ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra řízení

Mobilní sběr a archivace dat pomocí FPGA Spartan 3

Červen 2006

Vypracoval: Jaroslav Stejskal
Vedoucí práce: Ing. Jiří Kadlec, CSc

Poděkování

Zde bych rád poděkoval svému vedoucímu bakalářské práce Ing. Jiřímu Kadlecovi, CSc. za cenné rady a ochotu během zpracování práce.

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně s přispěním vedoucího práce a použil jsem pouze podklady (literaturu, software, projekty atd.) uvedené v přiloženém seznamu.

V Praze dne …… ……

podpis

Anotace

Tato bakalářská práce se zabývá vývojem mobilního zařízení pro sběr a archivaci naměřených dat. Zařízení je realizováno pomocí FPGA čipu Spartan 3 na vývojové desce RC10 firmy Celoxica. Cílem je získat naměřená data v okolí uživatelem stanovené události. Data se ukládají do paměti FLASH a mohou být následně přenesena do PC přes rozhraní USB pro další zpracování. Zařízení může běžet samostatně z bateriového zdroje. Pro ovládání a prohlížení naměřených dat je využito VGA rozhraní k připojení monitoru.

Annotation

This bachelor project deals with making mobile device for measured data capturing and archiving. The device is realized by FPGA chip Spartan 3 on evaluation board RC10 from Celoxica company. The main goal of this work is to get measured data around some event, which is specified by user. Data are stored to the FLASH memory and it can be transfer to PC trough USB interface for future processing. The device can run from floating battery. It is used VGA interface for controlling the device and exploring measured data.

OBSAH

1. ÚVOD

Cílem této bakalářské práce je vytvořit zařízení, které periodicky ukládá naměřené průběhy vstupních signálů při výskytu uživatelem zvolené události. Zařízení tohoto typu se nazývá data-logger.

Zařízení bude realizováno na vývojové desce RC10 firmy Celoxica, která disponuje mnoha vhodnými periferiemi připojených na čip FPGA (Field Programmable Gate Array). Vyvíjené zařízení by mělo využít dva analogové vstupy s analogově digitálními převodníky pro měření vstupních signálů s přesností 10 bitů a vzorkovací frekvencí až 64MHz. Dále osm digitálních (logických) vstupů s vzorkovací frekvencí až 64MHz. Naměřené průběhy signálů by mělo být možné ukládat do paměti typu FLASH, která je součástí vývojové desky. Do této paměti by se mělo uložit alespoň jeden milion (10^6) vzorků. Zařízení by mělo být vybaveno grafickým uživatelský rozhraním s využitím analogového VGA výstupu vývojové desky. Uložené průběhy signálů v paměti FLASH by mělo být možné opět zobrazit samotným zařízením. Důležitou součástí zařízení je přenos naměřených dat do PC pomocí rozhraní USB a jejich následná rekonstrukce a vizualizace v programovém prostředí Matlab. Ovládání zařízení je zprostředkováno pomocí pěti směrového ovladače, který je také součástí vývojové desky.

Důležitým požadavkem na zařízení je vysoká vzorkovací frekvence v řádech desítek MHz, proto je potřeba pro zpracování této úlohy využít programovatelné hradlové pole tedy čip FPGA (Field Programmable Gate Array). Mikroprocesory nejsou na tuto úlohu vhodné, protože jedna jejich instrukce trvá několik strojových cyklů, a nedokážou dosahovat takové rychlosti zpracování signálů jako zmíněné čipy FPGA.

Pro návrh a realizaci tohoto zařízení bude použita již zmíněná vývojová deska RC10 od firmy Celoxica UK s FPGA čipem Xilinx Spartan 3 XC3S1500L-4. Z hlediska programového vybavení bude využito vývojové programové prostředí DK 4.1 (Development Kit) a PDK 4.1 (Platform Development Kit) od firmy Celoxica (software přímo určený pro návrh aplikací na vývojové desky typu RC s čipy FPGA) využívající programový jazyk Handel-C. Dále bude použito návrhové prostředí ISE 7.1 firmy Xilinx, které umožňuje vygenerovat konečný konfigurační soubor pro čip FPGA.

2. POUŽITÝ HARDWARE

2.1. Vývojová deska RC10

Pro řešení úlohy měřícího zařízení data-loggeru byla zvolena vývojová deska RC10 od anglické firmy Celoxica (www.celoxica.com). Předností jsou její bohaté periferie, velmi dobré parametry a bohatá softwarová podpora pro vývoj aplikací. Vývojová deska obsahuje mnoho periferií na jednom tištěném spoji, které jsou připojeny přímo nebo přes další pomocné integrované obvody k FPGA čipu viz. ***Obrázek 1***. Použitý čip FPGA bude blíže popsán v kapitole 2.1.2.

Obrázek 1 *Vývojová deska RC10 firmy Celoxica*
s FPGA čipem řady Xilinx Spartan 3L
(zdroj dokumentace vývojové desky RC10)

Vytváření samotného hardware pro čip FPGA se děje ve vývojovém prostředí DK, kde je psán program připomínající program ve vyšším programovacím jazyku. Tento zdrojový kód je po poměrně složité kompilaci přeložen do fyzického zapojení konkrétního FPGA čipu. Je tedy generováno fyzické zapojení základních elementárních funkčních bloků, připomínající nejjednodušší logická hradla, až po složitější bloky, konkrétního čipu FPGA. Podle tohoto výsledného zapojení se FPGA čip nakonfiguruje, což je velice rychlý proces, a čip okamžitě vykonává svou úlohu.

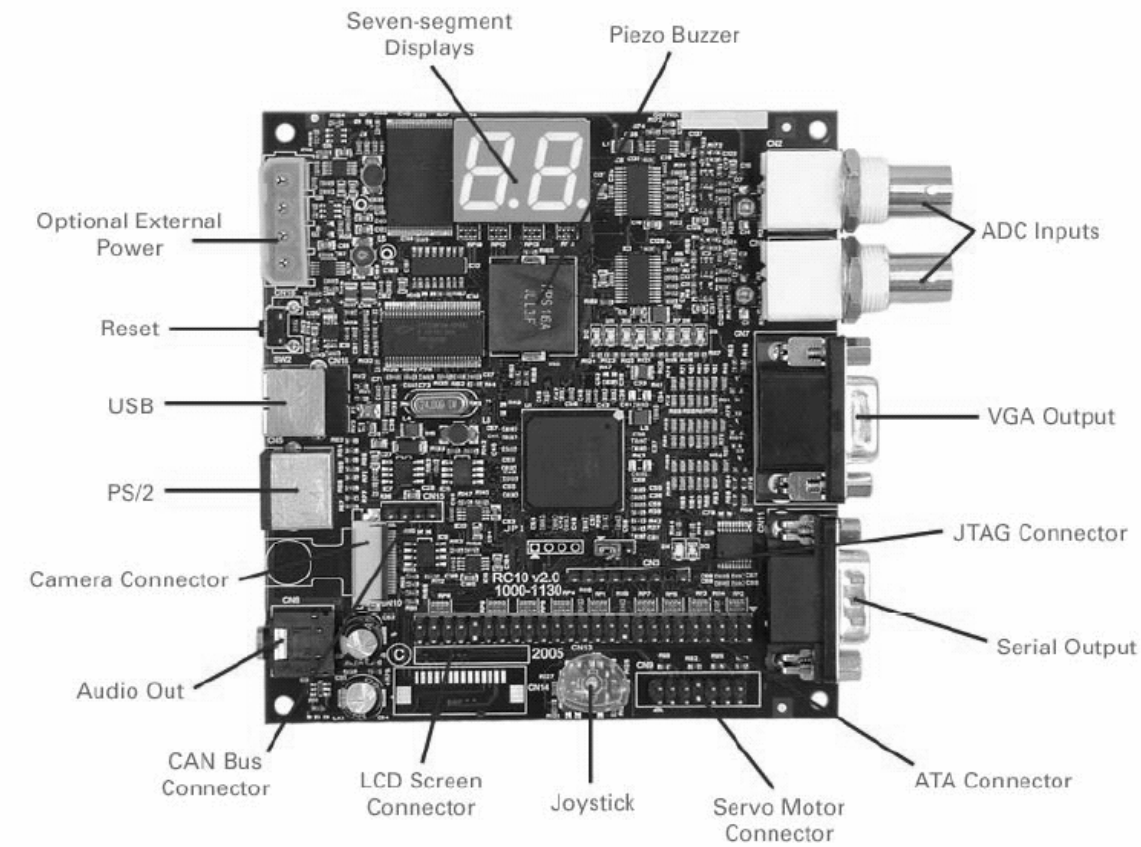
Programování ve vývojovém prostředí DK je podstatně jednoduší, než vytváření kódů v jazycích typu HDL (Hardware Description Language), ze kterých jsou nejznámější jazyky VHDL a Verilog. V prostředí DK se vytváří zdrojové kódy programovacího jazyka Handel-C založeného na jazycích C/C++.

Překlad zdrojových kódů a generování výsledného hardware pro čip FPGA je poměrně složitý, u větších projektů i časově náročný proces. Kompilace prochází několika částmi, které budou vysvětleny v kapitole 3.1.

Díky bohaté podpoře použitých periferií vývojové desky knihovnami PAL (Platform Abstract Layer) a PSL (Platform Support Library) je práce s těmito periferiemi poměrně snadná. Použití knihoven je dostatečně popsáno v dokumentaci, která je též součástí balíčku PDK, více viz. kapitola 3.2.

2.1.1. Specifikace a periferie

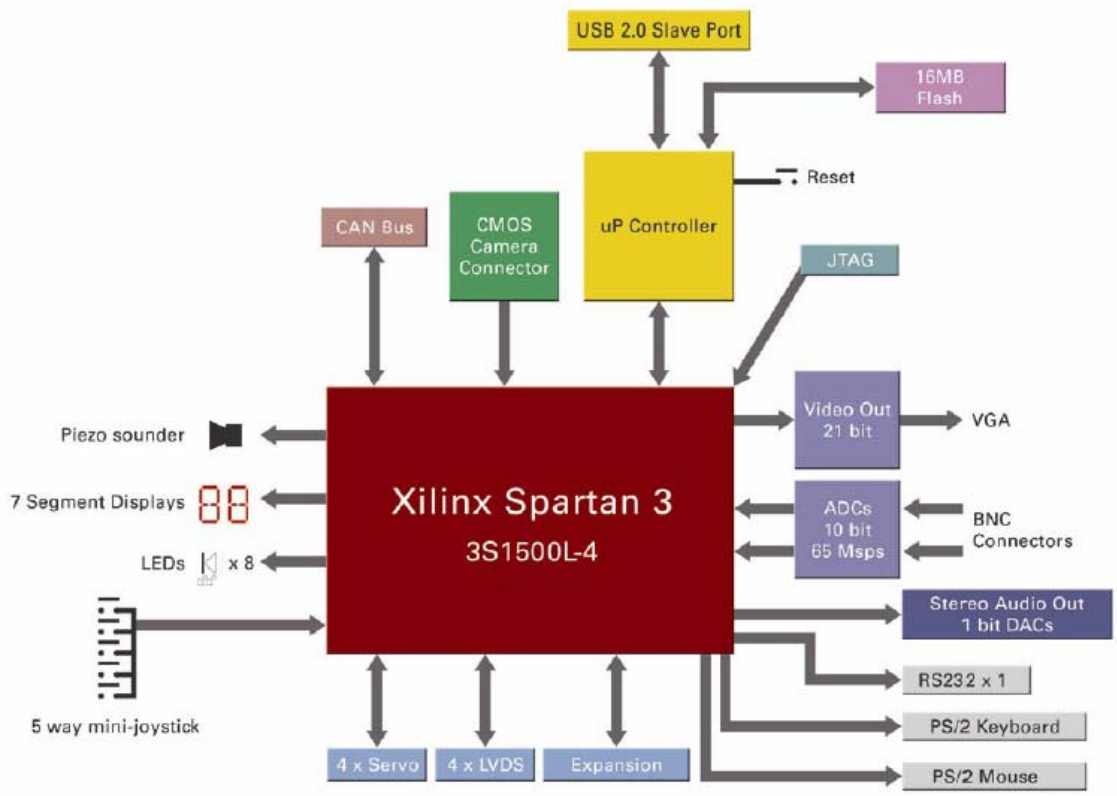
Vývojová deska RC10 disponuje mnoha periferii, které jsou znázorněny na originálním blokovém schématu viz. ***Obrázek 3,*** a na osazeném tištěném spoji viz. ***Obrázek 2***. Periferie zde budou uvedeny všechny, avšak podrobněji popsané budou v následujících kapitolách jen ty důležité pro samotné zařízení data-logger.



