TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 1802T007 – Informační technologie

Obvod FPGA jako zdroj obrazových dat pro operační systém Microsoft Windows

FPGA circuit as a source of visual data for the Microsoft Windows operating system

Diplomová práce

Autor: **Bc. Zdeněk Kořínek**
Vedoucí práce: Ing. Jan Václavík
Konzultant: Ing. Ondřej Zelinka, Ph.D.

V Liberci 11. 09. 2012

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Datum: 11. 09. 2012

Podpis

Poděkování

Nejprve bych chtěl poděkovat Ing. Janu Václavíkovi za podnětné rady, poskytnuté vědomosti a příkladné odborné vedení po celou dobu práce. Moje další poděkování patří doc. Ing. Milanu Kolářovi, CSc. za odborné a věcné připomínky. Závěrem děkuji svým rodičům za podporu při mých studií.

Abstrakt

Tato práce se zabývá tvorbou zařízení, které by poskytovalo vědecká obrazová data získaná z atypického senzoru a tato data transportovalo po vhodné sběrnici k počítačové platformě, kde by data byla předána do struktury operačního systému Windows. V práci je popsán protokol sběrnice USB, jakožto sběrnice zvolené pro tvorbu takového zařízení, hlavně pak požadavky nutné k zajištění komunikace zařízení po sběrnici USB. Dále je v práci popsána třída USB Video Device Class sloužící pro tvorbu aplikací přenášejících obrazová data po sběrnici USB. V jazyce VHDL byla vytvořena aplikace, která získává ze senzoru obrazová data a tato data dále předává pro transport po sběrnici USB. Tato aplikace byla implementována v FPGA obvodu. S využitím vhodného USB kontroléru bylo vytvořeno zařízení, implementující třídu USB Video Device Class, které data přenáší po sběrnici USB.

Klíčová slova: UVC, USB, FPGA

Abstract

The topic of this thesis is the design of a device that will supply visual scientific data gathered from an atypical sensor and transport this information via an appropriate bus to a computer platform, where it would be transferred to the structure of the Windows operating system. The thesis describes the USB protocol as a bus chosen for this design, especially the requirements necessary for establishing communication with the device via a USB bus. Furthermore it describes the USB Video Device Class, which serves the design of applications transferring image data via a USB bus. An application has been coded in the VHDL programming language which gathers visual data from the sensor and passes it on for transfer over a USB bus. This application has been implemented in a FPGA circuit. Using an appropraite USB controller and implementing the USB Video Device Class, a device has been created which transfers the data via a USB bus.

Keywords: UVC, USB, FPGA

Obsah

Seznam nejpoužívanějších symbolů, zkratek a termínů

USB	…	Universal Serial Bus
FPGA	…	Field-programmable gate array
UVC	…	USB Video device class
VHDL	…	VHSIC hardware description language
ID	…	Identifikační číslo
A/D převodník	…	Analogově digitální převodník
GUID	…	Globally unique identifier
GPIO pin	…	General Programable Input Output pin
MSB	…	Most significant bit
LSB	…	Least significant bit

Seznam tabulek

Seznam obrázků

1 Úvod

Zpracování obrazu je obor, který zasahuje do mnoha aspektů lidského života, nejčastěji se s ním ale můžeme setkat v prostředí průmyslu a prostředí vědeckém. Obzvláště pak ve vědeckém prostředí je mnohdy potřeba získávat a zpracovávat obrazová data z atypických senzorů. Kromě senzorů, které vytvářejí klasická obrazová data, je možné narazit i na senzory, které vytvářejí data, která svým charakterem připomínají obrazová data, a proto je možné použít podobných metod pro jejich zpracování. Pro získání dat z takovýchto atypických snímačů existuje velké množství řešení. Nejjednodušší způsob je zakoupení specializované kamery obsahující požadovaný senzor. Toto řešení s sebou ale přináší několik nevýhod, konkrétně často velkou pořizovací cenu specializovaného kusu hardwaru nebo i nutnost instalace speciálních ovladačů zařízení. Může se samozřejmě ale stát, že vhodná specializovaná kamera není průmyslově vyráběna. Dalším možným řešením, které se nabízí, je tedy vytvoření vlastního zařízení, které by splňovalo potřebné funkce. Toto řešení s sebou přináší i možnost implementování algoritmů, které by obraz předzpracovávaly již v hardwarové části, což by vedlo ke snížení nároků na platformu zpracovávající data a možnému zrychlení samotného zpracování dat.

Tato práce si klade za cíl vytvoření rozhraní, které by umožňovalo práci s atypickým optickým senzorem a předávání dat získaných z tohoto senzoru do struktury operačního systému Microsoft Windows přes běžné komunikační rozhraní. Pro jádro tohoto zařízení byl zvolen FPGA obvod, který nabízí velkou rychlost zpracování dat danou realizací funkcí pomocí vnitřních programovatelných logických obvodů zařízení. Toto řešení umožňuje paralelní zpracování některých operací, což může být v případě obrazových dat, nebo dat majících charakter obrazových dat, velice výhodné. Tohoto paralelního zpracování by bylo možné využít v případě, že by obvod měl data nějakým způsobem předzpracovávat. Neposlední výhodou FPGA obvodu je možnost daný obvod přeprogramovat dle potřeb uživatele, tedy v případě, že by bylo třeba využít senzoru s jinými parametry, nebo přidání algoritmu předzpracovávajícího data. Nevýhodou FPGA obvodu je bohužel často poměrně malá vnitřní paměť obvodu, která by pravděpodobně nebyla dostačující, pokud by bylo nutné získávat obrazová data s vysokým rozlišením. V takovém případě by pak bylo nutné využití externí paměti.

Pro transport dat ze zařízení do počítačové platformy je možné vybírat ze široké škály sběrnic, ať už starších typů jako sériový port RS-232 nebo paralelní port IEEE 1284, nebo standardních sběrnic USB nebo FireWire (IEEE 1394). Kromě těchto sběrnic je možné pro transport dat využít i ethernet.

Při výběru vhodného prostředku pro transport dat ze zařízení do počítačové platformy je potřeba nahlížet i na samotnou koncepci předávání dat. Je možné vytvořit zařízení, které bude vyžadovat vlastnoručně napsaný driver, který by odpovídal specifikaci Windows Driver Model a zajišťoval transport do struktury operačního sytému Windows. Toto řešení je poměrně elegantní, ale vyžaduje znalost vnitřní hierarchie tohoto modelu. Kromě toho by bylo nutné tento ovladač instalovat na veškeré stanice, kde by bylo třeba zařízení využívat. Dalším způsobem je využití obecného ovladače sběrnice a vytvoření aplikace, která by odeslaná data sbírala a následně předávala do struktury operačního systému. Toto řešení má opět nevýhodu v tom, že vyžaduje aplikaci nainstalovanou na každé stanici, na které chceme zařízení využívat. Kromě toho je data často potřeba ke zpracování dat použít výpočetní software, například Matlab, který umožňuje práci se zdrojem obrazových dat, které ale musí vyhovovat standardu Windows a pak je tento způsob nepoužitelný. Poslední možnost, kterou nabízejí některé sběrnice, je možnost využít ovladače pro zařízení vytvářející obrazová data, který je již implementován do systému Windows. Obrovskou výhodou tohoto řešení je to, že na počítačovou platformu, která bude obrazová data zpracovávat, není nutné instalovat žádný dodatečný software. Nevýhodou je pak to, že toto řešení může klást na vývoj zařízení některé omezující nároky a je nutné se přizpůsobit přesně definované struktuře.

Po zvážení dostupných možností a všech parametrů byla jako sběrnice zvolena sběrnice USB a to hned z několika důvodů. Prvním z důvodů je dostupnost této sběrnice na drtivé většině dnešních počítačových platforem a podpora mechaniky Plug and Play, tedy připojení zařízení za chodu počítače bez nutnosti dodatečné konfigurace. Dalším důvodem je pak to, že USB nabízí třídu USB Video Device Class (též USB Video Class nebo UVC), která nabízí možnost popisu video zařízení vytvářejícího proud obrazových dat, přičemž pro tuto třídu je dostupný ovladač Usbvideo.sys, který je již součástí operačního systému Windows. Tento ovladač je dostupný pro verzi USB Video Class 1.0 v operačním systému Windows Vista, Windows XP service pack 2 a v libovolném novějším operačním systému Microsoft Windows a nabízí podporu pro rozhraní Media Foundation a DirectShow. [1] [2] [3]

Pro realizaci samotné USB komunikace byl zvolen vývojový kit obsahující kontrolér EZ-USB FX3 nabízející podporu USB verze 3.0 s podporující vysokorychlostní přenos obrazové informace a také podporu velkého množství komunikačních sběrnic pro připojení FPGA obvodu.

Jako vzorový zdroj produkující obrazová vědecká data byl vybrán pyroelektrický řádkový senzor 128LTI se 128 elementy a k tomuto senzoru byl zvolen A/D převodník ADS8363EVM umožňující konverzi hodnoty každého elementu na 16 bitovou hodnotu.

Teoretická část práce se zabývá rešerší požadavků pro implementaci obecného USB zařízení pro verzi USB 2.0 a dále samotné třídy USB Video Device Class. Praktická část je pak rozdělena na dvě části. První část se zabývá návrhem aplikace v jazyce VHDL, která slouží pro ovládání senzoru a A/D převodníku a sběr obrazových dat a předávání těchto dat pro transport po sběrnici USB. Druhá část se zabývá implementací komunikace po sběrnici USB.

Teoretická část

2 Universal serial bus

2.1 Charakteristika sběrnice a protokolu

Universal serial bus (USB) je, jak napovídá anglický název, koncipováno jako univerzální sběrnice, která by umožnila připojit libovolné zařízení bez nutnosti větší konfigurace systému a přenášet data vysokou přenosovou rychlostí. Díky této filozofii je dnes USB sběrnice rozšířena na drtivé většině platforem a zařízení.

Data jsou po sběrnici přenášena sériově, přičemž verze USB 1.1 umožňuje dva základní módy přenosu, low speed a full speed. Low speed mód umožňuje přenosovou rychlost až 1.5Mbits/s, full speed mód potom přenosovou rychlost až 12Mbits/s. Verze USB 2.0 přinesla nový mód high speed, kterým je možné přenášet data rychlostí až 480Mbits/s. Jakékoliv zařízení implementující verzi 2.0 je zpětně kompatibilní s verzí 1.1. Verze USB 3.0 potom přináší dosud poslední mód známý jako SuperSpeed. Tento mód nabízí přenosovou rychlost až 5 Gbit/s. Opět platí, že libovolné zařízení verze 3.0 by mělo být kompatibilní s libovolnou nižší verzí. [4] [5]

USB sběrnice pracuje na principu klient – host, kdy strana hosta tzv. hostuje klienta, tedy kontroluje všechny parametry přenosu. U každého přenosu může být vždy pouze jeden host. Po připojení zařízení host detekuje připojení nového zařízení a po sérii vyjednávání se zařízením vybere vhodné ovladače zařízení. Tento způsob klade větší nároky na aplikaci, která se chová jako host, ale ulehčuje vývoji zařízení na druhé straně transakce. [4]

Aby bylo možné snadno rozeznat libovolné zařízení a dle toho mohl host vybrat parametry transakce, definuje USB protokol takzvané deskriptory. Deskriptory mají vnitřní hierarchii a obsahují informace o zařízení, například informaci o výrobci zařízení, jakou verzi USB zařízení podporuje, jaké jsou možné konfigurace zařízení, zda má být zařízení napájeno ze sběrnice, či má vlastní napájení, a tak dále. Kromě výše zmíněného umožňují deskriptory zařadit zařízení do nějaké třídy, tedy skupiny zařízení splňujících nějakou konkrétní funkci, například třída Human Interface Device (HID) sloužící pro popis zařízení pro interakci uživatele s počítačovou platformou, tedy myš, klávesnice, atd. Každá třída pak přináší nároky

na konfiguraci parametrů zařízení, například nutnost další skupiny deskriptorů a/nebo implementovaných metod. V této práci je využita třída USB Video Device Class, která spojuje skupinu zařízení sloužících pro přenos obrazové informace. [1] [2]

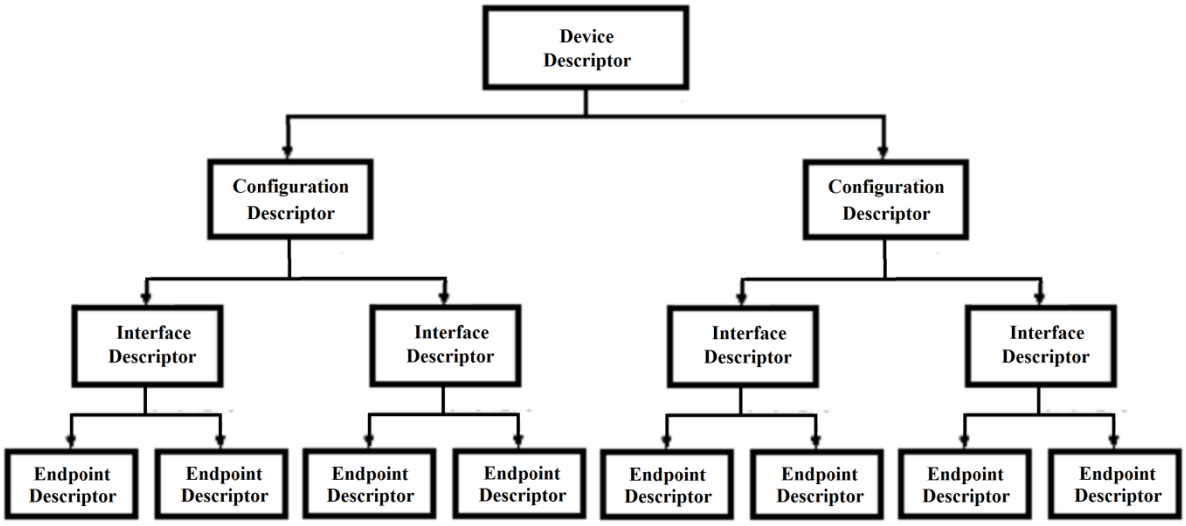
Každé klientské zařízení má jeden a více koncových bodů (endpoint), které slouží pro příjem nebo vysílání dat. Tyto koncové body mohou být různého typu, přičemž dle zvoleného typu jsou pak různé parametry transakce informací přes tento koncový bod. Každý z těchto bodů má svoje přiřazené číslo, přičemž platí, že každé zařízení musí mít koncový bod 0. Pomocí tohoto koncového bodu pak probíhá veškerá komunikace, která se týká nastavení parametrů spojení a komunikace. [1]

