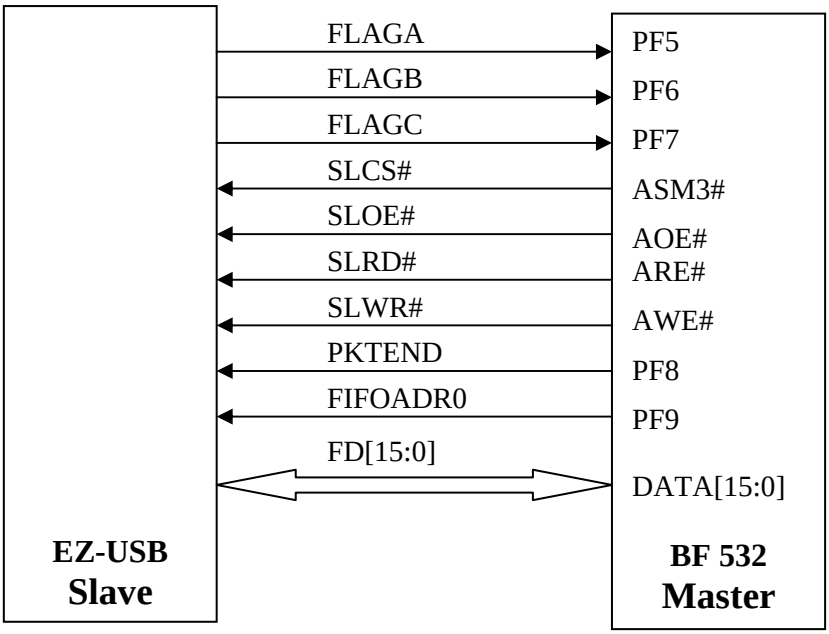
Pro správnou komunikaci mezi PC a snímací deskou je potřeba, aby PC podporovalo USB2.0, jinak komunikace nebude probíhat korektně mezi mojí aplikací a procesorem (odzkoušeno).

EZ-USB má naprogramované následující vlastnosti:

- Podpora čtyř endpointů, vstupní endpoity INEP1, INEP2 a výstupní endpoitny OUTEP1, OUTEP4. Všechny endpointy pracují v bulk režimu.
    1. INEP1 a OUTEP1 zprostředkovávají komunikaci mezi PC a obvodem Cypress. Díky tomu mohu číst vnitřní registry a FIFO paměti EZ-USB a vypisovat si je na monitoru a tím tak sledovat chod komunikace. Oba endpointy mají velikost bufferu 64B.
    2. Přes OUTEP4 zasílám příkazy do procesoru BF532. Velikost jednoho bufferu je 512B.
    3. Přes INEP2 přijímám nasnímané řádky, vypočtené hodnoty a souřadnice nasnímaných objektů, informace o procesoru. Velikost jednoho buffer je 512B.
- Komunikace mezi procesorem a EZ-USB je v asynchronním módu po 16 bitové sběrnici, z čehož vyplývá maximální přenosová rychlost 16,67 MB/s.
- Dvojnásobné bufferování pro endpointy EP2 a EP4, díky čemuž se muže zapisovat (číst) do (z) té samé FIFO a zároveň se mohou data přenášet, protože naplní-li se jeden buffer FIFO EZ-USB automaticky přepne do druhého bufferu a může se zapisovat dál a první buffer již zasílá (čte) data do (z) PC.
- Tři informační signály o stavech FIFO pamětech, které jsou přivedeny na piny procesoru, který na základě jejich významu vyvolá příslušné přerušení. Každý pin má svůj naprogramovaný „význam“ v EZ-USB.
    1. FLAGC informuje o stavu FIFO EP2. Pokud je FIFO plná (oba buffery), nahodí se tento signál do aktivní úrovně, čímž informuje procesor, že musí chvíli počkat s dalším zapisováním do FIFO.
    2. FLAGB nemá zatím určení.
    3. FLAGA informuje procesor o příchozím příkazu přijatém přes EP4. Tento signál se nahodí, pokud délka příchozích dat je vetší než 11B, protože každý můj příkaz vysílaný z PC má délku 12B.
- Externí 32KB EEPROM paměť pro uložení deskriptorů a nastavení jednotlivých endpointů a celé komunikace vůbec

3.7.1 Nastavení komunikace mezi CY7C68013A a BF532 pro přenos dat

EZ-USB je nastaveno ve slave módu (čili správcem přenosu je procesor) a způsob připojení k procesoru je na *Obr. 3-23*



Obr. 3-23 Blokové schéma připojení EZ-USB Slave Mode

Význam jednotlivých pinů:

FLAGA, FLAGB, FLAGC - uživatelem definované status flag, které informují o stavech FIFO, jako například status plný, prázdný. Jsou připojeny na PF piny procesoru (Programmable Flag).

SLCS# - jedná se o „chip select“ vodič celého obvodu EZ-USB, který je aktivní v L a řízen vodičem ASM3# procesorem BF532.

SLOE# - jedná se o povolení výstupních datových vodičů. Není-li aktivní v L, jsou datové výstupy FD[15:0] ve vysoké impedanci.

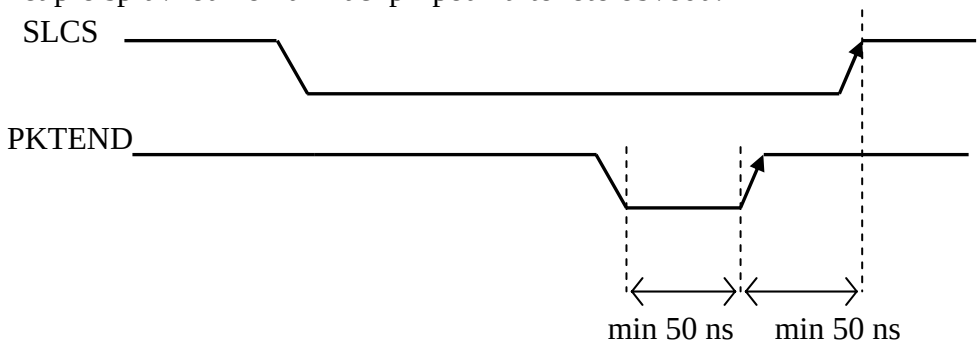
SLWR# - zápis do paměti FIFO

SLRD# - čtení z paměti FIFO

FIFOADR0 – tímto vodičem se vybýrá FIFO paměť náležící příslušnému ednpointu. Na snímací desce je druhý pin FIFOADR1 trvale uzemněn, takže logickou hodnotou signálu FIFOADR0 se výbíra buď FIFO endpoint0 0- EP0 (FIFOADR0=L), nebo FIFO paměť endpointu 4- EP4 (FIFOADR0=H).

PKTEND – toto je informační signál pro USB, když chce MASTER poslat krátký packet, jehož velikost je menší než velkost daného buffer, protože jinak USB posílá data až když je

daný buffer plný. Ná následujícím Obr. 3-24 si ukážeme časový diagram, který je nutný dodržet pro spravnou komunikaci při použití tohoto obvodu.

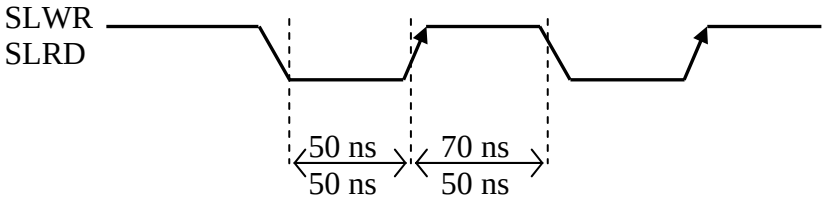


Obr. 3-24 Časový diagram SLCS a PKTEND

Z Obr. 3-24 je patrné, že signál PKTEND musí trvat minámálně 50ns v jeho aktivní úrovní, v tomto případě v L (aktivní úroveň – polaritu lze nastavit v registrech Cypressu) a musí skončit minimálně 50 ns před deaktivací „chip select“ signálu SLCS. Jelikož PKTEND je připojen na PF procesoru, není aktivní v době kdy přistupujeme k FIFO paměti a musíme to softwareově obejít, že budeme do FIFO zapisovat prázdná data a mezitím aktivujeme a deaktivujeme PKTEND. Jelikož je tento přístup z hlediska programu o něco složitější, tak tento signál nepoužívám a raději zapisuji do FIFO prázdná data, dokud se FIFO nenaplní na plný paměťový rozsah 512B a pak se automaticky pošle sama.

FD[15:0] – datové vodiče. EZ-USB podporuje jak 8b přenos, tak i 16b, či-li pro komunikaci EZ-USB s procesorem by tedy stačilo 8b, avšak pochopitelně s poloviční přenosovou rychlostí.

EZ-USB je nastaveno pro komunikaci s procesorem v asynchronním režimu, takže přenos jednotlivého slova (16b) se děje strobováním signálu SLWR#/SLRD# (zápis/čtení), čímž se FIFO pointer automaticky inkrementuje.



Obr. 3-25 Asynchronní režim - obecný model časování signálu pro čtení a zápis

V asynchronním režimu musí být délka doby zapisovacího pulzu SLWR nejméně 50ns v L a 70ns v H. Délka doby čtecího pulzu SLRD musí být nejméně 50ns v L a 50ns v H. Takže celková doba zapisovacího/ čtecího pulzu (SLWR/SLRD) je minimálně 120ns/100ns. Což

zároveň vymezuje maximální teoretickou přenosovou datovou rychlost. Důkladnější rozbor v [5],[6].

Podrobnější popis nastavení registrů zde nebudu uvádět vzhledem k větší rozsáhlosti a možnosti si to prohlédnout jak ve zdrojovém kódu, tak v jiných dokumentacích popisující práci s tímto obvodem. Jinak jednou nastavené registry již není po zapnutí či restartu dále nikterak upravovat, jelikož se přehrají z přiložené EEPROM přes IIC rozhraní, kterým Cypress disponuje. Dále přístup k EZ-USB s pohledu procesoru (míněno zápis a čtení) je již standardní přístup procesoru k periferii a nepotřebuje další komentář.

3.7.2 Příkazový rámec

Pro řízení přenosu a nastavení různých parametrů a proměnných v procesoru BF532, jsem si definoval následující rámec, který je výsílán přes OUTEP4 z aplikace v PC.
Délka a parametry mého příkazového rámce zasílaného do procesoru přes EZ-USB jsou následující:

LENGHT	COMMAND	DATA 1	DATA 2	DATA 3	DATA4

LENGHT …. Má šířku 2B, obsahuje kolik se přenáší bajtů. Defaultně je to nastaveno na 12B pro jeden rámec, ale pakliže se přenáší více takových to rámců, je LENGHT větší než 12B, z čehož se dá zjistit počet rámců. Následující rámce (je-li jich přenášeno více) mají LENGHT tedy rovnu 12B.

$$Počoč rámců = \frac{\text{LENGHT}}{12}$$

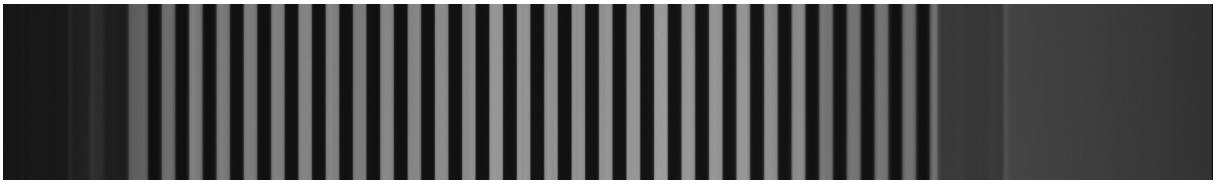
COMMAND …. Má šířku 2B. Číselně vyjadřuje příkaz pro procesor, který se má provést.
DATA1, DATA 2, DATA 3, DATA 4 …. Každá z těchto položek má velikost 2B a je určena pro přenos hodnot určených pro proměnné, ke kterým se přistupuje pomocí hodnoty obsažené v COMMAND. Nejsou-li všechny položky využity, jsou zbývající položky nulové.

Z délky mnou definovaného rámce je patrné, proč je FLAGA nastavován EZ-USB do aktivní úrovně při velikosti příchozího packetu většího než 11B (větší rovno 12B). Tedy po příchodu jednoho příkazového rámce se vyvolá v procesoru přerušení určené pro obsluhu vyskytnuté údálosti, tj. příchod mého příkazového rámce. Jednotlivé rámce jsou z FIFO paměti EP4 čteny vždy až po zpracování příkazu předchozího rámce. Jeden 512B packet může obsahovat celkem 42 mých příkazových rámců.

4 ZHODNOCENÍ VÝSLEDKŮ JEDNOTLIVÝCH METOD

V této kapitole zhodnotím výsledky, které jsem získal použitím metod uvedených v kapitole 2. Jedná se především o dobu nutnou pro provedení zvolené operace a dále dobu mezi pořízením dalšího řádku, nad kterým bude rovněž provedena příslušná operace (metoda obrazového zpracování). Z těchto hodnot času lze potom garantovat maximální dobu odezvy od pořízení řádku potažmo výskytem či absencí objektu pod kamerou, a dále vhodnost či nevhodnost použité metody pro snímanou scénu, za daných či proměnných snímacích podmínek pro dosažení co nejlepších výsledků – kompromis mezi rychlostí a kvalitou respektive přesností.

Základní atribut, který má zásadní vliv na rychlost odezvy systému, je integrační doba (doba potřebná pro dostatečné naakumulování náboje, který je generován dopadem světla-fotonů). Pro dostatečně dlouhou integrační dobu (jak dlouhou uvedu dále) lze jisté operace sparalelizovat. Či-li mohu během akumulace náboje, respektive vysouváním řádku (digitalizací řádku) zpracovávat data z minulého řádku a poté data posílat, než se začne digitalizovat řádek následující.



Obr. 4-1 Pořízený testovací obrázek jednořádkovým senzorem (2048x100, 8b)

Na Obr. 4-1 je zobrazen testovací obrázek – vzor, nad kterým byly zkoušeny obrazové metody a měřena doba jejich trvání. V následujících tabulkách si ukážeme časy příslušných metod. Doba byla měřena na osciloskopu, kde jsem sledoval dobu trvání aktivní logické úrovně, která trvala po celou dobu měřené metody. K tomuto účelu byl vyhrazen jeden pin procesoru. V zobrazených časech však není zahrnuta doba potřebná pro změnu logické úrovně tohoto pinu, která vzhledem ke své nepatrné velikosti (maximálně 145 ns) přesnost neovlivní, protože testované metody netrvali vždy přesně stejnou dobu (časová nestálost - „jitter“), ale doba se měnila hlavně vlivem osvětlovacích podmínek a celkovému nastavení (jako např. prahování jasové úrovně či jiné úrovně atd.) v rozmezí setin až desetin milisekundy. Pro různé integrační doby při stejných nastaveních a stejných osvětlovacích podmínek měly vždy stejnou dobu trvání.

Tab. 7 Doby trvání metod detekce hran - integrační doba 10ms

Detekce hrany	Level / σ	T_M [ms]
Threshold	0 / -	0,221
Threshold	50 / -	0,222
Threshold	100 / -	0,217
Threshold	150 / -	0,215
Threshold	200 / -	0,212
Laplacian	10 / -	1,023
Laplacian	30 / -	1,023
LoG	10 / 0,53	3,645
LoG	10 / 0,20	3,645
LoG	10 / 0,60	3,645
LoG	20 / 0,53	3,645
LoG	30 / 0,53	3,645
LoG	100 / 0,53	3,645
LoG	100 / 0,6	3,645
LoG	100 / 0,2	3,645
Prewitt	25 / -	0,95

První sloupce zobrazuje o jakou metodu detekce hrany se jedná. Sloupec Level / σ zobrazuje nastavení prahování a filtrace pro jednotlivé metody detekce hran (značka „-„ znamená, že pro danou metodu detekce hrany nemá hodnota σ vliv). Sloupec T_M zobrazuje trvání jednotlivé metody. Z naměřených hodnot je vidět, že detekční metoda založená na první a druhé derivaci, trvá za jakýchkoliv nastavení vždy stejnou dobu potřebnou pro zpracování celého řádku. Doba trvání detekce LoG operátorem je více jak trojnásobná oproti detekce Laplacianem a operaátorem Prewittové. To je způsobeno především tím, že v metodě s LoG operátorem se účasní konvoluce maska typu *float* (reálné číslo) oproti detekce pomocí Laplacianu a Prewittové, kde se účasní konvoluce maska typu *char*. Naproti tomu doba trvání metody Threshold je pro různé nastavení prahovací úrovně různá. Rozdíl je však v jednotkách až maximálně desítkách mikrosekund, což na celkovou dobu zpracování (jak si ukážeme dále) má nepatrný vliv. Čím je to způsobeno si netroufnu říci, jelikož softwareová implementace této metody je vskutku triviální. Zřejmě je to způsobeno vlastním nastavováním pinu, který umožňoval vlastní měření.