Obrázek 2 *Periferie vývojové desky RC10*
(zdroj www.celoxica.com)

- 5 - směrový mikro ovladač
- PS/2 port pro myš a klávesnici
- RS-232 sériový port

- Dva analogové digitální převodníky (ADC)
- VGA analogový výstup
- LCD video výstup
- Audio výstup (stereo pulsní šířková modulace, piezo reproduktor)
- USB mikroprocesor pro:
    - USB 2.0 rozhraní
    - konfiguraci FPGA
    - správu FLASH paměti
- Dva sedmi segmentové zobrazovače
- Osm zelených LED
- 50 – pinový konektor ATA rozhraní obsahující:
    - 33 – vstupně / výstupních pinů
    - 3 – napájecí piny (+12V, +5V, +3.3V)
    - 2 – piny s hodinovým signálem
- Konektor k řízení až čtyř servo motorů
- CAN bus konektor
- JTAG konektor

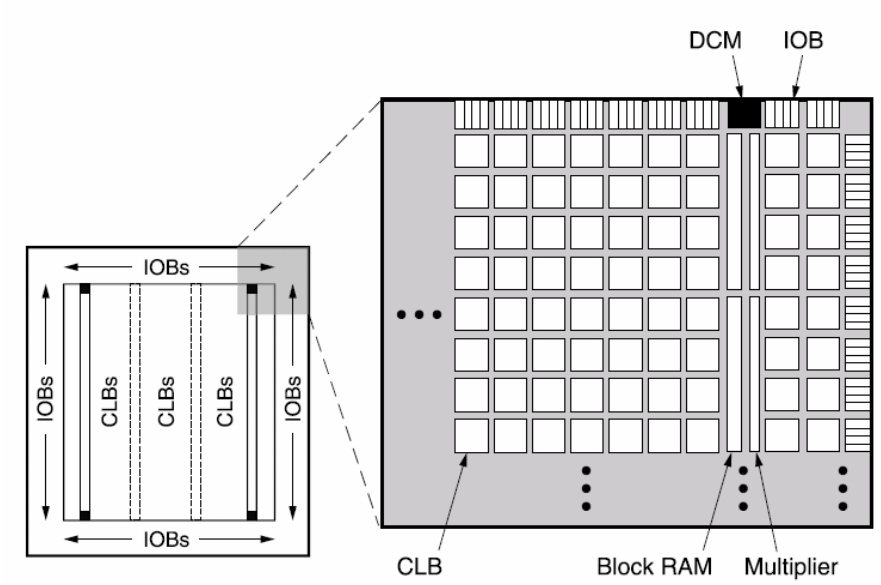


Obrázek 3 *Blokový diagram vývojové desky RC10*
(zdroj dokumentace vývojové desky RC10)

2.1.2. FPGA čip Spartan 3

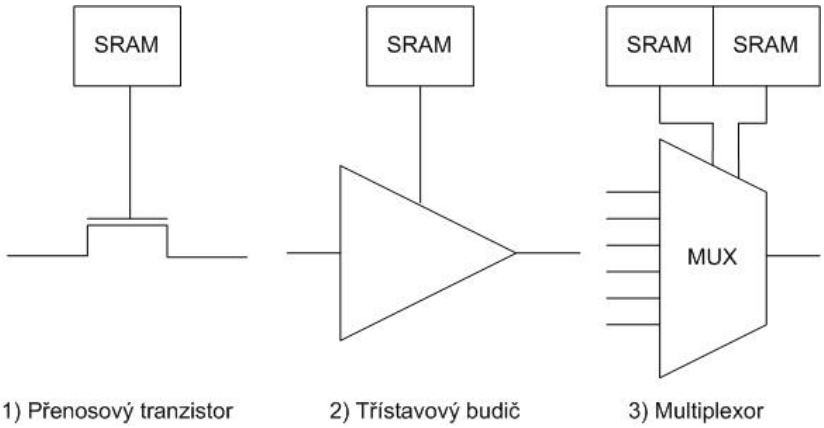
Obecně FPGA čipy jsou velká pole univerzálních logických bloků CLB a propojovacích vodičů. Propojování logických bloků pak tvoří různé logické funkce zadané návrhářem. Tyto logické bloky pak dohromady propojují konfigurační prvky (multiplexory) podle aktuální konfigurace nahrané do čipu. Dále čip obsahuje vstupně výstupní bloky IOB, paměťové bloky RAM, násobící bloky a bloky pro kmitočtovou syntézu DCM.

U rodiny čipů Xilinx Spartan je logický blok CLB (Configurable Logic Block) tvořen kombinací dvou vyhledávacích tabulek LUT (Look-up Table) a klopných obvodů typů D. Každá vyhledávací tabulka má čtyři vstupy. Těmito logickými bloky lze realizovat jednoduché kombinační obvody. Blok může být také použit jako přenosový prvek propojovací sítě. CLB logický blok je znázorněn na ***Obrázek 4***.



Obrázek 4 *Architektura FPGA čipu Spartan 3L*
(zdroj dokumentace Xilinx Spartan 3L)

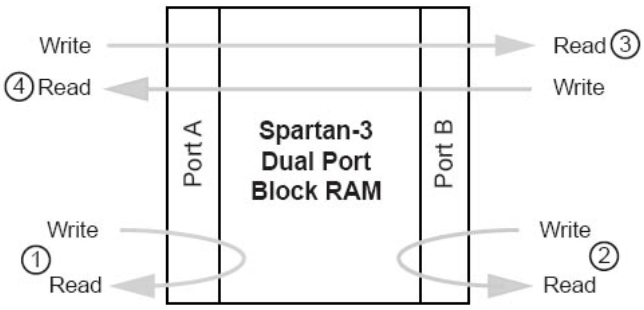
Propojovací síť je tvořena několika druhy vodičů podle jejich délek. Nejvíce obsahuje čip vodičů jednoduché délky (single length lines) pro propojení dvou sousedních logických bloků. Dále čip obsahuje vodiče dvojité délky (double length lines), které mají rozsah přes dva sloupce matice logických bloků na čipu. Třetí typ vodičů jsou dlouhé vodiče (long lines), které vystačí na vzdálenost všech sloupců nebo řádků celé matice logických bloků. Typ vodiče se určuje podle jeho použití. Aby bylo možné vodiče celé sítě mezi sebou různě propojovat, a také je připojovat k jednotlivým logickým blokům matice čipu, používají se programovatelné přepínače. Tyto přepínače jsou dvojího typu. První je přepínač typu PIP (Programmable Interconnection Point) sloužící k propojování vývodů logických bloků. Druhý typ jsou přepínače SMIP (Switching Matrix Interconnection Point) sloužící k propojování vodičové (signálové) sítě čipu.



Obrázek 5 *Programovatelné přepínače řízené paměťovou buňkou SRAM*

Aby bylo možné čip FPGA libovolně konfigurovat (přeprogramovat), jsou používané programovatelné přepínače tvořeny paměťovou buňkou SRAM. Tyto přepínače konfigurují funkci již zmíněného logického bloku nebo konfigurují propojení samotné sítě vodičů. Příklad použití programovatelného přepínače viz. ***Obrázek 5***, kde je řízena funkce přenosového CMOS tranzistoru 1), třístavového budiče 2) a multiplexoru 3). To má za následek tu nevýhodu, že při výpadku napájení je konfigurace čipu vymazána a čip je nutné znovu nakonfigurovat například ze záložní FLASH paměti.

Dalším důležitým prvkem FPGA čipu je vstupně výstupní blok IOB (input output block), který propojuje vnitřní logiku se vstupně výstupními piny čipu. IOB umožňuje třístavový obousměrný přenos dat.



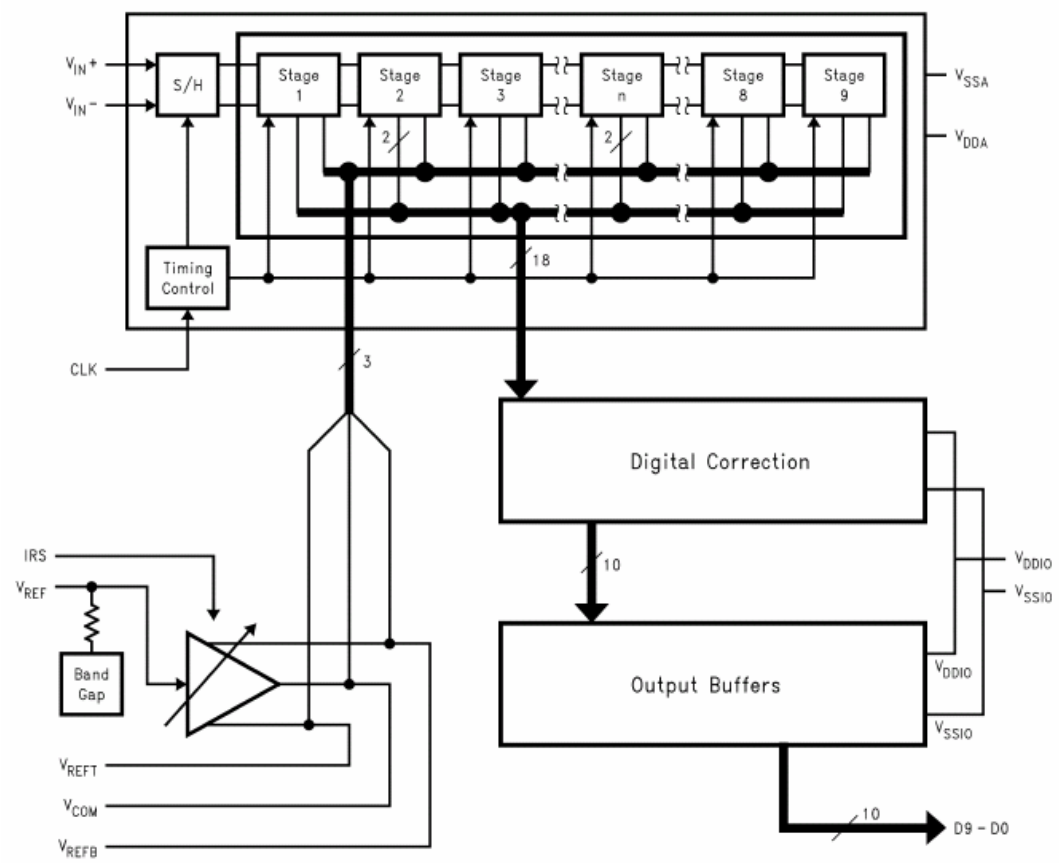
Obrázek 6 *Princip dvoubránové blokové paměti RAM v FPGA čipu Spartan 3L (zdroj dokumentace Xilinx Spartan Family)*

Aby byl čip schopný pracovat bez připojování externí RAM paměti, obsahuje přímo na čipu 18kbit velké paměťové bloky RAM. Těchto dvou-bránových bloků je na použitém čipu Spartan 3S1500L rovných 32, celkem tedy 576kbit RAM paměti. Použití těchto nezávislých dvou-bránových (Dual Port) pamětí má značné výhody při zpracování

popisovaného měřícího zařízení. Znamená to tedy, že může v jednom hodinovém cyklu do paměti psát nebo číst portem A, a zároveň na libovolné jiné místo číst nebo psát portem B viz. ***Obrázek 6***. Umístění bloku RAM pamětí na čipu viz. ***Obrázek 4***.

Vedle paměťových bloků RAM se ve sloupci viz. ***Obrázek 4*** nachází násobící bloky (Multiplier Block), které umožňují násobení dvou 18bitů širokých binárních čísel.

Posledním důležitým blokem na čipu FPGA je blok pro kmitočtovou syntézu DCM (Digital Clock Manager). Na čipu jsou čtyři DCM bloky distribuující hodinový signál po celém čipu. DCM umí dělením a násobením generovat široké spektrum různých výstupních hodinových signálů s různým posunem fází a eliminovat časové zpoždění.