Samotný transport informací probíhá ve formě paketů, přičemž protokol definuje tři základní druhy paketů. *Token packet*, který vždy generuje host a pomocí tohoto paketu sděluje klientskému zařízení, co bude následovat, tedy zda budou od klienta vyžadována nějaká data, zda budou data klientovi zaslána nebo zda bude další transakce sloužit pro úpravy nastavení klientského zařízení. Velikost tohoto paketu je vždy stejná pro libovolnou přenosovou rychlost. Speciální variantou token paketu je pak *Start of Frame Packet*, který se využívá pro synchronizaci hodin vysílače a přijímače. Tento paket má konstantní velikost, dle přenosové rychlosti se ale liší časové intervaly, v kterých je tento paket odesílán. Pro full speed je generován paket jednou za 1.00 ms ±0.0005 ms, pro high speed je tento interval 125 μs ±0.0625 μs. Dále pak protokol definuje takzvaný *Data packet*, tedy paket obsahující fyzická data. Velikost tohoto paketu se liší dle přenosové rychlosti. Posledním typem je *Handshake packet*, což je paket sloužící pro potvrzení transakce, tedy pro rozpoznání, zda transakce proběhla úspěšně, nebo zda je třeba operaci opakovat kvůli chybě při přenosu. Posledním typem je *Token packet*, který je pomocí svého vnitřního ID nastaven jako takzvaný *Setup packet* slouží pro požadavky (requesty) hosta na zařízení. Tyto requesty slouží pro nastavení parametrů spojení a komunikace. Protokol diktuje, že každé zařízení musí reagovat na všechny základní requesty společné pro libovolné zařízení. [1]

2.2 Deskriptory obecného zařízení

Každé obecné zařízení, chovající se jako klient, musí obsahovat některé deskriptory, které jsou společné pro libovolné klientské USB zařízení. Tyto deskriptory jsou pak postupně předány hostu transakce a ten z nich vybere nejvhodnější nastavení. Každý deskriptor se skládá z několika bytů a má stejnou základní strukturu, kde platí, že první byte každého deskriptoru obsahuje informaci o délce daného deskriptoru a druhý byte definuje typ deskriptoru. Pokud je délka deskriptoru menší, než by měla být dle specifikace deskriptoru daného typu, je tento deskriptor hostem ignorován. Pokud je délka větší, host ignoruje pouze nadbytečné byty. [1]

Jakékoliv USB klientské zařízení musí obsahovat: device descriptor, tedy descriptor popisující základní parametry zařízení, configuration descriptor, tedy descriptor obsahující informace o jedné konkrétní konfiguraci zařízení, interface descriptor, tedy descriptor obsahující informace o jednom konkrétním rozhraní jedné konfigurace zařízení, a nakonec endpoint descriptor, tedy descriptor obsahující informace o jednom konkrétním endpointu, tedy vstupu nebo výstupu zařízení, z kterého nebo do kterého proudí data. Přesná struktura této hierarchie je zachycena na obrázku 2.1. [1]



Obr. 2.1 Hierarchie základních deskriptorů libovolného USB zařízení [4]

Na tomto obrázku je jasně vidět, že každé zařízení musí mít přesně jeden device descriptor, ale dále může zařízení implementovat více různých konfigurací a každá z těchto konfigurací může obsahovat více rozhraní, každé pro více endpointů.

2.2.1 Device descriptor

Každé zařízení musí mít přesně jeden device descriptor. Tento deskriptor přímo popisuje základní parametry klientského zařízení jako verzi USB, pro kterou je zařízení koncipováno, informaci o výrobci, ID zařízení důležité pro přiřazení správného ovladače a počtu dostupných konfigurací. Pokud je v descriptoru uveden menší počet konfigurací, než kolik jich je dostupných, budou nadbytečné konfigurace ignorovány. Tento deskriptor také obsahuje informaci o tom, zda je zařízení koncipováno dle nějaké konkrétní třídy. Je tedy možné určit třídu zařízení již pomocí tohoto prvního deskriptoru, nicméně se spíše setkáme se zařízeními, která specifikují třídu až na úrovni rozhraní. Tento přístup je výhodný zejména z toho důvodu, že zařízení pak může plnit více rozličných funkcí, respektive může implementovat více tříd. Délka tohoto descriptoru musí být vždy přesně 18 bytů. [1]

Přesná struktura tohoto deskriptoru je zachycena v tabulce 1 v příloze A.

2.2.2 Configuration descriptor

Každé zařízení musí mít alespoň jeden Configuration descriptor. Tento deskriptor obsahuje informace o jedné konkrétní konfiguraci zařízení, a to informaci o způsobu napájení zařízení, tedy zda je napájeno ze sběrnice nebo zda má vlastní napájení, jaká je maximální spotřeba zařízení v případě, že je napájeno přímo ze sběrnice, kolik konfigurace obsahuje možných rozhraní a celkovou délku všech descriptorů, respektive délku všech deskriptorů rozhraní a koncových bodů, vztahujících se k této konfiguraci. [1]

Poté, co host získá od zařízení informaci o všech dostupných konfiguracích zařízení, vybere z této skupiny konfiguraci pro něj nejvhodnější. [1]

Přesná struktura tohoto deskriptoru je zachycena v tabulce 2 v příloze A.

2.2.3 Interface descriptor

Tento deskriptor popisuje jedno konkrétní rozhraní, tedy skupinu koncových bodů zařízení, které jsou určeny pro nějakou konkrétní činnost zařízení. Každá konfigurace může mít více rozhraní a každé rozhraní může spojovat větší množství koncových bodů, přičemž jednotlivá rozhraní jedné konfigurace mohou vykonávat různou funkci. Vzhledem k tomu, že host může libovolně vybírat mezi rozhraními jedné konfigurace, je tento přístup výhodný, neboť umožňuje, aby zařízení při jedné konfiguraci mohlo vykonávat více funkcí. Samotný deskriptor pak obsahuje informace o počtu koncových bodů tohoto rozhraní a případně třídě, dle které je rozhraní implementováno. [1]

Přesná struktura tohoto deskriptoru je zachycena v tabulce 3 v příloze A.

2.2.4 Endpoint descriptor

Tímto deskriptorem je popsán jeden koncový bod jednoho rozhraní. Každý koncový bod, respektive jeho deskriptor, má svoje číslo, pomocí kterého je možné tento koncový bod adresovat. Platí, že není třeba popisovat vlastnosti koncového bodu 0. Vlastnosti tohoto bodu jsou automaticky nastaveny dle parametrů, které diktuje protokol, ihned po připojení zařízení. Deskriptor dále obsahuje typ koncového bodu dle způsobu přenosu dat, který tento koncový bod umožňuje, maximální velikost packetu, kterou koncový bod může zpracovat, a časový interval jednotlivých přenosů u některých typů přenosu. [1]

Přesná struktura tohoto descriptoru je zachycena v tabulce 4 v příloze A.

2.2.5 String Descriptor

Posledním typem základního deskriptoru je takzvaný *String descriptor*. Ten nemá z hlediska vlastností zařízení žádnou funkci, ale poskytuje uživateli textovou, snadno čitelnou informaci o zařízení. Text musí být v kódování Unicode, přičemž je možná podpora vícejazyčného deskriptoru. [1]

Vzhledem k tomu, že string descriptor není pro funkci zařízení důležitý, nebude v této práci více popisován.

2.3 Typy koncových bodů

Každý koncový bod může být jednoho ze čtyř možných typů určených dle způsobu přenosu, který přes tento koncový bod bude probíhat. Jednotlivé typy jsou:

Řídicí přenos (tzv."control transfer") se využívá pro řízení zařízení, nejčastěji během tzv. enumerace. Tento typ přenosu obsahuje kontrolu chyb. Koncový bod 0 je vždy nastaven jako koncový bod pro řídicí přenos. [6]

Přenos při přerušení (tzv. "interrupt transfer") je periodický způsob přenosu, kdy host zjišťuje stav zařízení. Přenos nastane pouze tehdy, když zařízení předá žádost o interakci. Typicky se využívá pro klávesnice nebo myš. [6]

Hromadný přenos (tzv. "bulk transfer") je využíván v případě, že je třeba předávat velké množství dat, u kterého je nutné kontrolovat chyby, ale není prioritní čas vyřízení. Tento způsob je typický pro tiskárny nebo scannery. [6]

Izochronní přenos (tzv. "isochronous transfer") je periodický způsob přenosu, u kterého je rozhodující, aby data byla předána v určitém časovém intervalu. Je možné přenášet velké množství dat, ale tento způsob přenosu neobsahuje kontrolu chyb. Typicky se využívá při přenosu audio nebo video signálu. [6]

2.4 Struktura paketů

Jak bylo již zmíněno výše, komunikace po USB sběrnici probíhá pomocí paketů, přičemž protokol definuje tři základní druhy paketů, jedná se o tzv. Token paket, Data paket a Handshake paket. Speciální formou token paketu je paket Start of frame s lehce odlišnou strukturou. Každý z těchto typů paketu se skládá z několika polí, přičemž každé pole obsahuje

specifickou informaci. Vzhledem k tomu, že některá pole jsou společná pro více druhů paketů, budou nejprve zmíněna všechna pole a až poté struktura jednotlivých typů paketů. [1]

2.4.1 Pole v paketech

Sync – Každý paket musí začínat polem sync, které slouží pro synchronizaci vnitřních hodin přijímače a vysílače. Velikost tohoto pole se liší dle přenosové rychlosti. Pro módy low speed a full speed je to 8 bitů, pro mód high speed je to 32 bitů. Poslední dva bity tohoto pole indikují, kde začíná další pole. [1]

PID (Packet ID) – Toto pole slouží pro identifikaci typu paketu. Jeho velikost je vždy 8 bitů, respektive 2*4 bity, přičemž obě čtveřice obsahují stejnou informaci. Tato redundance je zde z důvodu kontroly. Všechny varianty PID jsou zobrazeny v tabulce 2.1. [1]

Typ paketu	PID	Označení	Popis
Token	0001B	OUT	Transakce host-zařízení
	1001B	IN	Transakce zařízení-host
	0101B	SOF	Start of frame
	1101B	SETUP	Setup paket
Data	0011B	DATA0	Sudý blok dat
	1011B	DATA1	Lichý blok dat
	0111B	DATA2	Paket dat pro isochronní high speed mód
	1111B	MDATA	Paket dat pro isochronní high speed mód
Handshake	0010B	ACK	Příjem bez chyb
	1010B	NAK	Nebylo možné přijmout nebo odeslat data
	1110B	STALL	Není možná komunikace
	0110B	NYET	Zatím bez odpovědi (No response yet from receiver)
Speciální	1100B	PRE	Host-issued preamble
	1100B	ERR	Split Transaction Error Handshake
	1000B	SPLIT	Speciální token pro mód high speed
	0100B	PING	Speciální token pro mód high speed
	0000B	Reserved	Rezervováno

Tab. 2.1 PID [1]

ADDR – Pole pro adresu zařízení, pro které je paket určen. Toto pole má vždy velikost 7 bitů, přičemž hodnota 0000000b není přípustná. [1]

ENDP – Pole obsahující číslo koncového bodu zařízení, přičemž pro full speed a high speed zařízení je velikost tohoto pole čtyři bity, tedy maximálně 16 koncových bodů. [1]

CRC (Cyclic Redundancy Check) – Pole sloužící pro kontrolu. U datových paketů je velikost tohoto pole 16 bitů, u ostatních typů paketu pak pouze 5 bitů. [1]

EOP (End of packet) – Pole označující konec paketu. [1]

2.4.2 Skladba paketů

Struktura Token paketu:

Sync	PID	ADDR	ENDP	CRC5	EOP

Struktura Data paketu:

Sync	PID	Data	CRC16	EOP

Pole Data obsahuje informaci, která se má přenášet. Velikost tohoto pole je omezena dle přenosové rychlosti, pro low speed mód je to 8 bajtů, pro full speed 1023 bajtů, pro high speed 1024 bajtů. [1]

Struktura Handshake paketu:

Sync	PID	EOP

Struktura paketu Start of frame:

Sync	PID	Číslo framu	CRC5	EOP

Pole číslo framu obsahuje číslo zapsané pomocí jedenácti bitů. [1]

2.5 Requesty

Request je požadavek hosta na zařízení prostřednictvím specifického Token paketu. Tento paket má pevnou délku 8 bytů a pevnou strukturu, která je zobrazena v tabulce 2.2.

Offset	Označení pole	Velikost v bajtech	Typ hodnoty	Popis
0	bmRequestType	1	Bitmapa	Bitmapa ve tvaru: B7 – Směr případné transakce datového paketu: 0 = směr host -> zařízení 1 = zařízení -> host B6..5 – Typ requestu: 0 = Standardní 1 = Interface 2 = Vendor 3 = Rezervováno B4..0 – Přijemce requestu: 0 = Zařízení 1 = Rozhraní 2 = Koncový bod 3 = Jiné 4 až 31 = Rezervováno
1	bRequest	1	Hodnota	Hodnota tohoto pole je závislá na typu requestu
2	wValue	2	Hodnota	Obsah se mění dle typu requestu
4	wIndex	2	Index nebo Offset	Obsah se mění dle typu requestu
6	wLength	2	Počet	Určuje délku dat pro přenos, pokud má následovat přenos dat

Tab. 2.2 Struktura setup paketu [1]

Na základě typu requestu pak zařízení pošle hostu nějakou informaci, nebo naopak změní své nastavení podle požadavku hosta. Standardní requesty se dělí do třech skupin:

requesty pro zařízení (Device Requests), requesty pro rozhraní (Interface Requests) a requesty pro koncový bod (Endpoint Requests), přičemž tyto requesty musí implementovat každé USB zařízení nehledě na typ. Kromě toho pak může zařízení implementovat další requesty v závislosti na třídě zařízení. [1]

Seznam standardních requestů a jejich popis je zachycen v tabulce 1, tabulce 2 a tabulce 3 v příloze B.

2.6 Enumerace

Jedná se o proces, který host vykoná poté, co je detekováno připojení nového zařízení do systému, přičemž tento proces probíhá v několika krocích.