Jak jsem již uvedl v kapitole 2.1.2, tak jsem ve výsledku implementoval pouze detekci hran na základě aproximace první derivace – operátor Prewittové. Důvod je z předchozí tabulky jasný a to dosažení co nejmenších dob trvání celého zpracování. Je jasné, že hledání hran pomocí Laplacianu je v podstatě stejné, jako hledání pomocí operátoru Prewittové.

Avšak složitost, citlivost na šum a doba potřebná pro nalezení vypočtených hodnot jednoznačně upřednostňuje detekci operátorem Prewittové.

Tab. 8 Doby trvání metod filtrace obrazu

Filtrace	Integrační doba [ms]	T_M [ms]
Median	10	0,96
Průměrování x3	10	30
Průměrování x4	10	40
Průměrování x12	10	320
Průměrování x3	1	4
Průměrování x4	1	5
Průměrování x8	1	9
Průměrování x58	1	59

Z naměřených hodnot v Tab. 8 je patrné, že doba trvání metody průměrování je přímo závislá na integrační době, jelikož se jedná o mezi řádkové zpracování. Pro kratší doby integrace je nutné brát v potaz dobu nutnou pro vlastní zpracování. Pro průměr tří řádků je nutné provést celkem 3x2048 = 6144 součtů a 2048 podílů, kde operace dělení je vždy časově náročná matematická operace. Pro delší doby integrace je již čas potřebný pro zpracování dostatečně dlouhý. Naproti tomu medianová filtrace trvá vždy stejnou dobu, avšak v sobě skrývá nevýhody zmíněné v kapitole 2.1.1.1 – rozmázávání (rozostřování) hran.

Tab. 9 Doba trvání metody pro nalezení oblasti (funkce FindArea)

Level / σ	T_{MB} [ms]	T_{MS} [ms]
1 / -	0,72	1,02
5 / -	0,68	1,07
10 / -	0,80	1,13
20 / -	0,80	1,14
50 / -	0,83	1,15
100 / -	0,85	1,20
200 / -	0,68	1,02

Jak bylo uveden v 2.1.2.5, tato funkce hledá oblasti a případně upravuje řádek pro danou metodu hledání a zpracování objektů. Doba trvání T_{MB} ukazuje časy bez vyplňování oblastí mezi levou a pravou hranou – určené pro metodu „překrývajících se pruhů“. Kdežto doba T_{MS} ukazuje časy s vyplněním oblasti mezi levou a pravou hranou. Toto vyplňování je určené pouze pro metodu „Labeling“. Vyplňováním se míní, že všechny hodnoty pixelu mezi levou a pravou hranou (včetně) mají stejnou hodnotu a to 1, která deklaruje detekovanou a nezpracovanou oblast.

Tab. 10 Doba trvání metody „překrývajících se pruhů“

Filtrace	Detekce hrany	Level / σ	T_M [ms]	T_{INT} [ms]
Bez filtrace	Laplacian	17 / -	2,2	10
Median	Laplacian	17 / -	3,2	10
Průměrování x3	Laplacian	17 / -	33	40
Průměrování x32	Laplacian	17 / -	320	330
Bez filtrace	LoG	17 / 0,53	4,85	10
Median	LoG	17 / 0,53	5,85	10
Průměrování x3	LoG	17 / 0,53	30	320
Průměrování x32	LoG	17 / 0,53	320	330
Bez filtrace	Prewitt	25 / -	2,15	5
Median	Prewitt	25 / -	3,05	5
Průměrování x4	Prewitt	25 / -	20	5

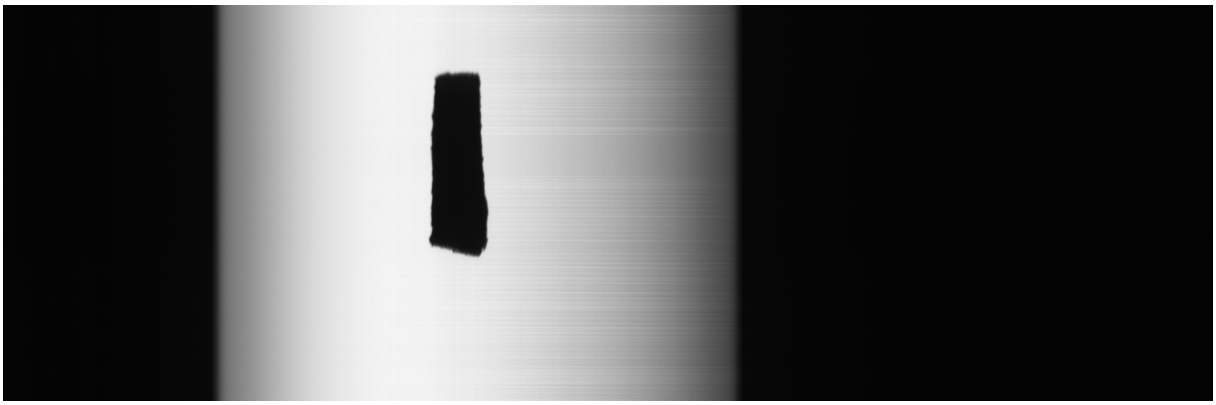
Tab. 11 Doba trvání metody "Labeling"

Filtrace	Detekce hrany	Level / σ	T_M [ms]	T_{INT} [ms]
Bez filtrace	Threshold	80 / -	1,4	10
Median	Threshold	80 / -	2,3	10
Bez filtrace	Laplacian	17 / -	3,4	10
Median	Laplacian	17 / -	4,5	10
Bez filtrace	LoG	12 / 0,45	7	10
Median	LoG	12 / 0,45	7,6	10
Bez filtrace	Prewitt	25 / -	2,83	5
Median	Prewitt	25 / -	3,8	5

Z Tab. 10 a Tab. 11 je vidět, a potvrzuje to i tvrzení uvedené v kapitole 2.2, že metoda „Labeling“ je pomalejší oproti metodě „překrývájících se pruhů“. Pro obě metody je zvláště zpomalující kombinace filtrace s mediánem a hranová detekce s LoG operátorem. Proto jak již bylo uvedeno detekční metodu LoG a Laplacian jsem vypustil. Tím to jsem schopen garantovat maximální dobu trvání zpracování s příslušnou dobou odezvy maximálně **5 ms** pro senzory s rozlišením 2048 pixelů I když skutečná doba bude asi kratší, maximálně 4ms pro kombinaci Prewitt + Median + Labeling. Hodnota 5ms tak vymezuje maximální řádkovou frekvenci pro zpracování 200Hz. Samozřejmě pro filtraci průměrováním bude doba trvání odezvy záviset na počtu průměrování. Dále není brán v potaz malé rychlosti reakce systému (PC), který ve výsledku může reakční dobu výrazně prodloužit, protože se čeká, až si PC vyzvedne všechna data s FIFO paměti. Jelikož lze snímací desku upravit tak, že se bude chovat jako autonomní systém, protože program beží ve smyčce, není proto nutné tuto reakční dobu do výsledku zahrnovat. Dále jsem zde neuvedl minimální dobu obyčejného snímání (scann-mod), která činí 0,75 ms, čili minimální doba potřebná pro vysunutí celého řádku.

V následujících tabulkách a na obrázku uvedu výsledky, které jsem dostal od výše popsaných metod a přístupů.

Na nakloněné podsvícené rovině jsem pouštěl předměty a snímal je pomocí mé snímací desky, která potom posílala hotové výsledky do PC (mé aplikace). Uvádím jenom, že toto je pouze demonstrativní příklad a hodnoty zde uvedené nemají žádný rozměr, pouze číselně vyjadřují indexy pixelů. Rozměr by měli, kdyby se změřila vzdálenost kamery od snímané scény, typ objektyvu aj. Tím bychom získaly převodní konstantu, která by se vynásobila s výsledky poslanými od snímací desky. V důsledku ryclejšího pádu získáme pár řádků, kde na jeden řádek potřebuji dobu 5ms. Na následujícím obrázku je zobrazen padající objekt.



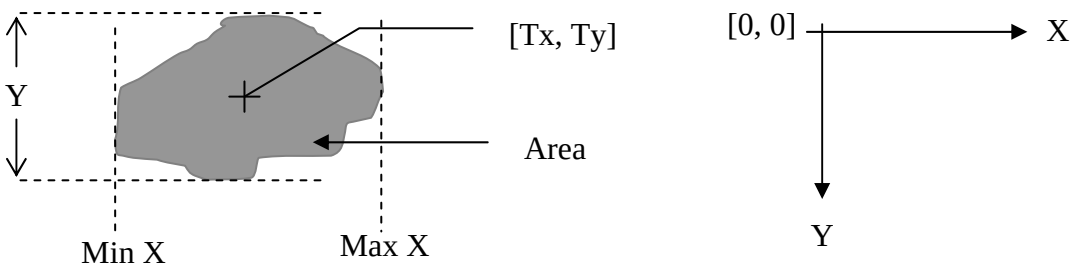
Obr. 4-2 Nasnímaný pád objektu na nakloněné podsvícené rovině

Tab. 12 Příklad nasnímaných hodnot metodou „překrývajících se pruhů“ - Prewitt = 25

Min x	Max x	Tx	Ty	Area	Y
827	928	873,5	8,46	1232	15
771	785	752,5	13,34	1152	24

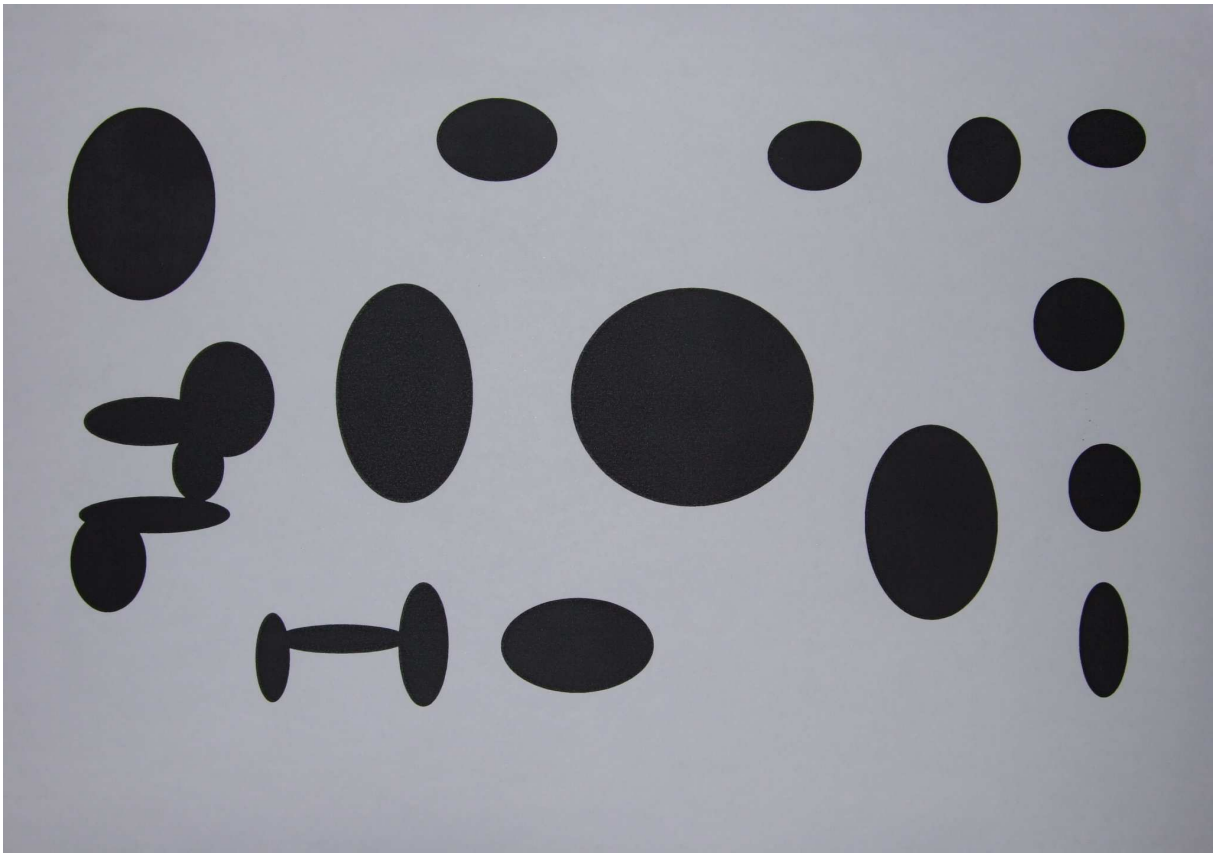
V Tab. 12 je uveden příklad zpracovaných hodnot při snímání pádu dvou objektů. Oba současně padající objekty měli podobný tvar jako objekt na Obr. 4-2. Velikost ploch, ale i délky (Y) je dána především rychlostí pádu a doby snímání jednoho řádku.

Význam hodnot z Tab. 12 je patrný z následujícího obrázku:



Obr. 4-3 Získané hodnoty ze zkoumaného objektu

Na následujícím Obr. 4-4 opět otestuji metody hledání objektu.



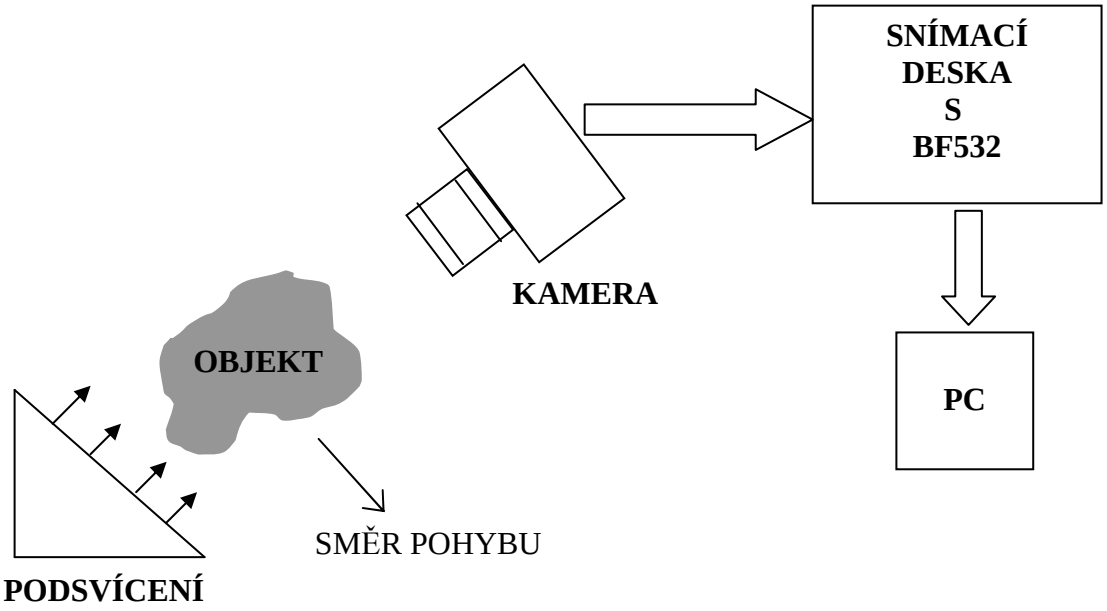
Obr. 4-4 Vytištěná předloha pro testování metod a hledání objektů

Po nasnímání předlohy jsem získal následující data uvedené v Tab. 13.

Tab. 13 Hodnoty získané metodou Labeling - threshold = 25

N	OBJECT	MIN_X	MAX_X	Tx	Ty	Area	Y
1	2	644	742	694	77	11080	151
2	5	1148	1210	1178	44	4713	92
3	3	908	981	943	62	6714	119
4	4	1047	1106	1076	63	5943	126
5	4	1104	1182	1140	158	14230	298
6	4	443	517	477	42	4225	72
7	5	442	508	466	35	2990	79
8	2	553	677	612	288	42300	544
9	3	750	945	839	266	63260	487
10	6	1150	1211	1180	61	5322	116
11	2	989	1096	1043	131	20390	261
12	3	701	824	762	69	11010	129
13	2	507	662	599	88	10310	171
14	5	1157	1197	1177	79	4376	150

Funkčnost celého celku byla ověřena (deska s PC aplikací) tak, že jsem snímal padající objekty na podsvícené nakloněné rovině a na monitoru si zobrazil získané souřadnice padajících objektů. Sestavu si lze ukázat na následujícím obrázku.