Obrázek 7 *Blokové schéma AD převodníku ADC10065*
(zdroj dokumentace AD převodníku ADC10065)

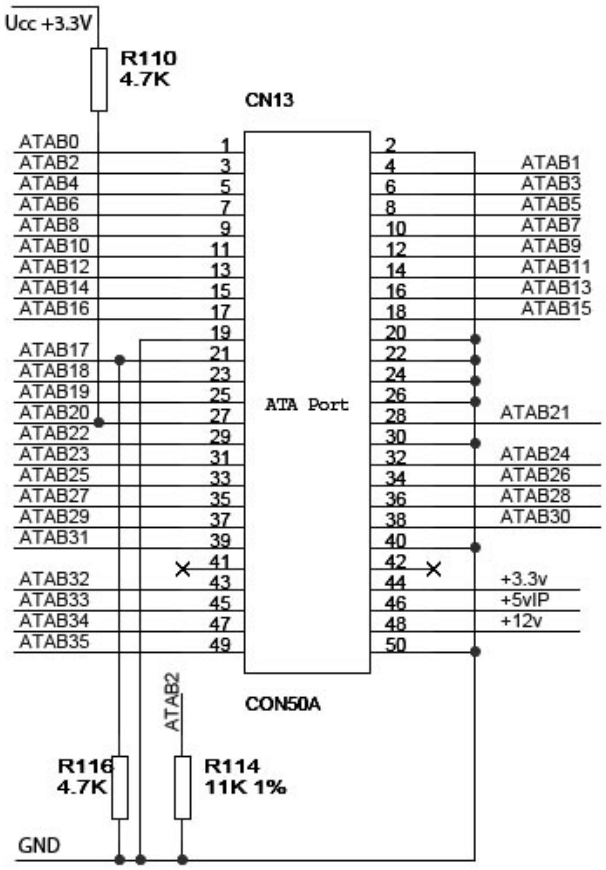
2.1.3. AD převodníky

Vývojová deska RC10 používá dva analogové digitální převodníky typu ADC10065 od firmy National Semiconductor. Převodník má rozlišení 10 bitů s rychlostí převodu 65MSPS (mega ssmples per second) tedy 65 miliónů vzorků za vteřinu. Napájecí napětí převodníku je +3.0V a jeho spotřeba při rychlosti vzorkování 65MHz je 68.4mW. Převodní umožňuje přechod do režimu STANDBY, kdy klesne jeho spotřeba na pouhých 14.1mW.

Převodník používá jednotku S/H (sample-and-hold) a několika úrovňové zřetězení (pipeline) s digitální korekcí přímo na čipu viz. blokové schéma ***Obrázek 7***.

2.1.4. ATA rozhraní

Vývojová deska RC10 má implementované rozhraní kompatibilní se sběrnicí ATA. Toto rozhraní má celkem 50 pinů, z toho je 34 pinů je vstupně-výstupních (I/O) označených jako ATAB0 až ATAB33 viz. ***Obrázek 8***. Dále pak tři napájecí piny (+3.3V, +5V a +12V) a dva hodinové piny označené jako ATAB47 a ATAB49 viz. opět ***Obrázek 8***. Logické vstupně-výstupní piny (I/O) lze použít pro libovolnou funkci, při maximálním přiváděném napětí +3.3V. Pro popisované zařízení data-logger je využito pro měření prvních osm logických vstupů.



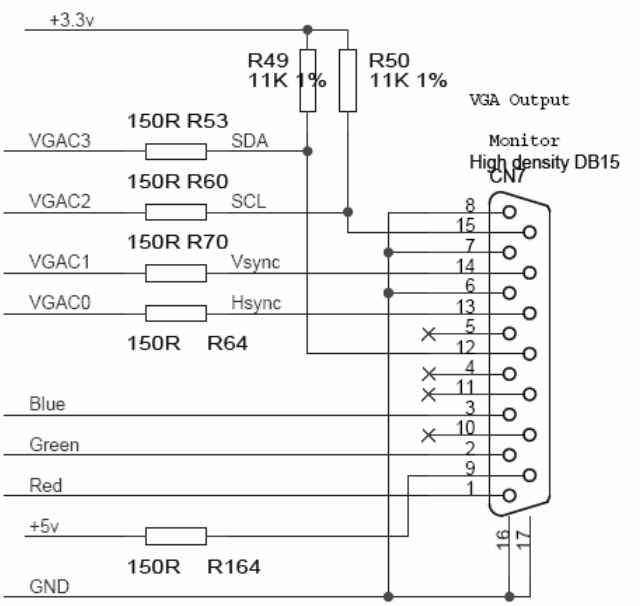
Obrázek 8 *Zapojení vývodů ATA konektoru na vývojové desce RC10*
(zdroj dokumentace vývojové desky RC10)

2.1.5. VGA analogový výstup

Analogový VGA výstup je přímo připojen na vývody FPGA čipu, tedy není použit žádný integrovaný obvod generující rozkladové signály a podobně. FPGA čip musí tedy pro provoz VGA výstupu generovat horizontální a vertikální synchronizační signály (Hsync a

Vsync) a případně signály SDA a SCL pro obousměrnou synchronní komunikaci mezi zařízením a použitým monitorem viz. ***Obrázek 9***.

Signály určující analogovou hodnotu barvy paprsku jsou označeny Red, Green a Blue viz. ***Obrázek 9***. Každý z nich využívá sedm vývodů čipu FPGA. Přes několika stupňový odporový dělič je tato sedmi bitová hodnota převedena na hodnotu amplitudy napětí (jedná se o jednoduchý digitálně analogový převodník). Celá RGB složka je tedy 21 bitů široká a je možné generovat zhruba přes dva miliony barevných úrovní paprsku v monitoru.



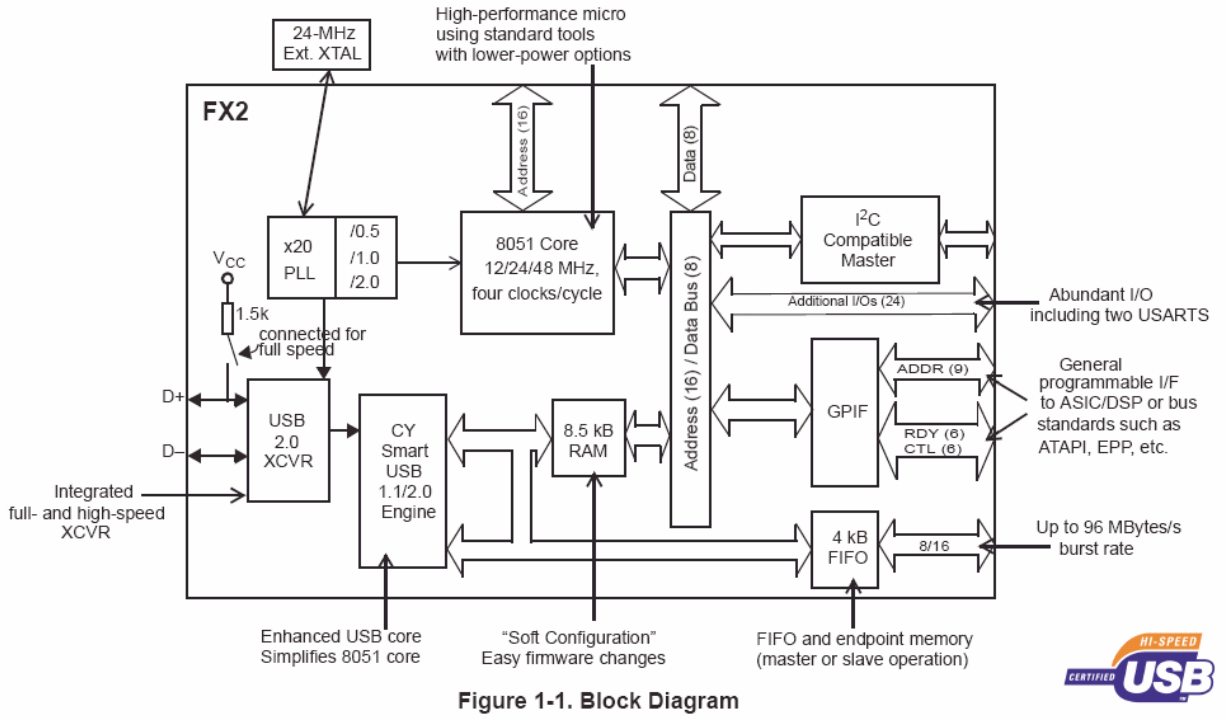
Obrázek 9 *Zapojení analogového výstupu VGA*
(zdroj dokumentace vývojové desky RC10)

2.1.6. USB 2.0 rozhraní

USB rozhraní vývojové desky R10 je poskytováno mikroprocesorem CY7C68013-56pvc FX2 značky Cypress, který je založen na rozšířené architektuře mikroprocesoru řady 8051 viz. ***Obrázek 10*** blok 8051 Core. Mikroprocesor má integrovaný USB 2.0 vysílač-přijímač (transciever) s maximální možnou přenosovou rychlostí 56MB/s viz. ***Obrázek 10*** blok USB2.0 XCVR. Přes USB rozhraní je tedy možné čip konfigurovat souborem typu bit stream nebo z něj aktualní konfiguraci smazat pomocí aplikace FTU3 (File Transfer Utility), která je součástí balíčku PDK (Platform Development Kit) viz. kapitola 3.4. Přez USB rozhraní je vývojová deska RC10 napájena, pokud není přivedeno externí napájení na napájecí konektor vývojové desky.

2.1.7. FLASH paměť

Stejným mikroprocesorem uvedeným v předchozí kapitole 2.1.6 je zprostředkován přístup do paměti FLASH na vývojové desce RC10. Je tedy možno přes rozhraní USB číst a zapisovat do paměti FLASH konfigurační soubory typu bit stream pro FPGA čip přímo aplikací FTU3 (File Transfer Utility) z hostitelského PC viz. kapitola 3.4. FLASH paměť má kapacitu 16MB a mikroprocesor ji implementoval základní souborový systém. Do paměti lze tedy zapsat až 255 různých souborů tak, že jeden záznam může zabírat kapacitu celých 16MB a žádný další už se nevejde. Paměť FLASH je přístupná čipu FPGA pouze přes výše uvedený mikroprocesor.



Obrázek 10 *Blokové schéma USB mikroprocesoru Cypress CY7C68013-54pvc FX2 (zdroj dokumentace USB mikroprocesoru CY7C68013)*

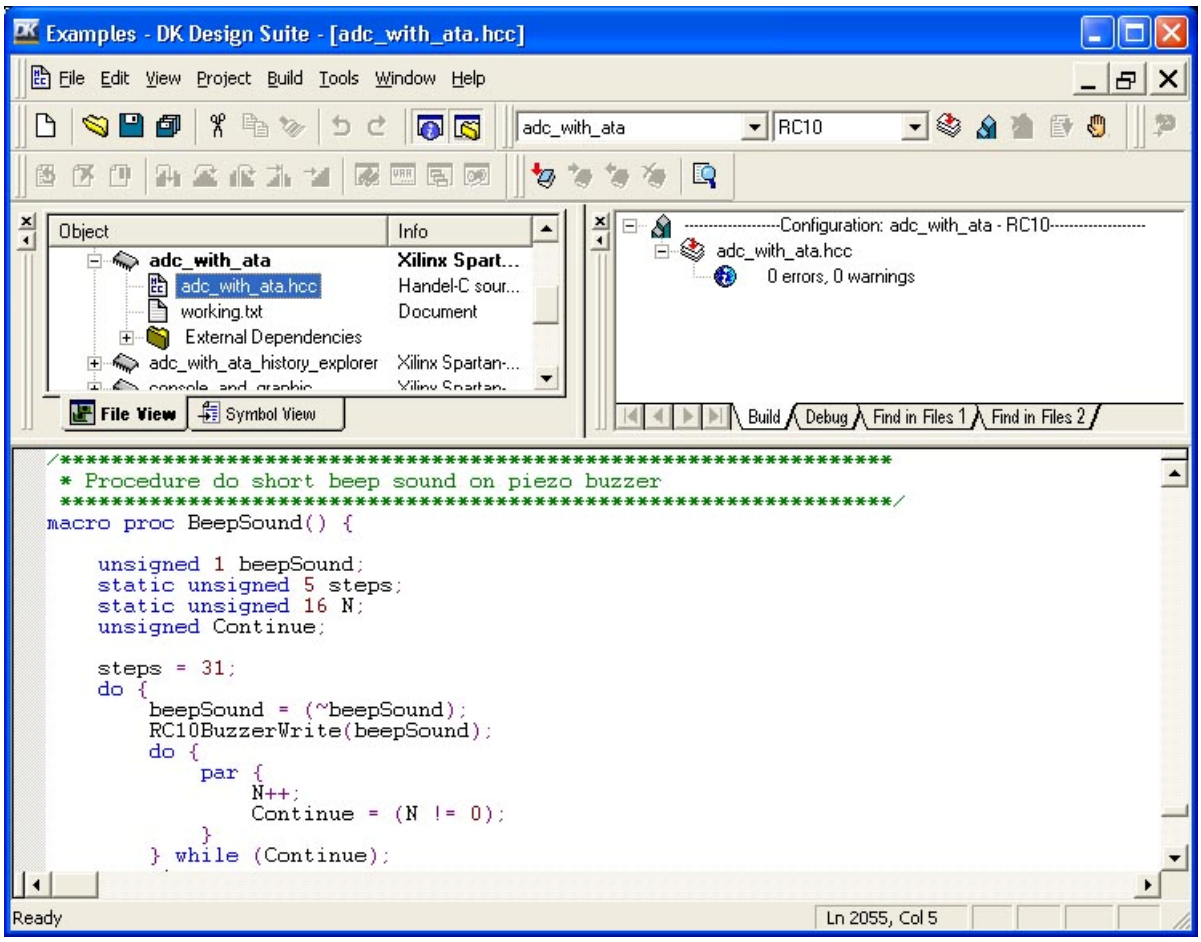
2.1.8. Sedmi segmentový zobrazovače, pěti cestní ovladač, LED a piezo reproduktor

Na vývojové desce jsou umístěny dva sedmi segmentové zobrazovače, na kterých data-logger ukazuje aktuální hodnoty některých parametrů viz. ***Obrázek 2*** (popisek Seven Segment). Dále pěti cestný mikrospínačový ovladač, který je využit pro ovládání data-loggeru viz. ***Obrázek 2*** (popisek Joystick). Osmice LED je využita pro chybové hlášky data-loggeru. Piezo reproduktor je využitý pro zvukovou signalizaci při používání pěti cestného ovladače viz. ***Obrázek 2*** (popisek Piezo Buzzer).