Host zašle request Get Descriptor koncovému bodu 0 na adrese 00h. Jelikož každé zařízení musí odpovědět na tento request, je zajištěno, že nové zařízení pošle jako odpověď svůj Device deskriptor. Z tohoto descriptoru získá host informaci o maximální velikosti paketu a poté přidělí zařízení konkrétní unikátní adresu pomocí zaslání requestu Set Adress. Od toho okamžiku probíhá veškerá komunikace se zařízením pomocí této přidělené adresy až do odpojení zařízení od hubu. [7]

V dalším kroku pošle host znovu request Get Descriptor, přičemž tentokrát jsou získány všechny základní informace o zařízení, hlavně pak počet možných konfigurací. [7]

Host poté pomocí dalších requestů Get Descriptor získá informace o všech dostupných konfiguracích, které zařízení umožňuje. Pokud je hostem operační systém Windows, probíhá získání konfigurace ve dvou krocích, kdy nejprve je provedena žádost o prvních 9 bytů Configuration deskriptoru, v nichž je uložena informace o celkové délce tohoto deskriptoru, ale také všech dalších deskriptorů spadajících pod tuto konfiguraci, tedy všech příslušných Interface a Endpoint deskriptorů. Poté je již zaslán request o kompletní sadu deskriptorů konkrétní konfigurace. Tato sekvence kroků je důležitá hlavně proto, že některé konfigurace mohou obsahovat další deskriptory specifické pro některé konkrétní třídy. [7]

Dle informací získaných v předešlých krocích host přidělí zařízení dostupný ovladač. K přidělení ovladače systém Windows prohledá dostupné INF soubory, přičemž hledá shodu s parametry Vendor ID a Product ID obsaženými v Device deskriptoru. [7]

Po přidělení a nahrání ovladače je vybrána nejlepší dostupná konfigurace a host zašle request Set Configuration obsahující číslo zvolené konfigurace. Poté co zařízení implementuje vybranou konfiguraci, je připravené k transakcím. [7]

3 USB video device class

3.1 Charakteristika třídy

USB video device class (též USB video class, zkr. UVC) je třída pro zařízení, která přenášejí obrazovou informaci po sběrnici USB, tedy například webkamery nebo televizní tunery. Pokud je zařízení implementováno dle tohoto standardu, je pro něj v systému Windows přímo dostupný ovladač Usbvideo.sys a vývojáři tedy odpadá nutnost vytvářet ovladač vlastní. V případě potřeby je ale možné provést i rozšíření tohoto dostupného ovladače o funkce specifické pro dané zařízení. [2] [3]

Poslední vydanou verzí specifikace třídy UVC je verze 1.5. Tato práce se ale zabývá verzí předchozí, to znamená verzí 1.1, neboť verze 1.5 byla vydána krátce před dokončením této práce. Verze 1.1 byla vydána 1. června 2005, podporuje přenos komprimovaného videa ve formátech MPEG-2 TS, H.264, MPEG-4 SL, SMPTE VC1 a MJPEG. Dále pak nekomprimovaného videa zakódovaného ve formátu YUV, respektive odvozených formátech YUY12 a NV12. Ovladač Usbvideo.sys plně podporující verzi 1.1 je dostupný v operačním systému Windows 7 a novějším. Pro verzi 1.5 je to pak operační systém Windows 8. Operační systém Windows Vista poskytuje ovladač pro verzi 1.0 a stejně tak Windows XP od service packu 2. [2] [3]

Pro třídu UVC je dostupný ovladač i pro operační systém Linux. Ve verzích Linux 2.6.26 a novějších je ovladač dostupný nativně. [13]

Každé zařízení, které má splňovat specifikaci UVC, musí obsahovat jedno rozhraní VideoControl (VC) sloužící pro ovládání zařízení a může obsahovat více VideoStreaming (VS) rozhraní. VS rozhraní slouží pro transport datové toku z nebo do zařízení. Kolekci VC rozhraní a všech VS rozhraní jednoho zařízení je označováno jako Video Interface Collection (VIC). K popisu VIC pak slouží Interface Association deskriptor. Tento deskriptor a deskriptory související s rozhraními VC a VS jsou důležité zejména v tom, že na úrovni právě těchto deskriptorů se deklaruje třída UVC. Jednotlivé deskriptory rozhraní jsou pak rozlišeny pomocí takzvané podtřídy. [2]

Každé zařízení představuje funkci, která nějakým způsobem pracuje s proudem obrazových dat. Aby bylo možné manipulovat s fyzickým nastavením této funkce, je třeba ji rozdělit do adresovatelných bloků. K tomuto definuje UVC dvě základní skupiny bloků a to unity a terminály. Terminály představují vstupní nebo výstupní body video funkce tedy body, skrze které proudí obrazová data do nebo z funkce. Unity pak představují základní stavební bloky obsahující jednoduché funkce. Každá unita má vždy jeden výstupní pin a jeden nebo více vstupních pinů. Výsledná funkce potom vzniká postupným propojováním vstupních a výstupních pinů jednotlivých unit, případně terminálů. Při spojování pinů platí, že jeden výstupní pin může být připojen na více vstupních, ale na jeden vstupní pin může být vždy připojen pouze jeden pin výstupní. Smyčky uvnitř funkce jsou zakázány. Aby bylo možné piny propojovat, má každý pin každé unity přiděleno svoje číslo, přičemž se piny číslují vždy od nuly. To samé pak platí pro terminály. Je možné takto vytvořit funkci, která bude pracovat s čistě digitálními informacemi, ale také je možné vytvořit zařízení pracující s informacemi analogovými a dokonce je možné popsat funkci pracující s hybridní informací. Pro popsání celé struktury jsou definovány takzvané Unit deskriptory a Terminal deskriptory. Unit deskriptory popisují vždy typ dané unity a tím i její funkci, dále také způsob, jakým jsou propojeny jednotlivé piny unity. To samé platí i pro deskriptory jednotlivých terminálů. [2]

Kromě standardních requestů daných protokolem USB, které musí každé zařízení implementovat, definuje třída UVC další dvě skupiny requestů, které může zařízení implementovat. Jedná se o skupinu VideoControl requestů, sloužících pro ovládání vlastností jednotlivých unit a terminálů a skupinu VideoStreaming requestů, které slouží pro ovládání parametrů vlastního přenosu obrazové informace. Některé tyto requesty je povinné implementovat, zatímco některé jsou pouze volitelné. [2]

3.2 Struktura UVC zařízení

3.2.1 Video Interface Collection

Jak již bylo zmíněno v předchozí kapitole, každé zařízení třídy UVC musí obsahovat jedno rozhraní VideoControl (VC) a může obsahovat více VideoStreaming rozhraní (VS). Společně tato rozhraní tvoří tzv. Video Interface Collection (VIC), která je popsána pomocí

Interface Association deskriptoru. Tento deskriptor musí být umístěn nejvýše v celé hierarchii UVC deskriptorů a musí být vrácen jako součást konfigurace zařízení v odpovědi na request GetDescriptor s požadavkem na konfiguraci zařízení. [2]

Struktura tohoto deskriptoru je zachycena v tabulce 3.1.

Offset	Označení pole	Velikost v bajtech	Typ hodnoty	Popis pole
0	bLength	1	Číslo	Velikost deskriptoru v bajtech tj. 8
1	bDescriptorType	1	Konstanta	Konstanta označující typ deskriptoru - Interface association descriptor
2	bFirstInterface	1	Číslo	Číslo prvního VC deskriptoru této funkce
3	bInterfaceCount	1	Číslo	Počet souvisejících VS rozhraní této funkce
4	bFunctionClass	1	Třída	Kód označující třidu UVC
5	bFunctionSubClass	1	Podtřída	Kód označují VIC
6	bFunctionProtocol	1	Protokol	Kód označující protokol, v tomto případě není pole využito a musí být nastaveno na hodnotu nedefinováno
7	iFunction	1	Index	Číslo souvisejícího string deskriptoru

Tab. 3.1 Struktura Interface Association deskriptoru [2]

3.2.2 VideoControl interface

K ovládání nastavení video funkce, respektive nastavení unit a terminálů, je třeba rozhraní VC. Toto rozhraní v sobě musí obsahovat koncový bod s označením 0, který bude nastaven pro řídicí přenosy. Tímto koncovým bodem pak prochází veškerá data sloužící pro manipulaci s nastavením funkce. Dále v sobě toto rozhraní může obsahovat koncový bod nastavený pro přenos při přerušení. Toto rozhraní zaštiťuje celou vnitřní strukturu funkce složenou z unit a terminálů. [2]

K popsání tohoto rozhraní je třeba využít hned dvou deskriptorů. Jsou to standardní deskriptor VC rozhraní (Standard VC Interface Descriptor), který popisuje vlastnosti samotného rozhraní. Tento deskriptor je svojí strukturou totožný se standardním rozhraním

obecného zařízení pouze s tím rozdílem, že některá pole musí obsahovat specifickou informaci. Dále pak deskriptor VC rozhraní specifický vzhledem ke třídě (Class-Specific VC Interface Header Descriptor). Ten již popisuje vnitřní strukturu video funkce, tedy všechny unity a terminály a jejich možnosti. Z tohoto vyplývá, že délka deskriptoru je závislá na počtu unit a terminálů, které jsou ve funkci obsaženy a musí být nastavena tak, aby její délka odpovídala délce deskriptoru rozhraní a délkám všech deskriptorů podřízených unit a terminálů. [2]

Struktury těchto dvou deskriptorů jsou zachyceny v tabulkách 3.2 a 3.3.

Offset	Označení pole	Velikost v bajtech	Typ hodnoty	Popis pole
0	bLength	1	Číslo	Velikost deskriptoru v bajtech tj. 9
1	bDescriptorType	1	Konstanta	Konstanta označující typ deskriptoru - Interface descriptor
2	bInterfaceNumber	1	Číslo	Číslo identifikující toto rozhraní
3	bAlternateSetting	1	Číslo	Číslo identifikující alternativní nastavení tohoto rozhraní
4	bNumEndpoints	1	Číslo	Počet koncových bodů mimo bodu 0
5	bInterfaceClass	1	Třída	Kód označující třidu UVC
6	bInterfaceSubClass	1	Podtřída	Kód označující VC rozhraní
7	bInterfaceProtocol	1	Protokol	Kód označující protokol, v tomto případě není pole využito a musí být nastaveno na hodnotu nedefinováno
8	iInterface	1	Index	Číslo souvisejícího string deskriptoru

Tab. 3.2 Struktura Standard VideoControl Interface deskriptoru [2]

Offset	Označení pole	Velikost v bajtech	Typ hodnoty	Popis pole
0	bLength	1	Číslo	Velikost tohoto deskriptoru v bajtech tj. 12 + n
1	bDescriptorType	1	Konstanta	Konstanta označující typ deskriptoru - Interface
2	bDescriptorSubType	1	Konstanta	Konstanta označující podtyp deskriptoru - Header
3	bcdUVC	2	BCD	Specifikace verze UVC
5	wTotalLength	2	Číslo	Celková velikost tohoto deskriptoru a všech deskriptorů souvisejících unit a terminálů

7	dwClockFrequency	4	Číslo	Slouží k nastavení hodin zařízení v jednotkách Hz
11	bInCollection	1	Číslo	Počet VC rozhraní v této VIC, ke kterým patří toto rozhraní
12	baInterfaceNr(1)	1	Číslo	Číslo prvního VS rozhraní v této kolekci
…	…	…	…	…
12 + (n-1)	baInterfaceNr(n)	1	Číslo	Číslo posledního VS rozhraní v této kolekci

Tab. 3.3 Struktura Class-Specific VC Interface Header deskriptoru [2]

Jak již bylo zmíněno v předchozím textu, VC rozhraní a konkrétně Class-Specific VC rozhraní představuje hlavičku pro vnitřní uspořádání funkce, tedy strukturu složenou z terminálů a unit. Tyto unity a terminály jsou pospojovány pomocí svých vstupních a výstupních pinů a tím tvoří výslednou video funkci. K jejich popisu pak slouží deskriptory, přičemž v hierarchii deskriptorů následují deskriptory unit a terminálů přímo za Class-Specific VC Interface Header deskriptorem, přičemž nezáleží na pořadí, v jakém jsou deskriptory jednotlivých unit a terminálů deklarovány. [2]

3.2.2.1 Terminály

UVC verze 1.1 specifikuje dva základní druhy terminálů, vstupní (Input) terminál (IT) a výstupní (Output) terminál (OT). Dále pak specifikace definuje dva druhy speciálních teminálů, a sice Camera a Media Transport. Tyto speciální druhy se pak liší přidanou funkcionalitu a také upravenou verzí deskriptoru, který je rozšířen o některá pole. [2]

Vstupní terminál slouží jako vstup obrazové informace do vnitřku video funkce. Terminál má jeden výstupní pin. K popisu terminálu slouží Input Terminal deskriptor, jehož standardní délka je 8 bajtů. Je ale možné mít vstupní terminál rozšířen o některé další funkce. V takovém případě pak mívá deskriptor další pole a tedy i jeho délka může být větší. Přesná struktura deskriptoru běžného vstupního terminálu je popsána ve specifikaci *USB Device Class Definition for Video Devices, Revision 1.1* v kapitole *3.7.2.1 Input Terminal Descriptor*. [2]

Výstupní terminál slouží jako výstup obrazové informace z vnitřku video funkce. Terminál má jeden vstupní pin. K popisu terminálu slouží Output Terminal deskriptor, jehož standardní délka je 9 bajtů. Je ale možné mít výstupní terminál rozšířen o některé další funkce. V takovém případě pak mívá deskriptor další pole a tedy i jeho délka může být větší. Přesná struktura deskriptoru běžného výstupního terminálu je popsána ve specifikaci *USB Device Class Definition for Video Devices, Revision 1.1* v kapitole *3.7.2.2 Output Terminal Descriptor.* [2]

3.2.2.2 Unity

UVC verze 1.1 specifikuje tři základní druhy unit, jsou to Selector Unit, Processing Unit a Extension Unit. [2]

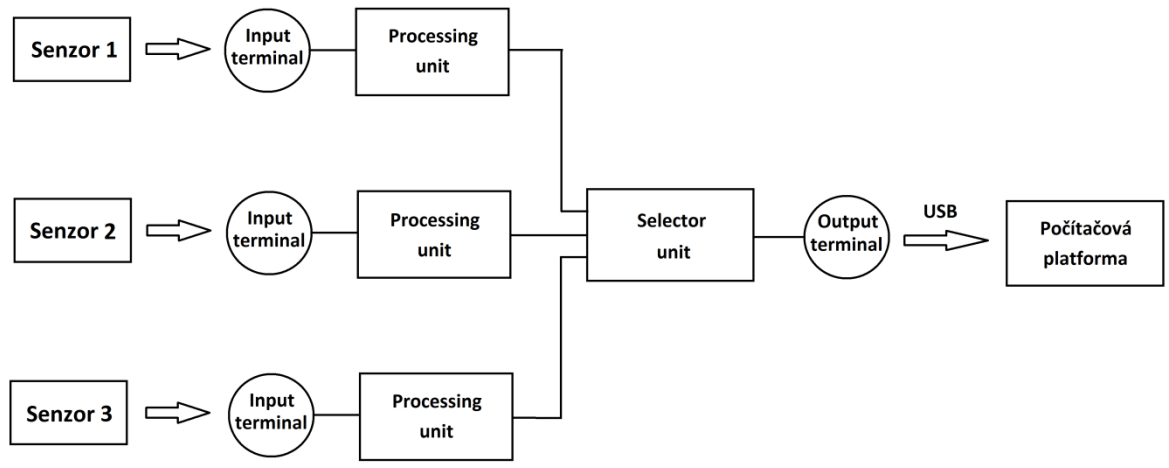
Selector Unit představuje funkci, která vybírá z několika vstupních datových proudů, které jsou připojeny na vstupní piny unity a jeden z nich přenáší na svůj výstupní pin. Z toho tedy vyplívá, že tato unita má jeden výstupní pin a jeden nebo více pinů vstupních. Délka deskriptoru této unity je proměnlivá a závisí na počtu vstupních pinů. Pro jeden vstupní pin je tato délka 6 bajtů, za každý vstupní pin se pak délka prodlužuje o jeden bajt. Přesná struktura tohoto deskriptoru je popsána ve specifikaci *USB Device Class Definition for Video Devices, Revision 1.1* v kapitole *3.7.2.4 Selector Unit Descriptor.* [2]

Processing Unit je funkcí, která ovládá parametry snímků, které skrze ni prochází jako proud dat. Unita má tedy pouze jeden vstupní a pouze jeden výstupní pin. Pomocí unity je možné ovládat jas, kontrast, saturaci barev, odstín snímku, ostrost, gammu, digitální zoom, je možné nastavovat vyvážení bílé barvy, ale také například zesílení signálu. Délka deskriptoru této unity závisí na počtu parametrů, které chceme pomocí unity kontrolovat. Základní délka je 10 bajtů a za každou vlastnost, kterou chceme být schopni kontrolovat, se zvětšuje o 1 bajt. Přesná struktura tohoto deskriptoru je popsána ve specifikaci *USB Device Class Definition for Video Devices, Revision 1.1* v kapitole *3.7.2.5 Processing Unit Descriptor.* [2]

Extension Unit umožňuje doplnit funkci o libovolnou uživatelsky definovanou funkcionalitu. Tato unita může mít jeden výstupní a jeden či více vstupních pinů. Ovladač dostupný v operačním systému sice nedokáže rozeznat, co funkce obsažená v unitě vykonává,

ale umožní některé metody pro komunikaci s uživatelským softwarem. Délka deskriptoru je velice proměnlivá, neboť záleží na počtu výstupů a na množství funkcí, které unita vykonává. Přesná struktura tohoto deskriptoru je popsána ve specifikaci *USB Device Class Definition for Video Devices, Revision 1.1* v kapitole *3.7.2.6 Extension Unit Descriptor.* [2]

Na obrázku 3.1 je zachycen příklad možné funkce, realizovatelné pomocí unit a terminálů. Schéma popisuje funkci, která má tři různé dostupné zdroje obrazových dat ze třech různých senzorů. Funkce pak umožňuje nastavovat obrazové vlastnosti každého proudu obrazových dat zvlášť pro každý proud pomocí jednotlivých Processing unit. Ze třech datových proudů pak může host vybírat požadovaný proud skrze Selector unit. Vybraný datový proud je pak předáván ven z funkce Output terminálem.