Obr. 4-5 Schéma zkoušené sestavy

5 ZÁVĚR A DOPORUČENÍ

Náplní této diplomové práce bylo navržení metod zpracování obrazu - nalezení objektů a určení jejich težiště a plochy. Tyto metody byly implementovány do signálového procesoru AD-BF532. Dále byla navrhnuta a osazena deska s tímto procesorem. Tato deska táke obsahuje blok generátoru řídících signálů realizováného CPLD (naprogamováno ve VHDL) obvodem, který řídí CCD senzor a AD převodník v podobě AFE (AD9822) a tím umožňuje získání videoinformace (obrazového řádku) pro další zpracování procesorem. Dále deska rovněř obsahuje obvod umožňující a sprostředkovávající komunikaci přes USB rozhraní. Toto rozhraní je realizováno obvodem Cypress, který lze přes USB jednoduše přeprogramovat, nebo si tento obvod může naprogramované nastavení „nabootovat“ s přiložené EEPROM paměti. Díky USB rozhraní může ve výsledku deska pracovat i v jednoduchém režimu, kdy získané řádky nezpracovává, ale pouze je posílá přes USB a funguje tak jako inteligentní scanner s možností si navolit rozlišení. Možnost volby řídit senzory s různým rozlišením (počet pixelů) umožňuje právě obvod CPLD, který je schopen po nastavení či přeprogramování generovat signály pro různé CCD senzory (černobílé či barevné). Ve výsledku jsou získaná data, ať už v podobě obrázku či souřadnic nalezených objektů, poslána přes USB do PC a v mnou vytvořené „user-friendly“ aplikaci zobrazena. Tato aplikace rovňež umožnuje jednoduché ovládání a nastavení realizované desky.

Za účelem seznámení se s procesorem BF532 byla rovněž navrhnuta vyvojová deska (kit), která obsahovala procesor BF532 jehož všechny důležité periferie jsem vyvedl na příslušné piny a mohl tak testovat jeho chování. Zkoušet vlastní programy na tomto procesoru bylo možné pomocí externí EEPROM, kterou bylo možno přeprogramovat buď přímo na desce, nebo po vyjmutí z patice v externím programátoru pamětí.

Z výsledků získaných měřením je patrné, že garantovaná doba zpracování řádku je 5ms. Pro pomalejší pohyb zkoumaných objektů, jako jsou pády po nakloněných rovinách či pomalé dopravní pásy, je tato doba dostačující, avšak pro rychlé pohyby (např. přímý voný pád, rychlý dopravní pás aj.) je tato doba nedostačující a příliš velká. Proto možnou alternativou by bylo použití FPGA obvodu na místo procesoru. Ten by vhodným naprogramováním umožnil skutečné „real-time“ zpracování řádku i pro velmi malé integrační doby. Jelikož jsem měl za úkol použít signálový procesor jehož maximální pracovní frekvence je 396MHz, tak jsem dle mého názoru udělal maximum, abych dobu potřebnou pro zpracování co nejvíce minimalizoval. Možná, kdyby se použilo procesoru ze stejné rodiny Bkackfin, procesor BF533, jehož pracovní frekvence může být až 600MHz, tak dobu potřebnou na zpracování řádku bych podstatně zmenšil. Důvodem jeho nepoužití v této diplomové práci je fakt, že se vyrábí pouze v pouzdře BGA. Nicméně si myslím, že i tak se jedná o velmi výkoný a celkem univerzální signálový procesor (počet a možnost nastavení periférií), který se často používá pro zpracování obrazů.

Také navržené metody jsou naprogramovány tak, že je lze použít i na jíných procesorech.

Výsledná zařízení a programy fungují, tedy věřím, že se mi podařilo splnit zadání této diplomové práce v plném rozsahu.

6 LITERATURA

[1] Václav Hlaváč, Miloš Sedláček: Zpracování signálů a obrazů, Vydavatelství ČVUT, Praha 2005

[2] Datasheet: BF533_HRM_for_web.pdf

[3] Datasheet: AD9822.pdf

[4] Datasheet: ILX503.pdf

[5] Manuá:l EZ-USB_TRM - user manual.pdf

[6] Datasheet: CY7C68013A.pdf

[7] Diplomová práce Radil

[8] Datasheet: ILX553A.pdf

[9] Diplomová práce 2006, Lubomír Říha :Videoprocesor s FPGA pro zpracování signálu z obrazových CCD a CMOS snímačů.

[10] Internet

Seznam použitých symbolů a zkratek

CCD…Charge-Coupled Device, což v překladu znamená zařízení s vázanými náboji

AFE…Analog Front End

ISP…In-System programmable

SCLK…frekvence hodin periferií procesoru BF 532

CCLK…frekvence hodin jádra procesoru BF 532

GSS…generátor synchronizačních signálů

7 PŘÍLOHY

Tato kapitola se věnuje popisu a oživení desek plošných spojů, které jsem v rámci semestrální a diplomové práce navrhnul, oživil a osadil. Dále popisuje aplikaci, která komunikuje a nastavuje desku této diplomové práce.

7.1 Zkušební kit s procesorem BF532

Tato deska byla postavena za účelem bližšího seznámení se s procesorem BF532 a laborováním s ním. Z důvodu vysoké univerzálnosti byly vyvedeny skoro všechny důležité piny tohoto procesoru, na které jsem potom mohl připojovat další moduly a odzkoušet si tak zapojení následné diplomové práce před vlastní výrobou plošného obvodu. V této kapitole budou shrnuty základní technické parametry, především na piny vyvedené periférie.
Výrobní podklady a vrstvy TOB, BOT, SSTOP, SSBOT jsou vyexportovány v pdf souborech.

Napájení:

Napájet lze z libovolného stabilizované stejnosměrného zdroje s požadovaným výkonem.
Napájecí vstupní napětí je v rozsahu 6 V až 10 V (možno i 12 V). Vstup je chráněn diodou proti přepólování. Další stabilizaci napájení na 3,3V pro procesor a jiná zařízení na desce je použit stabilizátor LM3,3V.Napájení desky je indikováno LED diodoou.

JTAG:

JTAG je zapojen v doporučeném zapojení podle schématu ve vývojovém kitu s procesorem BF533 fy AD. Konektor pro připojení zařízení je klíčován.

SPORT0, SPORT1:

Tyto rozhraní mají vyvedené z procesoru všechny signálové piny.
RSCLK0, RFS0, DR0PRI, DR0SEC, TSCLK0, TFS0, DT0PRI, DT0SEC
RSCLK1, RFS1, DR1PRI, DR1SEC, TSCLK1, TFS1, DT1PRI, DT1SEC

Řídící adresová sběrnice:

Jsou vyvedeny všechny řídící sběrnice pro asynchronní přenos, či-li rozhraní Async. Memory Control. /BR, ARDY, /AMS0, /AMS1, /AMS2, /AMS3, /AOE, ARE, /AWE, /ABE0, /ABE1, /BGH, /BG

Adresová sběrnice:

Je vyvedeno prvních 8 pinů adresové sběrnice. AD1÷AD8

Datová sběrnice:

Je vyvedeno všech 16 pinů datové sběrnice DATA0÷DATA15.

UART:

Vyvedeny signály Txd, RxD na pinovou listu, včetně země a napajeni.

LED, BUTTON:

Tato LED je připojena na pin procesoru PF1. Tlačítko je připojené na pin procesoru PF11, kde tento pin je také sdílen ještě s rozhraním PPI.

BOOT EEPROM:

Do této 32KB EEPROM lze nahrát program, buď mimo desku (kdy se vyjme paměť s patice), nebo po přepojení pinů přímo na desku. Tedy vyjmutím jumperů pro boot s desky a připojením příslušných programovacích vodičů na piny z vnější strany.

Výběr tohoto obvodu procesorem se děje pinem PF2, který je vybrán pro bootování z vnějšku právě přes rozhraní SPI. Výběr způsobu bootování procesoru (v tomto případě přes SPI z flash nebo EEPROM) se volí přivedením vhodných napěťových úrovní na piny procesoru BMODE0, BMODE1 (piny 96,95), dále jen BMODE.

Procesor podporuje bootování přes SPI ze seriové EEPROM, kdy baudová rychlost je 500 kHz, adresování 8-ti, 16-ti nebo 24-ti bitové. Pro master mode boot je tedy nutné na BMODE pinech nastavit logické jedničky, tedy BMODE = 11.

Tab. 14 Booting modes

BMODE[1-0]	Description
00	Execute from 16-bit external memory (Bypass boot ROM)
01	Boot from 8-bit or 16-bit FLASH
10	Reserved
11	Boot from SPI seriál EEPROM (8-, 16-, or 24-bit address range

PPI:

Jsou vyvedeny všechny piny tohoto rozhraní.

Tedy PPI_CLK, PPI_0-PPI_15, PPI_FSS-PPI_FS3.

Frekvence pro toto rozhraní je možno získat přímo z krystalu pro procesor, na který je připojen obvod 74LVC04 Díky tomuto obvodu lze získat dva signály, které jsou porit sobě negované. Tyto dva signály jsou vyvedeny na vlastní lámací pinovou lištu.

RTC:

Připojen krystal pro použití obvodu realného času. Opět toto rozhraní zvyšuje aplikační použití.

NMI:

Nemaskovatelné přerušení. Vyhrazen pro to speciální pin.

RESET:

Resetovací obvod s tlačítkem. Doporučené zapojení.

Procesor má vyvedeny takřka všechny významnější piny a porty, pro univerzální používání a laborování. Samotná komunikace s procesorem je možná přes UART, SPORT0, SPORT1 či JTAG.

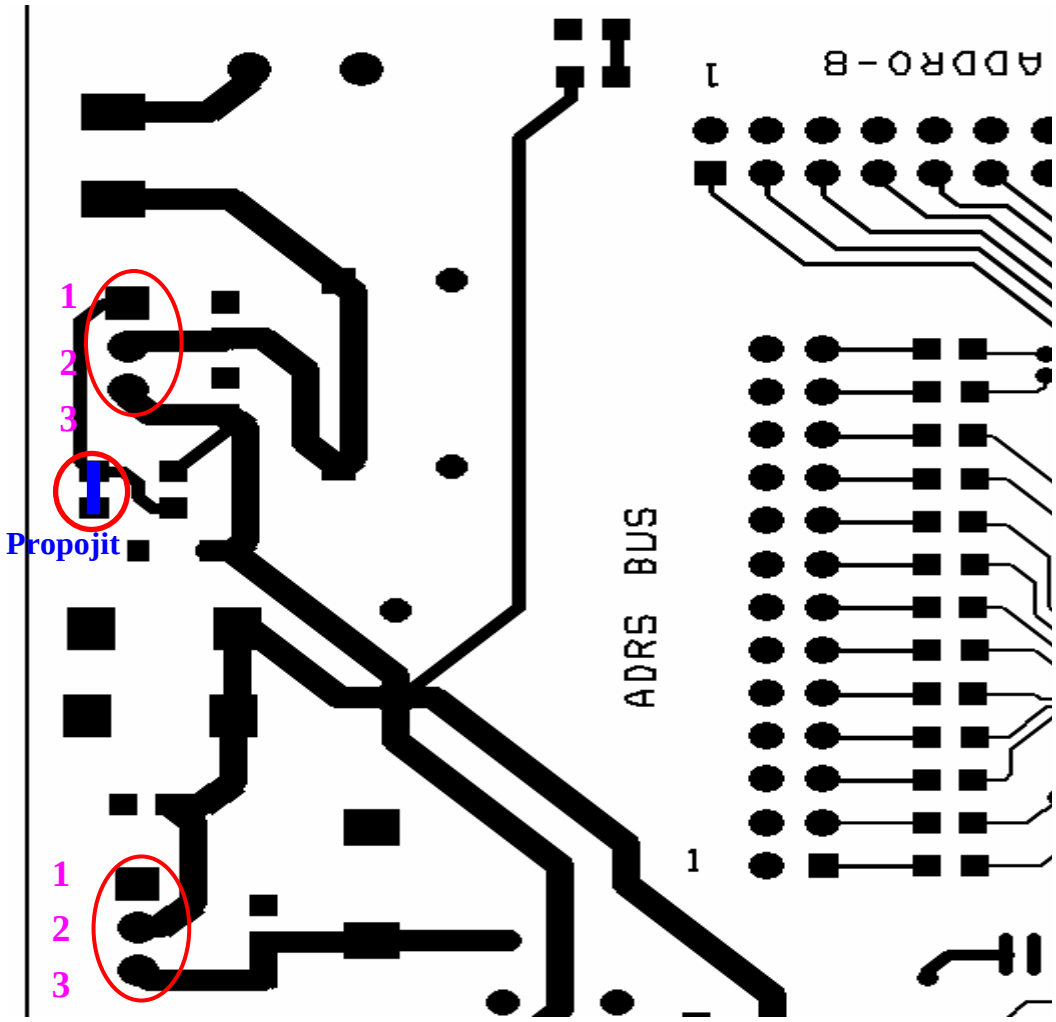
Testování a oživení desky:

Neosazenou desku bylo nutno nejdříve ručně multimetrem prověřit proti zkratům, ať už nerozpoznaných při samotném návrhu či vzniklých v průběhu výroby plošného spoje. Při negativním nálezu se mohlo přistoupit k samotnému osazení. Neprve se osadili SMD součástky, či-li samotný procesor, rezistroy, kondenzátory, jiné integrované obvody (např. resetovací obvod) atd.

Po kontrole napájení se deska připojila ke zdroji s proudovým omezením, z důvodu možného výskytu zkratu na desce, který mohl vzniknout třeba až v poslední fázi – osazení a pájení. Poté se již mohla naprogramovat EEPROM testovacím programem, například blikáním LED diody a pravidelným nastavováním výstupních obvodů Programmable Flags do logické nuly a jedničky.

Při spuštění programu jsem sledoval na zdroji odběr proudu kitu, který kolísal. Při blížším ohledání a opětovné kontrole návrhu jsem zjistil, že vstup č.51 je nesmyslně připojen na potenciál napájení. Jedná se o jeden z I/O periférie programmable flag, a to PF0. Při spuštění programu, který cyklicky měnil logickou hodnotu na výstupu tohoto pinu, nastala situace, že při logické nule se v podstatě tento výstup zkratoval, což mělo pochopitelně za následek vrůst odběru proudu.

Na následujících obrázcích jsou znázorněny návrhářské chyby testovací desky a jsou k nim uvedeny eventuální nápravy.



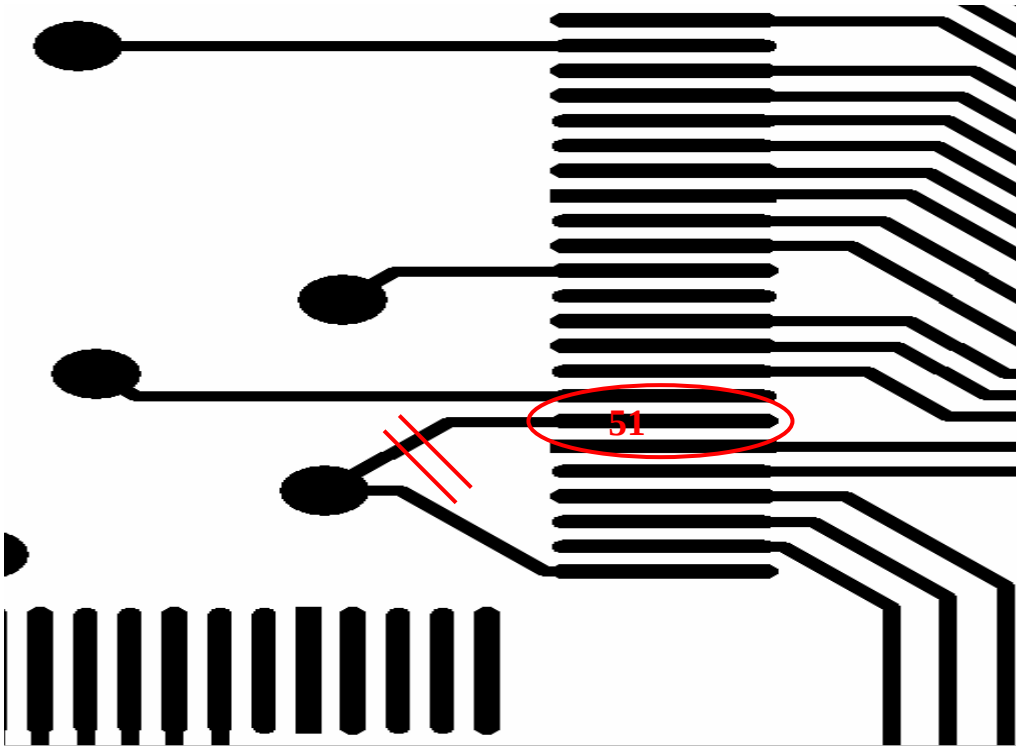
Obr. 7-1 Zobrazení úprav prohození nožiček stabilizátorů a propojení padů pro zapojení stabilizátoru s přímým napětím 3,3V na výstupu

Na Obr. 7-1 je znázorněna první návrhářská chyba, kterou je nutno při osazování desky vzíti v potaz. Z důvodu záměny vstupní a výstupní svorky (padu) stabilizátoru je nutné nožičky stabilizátoru prohodit, či-li překroutit tak, aby na padu č.2 byl vstup (tedy krajní nožička na stabilizátoru) a na padu č.3 byl výstup (tedy prostřední nožíčka stabilizátoru).

Následující Tab. 15 přibližuje výše zmíněný problém.