3. POUŽITÝ VÝVOJOVÝ SOFTWARE

3.1. Programovací Jazyk Handel C a vývoj. prostředí DK a PDK

Programové prostředí DK (Development Kit) s nástavbou PDK (Platform Development Kit) pro RC10 a další desky RC umožňuje vytvářet programové kódy v jazyku Handel-C. Ukázka prostředí DK/PDK viz. ***Obrázek 11***. Tento jazyk vychází ze standartu programovacího jazyka ANSI-C a je tedy podobný vyšším programovacím jazykům. Vývojové prostředí DK/PDK potřebuje pro svůj chod mít na PC nainstalované vývojové prostředí Microsoft Visual Studio .NET, protože využívá jeho C překladač a knihovny.



Obrázek 11 *Vývojový software DK / PDK a programovací jazyk Handel-C*

Na rozdíl od klasických programovacích jazyků se v jazyku Handel-C navrhuje digitální logika tedy hardware. PDK poskytuje tři funkční vrstvy pro návrh a simulace logiky napsané v jazyce Handel-C. Jsou to vrstvy DSM (Data Stream Manager), PAL (Platform Abstract Layer) a PSL (Platform Support Library). Pro návrh data-loggeru je využito vrstev PAL a PSL, které jsou popsány v následující kapitole 3.2.

Kompletním překladem (kompilací) zdrojového kódu napsaného v jazyce Handel-C je vytvořena logika resp. hardware (fyzické zapojení logických obvodů) pro daný čip FPGA. Toto vygenerované logické zapojení je uloženo do konfiguračního souboru typu BIT STREAM, kterým je možné daný čip FPGA nakonfigurovat. Kompilace provádí mnoho kroků (vytváření logiky ze základních logických prvků, mapování, drátování, generování konfiguračního souboru) a její časová náročnost roste se složitostí vytvářené logiky.

3.2. Vrstvy PAL a PSL programového prostředí PDK

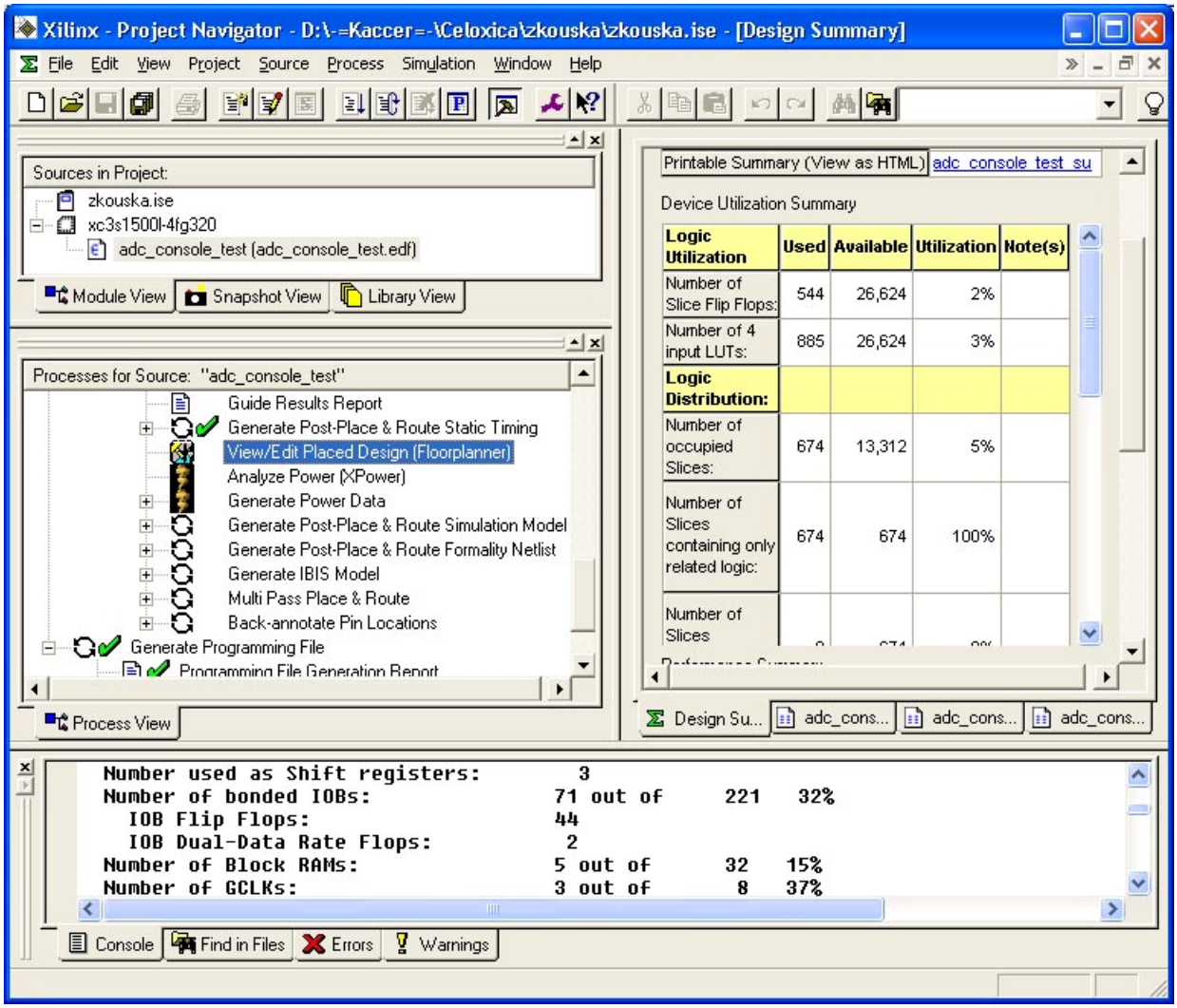
Vrstva PAL (Platform Abstartc Layer) obsahuje mnoho důležitých a potřebných konstrukčních prvků, které jsou použity pro vývoj zařízení data-logger. Vrstva PAL umožňuje používání základních společných periferií desek typu RC. Výhoda použití vrstvy PAL a jejích knihovny je v tom, že vytvořený hardware pro čip FPGA je možno simulovat na PC. Simulátor na PC obsahuje modely desek RC a jejich základních periferií. Simulace je velmi zpomalená, protože CPU osobního počítače nedokáže simulovat chování čipu FPGA v takové rychlosti jako sám čip. Diky tomu je zde možnost ladění vytvářeného hardware. Další výhodou vrstvy PAL je přenositelnost aplikací (programových kódů napsaných v jazyku Handel-C) mezi různými typy desek RC od firmy Celoxica, pokud je deska podporována vrstvou PAL.

Vrstva PAL je však omezená (právě kvůli zmíněné kompatibilitě desek RC) a nepodporuje všechny periferie vývojové desky RC10. Proto bylo při návrhu data-loggeru potřeba využít vrstvu PSL (Platform Support Library). Ta podporuje konkrétní typ desky a umožňuje používat její alternativní periferie.

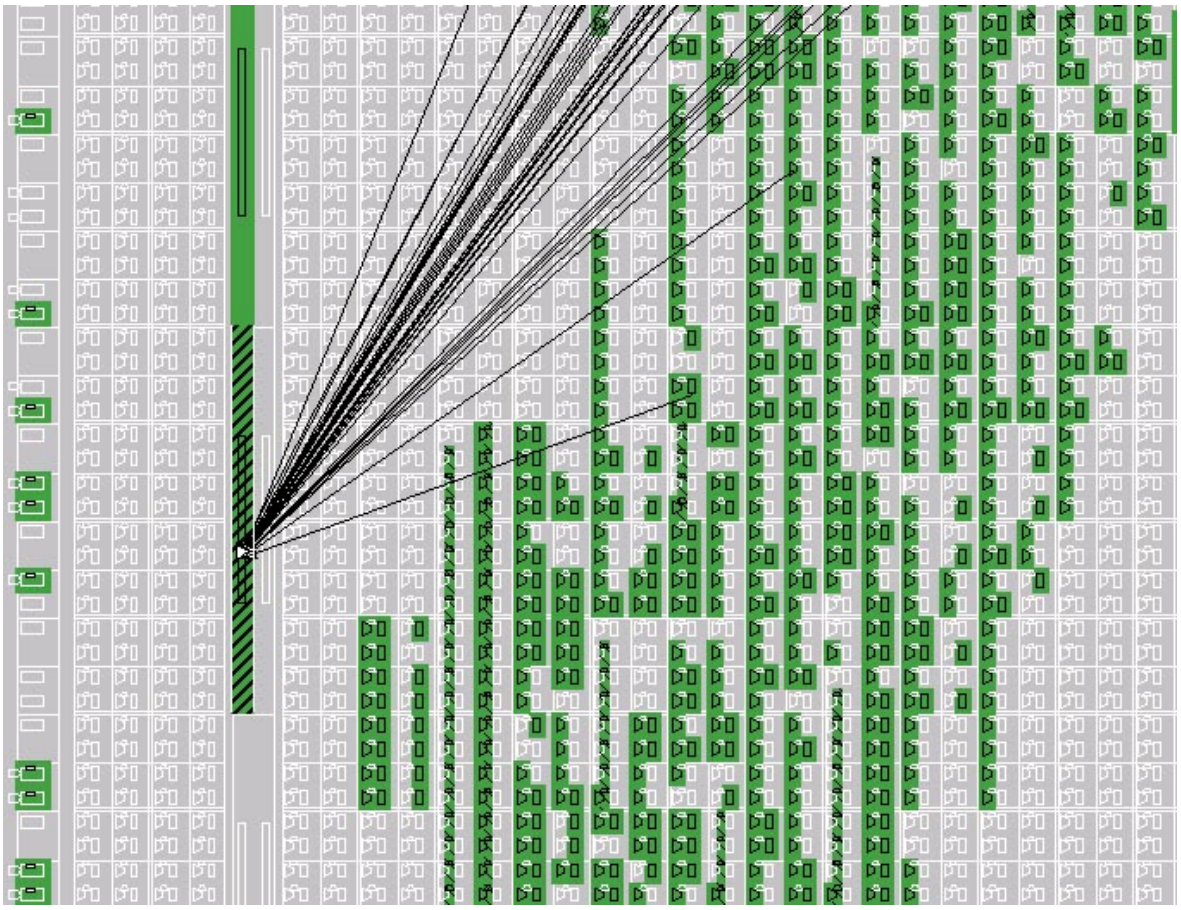
3.3. Prostředí Xilinx ISE 7.1

Prostředí Xilinx ISE 7.1 představuje nástroj pro vytvoření konfiguračního souboru (Bit Stream) čipu FPGA z popisu hardware HDL, EDIF a dalších. Prostředí PDK umožňuje provést kompletní kompilaci zdrojového kódu i bez použití tohoto nástroje. Xilinx ISE však musí být na PC nainstalován, protože PDK využívá jeho součásti. Pokud však pomocí prostředí PDK vytvoříme jen popis hardware typu EDIF ze zdrojového kódu, můžeme využít Xilinx ISE k vytvoření konfiguračního souboru čipu FPGA. Umožní nám tak definovat mnoho parametrů pro rozmísťování hardware na čipu, a dostáváme detailnější informace o průběhu mapování a umisťování logiky, drátování, zpoždění signálů a využitých prostředků čipu FPGA. Ukázka prostředí Xilinx ISE 7.1 viz. ***Obrázek 12***.