Obr. 3.1 Příklad funkce realizovatelné pomocí třídy UVC

3.2.3 VideoStreaming interface

VS rozhraní představuje bod, skrze který probíhá samotný přenos obrazových dat z nebo do funkce. Toto rozhraní není přímo povinné a je možné mít i více rozhraní, každé pracující s daty různé povahy a formátu. Každé rozhraní může obsahovat koncový bod nastavený pro izochronní nebo hromadný přenos. Pokud rozhraní obsahuje koncový bod pro izochronní přenos, je nutné, aby toto rozhraní implementovalo několik alternativních

nastavení, kdy jedno z těchto alternativních nastavení musí být nastaveno pro nulovou šířku pásma. Tento způsob pak umožní, že host může dočasně přerušit spojení zvolením tohoto alternativního nastavení. Toto alternativní nastavení pak neobsahuje deskriptor izochronního koncového bodu. Pokud rozhraní obsahuje koncový bod pro hromadný přenos, nesmí již obsahovat alternativní nastavení. [2]

VS rozhraní je stejně jako VC rozhraní popsáno pomocí dvou deskriptorů a to standardního VS interface deskriptoru (Standard VS Interface Descriptor) a specifického VS deskriptoru třídy, přičemž specifický VS deskriptor je složen z deskriptorů Class-specific VS Interface Input Header Descriptor, Class-specific VS Interface Output Header Descriptor, Payload Format Descriptor a Video Frame Descriptor. [2]

Deskriptor pro standardní VS rozhraní je standardní deskriptor rozhraní, tak jak je definován dle specifikace USB verze 2.0, a jeho struktura je tedy identická se strukturou standardního deskriptoru VC rozhraní, zachycenou v tabulce 3.2, pouze s tím rozdílem, že se liší obsah pole bInterfaceSubClass. Hodnota tohoto pole musí být nastavena tak, aby odpovídala kódu VS rozhraní. [2]

Input Header descriptor se používá u takových VS rozhraní, která obsahují koncový bod pro vstup proudu obrazových informací do funkce. Představuje hlavičku pro následující deskriptory popisující formát obrazových dat. Délka tohoto deskriptoru je proměnlivá. Přesná struktura je popsána ve specifikaci *USB Device Class Definition for Video Devices, Revision 1.1* v kapitole *3.9.2.1 Input Header Descriptor.* [2]

Output Header descriptor se používá u takových VS rozhraní, která obsahují koncový bod pro výstup proudu obrazových informací z funkce. Představuje hlavičku pro následující deskriptory popisující formát obrazových dat. Délka tohoto deskriptoru je proměnlivá. Přesná struktura je popsána ve specifikaci *USB Device Class Definition for Video Devices, Revision 1.1* v kapitole *3.9.2.2 Output Header Descriptor.* [2]

Payload Format descriptor popisuje formát obrazových dat. Jsou podporovány formáty MJPEG Video, MPEG1-SS, MPEG2-PS, MPEG-2 TS, H.264, SMTPE VC1, MPEG-4 SL, DV a nekomprimované formáty videa. Samotný deskriptor je pak specifický vzhledem k vybranému formátu. V případě nekomprimovaných formátů nabízí UVC verze 1.1 dva

formáty různé formáty standardu YUV, a to standard YUY2 a NV12. UVC verze 1.5 pak nabídku rozšiřuje od formát M420 a I420. Přesná struktura deskriptoru pro nekomprimované formáty je popsána v tabulce 5 v příloze A. Strukturu deskriptorů pro ostatní formáty je pak možno nalézt v jednotlivých souborech specifikace *USB Device Class Definition for Video Devices, Revision 1.1.* [2]

Frame descriptor obsahuje informace o dekódovaném obrazu. Proto je deskriptor opět specifický vzhledem k vybranému formátu. Přesná struktura deskriptoru pro nekomprimované formáty je popsána v tabulce 6 v příloze A. Strukturu deskriptorů pro ostatní formáty je pak možno nalézt v jednotlivých souborech specifikace *USB Device Class Definition for Video Devices, Revision 1.1.* [2]

Posledním deskriptorem nutným pro popsání struktury video funkce, je samotný koncový bod, skrze který proudí data do nebo z funkce. Jsou podporovány dva typy koncových bodů a to koncový bod pro izochronní přenos a koncový bod pro hromadný přenos. Přesná struktura deskriptoru koncového bodu je zobrazena v tabulce 3.4. Zásadní rozdíl je v poli bmAttributes, jehož struktura se liší dle typu přenosu. Pro hromadný přenos jsou důležité pouze poslední dva bity (D1..0), které musí být nastavené na hodnotu „10“ = Hromadný přenos. Ostatní bity jsou rezervované. Pro izochronní přenos pak vypadá struktura následovně: bity D3..2 obsahují typ synchronizace, nastavené na hodnotu „01“ = Asynchronní. Bity D1..0 nastavené na hodnotu „01“ = Izochronní přenos. Ostatní bity jsou opět rezervované.

Offset	Označení pole	Velikost v bajtech	Typ hodnoty	Popis pole
0	bLength	1	Číslo	Velikost deskriptoru v bajtech tj. 7
1	bDescriptorType	1	Konstanta	Konstanta označující typ deskriptoru - Endpoint
2	bEndpointAddress	1	Adresa	Adresa koncového bodu. Struktura: D7: Směr 0 = výstupní koncový bod 1 = vstupní koncový bod D6..4: Rezervováno, musí být 0 D3..0: Číslo koncového bodu
3	bmAttributes	1	Bitmapa	Viz výše
4	wMaxPacketSize	2	Číslo	Maximální velikost paketu, který je

				schopný koncový bod přijmout nebo odeslat
6	bInterval	1	Číslo	Interval přenosu dat, pro izochronní přenos musí být toto číslo v intervalu 1 až 16. Pro hromadný přenos pak v rozsahu 0 až 255

Tab. 3.4 Struktura Standard VS Video Data Endpoint deskriptoru [2]

3.3 Specifické requesty třídy UVC

Třída definuje velké množství requestů, přičemž některé jsou povinné, některé jsou volitelné. Většina requestů ale slouží ke zjišťování stavu nebo k nastavení některého ovládacího prvku. Tyto requesty by bylo možné rozdělit do dvou skupin a to na skupinu vztahující se k sekci VC a skupinu requestů vztahujících se k sekci VS. Jak už bylo zmíněno výše, VC sekce zařízení je složena z terminálů a unit. Requesty vztahující se k této sekci jsou vždy requesty směřující na jedno konkrétní VC rozhraní a nesou informaci pro identifikování konkrétní unity nebo terminálu, na který je request zaměřen. Requesty zaměřené na VS sekci zařízení jsou pak vždy směřovány přímo na samotné VS rozhraní. Jsou definovány dva základní typy requestu a to Set request a Get request. [2]

Set request slouží k nastavování atributů libovolné entity uvnitř video funkce. Struktura requestu odpovídá struktuře standardního requestu, přičemž obsah jednotlivých polí je následující: [2]

bmRequestType: možné hodnoty 00100001 nebo 00100010. Bit D7 (0) definuje, že se jedná o request nastavující parametry zařízení. Bity D6..5 (01) specifikují, že se jedná o request specifický pro třídu. Zbývající bity, specifikují zda je request mířen na VS nebo VC rozhraní (hodnota 00001) nebo na koncový bod VS rozhraní (hodnota 00010).

bRequest: obsahuje konstantu identifikující cílový atribut. Toto pole má jedinou přípustnou hodnotu: Current setting attribute (SET_CUR).

wValue: obsah a struktura tohoto pole se liší v závislosti na adresované entitě.

wIndex: dolní bajt tohoto pole obsahuje číslo zařízení nebo koncového bodu, který je adresován, horní bajt obsahuje ID entity, případně pokud má request adresovat přímo rozhraní, pak je horní bajt nulový.

wLength: hodnota je různá podle toho, jakou entitu request adresuje. [2]

Get request slouží pro získání informace o atributu entity. Struktura opět odpovídá struktuře standardního requestu, přičemž obsah jednotlivých polí je následující:

bmRequestType: možné hodnoty 10100001 nebo 10100010. Bit D7 (0) definuje, že se jedná o request získávající informace o zařízení. Bity D6..5 (01) specifikují, že se jedná o request specifický pro třídu. Zbývající bity specifikují, zda je request mířen na VS nebo VC rozhraní (hodnota 00001) nebo na koncový bod VS rozhraní (hodnota 00010).

bRequest: obsahuje konstantu identifikující cílový atribut. Přípustné hodnoty:

Current setting attribute (GET_CUR),
Minimum setting attribute (GET_MIN),
Maximum setting attribute (GET_MAX),
Default setting attribute (GET_DEF),
Resolution attribute (GET_RES),
Data length attribute (GET_LEN),
Information attribute (GET_INFO).

wValue: obsah a struktura tohoto pole se liší v závislosti na adresované entitě.

wIndex: dolní bajt tohoto pole obsahuje číslo zařízení nebo koncového bodu, který je adresován, horní bajt obsahuje ID entity, případně pokud má request adresovat přímo rozhraní, pak je horní bajt nulový.

wLength: hodnota je různá podle toho, jakou entitu request adresuje. [2]

Seznam dostupných requestů je příliš obsáhlý, aby bylo možné ho v této práci dokonale zmapovat a popsat. Přesnou strukturu pro jednotlivé entity a další informace je tedy nutné získat přímo ve specifikaci *USB Device Class Definition for Video Devices, Revision 1.1* v kapitolách 4.2 a 4.3. [2]

3.4 Video Payload Header

Každý přenos obrazové informace musí obsahovat hlavičku Video Payload Header. Hlavička slouží k oddělení jednotlivých obrazových snímků, ale obsahuje také další

informace. Délka této hlavičky se liší v závislosti na formátu obrazové informace a může se pohybovat v rozsahu 2, 6 nebo 12 bajtů. [2]

Struktura hlavičky Video Payload Header pro nekomprimované formáty je zachycena v tabulce 3.5.

Offset	Označení pole	Velikost v bajtech	Typ hodnoty	Popis pole
0	bHeaderLength	1	Číslo	Délka celé hlavičky v bajtech tj. 12
1	bmHeaderInfo	1	Bitmapa	D7: Konec hlavičky – nastaveno na hodnotu 1 D6: Označuje chybu při přenosu D5: 0 pokud snímek sloužil pro zachycení statického obrazu D4: Rezervováno D3: Označuje přítomnost pole scrSourceClock v hlavičce, v tomto případě tedy bude obsahovat hodnotu 1 D2: Označuje přítomnost pole dwPresentationTime v hlavičce, v tomto případě tedy bude obsahovat hodnotu 1 D1: Bit sloužící pro označení konce snímku D0: Bit sloužící pro označení počátku snímku
2	dwPresentationTime	4	Číslo	Toto pole obsahuje časovou známku snímku
6	scrSourceClock	6	Číslo	Bity D47..43 jsou rezervovány a musí být nastaveny na 0 Bity D31..0 obsahují časovou informaci vzorkovanou ze systémového času zdroje. Jelikož tento čas musí být korelován s časem USB sběrnice, obsahují bity D42..32 1KHz SOF čítač

Tab. 3.4 Struktura hlavičky Video Payload Header pro nekomprimované formáty [2]

Praktická část

4 Aplikace pro získání obrazových dat a jejich přenos do struktury operačního systému Windows

4.1 Charakteristika aplikace

Cílem práce bylo vytvoření aplikace, která by sloužila pro získání obrazových dat a předání těchto dat do struktury operačního systému Windows. Obsah práce byl rozdělen do dvou částí. První část, která z obrazového senzoru získává obrazová data a z těchto dat vytváří obrazový snímek, byla realizována v FPGA obvodu EP3C120F484C7 obsahujícím čip Cyclone III od společnosti Altera. Druhá část, která vytvořené snímky odesílá prostřednictvím sběrnice USB počítačové platformě, je pak realizována na vývojovém kitu CYUSB3KIT-001 od společnosti Cypress obsahující kontrolér EZ-USB FX3. Tato aplikace je implementována dle třídy UVC, čímž je zaručeno, že pro zařízení bude v operačním software Windows dostupný ovladač, který již zajistí předání dat do struktury operačního systému a bude možno k nim přistupovat skrze rozhraní Media Foundation a DirectShow. Rozdělení aplikace na dvě části a následná realizace na dvou oddělených obvodech byla provedena z důvodu snadnější realizace výsledné aplikace a rozdělení zátěže mezi dva specializované obvody. Kontrolér EZ-USB FX3 nabízí rozhraní pro snadnou implementaci zařízení dle třídy UVC, FPGA obvod potom nabízí rychlou práci s daty a možnost paralelního zpracování díky realizaci funkcí pomocí vnitřní logiky zařízení.

Pro vytvoření zkušební aplikace byl zvolen pyroelektrický senzor 128LTI a A/D převodník ADS8363EVM, který získanou obrazovou informaci převádí do digitální podoby.

V této kapitole je popsána aplikace, která ovládá optický senzor a přidružený A/D převodník a získanou obrazovou informaci dále zpracovává a předává aplikaci pro transport. Aplikace pro transport dat po sběrnici USB je popsána v kapitole 5.