Tab. 15 Zobrazení čísel nožiček a jejich přiřazení

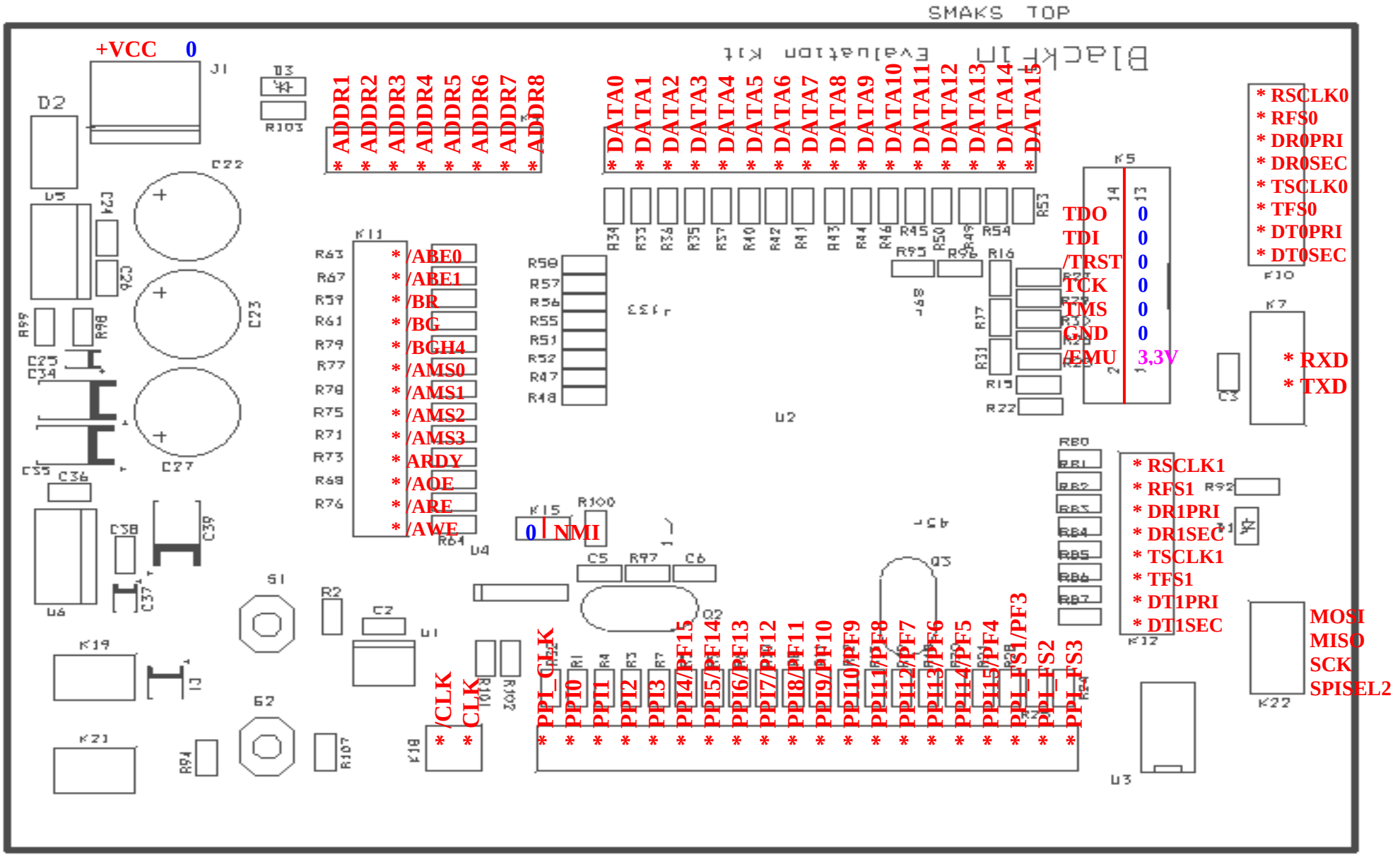
Číslování padu na desce	Číslování nožiček na stabilizátoru
1	1
2	3
3	2



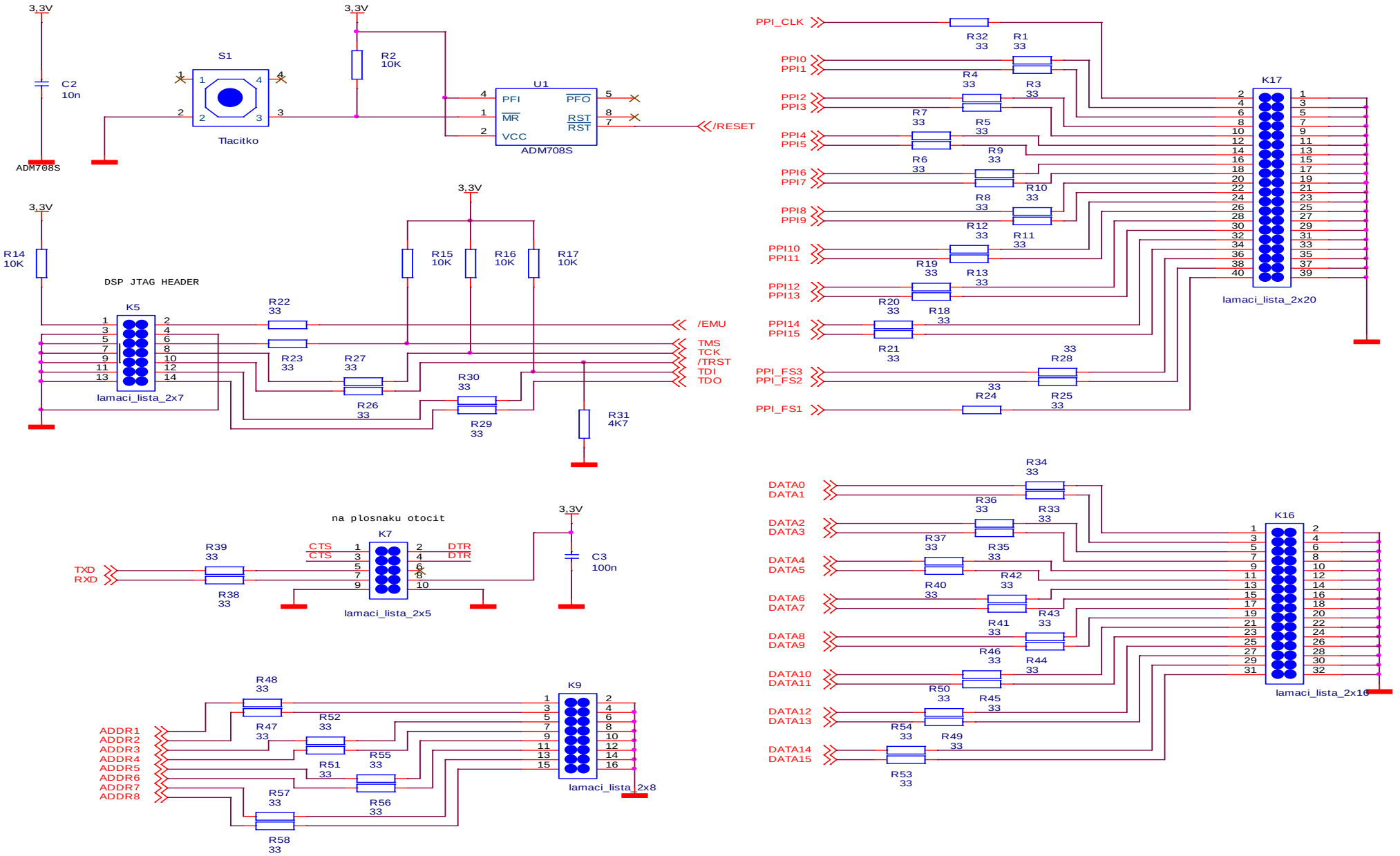
Obr. 7-2 Odstranění návrhářské chyby – přerušení spoje pod procesorem vedoucím k pinu č. 51

Na Obr. 7-2 je zobrazena další návrhářská chyba, kterou je nutno řešit z důvodu markantního nárůstu odběru proudu, který by při delším provozu procesoru mohl vést až k zničení výstupu na tomto pinu (číslo 51). Tuto chybu lze opravit tak, že se buď přeruší spoj jak je znázorněno na obrázku, či se nezapájí nožička č. 51 k padu desky.

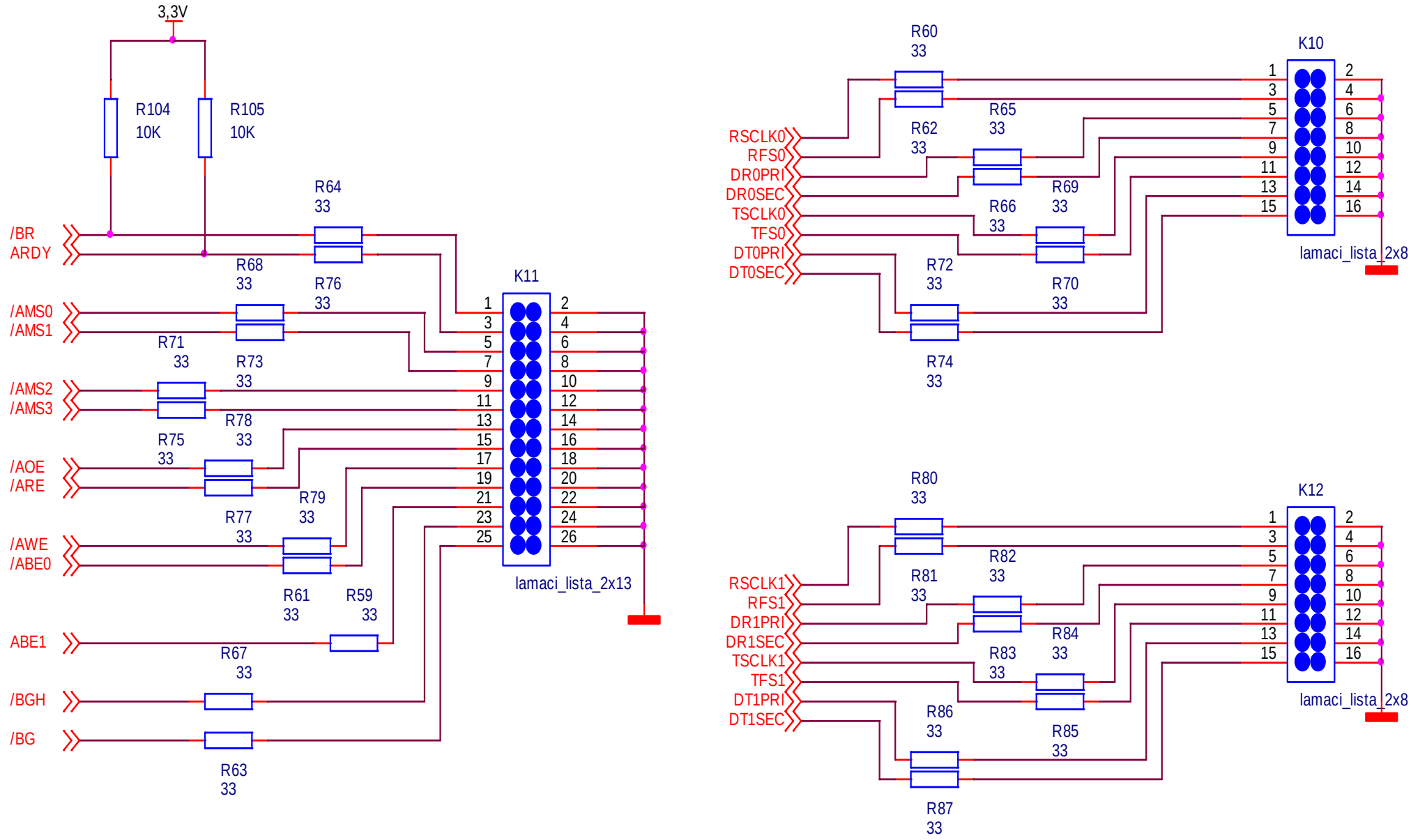
Na Obr. 7-3 jsou popsány piny lámací lišty, na které jsou připojena rozhraní procesoru.



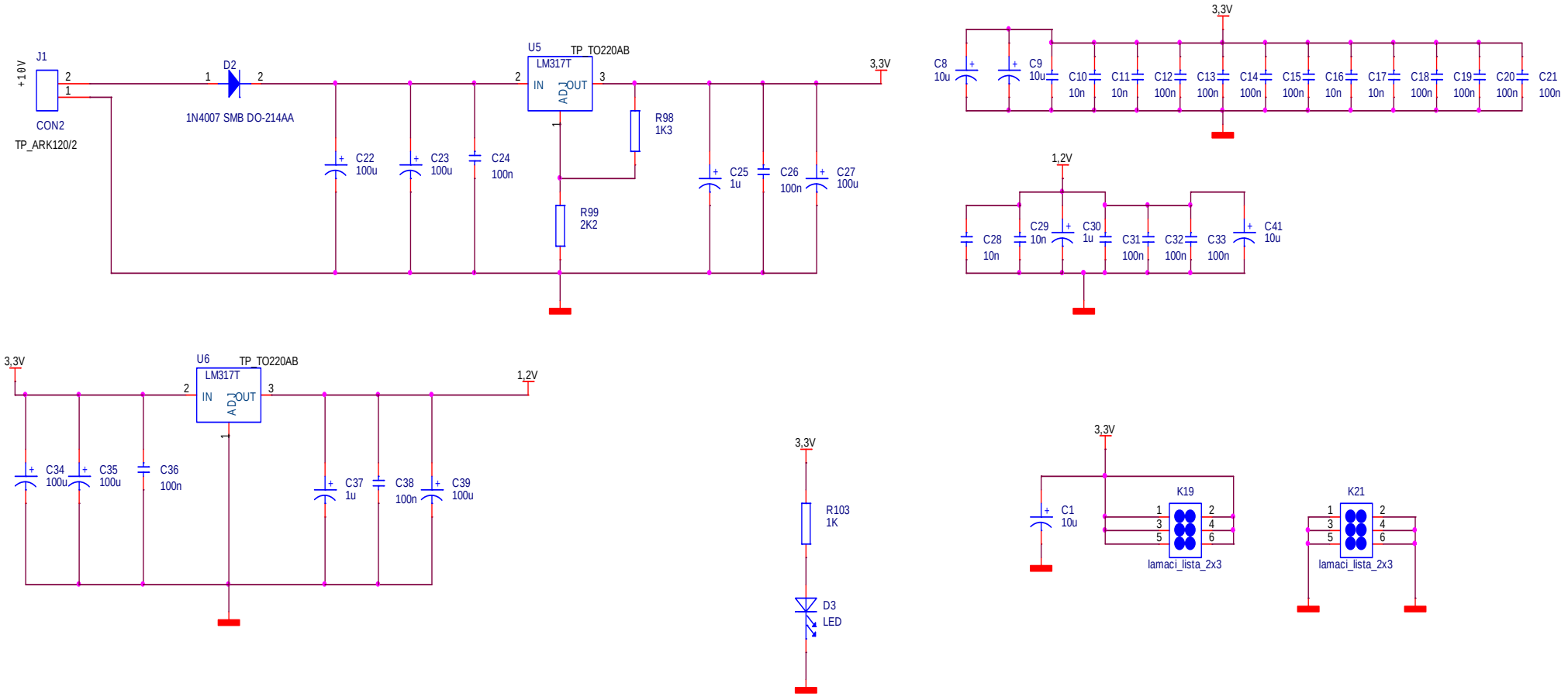
Obr. 7-3 Rozmístění jednotlivých periférií