Xilinx ISE 7.1 umožňuje zjednodušený náhled na rozmístění logiky (konfigurace) přímo na čipu FPGA viz. ***Obrázek 13***. Na obrázku jsou znázorněny konfigurovatelné logické bloky CLB a na levém okraji vstupně výstupní bloky IOB. Svislý obdélník představuje využitou blokovou paměť RAM. Černé vodiče naznačují propojení označeného bloku s jinými bloky.



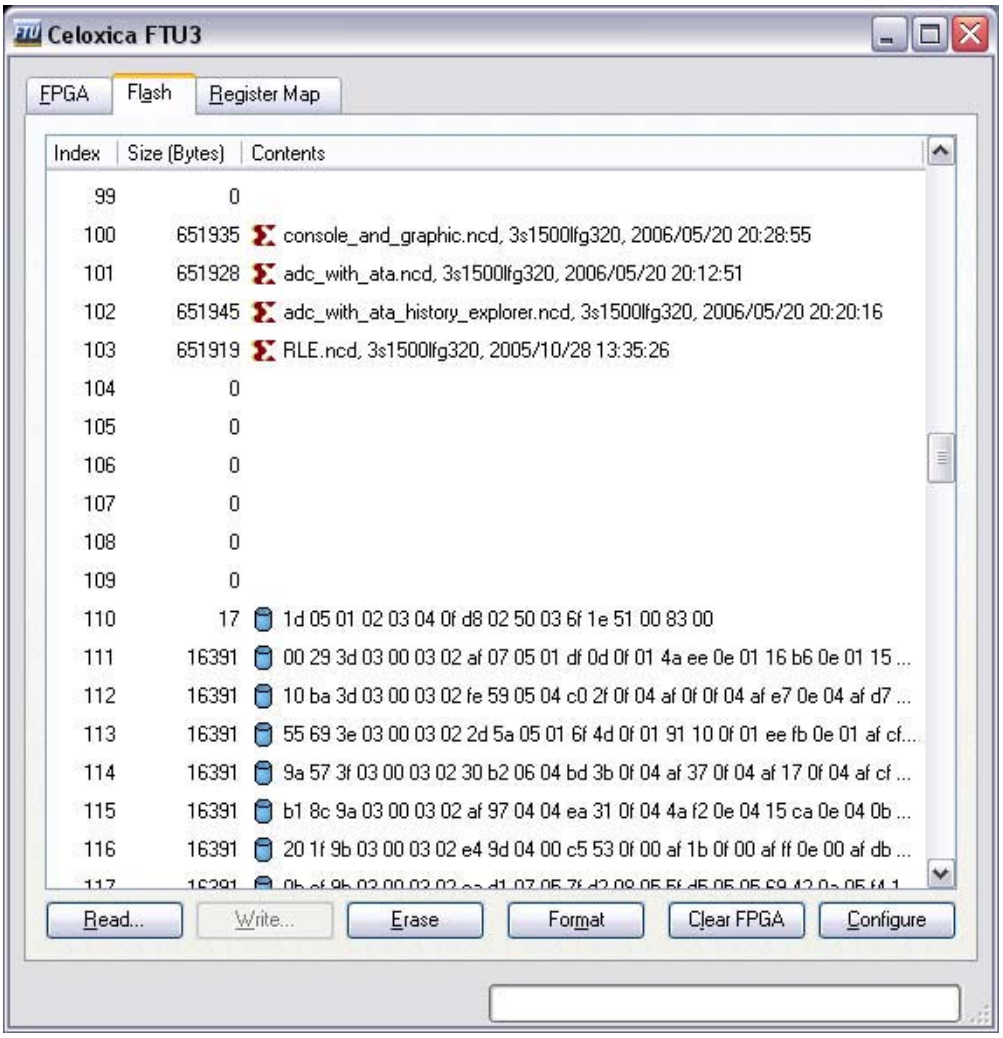
Obrázek 12 *Prostředí Xilinx ISE 7.1 pro generování konfiguračního souboru pro čip FPGA*



Obrázek 13 *Ukázka rozmístění logických bloků konfigurace na čipu FPGA*

3.4. Utilita FTU3

FTU3 (File Transfer Utility) program umožňuje ovládání vývojových desky typu RC a je součástí balíčku PDK. Veškerá komunikace s vývojovou deskou probíhá přes komunikační rozhraní USB. V následujícím popisu budou popsány funkce programu FTU3, které byly využity při vývoji data-loggeru na desce RC10.



Obrázek 14 *FTU3 - File Transfer Utility*
(obrázek spuštěné aplikace)

FTU3 umožňuje konfigurovat čip FPGA souborem s příponou *.bit* (bit stream), který dostaneme kompilací projektu ve vývojovém prostředí PDK. Opakem nakonfigurování FPGA čipu je jeho vyčištění (clear).

Nejdůležitější funkce programu FTU3 je procházení a práce s pamětí FLASH na vývojové desce RC10. Jak již bylo popsáno v kapitole 2.1.7, paměť FLASH obsahuje 255 souborů (files), které jsou zobrazeny v programu FTU3 jako řádky, kde je vidět jejich obsah. Pokud soubor obsahuje záznam typu *bit stream* (konfigurační soubor pro FPGA) je na řádku uvedeno jeho jméno. Pokud soubor obsahuje jiný záznam, je v řádku uvedeno několik prvních bajtů záznamu. U každého řádku, tedy souboru je uvedena celková délka záznamu v bajtech. Pohled na záložku s obsahem paměti FLASH viz. ***Obrázek 14***.

Pod libovolný soubor paměti FLASH je možno uživatelem uložit konfigurační souboru typu *bit stream* nebo obsah souboru vymazat. Celou paměť FLASH lze i naformátovat do výchozího stavu.

3.5. Programovací jazyk C++

Vývojové prostředí Microsoft Visual Studio .NET důležité pro chod vývojového prostředí DK/PDK nabízí vývojové prostředky pro vytváření aplikací v programovacím jazyce standartu ANSI-C. Aplikace uvedená v kapitole 5.1, byla vytvořena v tomto vývojovém prostředí s použitím standardních knihoven a knihovny podporující komunikaci s deskou RC10, která je součástí vývojového prostředí PDK.

3.6. Programovací prostředí Matlab

Programovací prostředí Matlab bylo využito pro tvorbu skriptu (funkce), který načte soubor vytvořený vyvinutou aplikací popsanou v kapitole 5.1. Skript vykreslí grafy průběhů naměřených signálů z hodnot uložených v tomto souboru. Prostředí Matlab bylo pro tuto funkci zvoleno, protože má velmi bohatou výbavu na kreslení grafů a případné další úpravy naměřených dat.

4. NÁVRH HARDWARE DATA-LOGGERU PRO ČIP FPGA

4.1. Struktura celého zařízení

Zařízení data-logger je realizované ze tří hardwarových částí, tedy ze tří konfiguračních souborů (Bit Stream) uložených v paměti FLASH přímo na vývojové desce RC10.

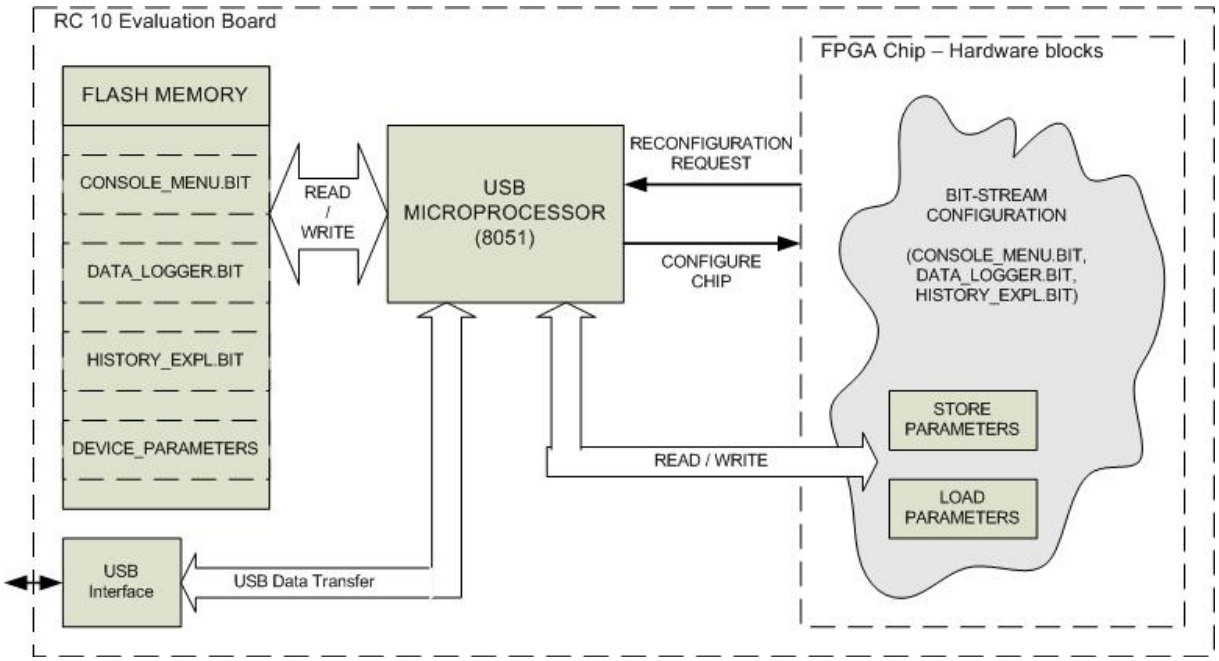
První částí zařízení je konzolové menu viz. ***Obrázek 16***, přestavující textovou obrazovku, ve kterém každý řádek představuje jeden parametr zařízení. Parametry lze měnit a tím přednastavit zařízení do požadovaného stavu. Detailnější popis je uveden v kapitole 4.2.

Druhou částí je vlastní měřící hardware data-loggeru, který se nastartuje v konfiguraci zvolené v konzolovém menu. Na obrazovce jsou zobrazeny dva analogové průběhy z analogově digitálních převodníků a osm průběhů snímaných z logických vstupů ATA rozhraní viz. ***Obrázek 18***. Dále jsou zobrazeno interaktivní menu, ve kterém se mohou měnit základní parametry snímaných průběhů. Interaktivní menu poskytuje i informační políčka jako aktuální čas, a aktuální událost. Měřící hardware je podrobněji popsán v kapitole 4.3.

Poslední, třetí částí zařízení je hardware umožňující prohlížení uložených průběhů událostí z paměti FLASH viz kapitola 4.4. Tato část je odvozená z měřícího hardware data-loggeru. Zobrazené průběhy jedné události jsou statické, lze je všechny snadno procházet a je použito méně parametrů v interaktivním menu než u měřícího hardware data-loggeru.

Čip FPGA je možné za chodu rekonfigurovat libovolným konfiguračním souborem uloženým v paměti FLASH, protože vývojová deska obsahuje Mikroprocesor CY7C68013-56pvc FX2 značky Cypress blíže popsaný v kapitole 2.1.6 a 2.1.7. Proces rekonfigurace čipu

FPGA je iniciován z aktuální konfigurace čipu. Běžící HW vyšle příkaz mikroprocesoru k přečtení nového konfiguračního souboru z paměti FLASH na požadované pozici, předá řízení mikroprocesoru a čeká na rekonfiguraci. Procesor čip FPGA požadovaným souborem nakonfiguruje viz. ***Obrázek 15***. Nový hardware FPGA se automaticky nastartuje a obnoví vedle dalších funkcí i komunikaci s procesorem. Aby bylo možné si pamatovat zvolené nastavení parametrů mezi různými hardwarovými konfiguracemi, musejí se parametry uložit a následně přečíst z paměti FLASH. To představují hardwarové bloky *Store Parameters* a *Load Parameters* viz. ***Obrázek 15***. Nastavení zvolené uživatelem v konzolovém menu se uloží do paměti a měřící hardware si ho po rekonfiguraci přečte. Zápis a čtení dat z paměti FLASH se provádí též přes mikroprocesor Cypress po jednotlivých bajtech do příslušného souboru.