Pro transport obrazových dat byl zvolen formát Uncompressed, tedy nekomprimovaná data. Třída UVC verze 1.1 nabízí pro tyto účely formát YUV, respektive odvozené formáty YUY12 a NV12. Z těchto dvou nabízených formátů byl zvolen formát YUY12. Jednomu

pixelu zakódovanému v tomto formátu pak odpovídá 16 bitů. Jeden celý řádek senzoru tedy představuje informaci o délce 256 bajtů. Při práci s daty získanými z pyroelektrického senzoru je třeba znát hodnoty naměřené při osvětlení senzoru a hodnoty naměřené při zatmění senzoru. Čidlo dále poskytuje informaci o teplotě snímače a též byl dán požadavek na možnost určení čísla snímku. Z těchto důvodů byl specifikován formát snímku, který vypadá následovně. Rozměry jednoho snímku jsou 128 * 4 pixely, kdy každý pixel je popsán 16 bity. První řádek obsahuje informaci o hodnotách získaných ze senzoru při osvětlení, druhý řádek obsahuje informaci o teplotě čidla při osvětlení. Tato informace je šestnáctibitová. Třetí řádek obsahuje informaci o hodnotách získaných ze senzoru při zatmění a poslední řádek obsahuje číslo každého snímku.

4.2 Hardware použitý pro získání obrazové informace

Pro vytvoření ukázkové aplikace produkující obrazová data, která by následně byla odesílána skrze realizovanou aplikaci pro transport obrazových dat po sběrnici USB, byl zvolen pyroelektrický řádkový senzor 128LTx se 128 elementy. Pro převedení signálu do digitální podoby byl zvolen A/D převodník ADS8363EVM, který umožňuje převod obrazové informace, ale také umožňuje pracovat se signálem z teplotního čidla, které je zabudováno ve zvoleném pyroelektrickém senzoru.

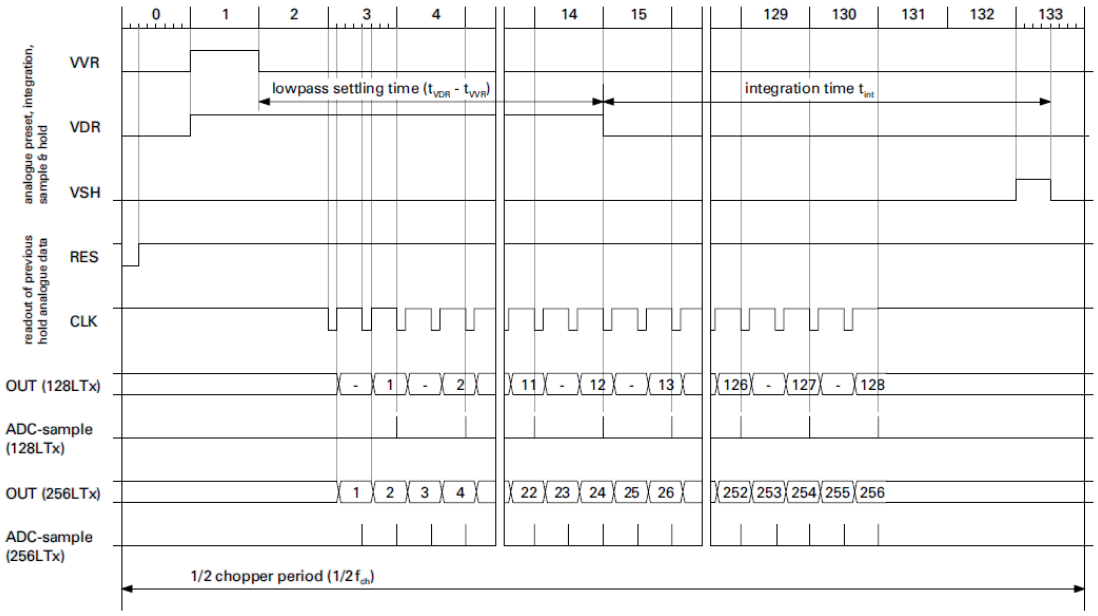
4.2.1 Pyroelektrický řádkový senzor 128LTI

128LTI je řádkový senzor obsahující 128 snímacích bodů a integrovaný CMOS multiplexor. Senzor v sobě dále obsahuje čidlo pro snímání teploty AD 590, které měří teplotu senzoru. Výstupem z čidla je proud proporčně odpovídající teplotě. Senzor vyžaduje napájecí napětí (VDD) v rozsahu -0.3 až 7V a napájení operačního zesilovače (VD2), které má hodnotu rovnou polovině napětí VDD. [8]

Činnost senzoru je řízena pěti digitálními vstupy CLK, VSH, VVR, VDR a RES. Maximální přípustné napětí pro tyto vstupy je pak odvozeno od napájecího napětí VDD a je

rovno právě hodnotě napájecího napětí + 0.3V. Logická nula pak odpovídá hodnotě 0 až 0.3*VDD, logická jedna hodnotě 0.7*VDD až VDD. [8]

Pro správnou činnost senzoru je třeba senzor střídavě zastiňovat, k čemuž se používá takzvaný chopper. Chopper tedy představuje vstupní signál, jehož hodnota odpovídá tomu, zda je chopper ve fázi, kdy je senzor osvětlen, nebo ve fázi, kdy je senzor zastíněn. Frekvence chopperu se musí pohybovat v rozsahu 10-512Hz. Od této frekvence se pak odvozují frekvence ostatních vstupních signálů. Přesný časový diagram je zachycen na obrázku 4.1. [8]



Obr. 4.1 Časový diagram řídicích signálů optického senzoru [8]

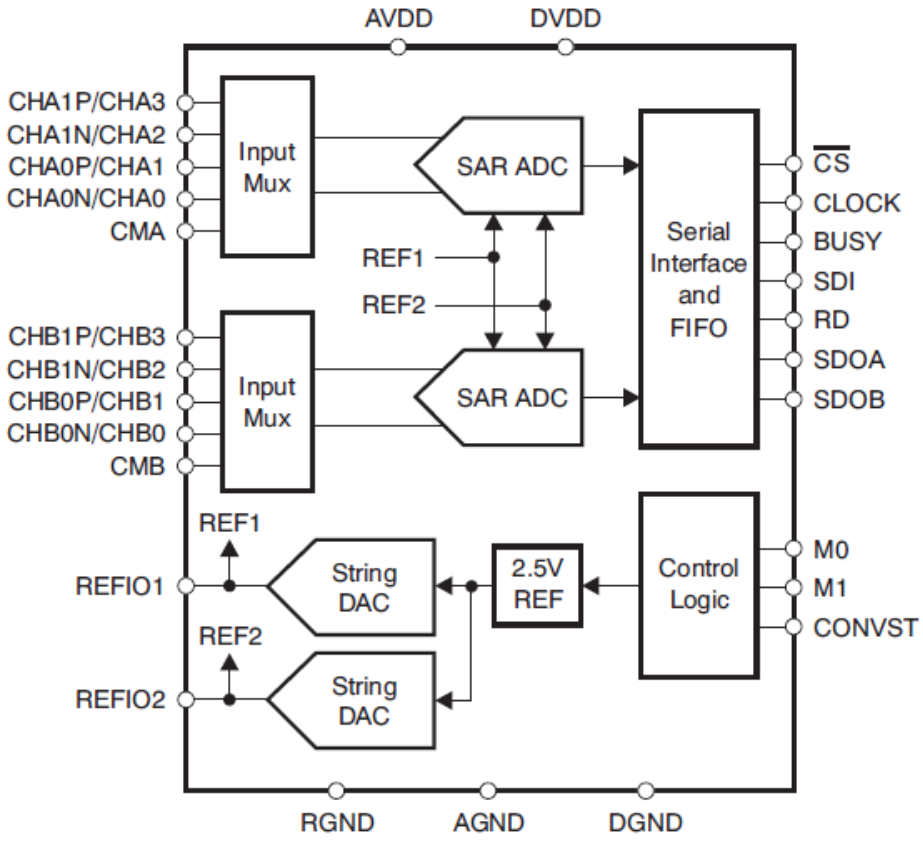
4.2.2 A/D převodník ADS8363EVM

Převodník ADS8363EVM je dvoukanálový šestnáctibitový A/D převodník nabízející osm pseudo diferenčních nebo čtyři plně diferenční vstupy seskupené do dvou skupin umožňujících simultánní převod. [9]

Převodník je třeba napájet napájecím napětím (AVDD) v rozsahu 2.7V až 5.5V vzhledem k zemi a napětím (DVDD) v rozsahu 2.3V až 5.5V. Převodník pak nabízí dvě různá programovatelná referenční napětí. Tato napětí jsou nastavena pomocí dvou deseti bitových

hodnot uložených ve dvou šestnáctibitových vnitřních registrech, které umožňují nastavovat referenční napětí v krocích po 2.44mV. Referenční napětí je pak dostupné na pinech označovaných jako REFIO1 a REFIO2. [9]

Převodník nabízí dva módy činnosti, takzvaný Half-Clock a Full-Clock mód. Pro řízení převodníku slouží šest digitálních vstupů M1, M0, SDI, CONVST, RD, $\overline{CS}$ a hodinový signál (CLOCK), jehož frekvence se musí pohybovat v rozmezí 1MHz až 40MHz pro Full-Clock mód a 0.5MHz až 20MHz pro Half-Clock mód. Kromě vstupních signálů obsahuje převodník také šestnácti bitový registr (CONFIG). Převedená hodnota je sériově přenášena na výstupní piny označené SDOA a SDOB. Blokové schéma převodníku je zachyceno na obrázku 4.2. [9]



Obr. 4.2. Blokové schéma převodníku ADS8363EVM [9]

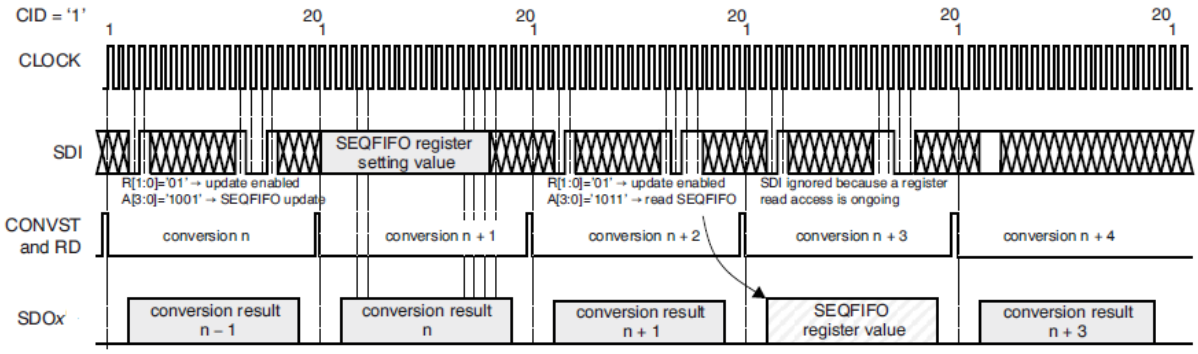
Jak již bylo zmíněno v předchozím textu, převodník obsahuje několik vnitřních registrů, které jsou nutné pro správnou činnost zařízení. Jsou to registry označené jako REFDAC1 a REFDAC2 sloužící pro nastavení referenčního napětí na pinech REFIO1 a REFIO2, a registr CONFIG sloužící pro nastavení činnosti zařízení. [9]

Hodnoty referenčních napětí byly v práci nastaveny na hodnotu 0111111111 odpovídající napětí 1,25V. [9]

Registr CONFIG je složen z těchto položek:

Bit 15..14 – C1..0 – bity ovládající vstupní multiplexory.
Bit 13..12 – R1..0 – ovládání přístupu k zápisu do registru CONFIG. Pro povolení přepisu celého registru je hodnota nastavena na 01, jinak hodnota 00 nebo 11.
Bit 11..10 – PD1..0 – ovládání power-down módů zařízení. Pro účely práce nehraje roli a tedy nastaveno na hodnotu 00.
Bit 9 – FE – ovládání FIFO registru. Nastaveno na hodnotu 0 pro vypnutí.
Bit 8 – SR – mód Special read mode. Nastaveno na hodnotu 0 pro vypnutí.
Bit 7 – FC – ovládání módu Full clock. Mód zakázán, pokud je bit nastaven na hodnotu 0.
Bit 6 – PDE – povolení pseudo diferenčního módu. Mód povolen, pokud je bit nastaven na hodnotu 1.
Bit 5 – CID – povolení informací o kanálu předcházejících každému výsledku konverze. Zakázáno bitem nastaveným na 1.
Bit 4 – CE – povolení vnitřního dvoubitového čítače. Nastaveno na hodnotu 0 pro vypnutí.
Bit 3..0 – A3..0 – tyto 4 bity umožňují přístup k ostatním registrům převodníku. Pro zápis do registru REFDAC1 při příští konverzi je třeba nastavit hodnotu x010, pro zápis do registru REFDAC2 je třeba nastavit x101. Při nastavení hodnoty x000 bude zapisováno přímo do registru CONFIG, přičemž tento zápis bude záviset na hodnotě vstupního signálu RD. [9]

Data jsou do registrů načítána ze vstupního signálu SDI. Pokud je na vstupním pinu $\overline{CS}$ hodnota *high*, pak je pin SDI ignorován. Princip programování registrů je zachycen na obrázku 4.3. [9]

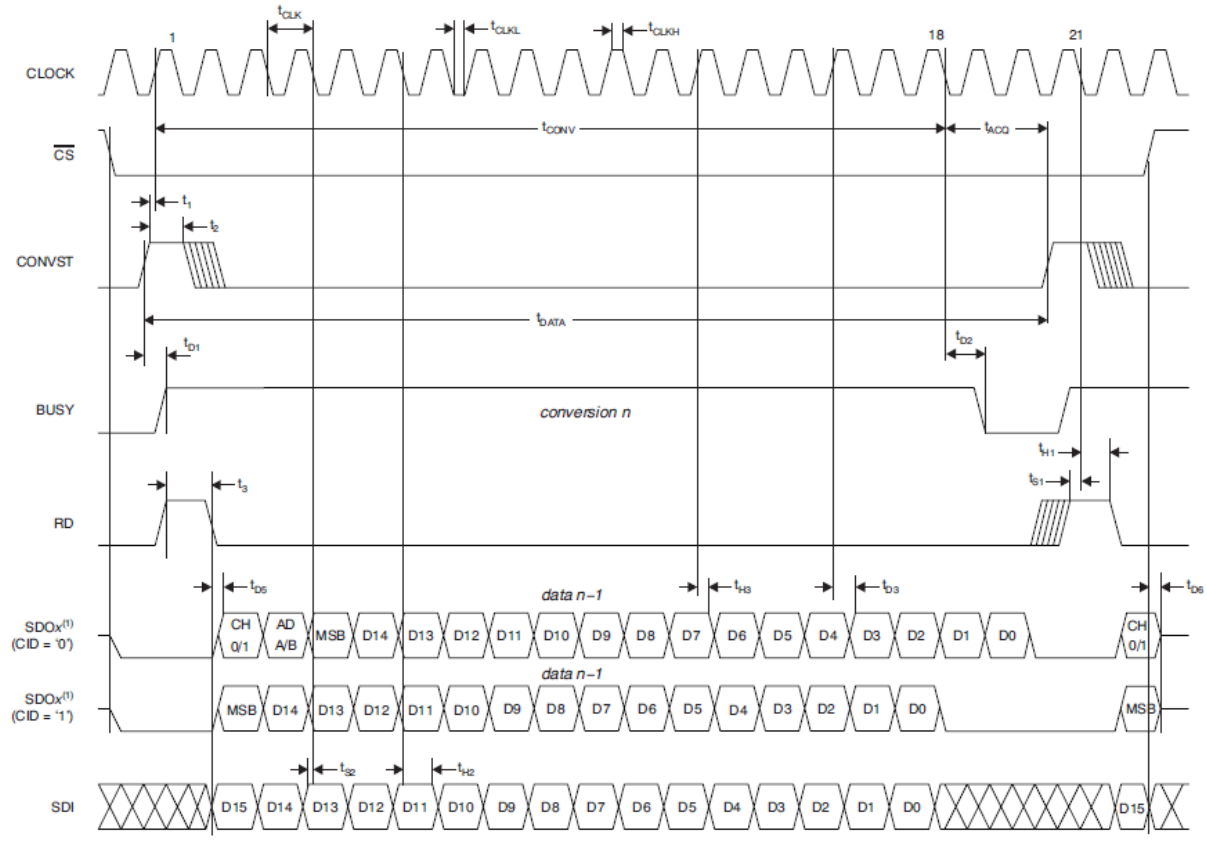


Obr. 4.3. Princip programování registru v módu Half-clock [9]