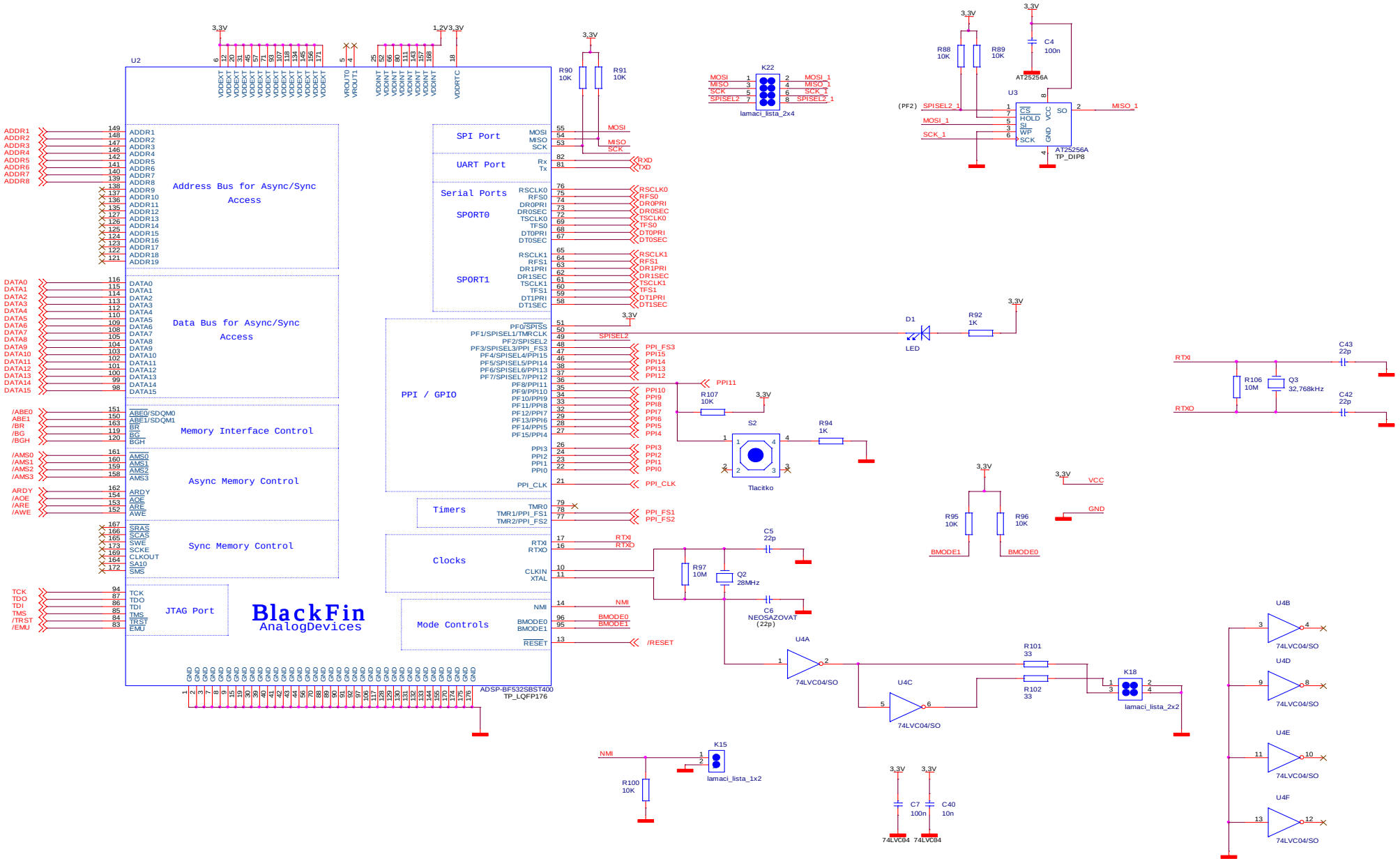
Obr. 7-4 Schéma zapojení resetovacího obvodu a konektorů pro JTAG, UART, nižší byte adresové sběrnice, PPI, datová sběrnice



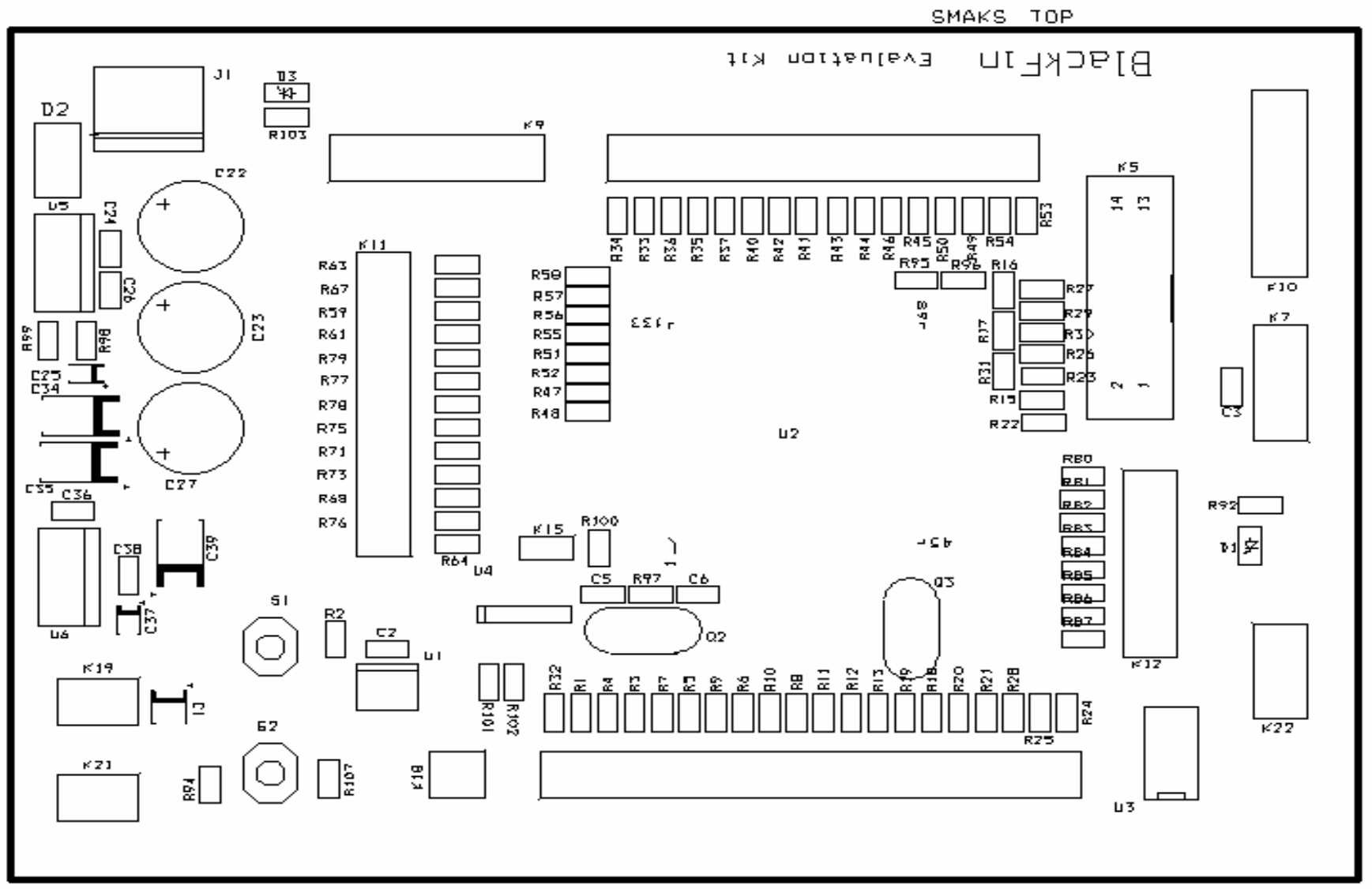
Obr. 7-5 Schéma zapojení konektorů pro SPORT0, SPORT1, řídící asynchronní sběrnice



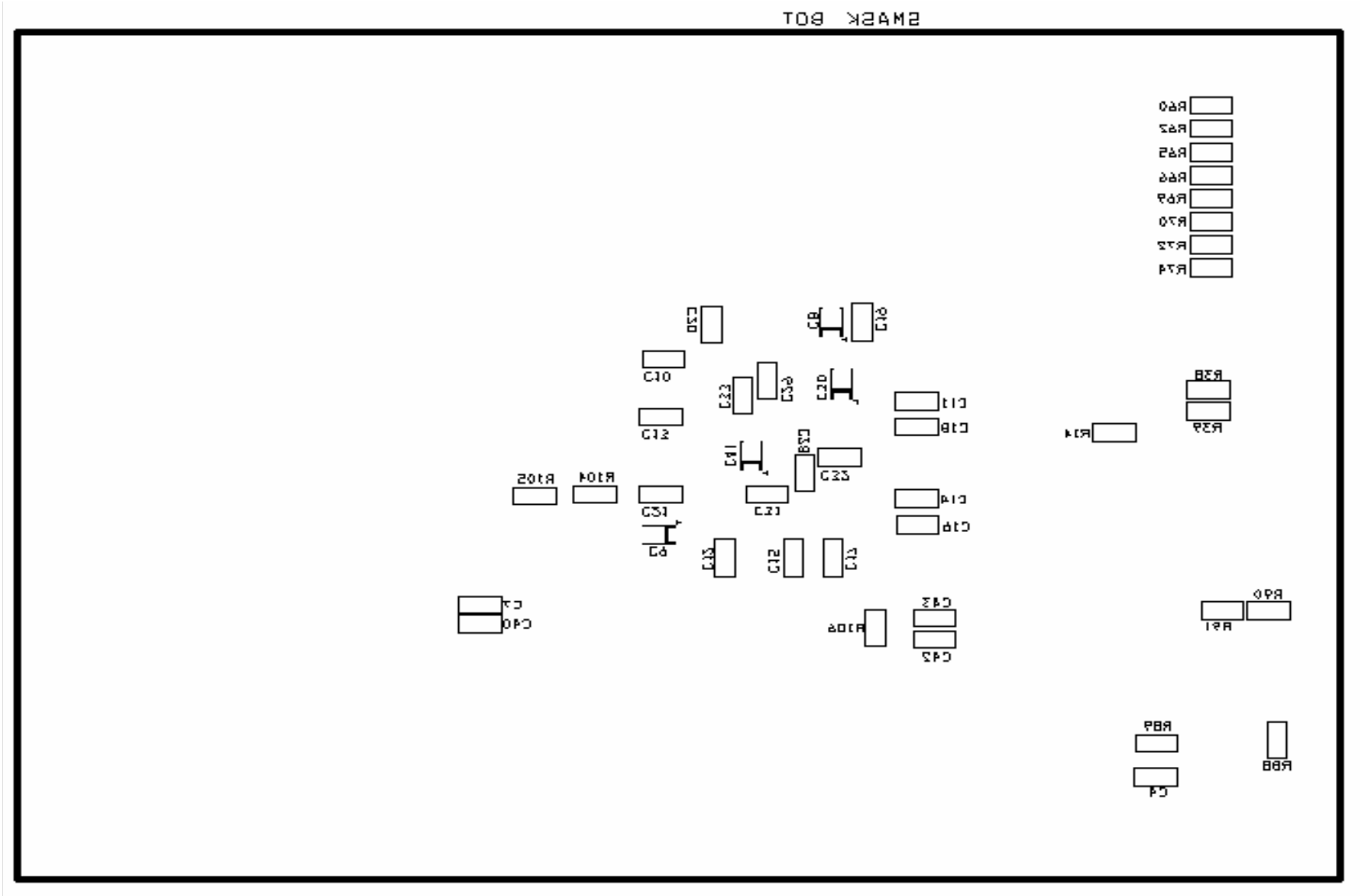
Obr. 7-6 Schéma zapojení napájecích a stabilizačních obvodů



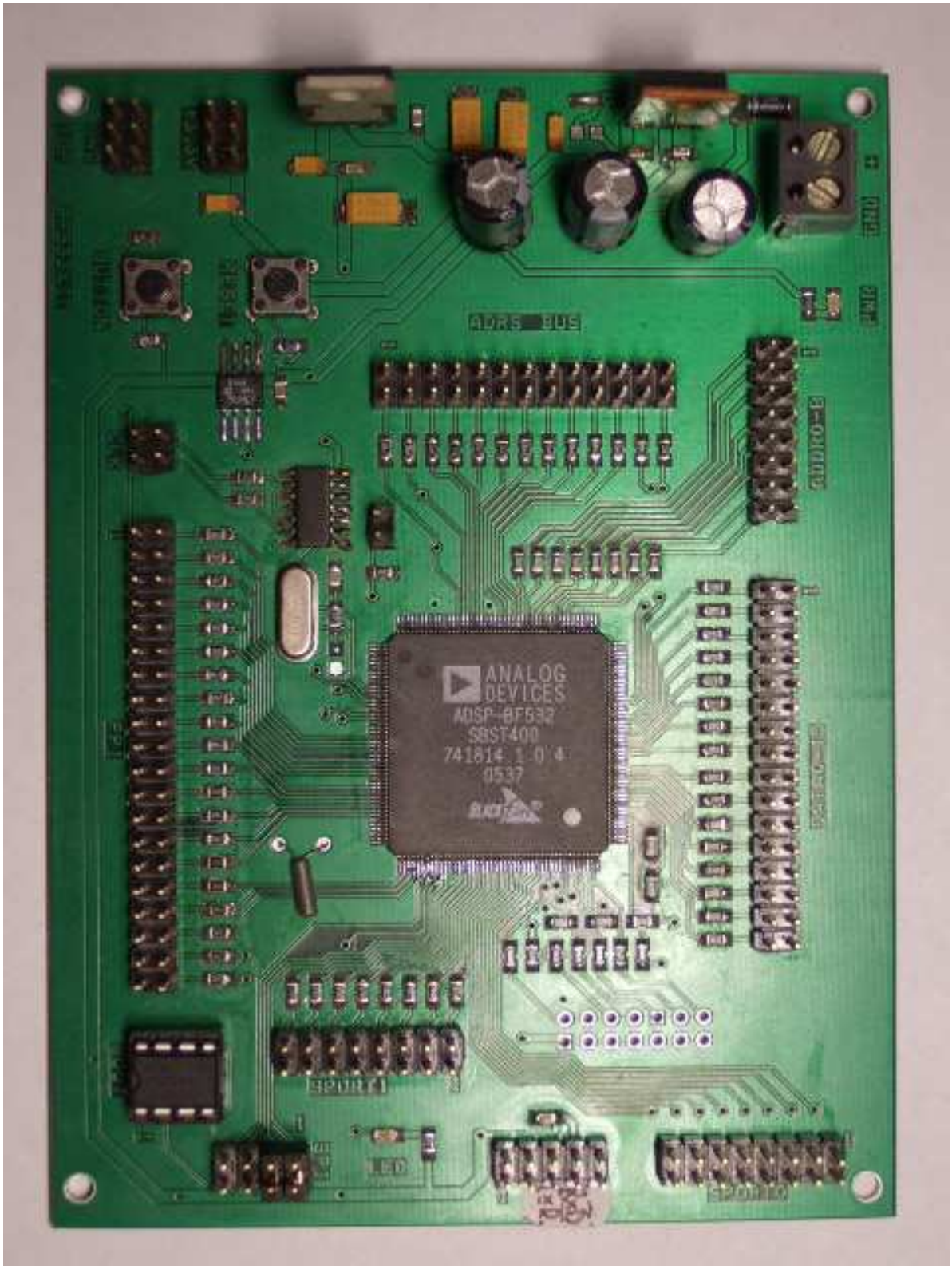
Obr. 7-7 Schéma zapojení procesoru BF532, hodinových obvodů, seriové EEPROM