Obrázek 15 *Princip rekonfigurace čipu FPGA*

4.2. Konzolové menu zařízení

4.2.1. Výchozí příklady

Při návrhu hardware realizujícího konzolové menu jsem vycházel z ukázkových příkladů vývojového prostředí PDK, které se nachází ve složce *…Pdk/Examples/Pal/* a *...Pdk/Examples/RC10/Psl/*. První ukázkový příklad nazvaný *Console* je hardware realizující textovou obrazovku přes rozhraní VGA, na kterou se vypisují texty a čísla počítaných posloupností. Druhým příkladem nazvaným *Expansion* je hardware, který na textovou obrazovku přes rozhraní VGA vypisuje aktuální logický stav rozhraní ATA. Každý logický vstup/výstup je možné nastavit do stavu logická 1 nebo 0 pomocí pěti cestného ovladače

(joystick). Příklady byly využity pro pochopení funkčnosti textové obrazovky (Console) a užitečných knihovním procedur.

4.2.2. Hardware konzolového menu

Vyvinutý hardware realizující konzolové menu generuje textovou obrazovku v rozlišení 800x600 bodů, na kterou je možné zobrazovat znaky a čísla viz. ***Obrázek 16***. Při tvorbě tohoto hardware byly využity konstrukční prvky textové konzole nabízené ve vrstvě (knihovně) PAL. Funkce a procedury využité pro návrh konzolového menu:

- `macro proc PalConsoleRun (ConsolePtr, Font, VideoOutHandleCT, ClockRate);`
  Hardware realizující textovou obrazovku na výstup VGA, do které lze vkládat znaky a čísla.
- `macro proc PalConsoleClear (ConsolePtr);`
  Smaže všechny znaky z textové obrazovky.
- `macro proc PalConsolePutChar (ConsolePtr, Char);`
- `macro proc PalConsolePutString (ConsolePtr, String);`
- `void PalConsolePutUInt (ConsolePtr *ConsolePtr, unsigned 32 Value);`
  Tyto dvě procedury a jedna funkce umožňují tisknou znaky (Char), texty (String) a čísla (Integer) na obrazovku.

Pomocí těchto funkcí a procedur je vytvořen program (hardware), který vypisuje parametry přístroje na obrazovku připojenou na výstup VGA. Protože některé z použitých funkcí a procedur mají vetší zpoždění (latency) vygenerované logiky, není možné je používat v programu příliš často. Při velkých zpožděních nelze dokončit kompilaci projektu a pokud ano, nemusí být výsledný hardware funkční. Tento problém se týká hlavně procedury `PalConsolePutString(ConsolePtr, String);`. Text vypisovaný na obrazovku není možné vypisovat po samostatných řádcích, a bylo tedy nutné jej rozdělit na max. 10 samostatných řetězců, které jsou vypisovány postupně za sebou. Řetězec znaků je reprezentován pamětí RAM organizované po bajtech (char = 8 bitů). Při změně parametru (slova, hodnoty) je přepsána požadovaná část RAM potřebnými znaky. Tyto řetězce jsou v nekonečné smyčce tisknuty na obrazovku. FPGA čip běží s touto konfigurací na frekvenci 40MHz. Obrazovka je tedy přepisována dostatečně rychle, aby se uživateli zdál text statický, a změny hodnot skokové.

4.2.3. Princip ovládání a jeho funkce

Konzolové menu se ovládá pomocí pěti směrového ovladače (joystick). Pohybem nahoru a dolů se vybírá příslušný řádek v menu, který přestavuje jeden parametr zařízení a je na začátku řádku označen hvězdičkou. Pohybem ovladače doleva a do prava se vybírá hodnota parametru. Uložení aktuálně nastavených parametrů může uživatel provést stisknutím ovladače (fire), nebo se uložení provede automaticky při přechodu na jinou hardwarovou část zařízení pomocí položky v menu.

V následujícím odstavci budou uvedeny parametry a volby konzolového menu tak, jak se nachází na obrazovce připojené k desce RC10 a čipu nakonfigurovaném tímto hardwarem viz. ***Obrázek 16***.

Parametry umožňující nastavení obou analogových vstupů a logických vstupů ATA rozhraní viz. ***Tabulka 1*** a ***Obrázek 16***.

Název parametru	**Hodnoty parametru**	**Význam parametru**
ADC0	[ON, OFF]	Zapnutí nebo vypnutí zobrazení průběhu v měřícím hardware
Amplitude	[2Vpp nelze měnit]	Rozsah analogového vstupu ADC0 (analog digital converter)
TimeBase	[0.5, 1, 5, 10, 20…90, 100 µs / Div]	Časová základna ADC0, ze které je odvozena rychlost vzorkování signálu
ADC1	[ON, OFF]	Zapnutí nebo vypnutí zobrazení průběhu v měřícím hardware
Amplitude	[2Vpp nelze měnit]	Rozsah analogového vstupu ADC1 (analog digital converter)
TimeBase	[0.5, 1, 5, 10, 20…90, 100 µs / Div]	Časová základna ADC1, ze které je odvozena rychlost vzorkování signálu
ATA	[ON, OFF]	Zapnutí nebo vypnutí zobrazení průběhu v měřícím hardware
TimeBase	[0.5, 1, 5, 10, 20…90, 100 µs / Div]	Časová základna ATA rozhraní, ze které je odvozena rychlost vzorkování signálu
Grid 1	[ON, OFF]	Zapnutí a vypnutí hlavní měřící mřížky (32 pixelů / Div)
Grid 2	[ON, OFF]	Zapnutí a vypnutí jemnější měřící mřížky (16 pixelů / Div)

Tabulka 1 *Parametry analogových (ADC0, ADC1) a digitálních (ATA) vstupů v konzolovém menu data-loggeru*

Další skupinu tvoří parametry pro nastavení startovacích událostí (EVENTS), při kterých jsou nabírané vzorky ukládané do paměti FLASH pro další zpracování. Význam jednotlivých parametrů viz. ***Tabulka 2*** a ***Obrázek 16***.

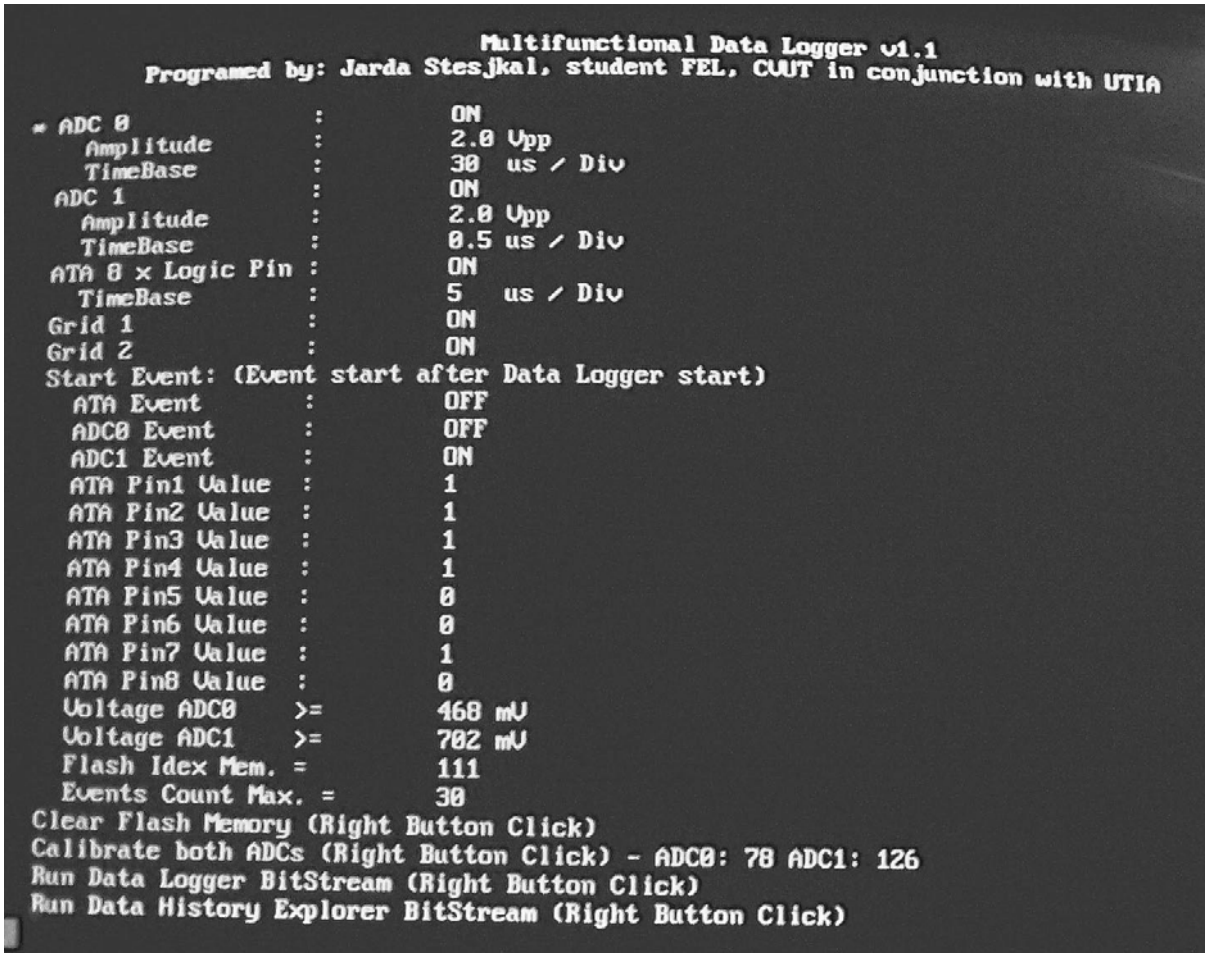
Název parametru	**Hodnoty parametru**	**Význam parametru**
ATA Event	[ON, OFF]	Zapnutí nebo vypnutí startovací události od ATA rozhraní při splnění logických úrovních na jednotlivých vstupech
ADC0 Event	[ON, OFF]	Zapnutí nebo vypnutí startovací události od analogového vstupu ADC0 při překročení nastavené napěťové úrovně
ADC1 Event	[ON, OFF]	Zapnutí nebo vypnutí startovací události od analogového vstupu ADC1 při překročení nastavené napěťové úrovně
ATA Pin1...Pin7 Value	[0, 1]	Logické úrovně jednotlivých vstupů ATA rozhraní při kterých se spouští událostní procedura
Voltage ADC0 ≥	[±1V po kroku 7mV]	Překročení této hodnoty napětí se spustí událostní procedura
Voltage ADC1 ≥	[±1V po kroku 7mV]	Překročení této hodnoty napětí se spustí událostní procedura
Flash Index Mem.	[111 … 249]	Adresa v paměti FLASH od které dochází k zápisu naměřených průběhů jednotlivých událostí
Events Count Max	[0 … 138]	Maximální počet ukládaných událostí od počáteční adresy v paměti FLASH

Tabulka 2 *Parametry pro nastavení startovací události v konzolovém menu data-loggeru*

Poslední čtyři položky (řádky) v konzolovém menu slouží pro ovládání zařízení a snadnému přechodu (rekonfiguraci) na další hardware tvořící data-logger. Podrobný popis viz. ***Tabulka 3***.

Název ovládacího prvku	Význam prvku
Clar Flash Memory	Smaže oblast paměti FLASH v rozmezí adres 111 a 249 určenou pro ukládání událostí
Calibrate both ADCs	Změří nulovou odchylku obou převodníků, která je uložena do paměti FLASH a následně používána pro kalibraci v měřícím hardware data-loggeru. Při použití kalibrace musí být oba analogové vstupy odpojeny (nejlépe připojeny na zem GND)
Run Data Logger BitStream	Nakonfiguruje čip na měřící hardware zařízení (přechod do měřící části zařízení)
Run Data History Explorer	Nakonfiguruje čip na hardware umožňující prohlížení uložených průběhů událostí

Tabulka 3 *Ovládací prvky zařízení v konzolovém menu data-loggeru*



Obrázek 16 *Snímek obrazovky připojené k vývojové desce RC10 při nakonfigurovaném čipu hardwarem konzolového menu*

4.2.4. Čtení a ukládání parametrů zařízení

Parametry nastavené v hardware konzolového menu musí být možné uložit do paměti FLASH a v dalších hardwarech je opět přečíst, aby hardware mohl v požadovaném nastavení pracovat. Pro přečtení parametrů slouží hardwarový blok označený jako *LOAD PARAMETERS* viz. ***Obrázek 15***, který přečte hodnoty parametrů z paměti FLASH a předá je do správných proměnných (registrů). Naopak pro uložení aktuálních hodnot parametrů slouží blok *STORE PARAMETERS* viz. ***Obrázek 15***. Uvedené bloky využívají následující procedury pro práci s pamětí FLASH:

- `macro proc RC10FlashAppendBegin (Index, Length);`
  Tato procedura zpřístupní zápis do libovolného souboru paměti FLASH s adresou uvedenou v proměnné *Index* a délkou zápisu určující proměnnou *Length*.
- `macro proc RC10FlashAppend (Value);`
  Po zpřístupnění souboru FLASH paměti procedurou `RC10FlashAppendBegin (Index, Length)` se musí zapsat do paměti přesný počet bajtů určený hodnotou *Length*. Hodnota aktuálně zapisovaného bajtu je v proměnné *Value*.
- `macro proc RC10FlashReadBegin (Index, Offset, Length);`
  Procedura zpřístupní soubor na adrese určené proměnnou *Index*. Bude se očekávat přečtení přesného počtu bajtů určeného proměnnou *Length*.
- `macro proc RC10FlashRead (ValuePtr);`
  Touto procedurou se čtou jednotlivé bajty do proměnné *ValuePtr*.