K řízení činnosti převodníku se používá šest digitálních signálů, které jsou označovány M1, M0, SDI, CONVST, RD, $\overline{CS}$. [9]

Signál CONVST značí začátek konverze. První bit převedené hodnoty se pak na výstupu objeví při další náběžné hraně signálu CLOCK. Signál RD slouží jako synchronizační pulz pro výstupy SDOA a SDOB a pro vstup SDI. Tento signál je ale ignorován, pokud signál $\overline{CS}$ má hodnotu *high*. Signál SDI se používá pro nastavení vnitřních registrů převodníku. M0 slouží pro přepínání mezi automatickým výběrem vstupního kanálu (hodnota 1) a ručním přepínáním (hodnota 0) skrze bity C1..0 registru CONFIG. Vzhledem k povaze úlohy je možné využít automatického přepínání. M1 pak značí, zda jsou výsledky konverze zobrazovány na obou výstupech SDOA i SDOB (hodnota 0) nebo pouze na výstupu SDOA (hodnota 1). [9]

Časový diagram všech signálů nutných pro řízení A/D převodníku ADS8363EVM je zachycen v obrázku 4.4. [9]



Obr. 4.4 Časový diagram řídicích signálů převodníku ADS8363EVM [9]

4.2 Aplikace pro získávání obrazových dat a jejich spravování

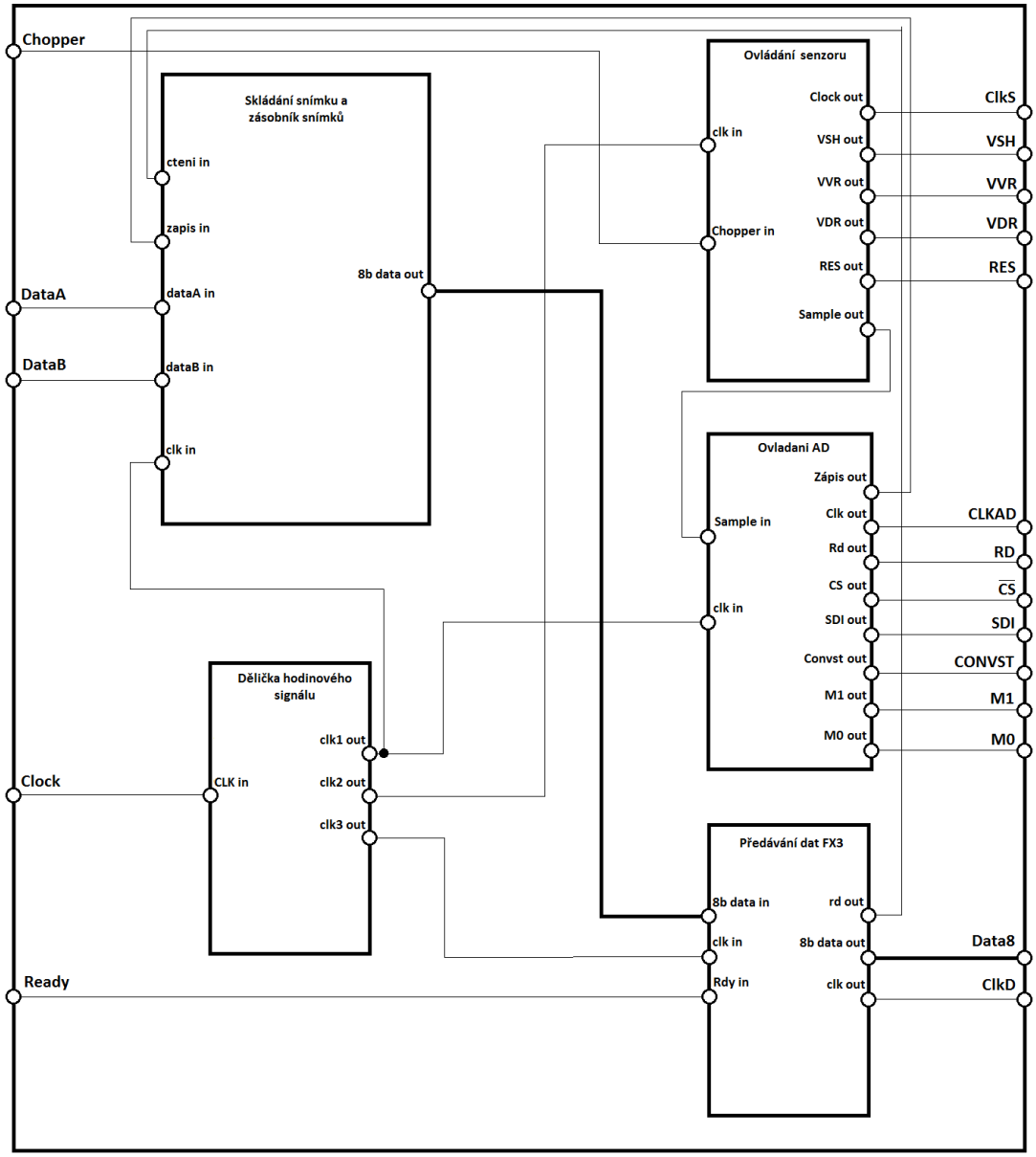
Aplikace pro získávání obrazových dat ze zvoleného senzoru, respektive přidruženého A/D převodníku, byla napsána v jazyce VHDL a implementována na FPGA obvodu EP3C120F484C7 obsahujícím čip Cyclone III od společnosti Altera.

Bylo vyžadováno, aby aplikace byla schopna ovládat optický senzor a k němu přidružený A/D převodník převádějící hodnotu získanou ze senzoru na digitální informaci, takto získanou digitální informaci vyčítat a skládat z ní obrazový snímek dle zvoleného formátu snímku, hotový snímek následně předat pro transport po sběrnici USB. Kromě těchto funkcí byl ještě požadavek na implementaci zásobníku, který by sloužil pro vyrovnávání

případných rozdílů rychlostí akvizice a transportu snímků. Pro tento zásobník mělo platit, že při naplnění kapacity bude docházet k odstranění nejstarších snímků.

Dle těchto požadavků byl vytvořen návrh aplikace a tento návrh byl následně realizován v jazyce VHDL. Celá aplikace byla rozdělena do pěti komponent, odpovídajících konkrétním požadovaným úlohám. Blokové schéma tohoto návrhu je zobrazeno na obrázku 4.5.

Aplikace má 5 digitálních vstupů a 21 digitálních výstupů. Výstupy slouží pro ovládání optického senzoru, A/D převodníku a k předávání dat pro odeslání po sběrnici. Na vstupy je třeba přivézt hodinový signál, signál z chopperu, oba datové kanály z A/D převodníku a řídicí signál z aplikace pro transport dat. Přesný popis vstupů je zachycen v tabulce 4.1, přesný popis výstupů v tabulce 4.2.



Obr. 4.5 Blokové schéma návrhu aplikace pro získávání a správu obrazových dat

Označení vstupu	Popis
Chopper	Vstup pro přivedení signálu, který generuje chopper. Frekvence tohoto signálu musí být v rozmezí 10 až 400Hz.
DataA	Vstup pro přivedení datového kanálu obsahující převedenou obrazovou informaci z A/D převodníku.

DataB	Vstup pro přivedení datového kanálu obsahující převedenou informaci o teplotě čidla z A/D převodníku.
Clock	Vstup pro přivedení hodinového signálu. Frekvence hodinového signálu musí být v rozmezí 2 až 80Mhz a platí, že frekvence signálu musí být v poměru 200000:1 k frekvenci signálu Chopper.
Ready	Vstup pro přivedení signálu z aplikace pro transport obrazových dat po sběrnici USB. Jedná se o signalizaci, že je obvod připraven přijímat data.

Tab. 4.1 Rozpis vstupů aplikace

Označení výstupu	Popis
ClkS	Hodinový signál pro pyroelektrický senzor
VSH	Signál pro stejnojmenný vstup pyroelektrického senzoru
VVR	Signál pro stejnojmenný vstup pyroelektrického senzoru
VDR	Signál pro stejnojmenný vstup pyroelektrického senzoru
RES	Signál pro stejnojmenný vstup pyroelektrického senzoru
CLKAD	Hodinový signál pro A/D převodník
RD	Signál pro stejnojmenný vstup A/D převodníku
$\overline{CS}$	Signál pro stejnojmenný vstup A/D převodníku
SDI	Signál pro stejnojmenný vstup A/D převodníku
CONVST	Signál pro stejnojmenný vstup A/D převodníku
M1	Signál pro stejnojmenný vstup A/D převodníku
M0	Signál pro stejnojmenný vstup A/D převodníku
ClkD	Hodinový signál pro řízení přenosu do obvodu sloužícího pro transport dat po sběrnici USB
Data8	Osm datových vstupů, dohromady tvoří osmibitovou sběrnici pro transport dat do aplikace předávající obrazovou informaci po sběrnici USB

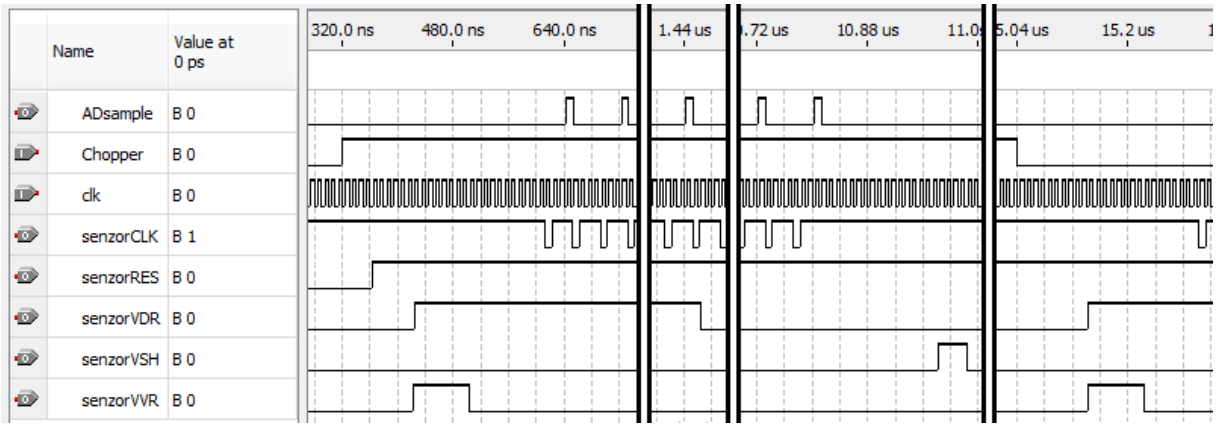
Tab. 4.2 Rozpis výstupů aplikace

4.2.1 Charakteristika komponent aplikace

Aplikace je rozdělena na pět komponent dle požadavků, které byly kladeny na aplikaci.

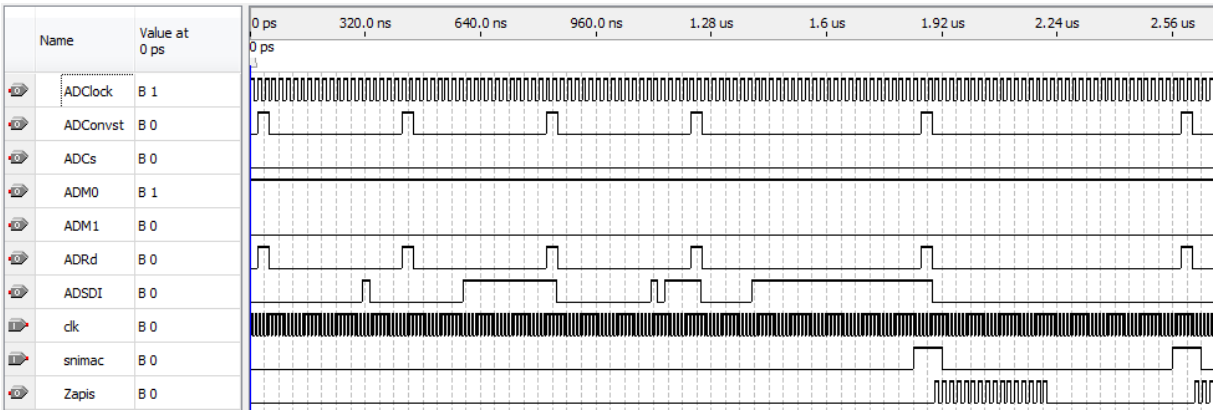
Základním kamenem je komponenta děličky hodinového signálu, která dělí hodinový signál tak, aby odpovídal požadovaným parametrů hodinových signálů jednotlivých komponent, respektive zařízení, které tyto komponenty řídí.

Komponenta sloužící pro ovládání senzoru generuje všechny požadované řídicí signály pro pyroelektrický senzor. Dále také generuje řídicí signál pro komponentu ovládající A/D převodník. Tento signál značí, že má dojít k převodu hodnoty jednoho pixelu. Komponenta je řízena hodinovým signálem a signálem, který vytváří chopper. Původně byl pro implementaci použit stavový automat. Tento přístup se ale ukázal jako nevhodný, neboť docházelo v obvodu k velkému zpoždění. Byla provedena simulace chování komponenty a výsledek simulace je na obrázku 4.6.



Obr. 4.6 Časový diagram komponenty pro ovládání senzoru

Komponenta, která slouží pro ovládání A/D převodníku je řízena hodinovým signálem z děličky a signálem značícím, že má dojít k procesu konverze. Komponenta pak generuje všechny řídicí signály potřebné k ovládání činnosti A/D převodníku a signál pro komponentu, která se stará o sběr dat z převodníku. Vzhledem k tomu, že A/D převodník vyžaduje nejprve nastavení hodnot jednotlivých registrů převodníku, musí ihned po spuštění dojít k sekvenci, která tyto registry nastaví. Vzhledem k vysoké frekvenci hodinového signálu, který ovládá chování komponenty, by nemělo dojít k tomu, že by tato sekvence měla vliv na chování celé aplikace. Simulací bylo ověřeno korektní chování komponenty. Výsledek simulace je na obrázku 4.7.