Obr. 7-8 Maska potisku vrstvy TOP



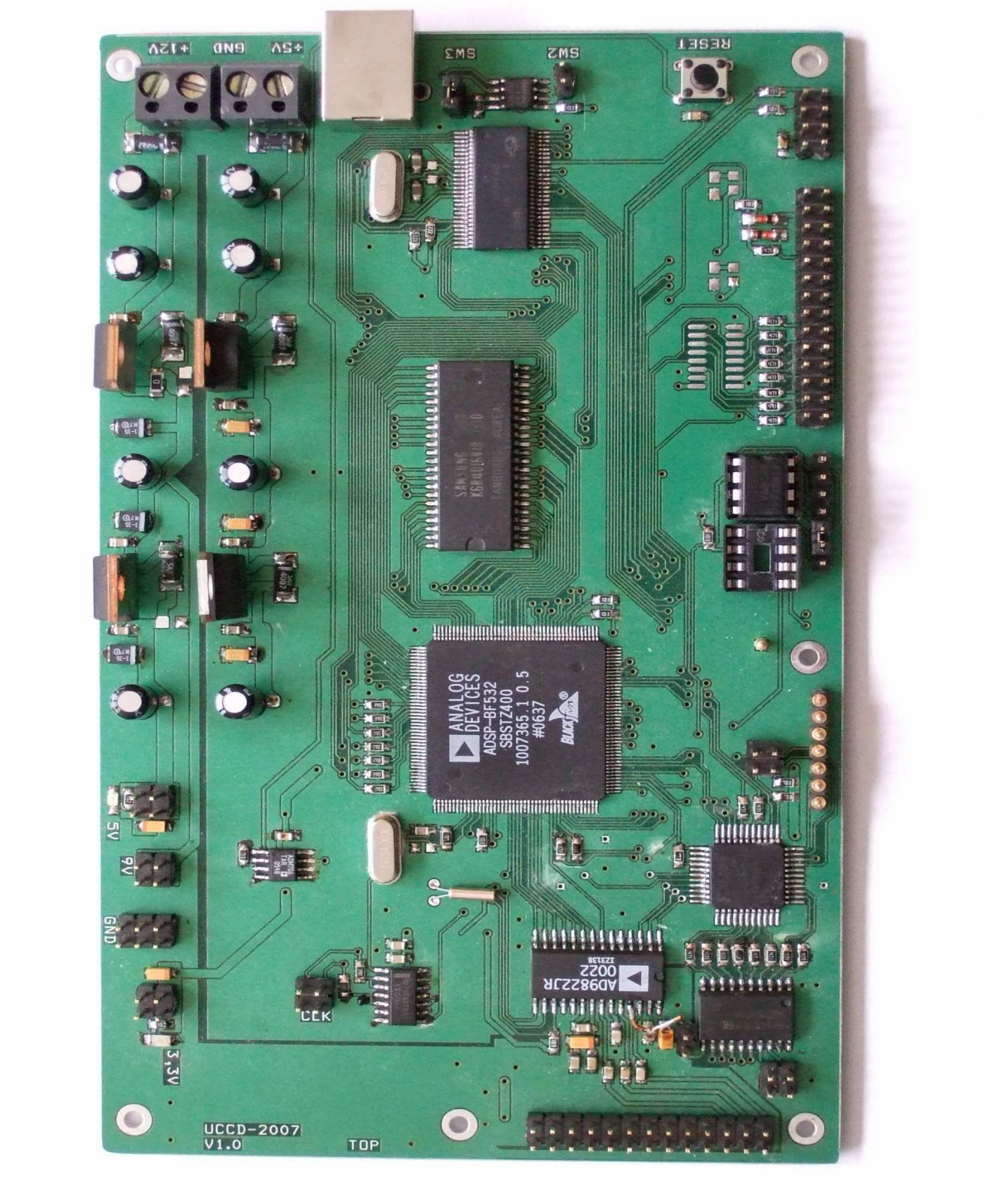
Obr. 7-9 Maska potisku vrstvy BOTTOM



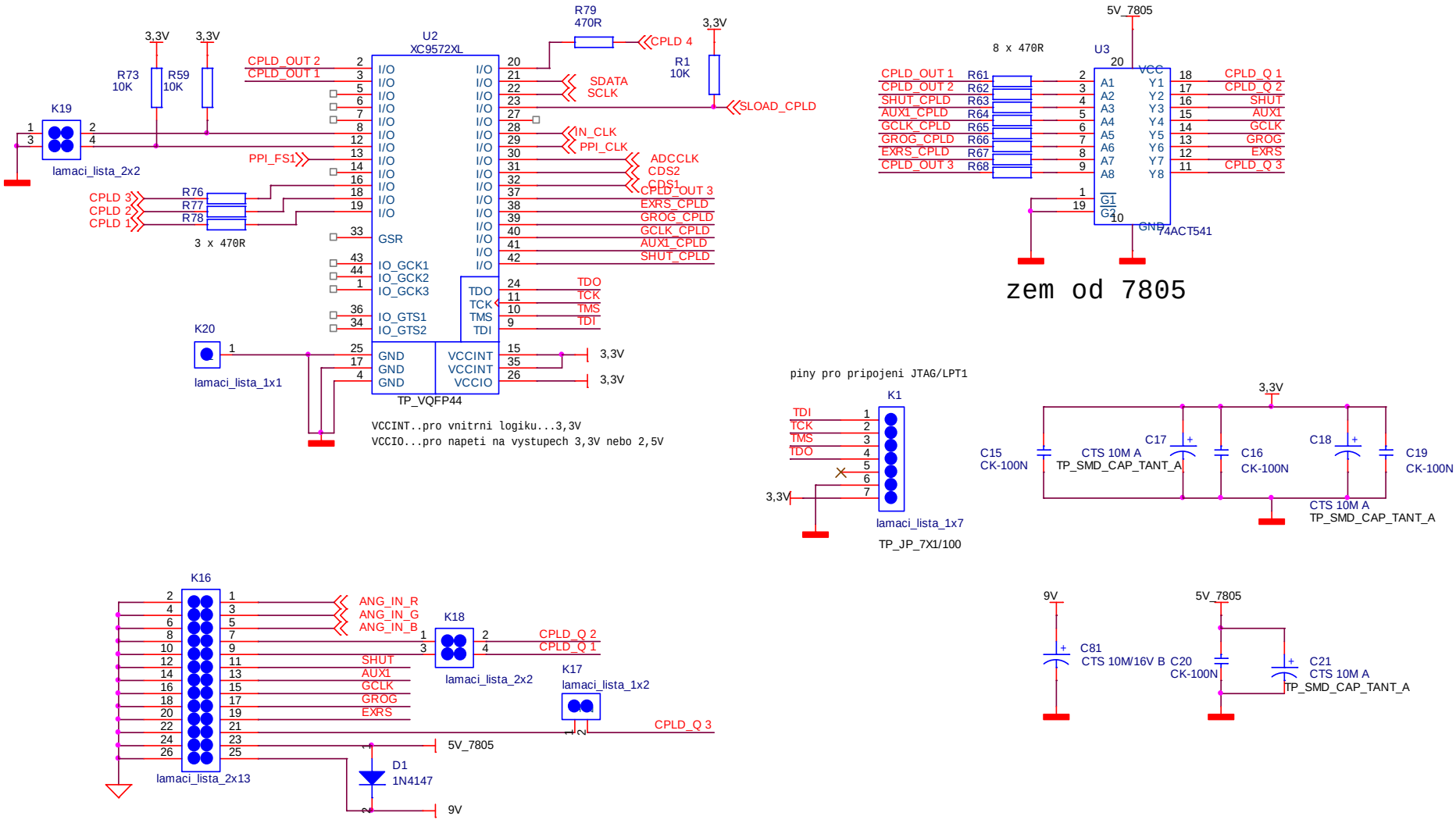
Obr. 7-10 Obrázek vývojové kit desky

7.2 Deska diplomové práce: UCCD-2007

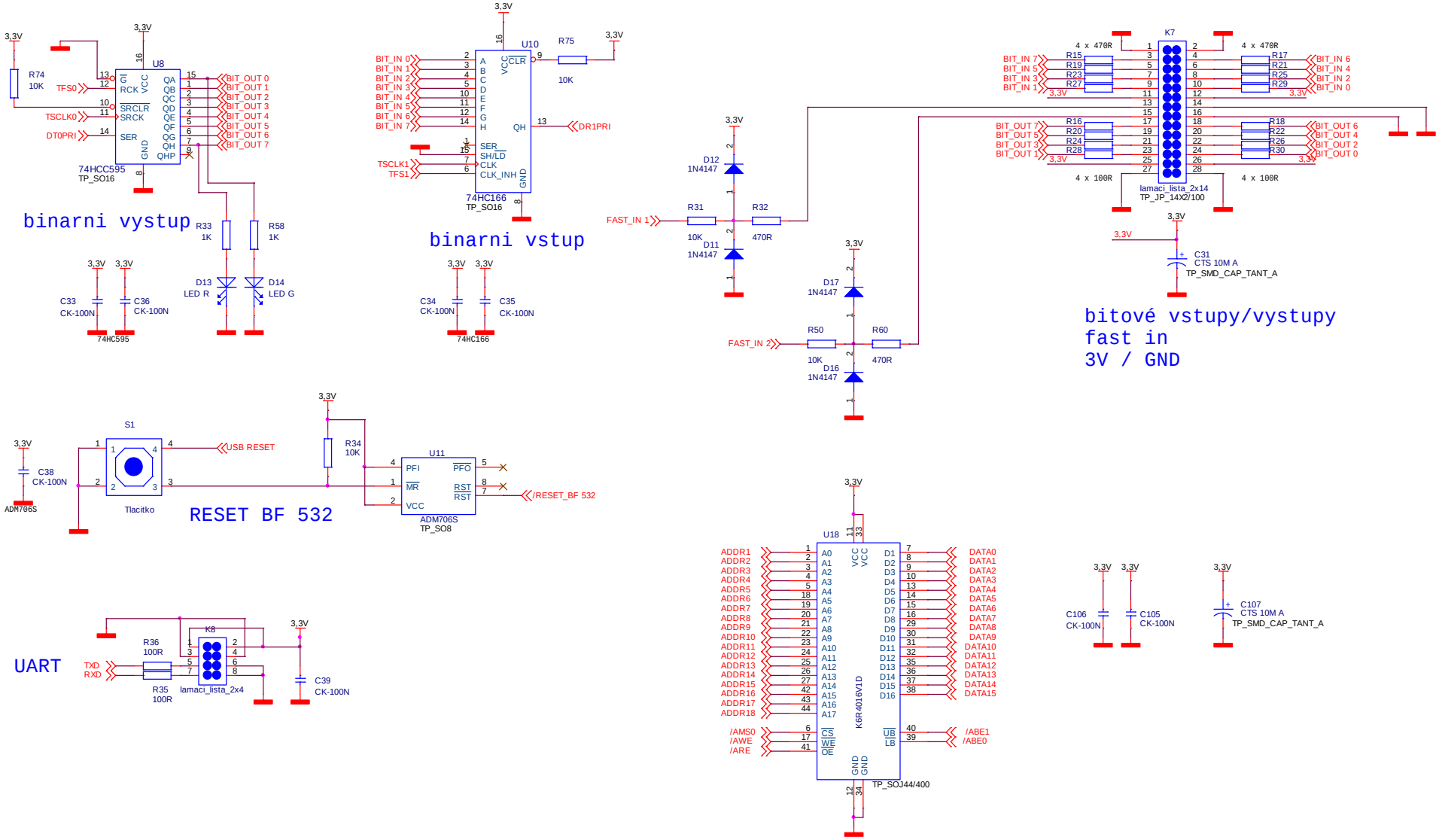
Vrstvy TOP, BOT, SSTOP, SSBOT jsou vyexportovány v pdf souborech.



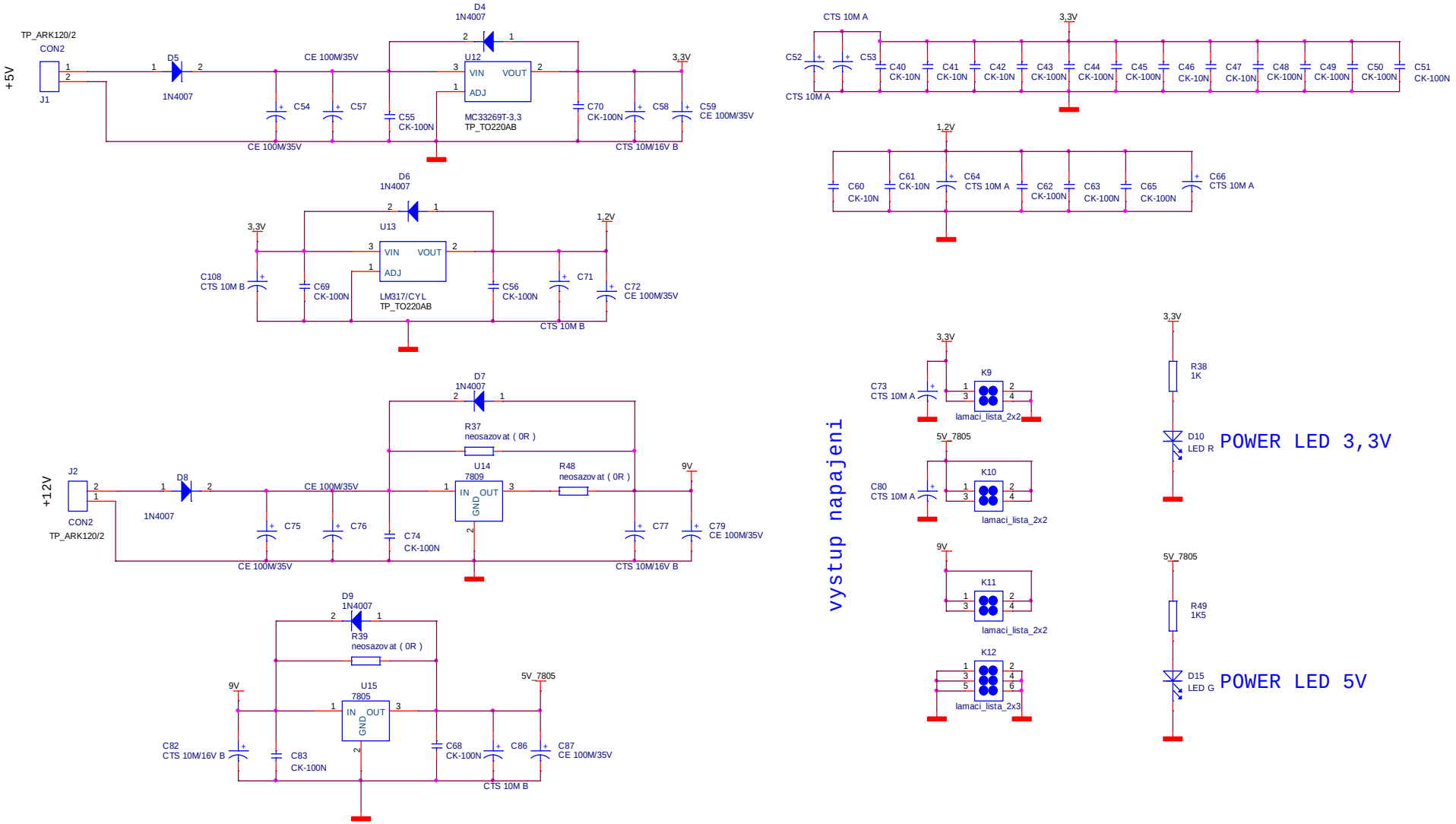
Obr. 7-11 Obrázek výsledné desky diplomové práce



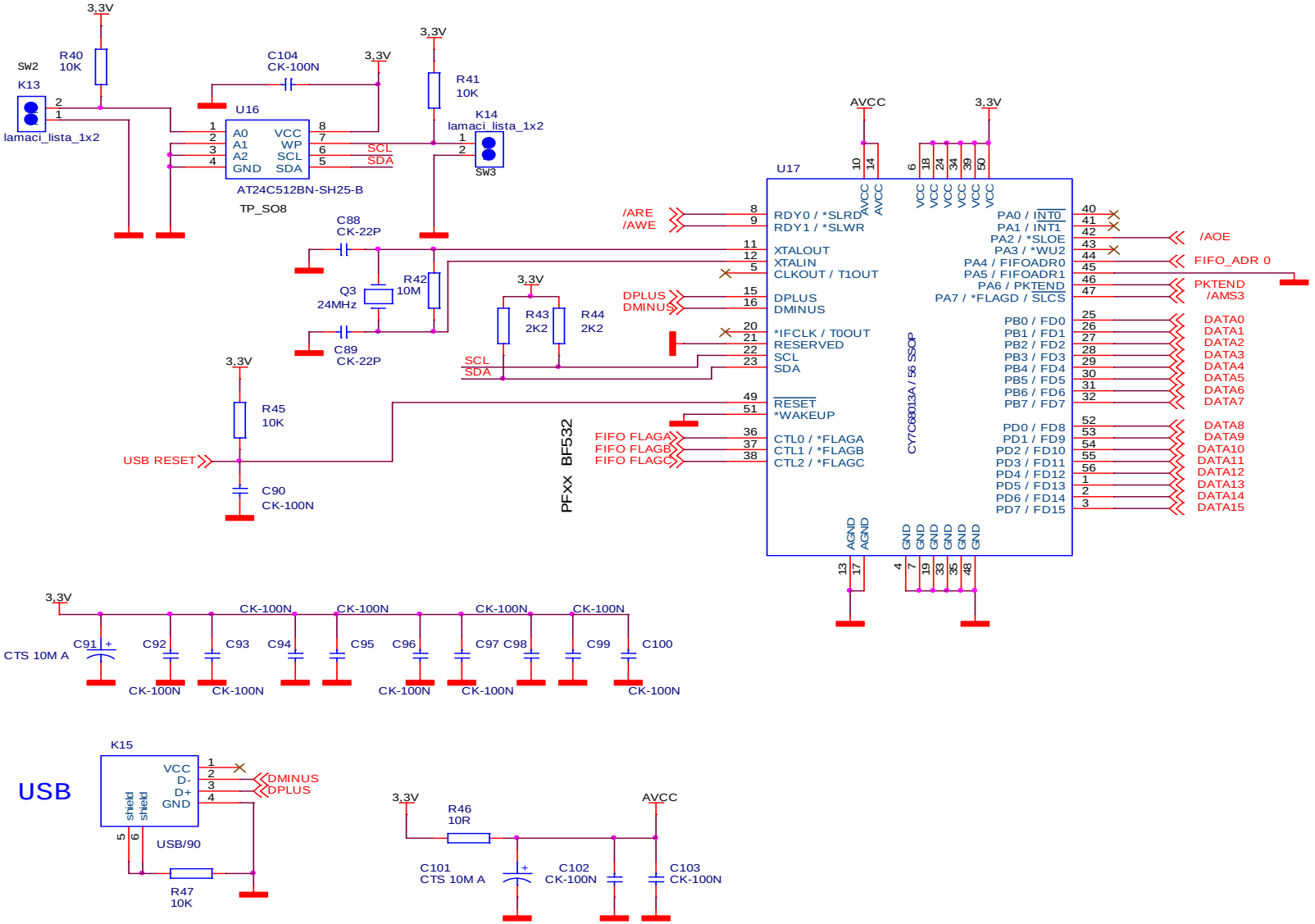
Obr. 7-12 Schéma zapojení obvodu GSS (CPLD) a konektoru pro připojení CCD snímače