Parametry jsou v paměti FLASH uloženy pod souborem s adresou 110. Soubor v paměti je orientován po jednotlivých bajtech (8 bitů), proto se hodnoty parametrů do paměti rozloží. Rozkládání a skládání vhodných bitů hodnot parametrů se provádí jednoduchými logickými funkcemi. Zvolená struktura je naznačena viz. ***Obrázek 17***, kde každý řádek představuje jeden bajt paměti od nejvyššího sedmého bitu po nejnižší nultý bit. Vpravo je uvedeno k jakému účelu parametr slouží. Celkem zabírají hodnoty parametrů v paměti pouhých 16B.

	BITS 7 - 0								Parameters For:
Byte 0	/	Clock Rate Index				Amplitude Index		ON/ OFF	ADC 0
Byte 1	/	Clock Rate Index				Amplitude Index		ON/ OFF	ADC 1
Byte 2	/	/	/	/	/	/	/	ON/ OFF	ATA
Byte 3	/	/	/	/	Clock Rate Index				ATA
Byte 4	/	/	/	/	/	/	ON/ OFF	ON/ OFF	GRIDS
Byte 5	/	/	/	/	/	ADC 1	ADC 0	ATA	EVENTS ENABLE
Byte 6	7	6	5	4	3	2	1	0	ATA EVENT VALUE
Byte 7	7	6	5	4	3	2	1	0	ADC 0 EVENT VALUE
Byte 8	/	/	/	/	/	/	9	8	
Byte 9	7	6	5	4	3	2	1	0	ADC 1 EVENT VALUE
Byte 10	/	/	/	/	/	/	9	8	
Byte 11	7	6	5	4	3	2	1	0	FLASH INDEX
Byte 12	7	6	5	4	3	2	1	0	EVENTS COUNT
Byte 13	7	6	5	4	3	2	1	0	ADC 0 CALIBRATION VALUE
Byte 14	/	/	/	/	/	/	9	8	
Byte 15	7	6	5	4	3	2	1	0	ADC 1 CALIBRATION VALUE
Byte 16	/	/	/	/	/	/	9	8	

Obrázek 17 *Struktura uložených hodnotu parametrů v paměti FLASH orientované po bajtech*

4.3. Měřící hardware zařízení

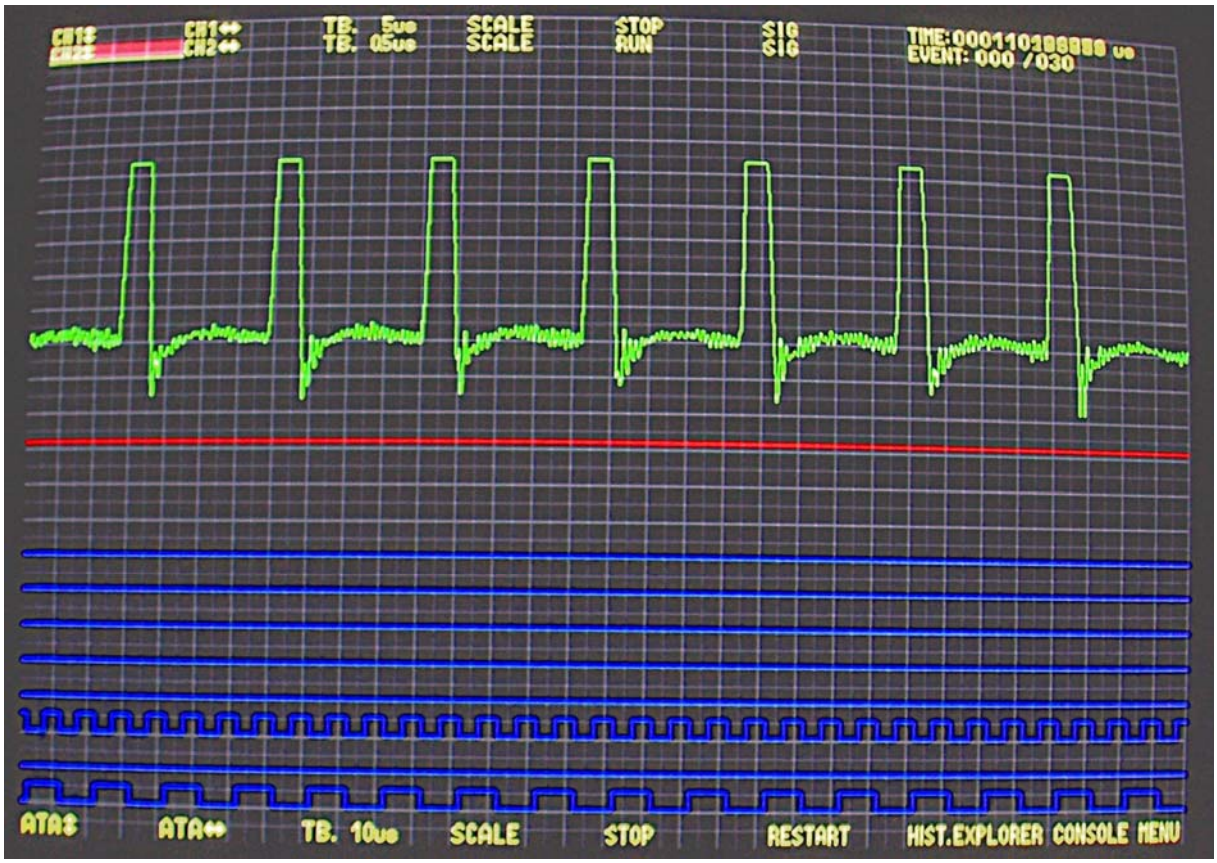
4.3.1. Výchozí příklad

Výchozím příkladem pro vytvoření měřícího hardware data-loggeru byl příklad nazvaný *ADC*, který je součástí vývojového prostředí PDK a nachází se ve složce *...Pdk/Examples/RC10/Psl/*. Příklad ukazuje použití analogových vstupů s analogově digitálními převodníky (ADC - Analog digital converter) a princip vytvoření grafického obrazu na výstupu VGA. Tyto principy jsou použity podobně i ve vyvinutém data-loggeru,

jsou značně rozšířeny a navíc s mnoha implementacemi. Pro snadné ovládání zařízení muselo být vyvinuto a implementováno interaktivní uživatelské menu, ovládané pomocí pěti směrového ovladače. Aby bylo zařízení schopné archivovat naměřená data, musel být implementován hardware ukládající vzorky průběhů do paměti FLASH.

4.3.2. Měřící hardware data-loggeru

Obrazovka připojená k VGA výstupu vývojové desky RC10 při konfiguraci měřícího hardware je zachycena na snímku viz. ***Obrázek 18***. Řádky se žlutými popisky představují interaktivní uživatelské menu pro ovládání zařízení. V pravém horním rohu je informace o aktuálním čase, který je spuštěn v okamžiku nakonfigurování FPGA čipu. Čas běží pokud bylo povoleno sledování události na některém ze vstupu, v opačném případě je čas zastaven a zařízení se chová jako základní osciloskop, kterým je možné pozorovat průběhy vstupních signálů. V režimu sledování událostí se vzorkují vstupní signály maximální možnou vzorkovací rychlostí a proto mohou zobrazované signály být na obrazovce poněkud rozkmitané, jako by nefungovala synchronizace.



Obrázek 18 *Snímek obrazovky připojené k vývojové desce RC10 při nakonfigurovaném čipu měřícím hardwarem*

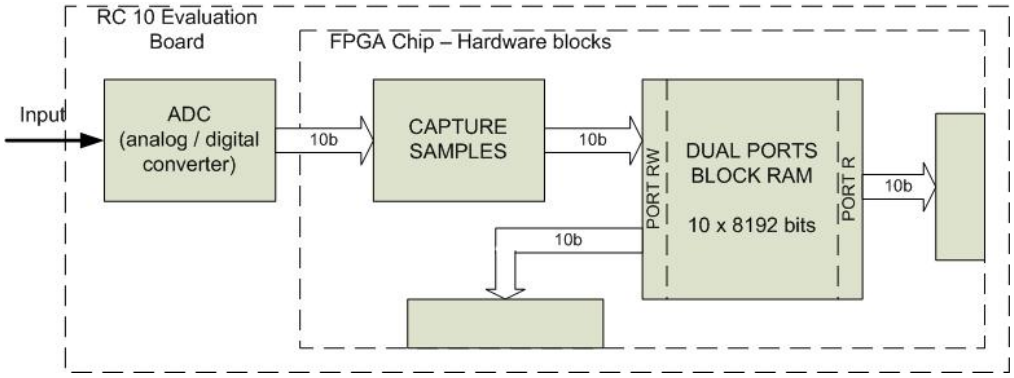
Pod aktuálním časem je informace o aktuální události. První číslo před lomítkem označuje počet uložených událostí v paměti FLASH a druhá číslice informuje o maximu možných událostí. Po uložení všech možných událostí je zařízení přepnuto do režimu osciloskopu.

Zelený signál viz. ***Obrázek 18*** je signál snímaný první analogově digitálním převodníkem. Pro ovládání tohoto vstupu slouží první řádek menu shora. Pro druhý vstup, červený signál, slouží druhý řádek menu. Modré průběhy jsou signály snímané z osmice vstupu ATA rozraní, kterému patří spodní řádek menu.

Poslední tři položky spodního menu slouží k restartování měřícího hardware a k snadnému přechodu (rekonfiguraci) na konzolové menu, nebo na hardware umožňující prohlížení uložených událostí.

4.3.3. Vzorkování vstupních signálů

První z dvou režimů měřícího hardware je již zmíněný řežím základního osciloskopu. Vstupní analogové signály jsou vzorkovány plnou vzorkovací rychlostí tj. 64MHz. Blok nazvaný *CAPTURE SAMPLES* viz. ***Obrázek 19*** vybírá 10 bitovou hodnotu aktuálního vzorku z AD převodníku a portem označeným *RW* ji ukládá do blokové dvou bránové paměti RAM přímo na čipu. Druhým portem *R* jsou hodnoty vzorků čteny a zpracovány blokem vytvářejícím grafické rozhraní, který bude popsán v kapitole 4.3.5. Vzorek je 10 bitů široký a nulová úroveň je na prostředku rozsahu 10 bitového čísla tj. hodnotě 512.



Obrázek 19 *Blokové schéma vzorkování analogového signálu AD převodníkem*

Blok snímající vstupní signál je opatřen automatickou synchronizací aby byly vstupní signály na obrazovce pozorovatelné. Signál je synchronizován tehdy, pokud prochází nulovou úrovní. Pokud k tomu u snímaného signálu nedojde, je po velmi krátkém časové intervalu (Timeout) za synchronizováno automaticky a následně nabráno 8192 vzorků do paměti RAM.

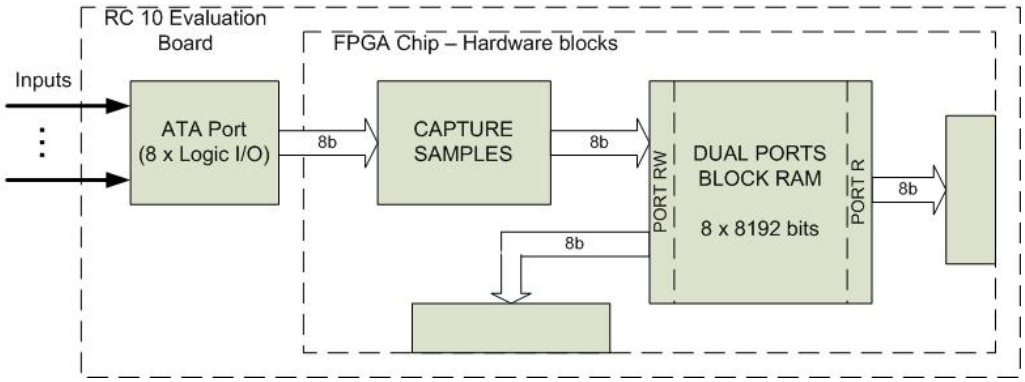
Blok obsahuje i korekci chyby nulové úrovně, která nemusí být u AD převodníku přesně na hodnotě 512. Kalibrační hodnota je získána z FLASH paměti, kam se uloží z hardware konzolového menu, při použití kalibrace AD převodníků.