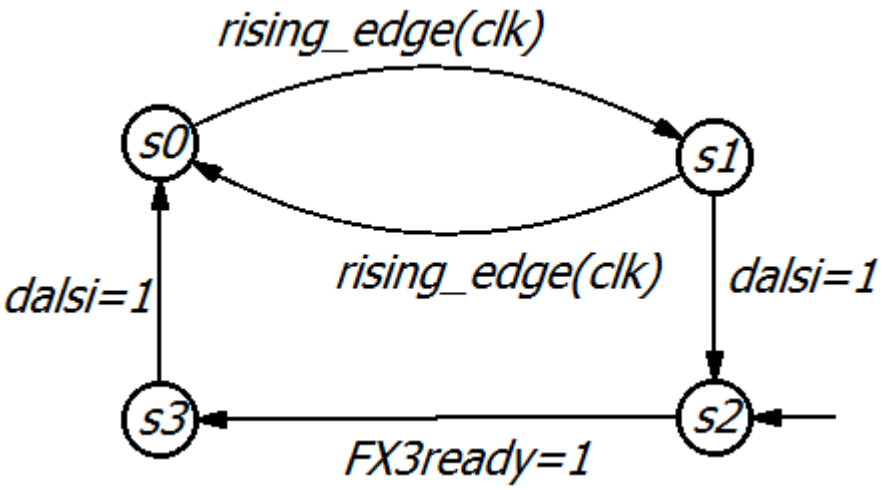


Obr. 4.7 Časový diagram komponenty pro ovládání A/D převodníku

Srdcem aplikace je komponenta, která z dat skládá snímek dle definované struktury. Hotový snímek pak uloží do zásobníku snímků, odkud jsou snímky dále na vyžádání předávány pro transport. Tuto komponentu by pravděpodobně bylo vhodnější rozdělit na dvě samostatné části, kdy jedna část by se starala pouze o skladbu snímku, druhá část by byla zásobník a přidružená rozhraní pro transport dat. Tato varianta by umožnila snadné vložení komponenty, která by s daty mohla nějakým způsobem manipulovat. Při implementaci se ale ukázalo, že tento způsob zpomaluje chování aplikace, a proto byly tyto dvě části sloučeny do jedné komponenty. Komponenta je ovládána hodinovým signálem a dvěma vstupními signály, které značí, že má dojít k transportu dat z nebo do struktury. Výstup pak tvoří osmibitová sběrnice sloužící pro transport dat komponentě, která se stará o transport k aplikaci, která data dále odesílá po sběrnici USB.

Pro ověření korektního chování komponenty byla provedena simulace. Výsledek simulace je uložen na přiloženém DVD. V práci není výsledek vložen z důvodu špatné přehlednosti v takto malém měřítku.

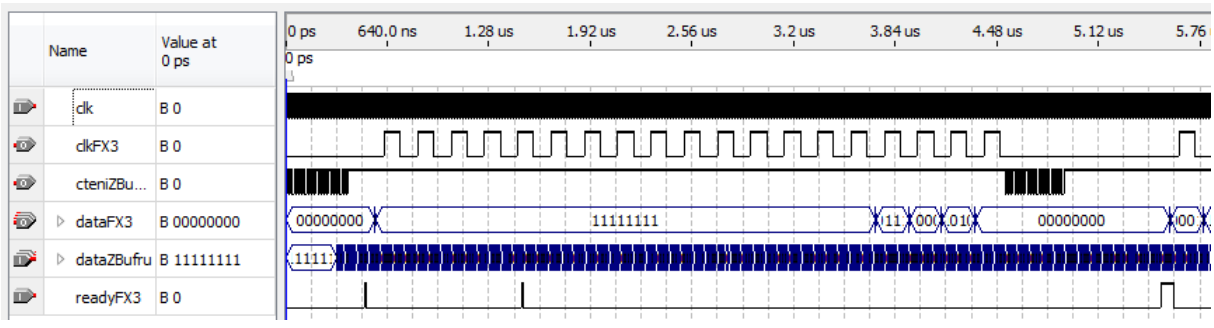
Poslední komponentou aplikace je část, která se stará o řízení transportu dat k aplikaci přenášející data po sběrnici USB. Komponenta je řízena signálem přicházejícím z aplikace sloužící pro přenos dat po sběrnici USB, který specifikuje, že aplikace je schopna přijímat data. Komponenta následně začne generovat synchronizační signál pro přenos a začne odesílat data. Po úspěšném odeslání zažádá o nová data ze zásobníku. Chování komponenty je řízeno stavovým automatem zobrazeným na obrázku 4.8.



Obr. 4.8 Stavový automat komponenty pro transport dat

Stavy s0 a s1 slouží pro načítání snímku ze zásobníku. Když je načten celý snímek (signál další nastaven na 1), je automat přepnut do stavu s2, který slouží pro čekání na to, až bude druhá strana transakce připravena (signál FX3ready nastaven na 1). Poté se automat přepne do stavu s3, kdy dochází k odesílání snímku. Když je odeslán celý snímek (signál další nastaven na 1), automat se přepne zpět do stavu s0 a dojde k načtení dalšího snímku ze zásobníku.

Aby byla ověřena funkčnost komponenty, byla provedena simulace. Pro přehlednost byl pro tuto simulaci změněn rozsah snímku z 1024 bajtů na 20. Výsledek simulace je na obrázku 4.6.



Obr. 4.9 Časový diagram komponenty pro transport dat

5 Aplikace pro transport dat po sběrnici USB

5.1 Charakteristika aplikace

Druhou částí práce bylo vytvoření aplikace, která by přebírala obrazovou informaci z FPGA obvodu skrze definované rozhraní a tuto informaci dále transportovala po sběrnici USB k cílové počítačové platformě. [10]

K implementování aplikace byl použit vývojový kit CYUSB3KIT-001 obsahující kontrolér EZ-USB FX3 od společnosti Cypress Semiconductor Corporation. Využití takového kontroléru je výhodné pro zjednodušení tvorby aplikace, neboť při vývoji aplikace není dále nutné řešit operace na nízkých úrovních protokolu. Tento konkrétní kontrolér byl vybrán z důvodu podpory USB 3.0, což by mohlo být výhodné v případě dalšího vývoje aplikace a dále programovatelného firmwaru, který zjednodušuje vývoj celé aplikace. [10]

K vytvoření aplikace byla použita ukázková úloha společnosti Cypress, která byla upravena tak, aby splňovala požadované funkce a obsahovala rozhraní pro propojení kitu s FPGA obvodem. Pomocí deskriptorů je popsáno zařízení pro transport dat po sběrnici USB, přičemž je implementována třída UVC. Zařízení má jeden koncový bod pro izochronní přenos, ten byl použit výhradně z toho důvodu, že izochronní přenos je spojen s přenosy obrazové informace, neboť u dat této povahy je kladen větší důraz na včasné doručení a menší již na kontrolu chyb při přenosu. Pokud by ovšem byl větší důraz kladen na bezchybný přenos a menší na včasné doručení, bylo by vhodnější využít koncový bod pro hromadný přenos. Přenášená data jsou ve formátu YUY2, formát dekódovaného snímku odpovídá výše popsanému formátu. Rozhraní pro připojení FPGA obvodu je realizováno pomocí takzvaných GPIO pinů, tedy pinů, které je možné naprogramovat jako vstupní nebo výstupní. Popis funkce jednotlivých pinů a jejich rozmístění na vývojovém kitu je v tabulce 5.1.

Číslo pinu	Umístění pinu na vývojovém kitu	Typ pinu	Popis
21	J100.2	Výstupní	Signalizace připravenosti obvodu přijímat data, signál má být přiveden na vstup označený Ready v návrhu na obrázku 4.5
45	TP13	Vstupní	Synchronizace transportu dat, na tento pin má být

			přiveden signál označený clkD v návrhu na obrázku 4.5
50	J20.4	Vstupní	Bit 7 (MSB) datové sběrnice označené jako Data8 v návrhu na obrázku 4.5
51	J20.5	Vstupní	Bit 6 datové sběrnice označené jako Data8 v návrhu na obrázku 4.5
52	J20.6	Vstupní	Bit 5 datové sběrnice označené jako Data8 v návrhu na obrázku 4.5
53	J101.2	Vstupní	Bit 4 datové sběrnice označené jako Data8 v návrhu na obrázku 4.5
54	J102.2	Vstupní	Bit 3 datové sběrnice označené jako Data8 v návrhu na obrázku 4.5
55	J103.2	Vstupní	Bit 2 datové sběrnice označené jako Data8 v návrhu na obrázku 4.5
56	J104.2	Vstupní	Bit 1 datové sběrnice označené jako Data8 v návrhu na obrázku 4.5
57	J20.7	Vstupní	Bit 0 (LSB) datové sběrnice označené jako Data8 v návrhu na obrázku 4.5

Tab. 5.1 Popis použitých GPIO pinů

Aplikace je rozdělena do čtyř souborů:

cyfxuvcdscr.c je zdrojový soubor napsaný v jazyce C a obsahuje všechny deskriptory používané aplikací.

cyfxuvcinmem.h je hlavičkový soubor jazyka C obsahující konstanty použité v aplikaci.

cyfxuvcvidframes.c je zdrojový soubor jazyka C, který obsahuje transportovaný snímek a další datové struktury spojené se snímkem.

cyfxuvcinmem.c je zdrojový soubor jazyka C. Tento soubor obsahuje veškerou funkcionalitu aplikace.

Funkcionalita je kompletně popsána v komentářích v kódu každého souboru.

6 Další úpravy aplikace

6.1 Formát YUV

YUV je označení pro barevný prostor. Na rozdíl od prostoru RGB, kde je jeden pixel popsán pomocí třech barevných složek, je tento popsán pomocí složek označených Y', U a V, kde Y' (luma) označuje jas, U a V jsou takzvané chromatické složky. V oboru digitálního videa označuje YUV barevný prostor pojmenovaný Y'CbCr, který pracuje na podobném principu. Přesto se ale jedná o dva různé formáty. [11] [12]

Velkou výhodou tohoto barevného prostoru je to, že je zpětně kompatibilní s černobílým vysíláním pro analogové televize. Další výhodou je, že díky vlastnostem lidského oka je často možné omezit rozsah chromatických složek a tím značně zmenšit objem přenášených dat. Pomocí tohoto principu bylo vytvořeno velké množství formátů, které se používají pro kódování digitálního videa. Tyto formáty se pak souhrnně označují jako YUV formáty. Každý formát je popsán pomocí identifikátoru GUID, který je součástí Format payload deskriptoru pro nekomprimované video. [11] [12]

YUV formáty se dělí do dvou skupin a to na skupinu formátů označovaných jako *packed* a skupinu označovanou *planar*. Formáty ze skupiny označené *packed* jsou charakteristické tím, že jednotlivé složky obrazu jsou uloženy ve formě takzvaných macropixelů do jednoho řetězce. Formáty *planar* mají všechny tři složky uložené zvlášť do separátních řetězců. [11] [12]

Třída UVC verze 1.1 podporuje formát YUY2 ze skupiny *packed* a formát NV12 ze skupiny *planar*. UVC verze 1.5 pak rozšiřuje podporované formáty o M420 ze skupiny *packed* a I420 ze skupiny *planar*. Bohužel žádný z těchto formátů není vhodný, pokud by bylo cílem zobrazovat data z pyroelektrického senzoru jako monochromatický obraz, neboť tím, že je informace rozdělena mezi jednotlivé složky, dojde při dekódování obrazu k milné interpretaci informace. Možným řešením by bylo uložit informaci pouze do složky představující jas (Y), pak by ale muselo dojít ke ztrátové konverzi hodnoty jasu. Druhým způsobem řešení je upravit aplikaci tak, aby data byla ve formátu Y16. Tento formát je monochromatický. Jasová složka (Y) je zakódována pomocí 16 bitů a složky U a V jsou ignorovány. V zařízení stačí pozměnit GUID v deskriptoru Video Format descriptor na GUID

20363159-0000-0010-8000-00AA00389B71. Po změně již systém detekuje, že zařízení produkuje obrazová data v požadovaném formátu. Pro zobrazení dat je pak nutné do systému Windows doinstalovat filtr příslušný pro tento formát. [2] [3] [11] [12]

6.2 Externí paměť

V případě, že by bylo třeba využít jiný senzor, který by měl několikanásobně vyšší rozlišení, by patrně nastal problém v podobě nedostatku vnitřní paměti FPGA obvodu. V takovém případě by bylo buď možné zasílat postupně části a celek z nich tvořit až na úrovni počítačové platformy. Druhým řešením, které by bylo nutné použít v případě, že by se s daty měla provádět nějaká operace již na úrovni FPGA obvodu, by bylo využít externí paměť, která by obsahovala snímek a FPGA obvod by operoval vždy jen s některou jeho částí. Společnost Altera nabízí několik kontrolérů DDR a DDR2 SDRAM paměti dostupných jako IP Core. V tomto případě by bylo samozřejmě nutné změnit celý návrh aplikace pro FPGA obvod tak, aby umožnil komunikaci skrze rozhraní, které definuje kontrolér. Ve vnitřní paměti by se pak ukládala pouze adresa, kde je v paměti snímek uložen. Funkční bloky, které slouží pro skládání a transport snímku by pak vždy obsahovaly adresu paměti, z které se mají předat data pro transport, nebo naopak kam se mají data zapsat při skládání snímku.

7 Závěr

Cíle stanovené na začátku práce byly v průběhu práce splněny. Byla provedena rešerše zabývající se předáváním dat z hardwarových prostředků do struktury operačního systému Windows po sběrnici USB, jakožto sběrnici vhodnou pro vytvoření aplikace sloužící pro transport dat do struktury operačního systému Windows. Byl prozkoumán protokol sběrnice USB a dále pak třída USB Video Device Class sloužící pro popis zařízení pracujícího s obrazovou informací přenášenou po sběrnici USB. V práci pak byly popsány získané informace tak, aby zachytily základní nutné požadavky pro vytvoření zařízení přenášejícího data po sběrnici USB a implementující třídu USB Video Device Class. Dále bylo takové zařízení implementováno na vhodném vývojovém kitu a později doplněno o rozhraní a funkcionalitu umožňující získávání dat z FPGA obvodu poskytujícího obrazová data. Dále byl vytvořen návrh na aplikaci, která by ze vzorového optického senzoru získávala obrazová data a tato data dále předávala pro přenos po sběrnici USB. Návrh byl poté implementován na vybraném FPGA obvodu a celý proces tvorby byl zdokumentován v této práci.

Vzhledem k tomu, že tato práce byla prvním krokem k vytvoření možného komplexního zařízení, zpracovávajícího a přenášejícího obrazová data do struktury operačního systému Windows po sběrnici USB, nabízí se několik možných způsobů, jak práci dále rozvíjet. Některé způsoby možného dalšího rozšíření byly popsány přímo v práci, a to rozšíření o externí paměť, která by umožnila zpracování dat ze senzoru s vysokým rozlišením a dále prozkoumání možnosti filtrů, které by umožnily operovat s daty zakódovanými pomocí monochromatických formátů. Kromě toho by bylo možné hlouběji proniknout do protokolu USB a doplnit celou aplikaci o funkcionalitu, která by umožnila ovládání parametrů akvizice a transportu obrazových snímků, ale také vlastností samotné obrazové informace. Tato práce by pak měla poskytnout dostatek počátečních informací pro libovolnou z těchto cest.