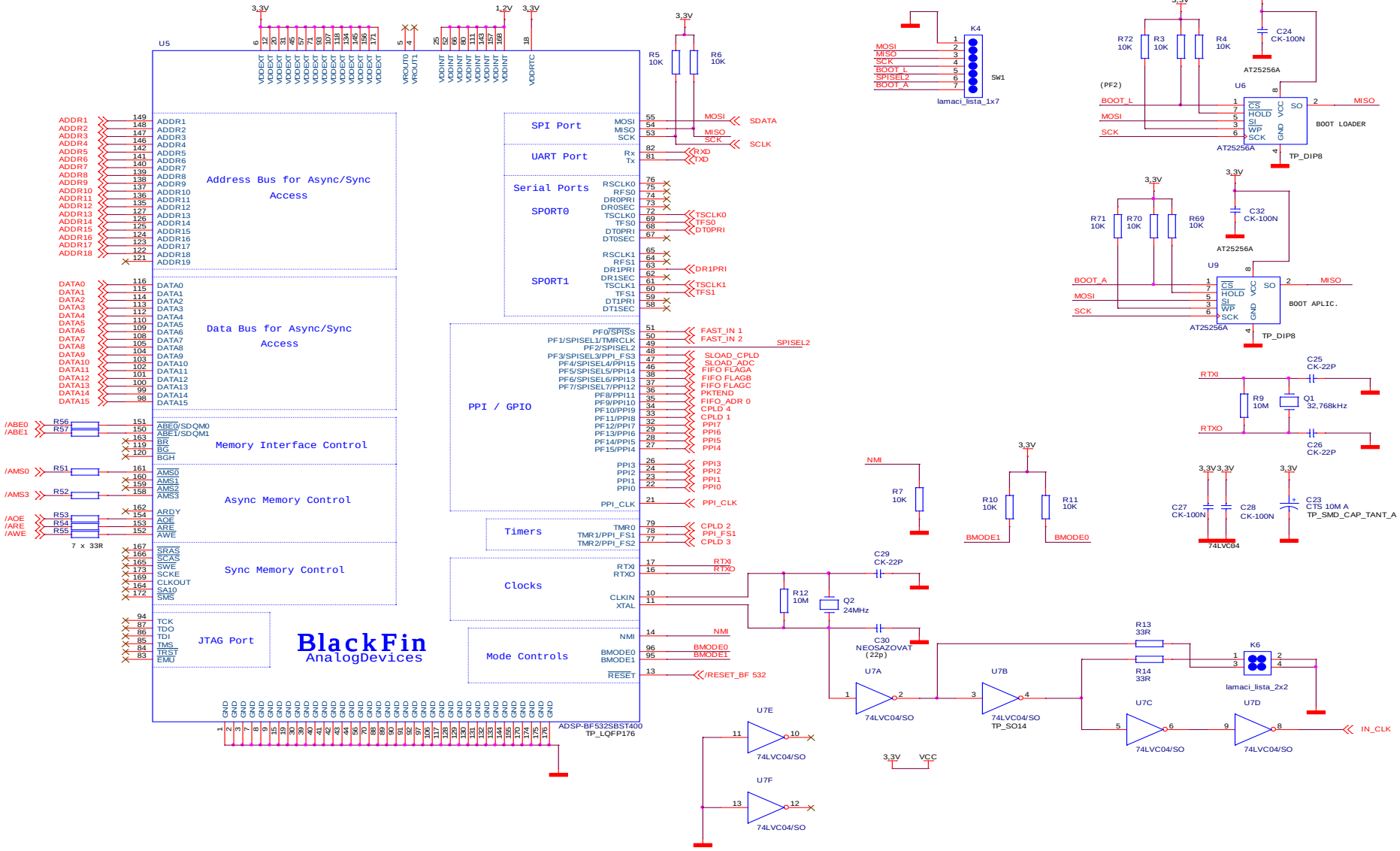
Obr. 7-13 Schéma zapojení binárních vstupu / výstupů, UART, resetovacího obvodu a SRAM paměti



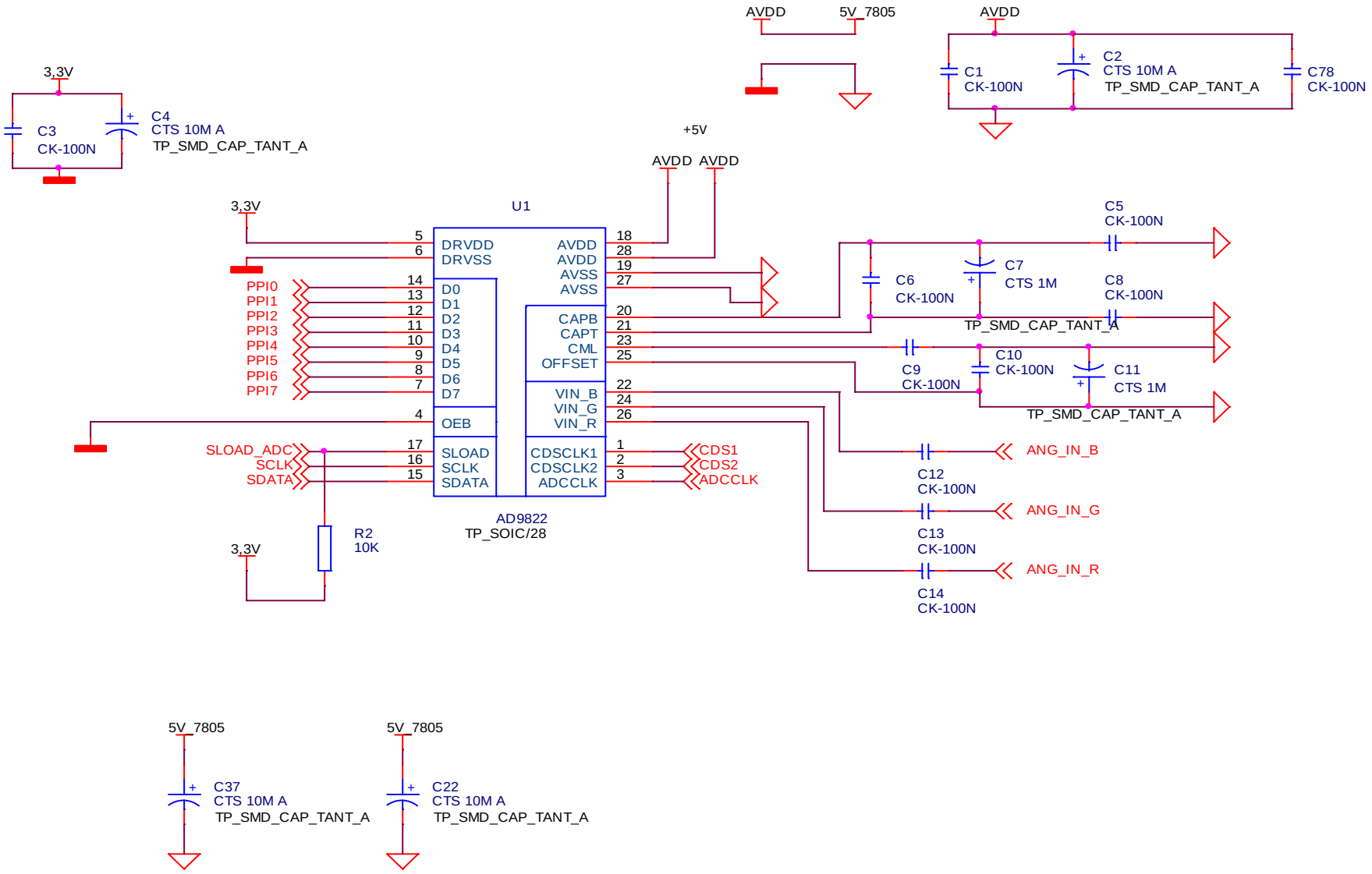
Obr. 7-14 Schéma zapojení napájení desky



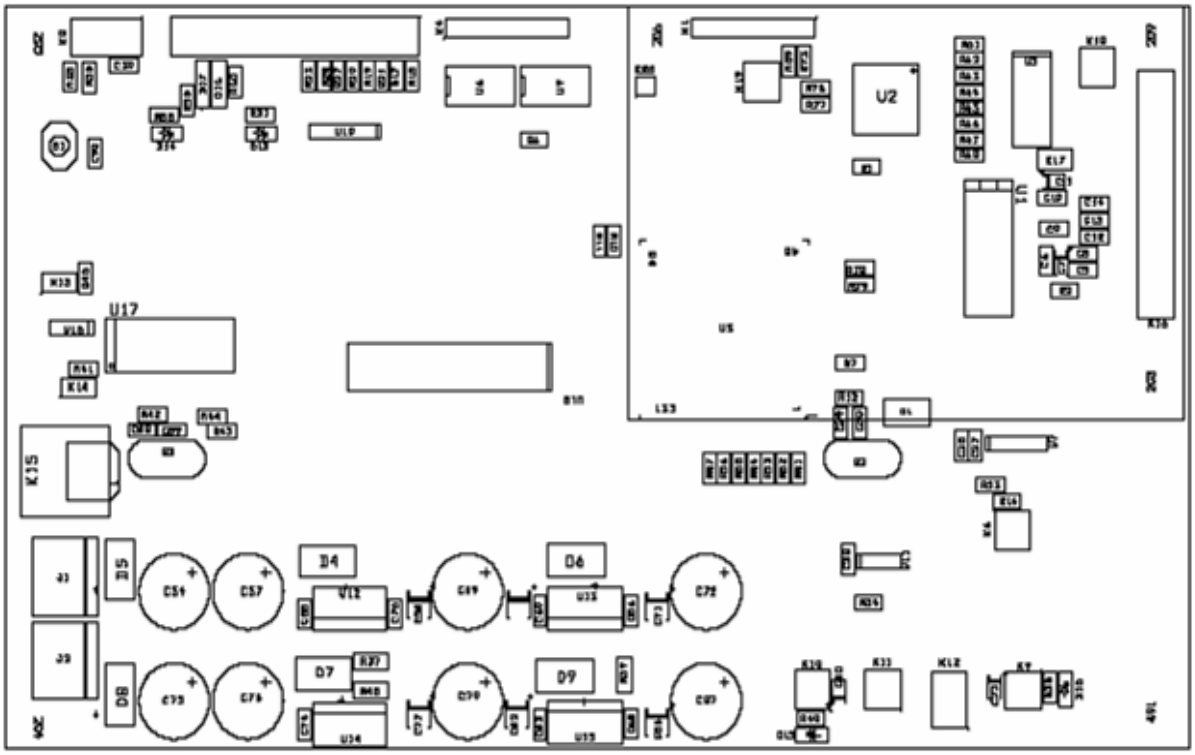
Obr. 7-15 SChéma zapojení USB rozhraní desky



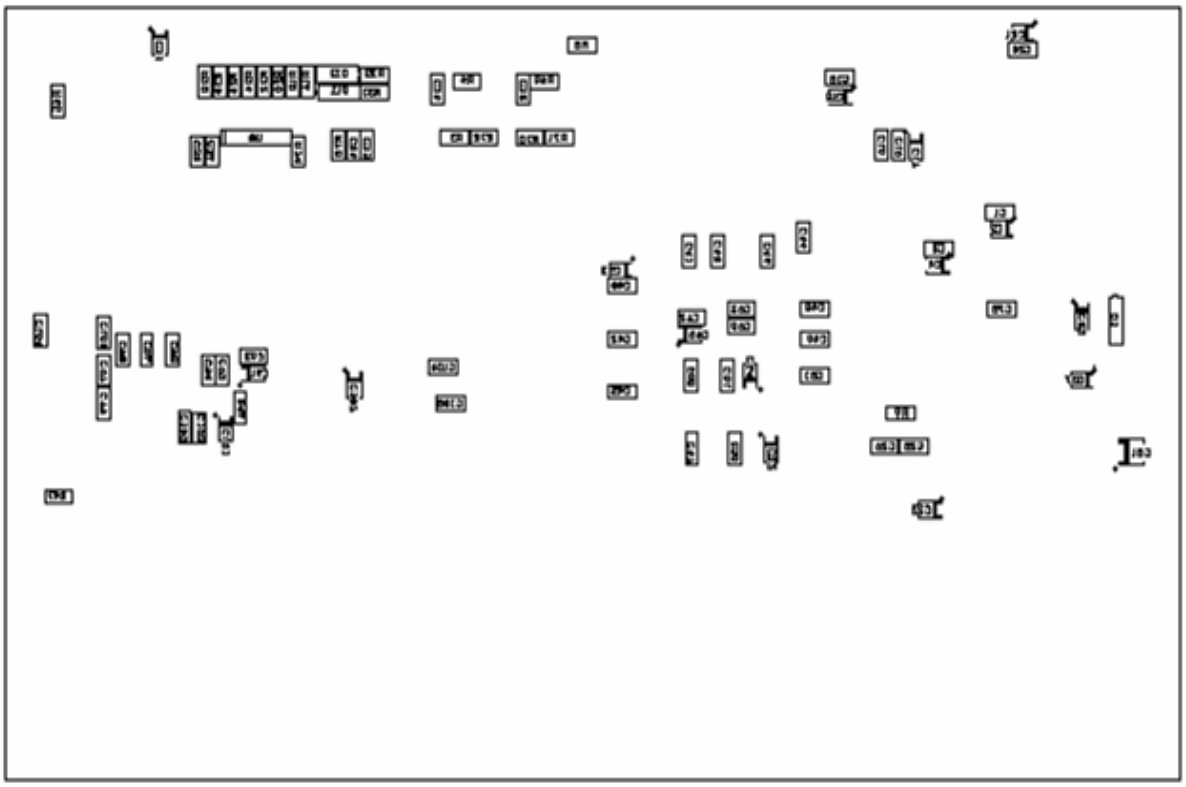
Obr. 7-16 Schéma zapojení procesoru BF532 a BOOT EEPROM



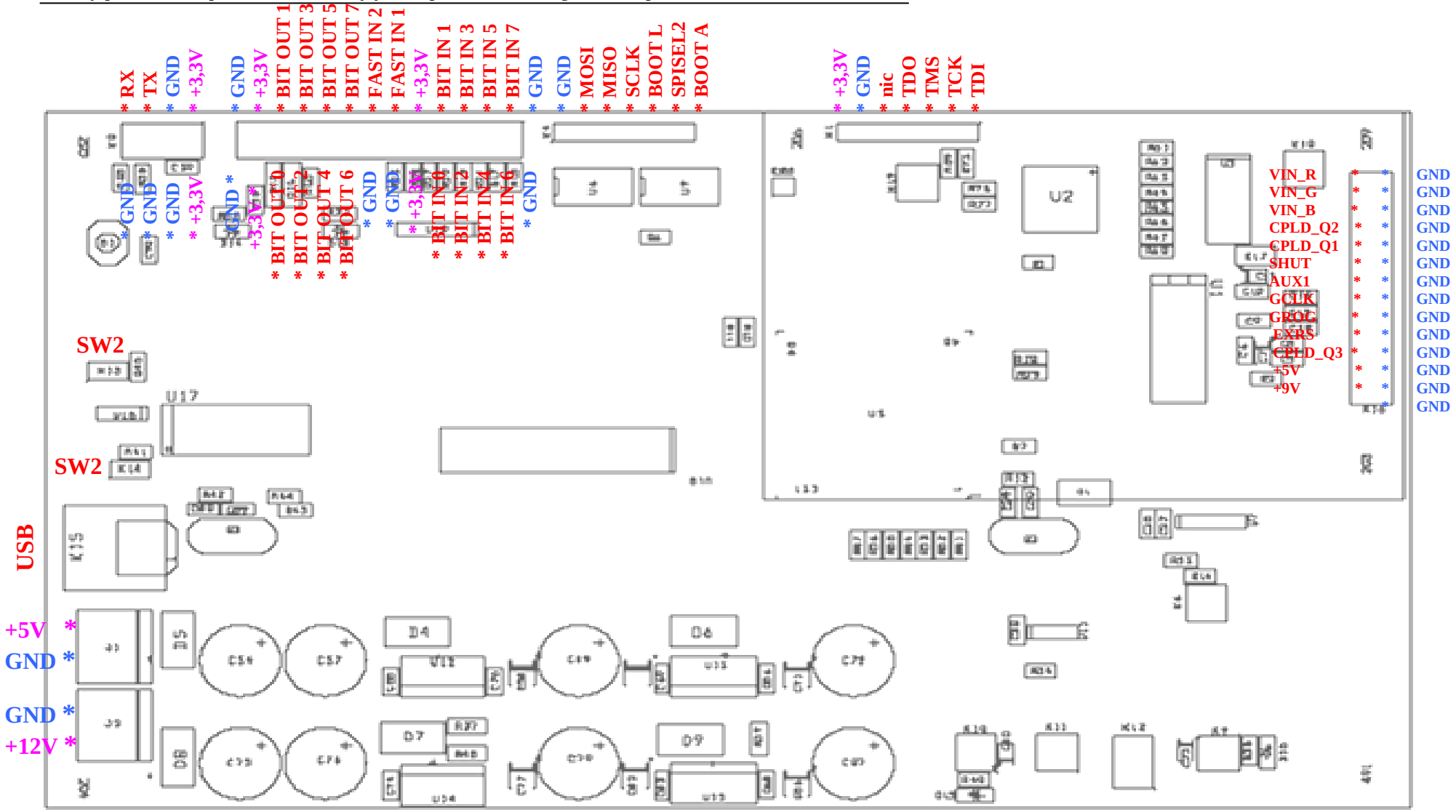
Obr. 7-17 Schéma zapojení obvodu AFE (AD9822)



Obr. 7-18 UCCD – 2007 SSTOP



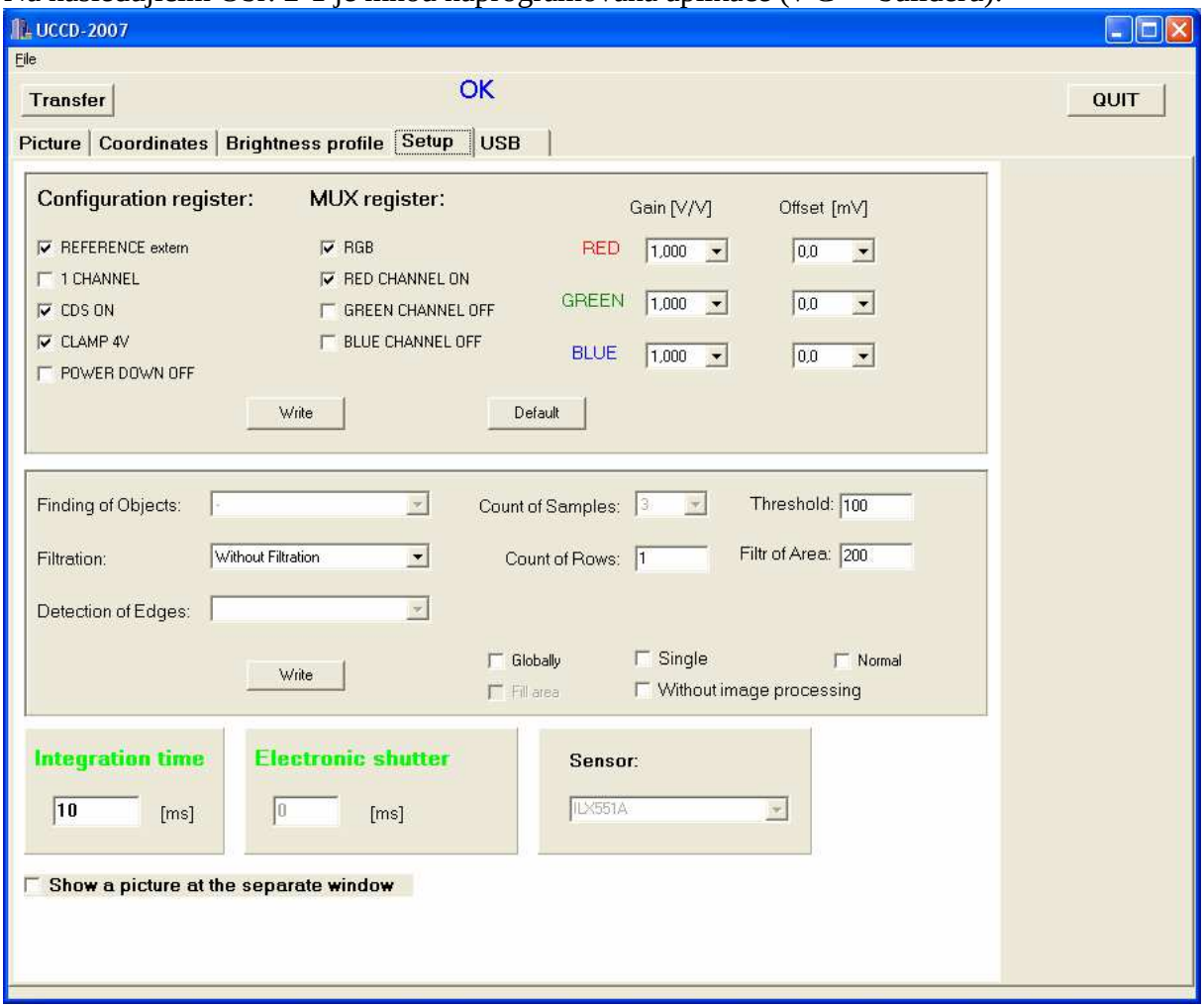
Obr. 7-19 UCCD - 2007 SSBOT



Obr. 7-20 Rozmístění jednotlivých periférií

7.3 Popis aplikace sprostředkovávají komunikace mezi deskou UCCD-2007 a PC přes USB

Na následujícím Obr. 2-1 je mnou naprogramovaná aplikace (v C++ builderu).



Obr. 7-21 Ukázka aplikace

Aplikace má pět záložek (**Picture, Coordinates, Brightness profile, Setup, USB**).

Záložka **Picture** ukazuje nasnímaný obrázek. Obrazek lze uložit kliknutím na obrázek pravým tlačítkem myši a kliknutím na **Copy**. Tím se obrázek přenese do systémové schránky a lze jej vložit např. do windowsovské aplikace Malování.

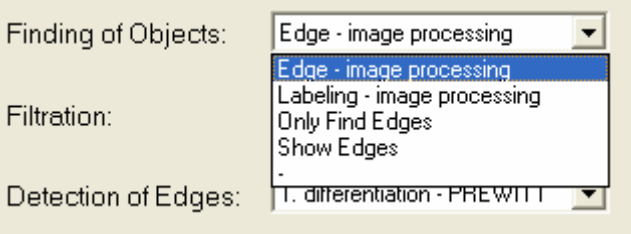
Záložka **Coordinates** zobrazuje souřadnice získané pomocí zvolených metod. Souřadnice lze uložit do souboru kliknutím na tlačítko **Save As**. Tabulka se maže kliknutím na tlačítko **Clear All**. Po zaškrtnutí položky **pix** se povolí přístup do editačního pole pro zadání převodní konstatny z pix na mm.

Záložka **Brightness profile** obsahuje graf zobrazující jasový profil řádku v rozmezí osmi bitové hloubky (0-255). Tlačítko **Clear Graf** maže graf.

Záložka Setup (viz Obr. 7-21) zobrazuje možnosti nastavení procesoru, AFE, volby metod aj.

Nastavení AFE se provádí v prvním panelu této záložky. Význam jednotlivých položek je uveden v kapitole 3.3.2. Pro provedení zvolených nastavení je nutné kliknou na tlačítko **Write**. Tím se přes USB přenesou jednotlivá nastavení AFE. Pro uvedení do defaultního (funkčního) nastavení AFE stačí zmáčknout tlačítko **Default**. Tím se přes USB do AFE zapíšou defaultní nastavení.

Volba metod a zobrazení se děje v následujícím panelu. Výběrové ple **Findnign of Object** umožňuje volbu mezi metodami hledajícími a zpracovavajícími objekty. Jak lze vidět na následujícím obrázku Obr. 7-22, je možno volit mezi těmito položkami:



Obr. 7-22 Položky Findnig of Object

Edge – image processing zpřístupňuje metodu „překrývajících se pruhů“.
Labeling – image processing zpřístupňuje metodu „Labeling“.
Only Find Edges zpřístupňuje zasílání souřadnic nalezených hran bez použití výše zmíněných metod nalezení objektů. Souřadnice se zobrazí v záložce **Coordinates.**
Show Edges zpřístupňuje zobrazování nalezených hran v záložce **Brightness profile** v podobě grafu, kde jednotlivou hrana zobrazí jako vertikální přímku. Dále nalezené hrany pro lepší názornost zobrazí v podobě obrázku v záložce **Picture** a umožní tak ukázat, jak jsou detekční metody nastavené (pro případnou korekci jejich nastavení).
Pro přístup k tomuto poli musí být zaškrtnuto **With image processing.**

Výběrové pole **Filtration** umožňuje volbu způsobu filtrace (**Median, Averaging**) či filtraci nepoužít (**Without Filtration**).

Výběrové pole **Detection of Edges** umožňuje volbu způsobu detekce (**Threshold, 1.differentation - PREWITT**). Pro přístup k tomuto poli musí být zaškrtnuto **With image processing.**

Výběrové pole **Count Of Samples** je aktivní pouze při volbě položky **Averaging** z výběrového pole **Filtration**, jinak je nepřístupné. Nastavuje počet průměrů (řádků účastňujících se průměru).

Editační pole **Count Of Rows** umožňuje nastavení počtu zobrazovaných řádků v záložce **Picture.**
Editační pole **Level** umožňuje nastavení prahovací úrovně pro metody **Threshold** a **1.differentation – PREWITT.**
Fill of Area umožňuje nastavení práhu plochy. Pokud metoda detekuje objekt s plochou menší, než je udáno v editačním poly **Fill of Area**, ignoruje tento objekt a nepošle **jeho** souřadnice do PC.

Zaškrtávací položka **Single** (nezaškrtnutá) nastavuje jednorázový odběr. Počet odběrů je zavislý na položce **Count Of Rows.** Pokud je odebráno tolik řádku, kolik je uvedeno

v **Count Of Rows,** snímání i přenos se zastaví. Je-li tato položka zaškrtnuta, zobrazí se jako **Continual**.

Zaškrtávací položka **Without image processing** (nezaškrtnutá) zpřístupňuje v procesoru pouze filtrační metody. Detekční a hledající metody nejsou aktivovány, pokud není tato položka zaškrtnuta (zabrazí se jako **With image processing**).

Zaškrtávací položka **Globally** umožňuje volbu mezi zasíláním souřadnic až po zkončení průchodu zkoumaného objektu pod kamerou (celkové – výsledné zpracování a posílání konečných a výsledných souřadnic), či jeho průběžném zasílání souřadnic, které se ale neustále s každým načítání dalšího řádku mění (**Continuously**).

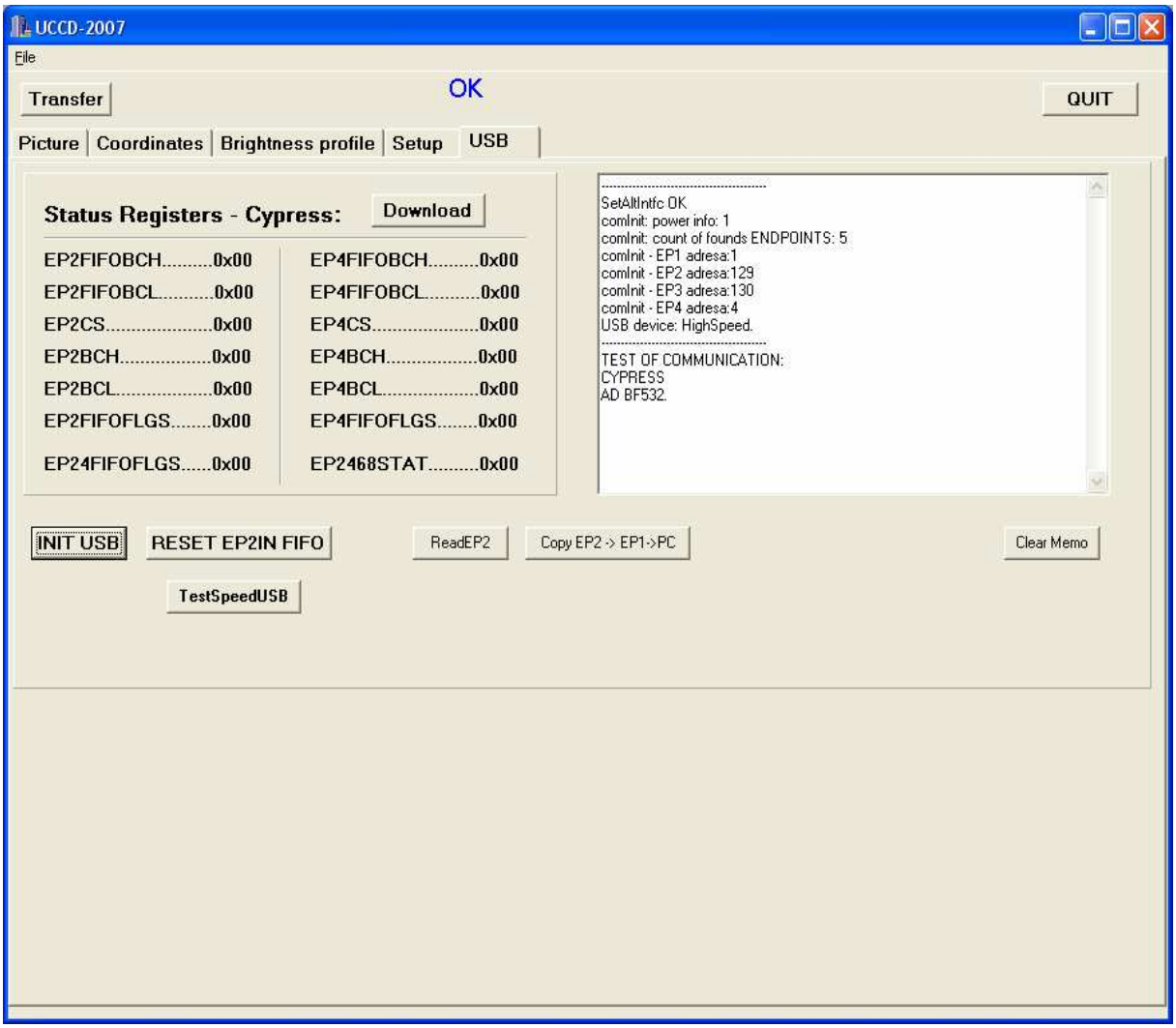
Fill Area po zaškrtnutí této položky se vyplní oblast mezi detekovanou levou a pravou hranou a zobrazí se výsledek v záložce **Picture** a **Brightness profile**. Tot je aktivní pouze pokud bylo zaškrtnuto **With image processing** a zvolena položka **Shod Edges** ve výběrovém poli **Findnig of Object.**

Zaškrtávací položka volí polaritu prahovací úrovně metody Threshold. Pro **Normal** je jasová hodnota pixelu menší rovna než hodnta uvedená v **Level** přepsána na nulu a větší než tato hodnota na jedničku. Pro **Negation** je tomu naopak.

Pro přenos vybraných nastavení přes USB do BF532 je nutné kliknout na tlačítkou **Write**.

Zadáním hodnoty do editační pole **Integration time** a potvrzením stiskem tlačítka na klávesnici Enter se provede nastavení integrační doby v časovači BF532.
Obdobně pro pole **Electronic shutter**, které je ale přístupné, pokud je vybrán příslušný senzor.
Senzor se volí (z bezpečnostních důvodů zatím nepřístupného) výběrového pole **Sensor**. Ten nastaví automaticky počet pixelů a pokud to senzor podporuje i editační pole pro elektronickou závěrku (respektive nastavení doby elektronické závěrky).

Záložka **USB** obsahuje atributy týkající se komunikace přes USB (Cypress).
Po stiku tlačítka **INIT USB** se provede inicializace komunikace USB – nastavení a přidělení adres v aplikaci, kontrola komunikace s obvodem cypress a sprocesorem AD BF532. Výsledek inicializace se zobrazí ve vedlejším informačním okně a modře zobrazeným slovem OK, viz. následující obrázek Obr. 7-23. V opačném přídě se zobrazý varovné hlášení, že se buď USB komunikace nezdařila, či se nezdařila jenom komunikace s procesorem (třeba je omylem v nekoečné smyčce) a aplikace se od USB odpojí a bude nutné provést inicializaci od začátku dalším stiskem tlačítka **INIT USB.** Pokud dojde během přenosu (po úspešné inicializaci) k nenadále komunikační chybě (cypress fifo je plná), zobrazí se červeně slovo FAILD místo modrého OK. Obnovit komunikaci lze opětovným stiskem tlačítka **Transfer**, či restartem celé desky a opětovnou inicializací (**INIT USB**).



Obr. 7-23 Inicializace USB

Stiskem tlačítko **Download se** zobrazí status registry obvodu Cypress.
Stiskem tlačítka **TestSpeedUSB** se zobrazí rychlost přenosu mezi procesorem a aplikací.
Stiskem tlačítka **ReadEP2** se přenesou data z FIFO EP2 do PC a zobrazí se výsledek v informačním okně. Pokud tento přenos nefunguje, ale podle stavových registrů víme, že ve FIFO stále něco je, tak stiskem tlačítka provedeme přenos přes endpoint EP1 do PC s tím ,že se data pouze zkopírují do PC, ale ve FIFO zůstanou (pouze takový test).
Stiskem tlačítka **Clear Memo** vymažeme informační okno.

Stiskem tlačítka **Transfer** spustíme přenos mezi snímací deskou UCCD-2007 a aplikací (PC). Opětovným stiskem přenos ukončíme, pokud byl zvolen nepřetržitý přenos zaškrtnutím položky **Continual.** Může se stát, že při jednorázovém odběru jedno řádku (**Single**) se nezobrazí jasový profil (zobrazí se nulová hodntoa jasu). To je z toho důvodu, že v procesoru BF532 zpracovávám a zároveň pořizuju obrázek najednou (paralelně – výhoda DMA). Proto při první přenosu se zobrazí nulová hodnota, protože se přenesl vynulovaný (inicializovaný) registr, kam řádek ukládám.
Stiskem tlačítka **QUIT** ukončíme aplikaci.