Literatura

[1] COMPAQ COMPUTER CORPORATION et al. *Universal Serial Bus Specification* [online]. 2000, vyd. 27.4.2000 [cit. 2012-8-20]. Dostupné z: http://www.usb.org/developers/docs/usb_20_110512.zip

[2] USB IMPLEMENTERS FORUM *Universal Serial Bus Device Class Definition for Video Devices* [online]. 2005, vyd. 1.6.2005 [cit. 2012-8-25]. Dostupné z: http://read.pudn.com/downloads77/ebook/293318/USB%20Video%20Class%201_1/USB_Video_Class_1.1.pdf

[3] MICROSOFT Je k dispozici aktualizovaný ovladač USB Video Class (UVC) pro systém Windows XP s aktualizací Service Pack 2. In: *Microsoft: Pomoc a podpora* [online]. 2011, vyd. 22.5.2011 [cit. 2012-8-25]. Dostupné z: http://support.microsoft.com/kb/899271

[4] PEACOCK, C. USB in a NutShell. *Beyond Logic* [online]. 2010, vyd. 2010 [cit. 2012-08-25]. Dostupné z: http://www.beyondlogic.org/usbnutshell/usb1.shtml

[5] HEWLETT-PACKARD COMPANY et al. *Universal Seriál Bus 3.0 Specification* [online]. 2011, vyd. 6.6.2011 [cit. 2012-8-25]. Dostupné z: http://www.usb.org/developers/docs/usb_30_spec_122012.zip

[6] PALKO, L. *Moderní trendy sériového rozhraní USB* [online]. 2005, vyd. 18.4.2005 [cit. 2012-8-25]. Dostupné z: http://www.elektrorevue.cz/clanky/05025/index.html#2.4.S%C3%A9riov%C3%BD%20p%C5%99enos

[7] AXELSON, J. *USB Complete: The Developer's Guide, Fourth Edition*. Fourth Edition. Lakeview Research, 2009. ISBN ISBN 978-1931448086.

[8] DIAS INFRARED GMBH *128LTI Hybrid pyroelectric linear array with 128 responsive elements and integrated CMOS multiplexer* [online] 4/2006, vyd. 4.2006 [cit. 2012-8-25]. Dostupné z: http://www.scitec.uk.com/infrared_detectors/datasheets/scitecdias_128lti_eng.pdf

[9] TEXAS INSTRUMENTS INCORPORATED *ADS7263/ADS8363EVM User Guide* [online] 2011, vyd. 11.1.2011 [cit. 2012-8-25]. Dostupné z: http://www.ti.com/lit/ug/sbau185/sbau185.pdf

[10] CYPRESS SEMICONDUCTOR CORPORATION *CYUSB3KIT-001 EZ-USB® FX3™ Development Kit* [online] 2012, vyd. 3.1.2012 [cit. 2012-8-25]. Dostupné z: http://www.cypress.com/?rID=58321

[11] SILICON.DK APS YUV pixel formats [online] 2011 [cit. 2012-8-25]. Dostupné z: http://www.fourcc.org/yuv.php

[12] MICROSOFT *About YUV Video (Windows)* [online] 2012 [cit. 2012-8-25]. Dostupné z: http://msdn.microsoft.com/en-us/library/windows/desktop/bb530104%28v=vs.85%29.aspx

[13] Linux UVC driver & tools. *Ideas on board* [online]. 2011 [cit. 2012-8-25]. Dostupné z: http://www.ideasonboard.org/uvc/

Přílohy

Příloha A – Deskriptory

Offset	Označení pole	Velikost v bajtech	Typ hodnoty	Popis pole
0	bLength	1	Číslo	Velikost deskriptoru v bajtech tj. 18
1	bDescriptorType	1	Konstanta	Konstanta označující typ deskriptoru - Device Descriptor (0x01)
2	bcdUSB	2	BCD	Číslo nejvyšší verze USB, kterou zařízení implementuje
4	bDeviceClass	1	Třída	Označení třídy, kterou zařízení implementuje – nastavení na hodnotu nula znamená, že je třída definovaná až na úrovni rozhraní
5	bDeviceSubClass	1	Podtřída	Označení podtřídy
6	bDeviceProtocol	1	Protokol	Označení protokolu
7	bMaxPacketSize	1	Číslo	Maximální velikost paketu, kterou umí pojmout endpoint 0, povolené hodnoty jsou 8, 16, 32, 64
8	idVendor	2	ID	Vendor ID, tedy číslo zařízení, toto číslo je přidělováno konsorciem USB Org
10	idProduct	2	ID	ID konkrétního zařízení
12	bcdDevice	2	BCD	Označuje verzi zařízení
14	iManufacturer	1	Index	Číslo String deskriptoru obsahujícího textovou informaci o výrobci
15	iProduct	1	Index	Číslo String deskriptoru obsahujícího textovou informaci o zařízení
16	iSerialNumber	1	Index	Číslo String deskriptoru obsahujícího textovou informaci o sériovém čísle zařízení
17	bNumConfigurations	1	Integer	Počet konfigurací zařízení

Tab. A.1 Device descriptor [1]

Offset	Označení pole	Velikost v bajtech	Typ hodnoty	Popis pole
0	bLength	1	Číslo	Velikost deskriptoru v bajtech tj. 9
1	bDescriptorType	1	Konstanta	Konstanta označující typ deskriptoru - Configuration Descriptor (0x02)
2	wTotalLength	2	Číslo	Délka deskriptoru a všech

				následujících deskriptorů vztahujících se k této konfiguraci
4	bNumInterfaces	1	Číslo	Počet rozhraní konfigurace
5	bConfigurationValue	1	Číslo	Číslo označující jednoznačně konkrétní konfiguraci, následně se používá při nastavování konkrétní zvolené konfigurace
6	iConfiguration	1	Index	Číslo String deskriptoru obsahujícího textovou informaci o této konfiguraci
7	bmAttributes	1	Bitmapa	D7 Rezervováno, nastavit na 1. (USB 1.0 Bus Powered) D6 Vlastní napájení D5 Remote Wakeup D4..0 Rezervováno, nastavit na 0
8	bMaxPower	1	mA	Maximální spotřeba energie v jednotkách 2mA

Tab. A.2 Configuration descriptor [1]

Offset	Označení pole	Velikost v bajtech	Typ hodnoty	Popis pole
0	bLength	1	Číslo	Velikost deskriptoru v bajtech tj. 9
1	bDescriptorType	1	Konstanta	Konstanta označující typ deskriptoru - Interface Descriptor (0x04)
2	bInterfaceNumber	1	Číslo	Číslo označující jednoznačně konkrétní rozhraní, následně se používá pro výběr rozhraní
3	bAlternateSetting	1	Číslo	Číslo alternativního nastavení
4	bNumEndpoints	1	Číslo	Počet koncových bodů rozhraní
5	bInterfaceClass	1	Třída	Kód třídy, je přidělován konsorciem USB Org
6	bInterfaceSubClass	1	Podtřída	Kód podtřídy, je přidělován konsorciem USB Org
7	bInterfaceProtocol	1	Protokol	Kód protokolu, je přidělován konsorciem USB Org
8	iInterface	1	Index	Číslo String deskriptoru obsahujícího textovou informaci o tomto rozhraní

Tab. A.3 Interface descriptor [1]

Offset	Označení pole	Velikost v bajtech	Typ hodnoty	Popis pole
0	bLength	1	Číslo	Velikost deskriptoru v bajtech tj. 9
1	bDescriptorType	1	Konstanta	Konstanta označující typ deskriptoru - Endpoint Descriptor (0x05)
2	bEndpointAddress	1	Číslo koncového bodu	Bity 0..3b Číslo koncového bodu Bity 4..6b Rezervováno, nastavit na 0 Bity 7 Směr 0 = Out, 1 = In
3	bmAttributes	1	Bitmapa	Bity 0..1 Typ přenosu 00 = Řídicí 01 = Izochronní 10 = Hromadný 11 = Při přerušení Bity 2..7 Jsou rezervovány, pokud nejde o koncový bod s izochronním přenosem. Pak: Bity 3..2 = Typ synchronizace 00 = Bez synchronizace 01 = Asynchronní 10 = Adaptivní 11 = Synchronní Bity 5..4 = Usage Type 00 = Data Endpoint 01 = Feedback Endpoint 10 = Explicit Feedback Data Endpoint 11 = Rezervováno
4	wMaxPacketSize	2	Číslo	Maximální velikost paketu, s kterou může koncový bod pracovat
6	bInterval	1	Číslo	Interval přenosu, hodnota se liší dle typu přenosu skrze koncový bod, pro hromadné a řídicí přenosy je pole ignorováno, pro izochronní musí být nastaveno na hodnotu 1, pro přenosy při přerušení se hodnota může pohybovat v rozsahu 1 až 255

Tab. A.4 Endpoint descriptor [1]

Offset	Označení pole	Velikost v bajtech	Typ hodnoty	Popis pole
0	bLength	1	Číslo	Velikost deskriptoru v bajtech tj. 27

1	bDescriptorType	1	Konstanta	Konstanta označující typ deskriptoru – Class Specific Interface Descriptor (0x24)
2	bDescriptorSubtype	1	Konstanta	Konstanta označující podtyp deskriptoru – Format Uncompressed (0x04)
3	bFormatIndex	1	Číslo	Číslo pro identifikaci tohoto deskriptoru
4	bNumFrameDescriptors	1	Číslo	Počet Frame deskriptorů náležících k tomuto
5	guidFormat	16	GUID	Globally Unique Identifier, používá se pro specifikování formátu, v kterém je obraz kódován
21	bBitsPerPixel	1	Číslo	Počet bitů na jeden pixel
22	bDefaultFrameIndex	1	Číslo	Optimum Frame Index, používá se pro nastavení rozlišení
23	bAspectRatioX	1	Číslo	Číslo určující X pro poměr X:Y výsledného obrazu
24	bAspectRatioY	1	Číslo	Číslo určující Y pro poměr X:Y výsledného obrazu
25	bInterlaceFlags	1	Bitmapa	Pokud zařízení podporuje mód skenování, pak je třeba nastavit jednotlivé parametry pomocí této bitmapy
26	bCopyProtect	1	Boolean	Určuje, zda je omezena duplikace datového proudu, 0 = bez restrikcí, 1 = duplikace zakázána

Tab. A.5 Format descriptor pro nekomprimované formáty videa [2]

Offset	Označení pole	Velikost v bajtech	Typ hodnoty	Popis pole
1	bDescriptorType	1	Konstanta	Konstanta označující typ deskriptoru – Class Specific Interface Descriptor (0x38)
2	bDescriptor Subtype	1	Konstanta	Konstanta označující podtyp deskriptoru – Frame Uncompressed (0x05)
3	bFormatIndex	1	Číslo	Číslo pro identifikaci tohoto deskriptoru
4	bmCapabilities	1	Číslo	Hodnota určuje možnost zachycovat obraz jako still image
5	mWidth	2	Číslo	Šířka dekódovaného snímku v pixelech
7	mHeight	2	Číslo	Výška dekódovaného snímku v pixelech

9	dwMinBitRate	4	Číslo	Minimální rychlost přenosu bitu
13	dwMaxBitRate	4	Číslo	Maximální rychlost přenosu bitu
17	dwVideoMax BufferSize	4	Číslo	Počet bajtů získaných kompresí videa, toto pole je doporučováno nepoužívat
21	dwDefaultFrame Interval	4	Číslo	Interval snímků, který bude zařízení brát jako základní
25	bFrameInterval Type	1	Číslo	Pole určuje, jakým způsobem budou programovány intervaly snímků, hodnota 0 označuje typ *Continuous,* hodnota 1 až 255 určuje počet diskrétních hodnot pro definici intervalu, na tomto poli je závislá další struktura deskriptoru.
Pro typ *Continuous* vypadá další struktura takto:				
26	dwMinFrame Interval	4	Číslo	Nejkratší podporovaný interval v jednotkách 100 ns
30	dwMaxFrame Interval	4	Číslo	Nejdelší podporovaný interval v jednotkách 100 ns
34	dwFrame IntervalStep	4	Číslo	Granularita intervalu v jednotkách 100 ns
Pomocí diskrétních hodnot vypadá další struktura takto:				
26	dwFrame Interval(1)	4	Číslo	Nejkratší podporovaný interval v jednotkách 100 ns
…	…	…	…	…
26 + (4*n) -4	dwFrame Interval(n)	4	Číslo	Nejdelší podporovaný interval v jednotkách 100 ns

Tab. A.6 Frame descriptor pro nekomprimované formáty videa [2]

Příloha B – Standardní requesty

bmRequest Type	bRequest	wValue	wIndex	wLength	Data
1000 0000b	GET_STATUS (0x00)	0	0	2	Status zařízení
0000 0000b	CLEAR_FEATURE (0x01)	Feature Selector	0	0	Data nejsou předávána
0000 0000b	SET_FEATURE (0x03)	Feature Selector	0	0	Data nejsou předávána
0000 0000b	SET_ADDRESS (0x05)	Adresa zařízení	0	0	Data nejsou předávána
1000 0000b	GET_DESCRIPTOR (0x06)	Typ a index deskriptoru	0 nebo ID jazyka	Délka deskriptoru	Deskriptor
0000 0000b	SET_DESCRIPTOR (0x07)	Typ a index deskriptoru	0 nebo ID jazyka	Délka deskriptoru	Deskriptor
1000 0000b	GET_CONFIGURATION (0x08)	0	0	1	Číslo konfigurace
0000 0000b	SET_CONFIGURATION (0x09)	Číslo konfigurace	0	0	Data nejsou předávána

Tab. B.1 Device Requests [1]

bmRequest Type	bRequest	wValue	wIndex	wLength	Data
1000 0001b	GET_STATUS (0x00)	0	Rozhraní	2	Status rozhraní
0000 0001b	CLEAR_FEATURE (0x01)	Feature Selector	Rozhraní	0	Data nejsou předávána
0000 0001b	SET_FEATURE (0x03)	Feature Selector	Rozhraní	0	Data nejsou předávána
1000 0001b	GET_INTERFACE (0x0A)	0	Rozhraní	1	Alternativní rozhraní
0000 0001b	SET_INTERFACE (0x11)	Alternativní nastavení	Rozhraní	0	Data nejsou předávána

Tab. B.2 Interface Requests [1]

bmRequest Type	bRequest	wValue	Windex	wLength	Data
1000 0010b	GET_STATUS (0x00)	0	Koncový bod	2	Status koncového bodu

0000 0010b	CLEAR_FEATURE (0x01)	Feature Selector	Koncový bod	0	Data nejsou předávána
0000 0010b	SET_FEATURE (0x03)	Feature Selector	Koncový bod	0	Data nejsou předávána
1000 0010b	SYNCH_FRAME (0x12)	0	Koncový bod	2	Číslo rámce

Tab. B.3 Endpoint Requests [1]

Příloha C – Struktura přiloženého DVD

Struktura DVD

- Složka Text – obsahuje elektronickou verzi práce ve formátu .docx a .pdf
- Složka FX3
    - Aplikace – kompletní aplikace pro vývojový kit CYUSB3KIT-001
- Složka VHDL
    - Složka Aplikace – kompletní aplikace pro FPGA obvod
    - Složka Komponenty – jednotlivé komponenty aplikace