Aby bylo možné nastavovat časovou základnu měřených signálů, musí být možné měnit vzorkovací rychlost. Protože však není možné měnit kmitočet čipu a tedy i AD převodníků, je do paměti RAM zapisován jen vhodný vzorek signálu. Pokud budeme zapisovat do paměti každý druhý vzorek signálu, bude vzorkovací rychlost poloviční. Tímto způsobem je vytvořená časová základna pro měřené průběhy signálů.

Způsob vzorkování osmice vstupů ATA rozhraní je obdobná jako u analogových vstupů viz. ***Obrázek 20***. Zapisovaný vzorek do paměti RAM je široký 8 bitů, kde každý bit vyjadřuje logickou úroveň konkrétního vstupu.

Liší se zde pouze synchronizace vzorkování. Vzorky jsou odebrány vždy, když dojde ke změně logické úrovně alespoň na jednom z osmi vstupů. Pokud tak nenastane, pak je nevzorkováno po uplynutí velmi krátkého intervalu, jako tomu bylo u synchronizace analogových vstupů.

Procedury jsou v kódu programu pojmenovány *macro proc CaptureDataFromADC (RAMPtr, ADCRun, ADCRead, ClockRate…);* pro vzorkování analogových vstupů a *macro proc CaptureDataFromATA (RAMPtr, ClockRate, Stop…);* pro vzorkování digitálních vstupů ATA rozhraní.



Obrázek 20 *Blokové schéma vzorkování logických signálů ATA rozhraní*

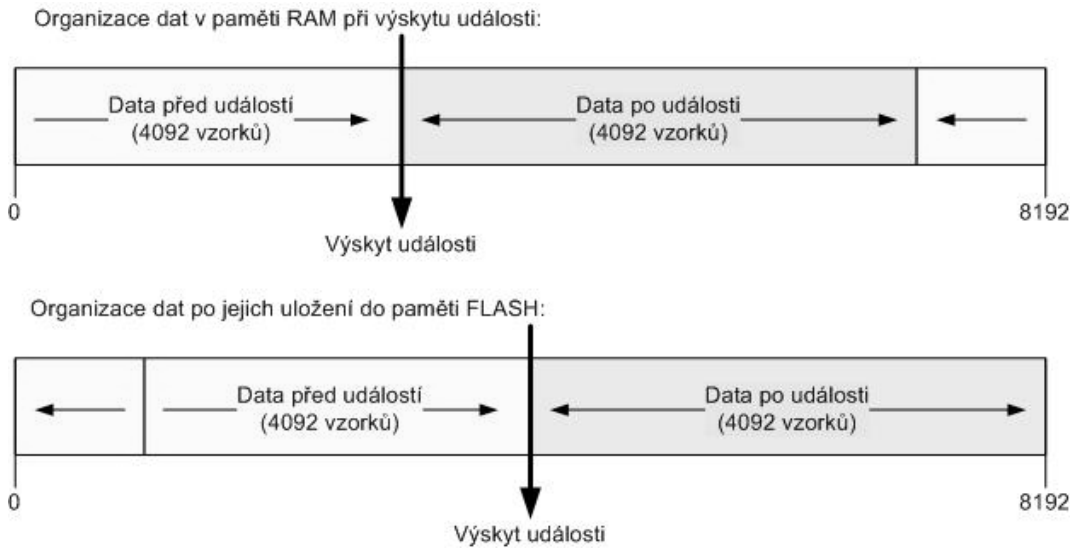
4.3.4. Zpracování událostí, synchronizace

Druhým již zmíněným režimem měřícího hardware je režim sledování událostí. V tomto režimu není využita žádná synchronizace signálů a je vzorkováno plnou rychlostí. Dvou bránová paměť RAM je vzorky neustále přepisována tak, že pokud se dostane ukazatel na konec, začne se přepisovat opět od začátku.

Uživatel si může v konzolovém menu zařízení zvolit jednu ze tří nabízených událostí při které jsou sejmuté vzorky uloženy do paměti FLASH. První a druhá událost (Event) je v principu shodná, a reaguje se na událost kdy napětí na jednom z analogových vstupů je větší než uživatelem nastavená hodnota. Třetí událost je zaměřena na osm logických vstupů, kdy se zvolí požadované slovo (Word = 8 bitů), při kterém je spuštěna událostní procedura.

Realizovanou architekturu snímání a zpracování událostí dokumentuje ***Obrázek 22***, kde jsou vstupní signály vzorkované bloky *CAPTURE ADC0 SAMPLES*, *CAPTURE ADC1 SAMPLES* a *CAPTURE ATA SAMPLES* a ukládány do dvou bránových pamětí RAM. Druhými porty označenými *R* jsou vzorky rovnou zobrazované pomocí bloku *GRAPHIC DISPLAY* a *PIXEL COLOR* na výstup VGA. Pokud nastane očekávaná událost, snímací bloky navzorkují polovinu pamětí RAM. Po dokončení vzorkování je každý snímací blok zastaven a portem *RW* jsou vzorky uloženy do paměti FLASH. Uložení vzorků do paměti FLASH pomocí *USB MIKROPROCESORU* provede blok *EVENT STORE*, který po vykonání této operace povolí spuštění snímacích bloků. Uspořádání naměřených dat v pamětích RAM a po uložení v paměti FLASH je naznačena viz. ***Obrázek 21***. Paměť je jakoby rozdělena na dvě poloviny. V první polovině paměti RAM je 4092 vzorků než nastala očekávaná událost a druhé polovině 4092 vzorků sejmutých po události. Po uložení obsahu pamětí RAM do paměti FLASH jsou vzorky seřazeny postupně za sebou.

Čas potřebný k uložení jedné události je závislí na zvolených časových základnách vstupních signálů (vzorkovacích frekvencích). Zařízení čeká na ovzorkování všech vstupních signálů, pak je spuštěn proces uložení vzorků do paměti FLASH, který trvá minimálně $8192/64\cdot10^{6}=128\mu s$. Výsledná doba zpracování jedné události je tedy dána součtem doby uložení všech vzorků do paměti FLASH a doby ovzorkování 4096 vzorků (poloviny paměti RAM) po výskytu události.



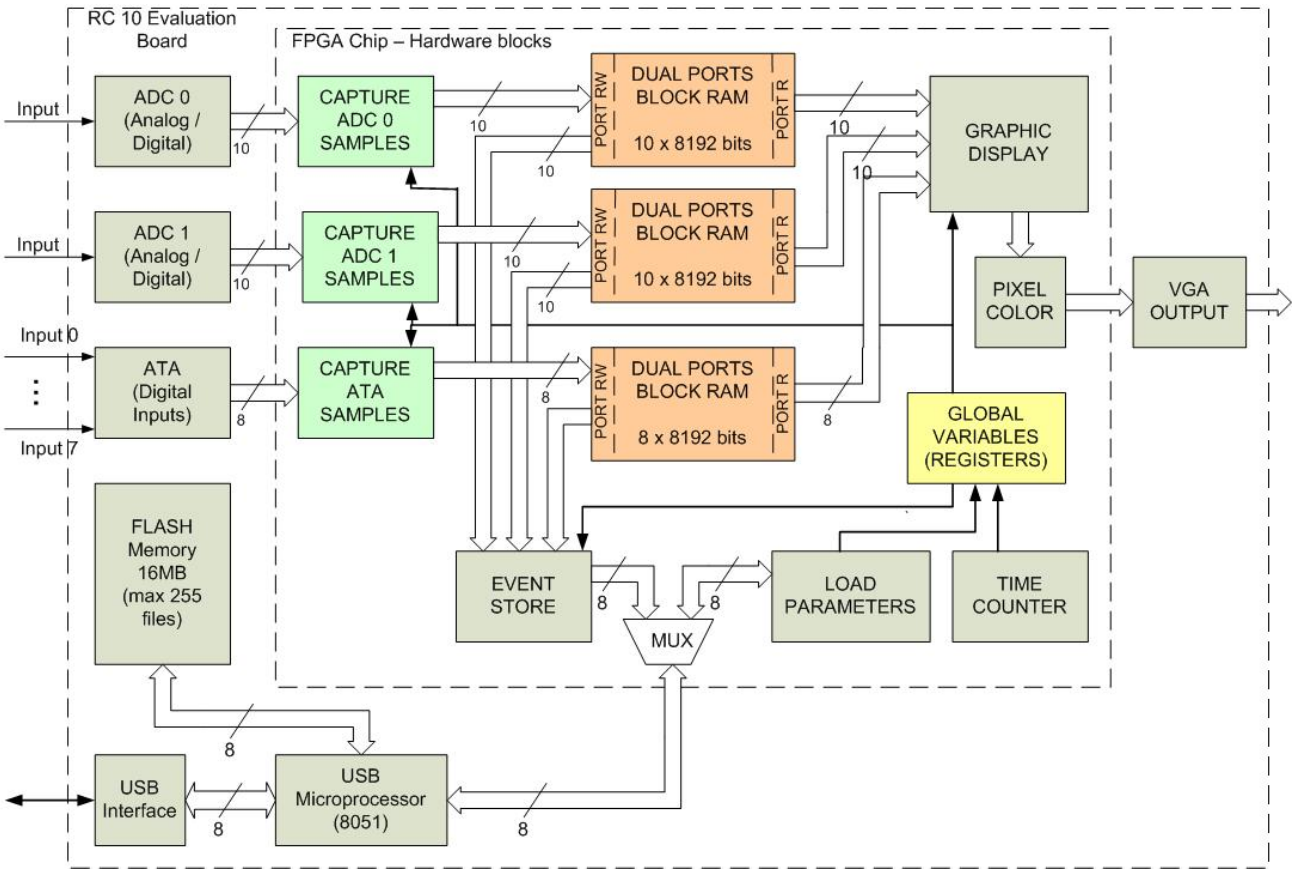
Obrázek 21 *Rozložení naměřených dat pamětích RAM a po uložení v paměti FLASH*

Spolupráce jednotlivých bloků je zajištěna přes globální proměnné (registry) naznačené blokem *GLOBAL VARIABLES.* Každý blok si kontroluje pro svou činnost důležité proměnné a podle nich řídí svou činnost. Synchronizace bloků je velmi důležitá, protože vzorkování vstupních signálů trvá různé doby v závislosti na zvolené časové základně.

Některé proměnné jsou nastaveny podle uživatelského nastavení přečteného z paměti FLASH blokem *LOAD PARAMETERS*.

Posledním důležitým blokem schématu viz. ***Obrázek 22*** je blok *TIME COUNTER*, který počítá čas v µs od nakonfigurování čipu měřícím hardwarem. Při výskytu události se uloží do paměti FLASH se vzorky signálů i čas, kdy k události došlo. V programu je tato procedura pod názvem *macro proc TimeGo ();*.

V programové kódu měřícího hardware je procedura realizující blok *EVENT STORE* nazvaná jako *macro proc StoreSamplesToFlash (FlashIndex, FlashIndexOffset);*. Struktura uložených vzorků v paměti FLASH je podrobněji popsána v kapitole 4.3.8.



Obrázek 22 *Blokové schéma popisující vzorkování vstupních signálů, ukládání událostí do paměti FLASH a zobrazování průběhů na výstup VGA*

4.3.5. Zobrazování snímaných průběhů na VGA výstup

Generování grafického výstupu a zobrazování snímaných průběhů signálů realizují bloky *GRAPHIC DISPLAY* a *PIXEL COLOR viz.* ***Obrázek 22***. Díky využití dvou bránových pamětí RAM, ve kterých jsou aktuálně snímané vzorky, odpadá problém se synchronizací zápisu a čtení z uvedených pamětí. Zobrazovací blok využívá pro přístup do pamětí port označený *R* a periodicky vykresluje vzorky na výstup VGA.

Blok GRAPHIC DISPLAY generuje grafický výstup po jednotlivých bodech (pixelech) v rozlišení 1024x768 bodů. Díky velké rychlosti čipu FPGA a určité setrvačnosti svitu bodu na monitoru je možné takto složit celý obraz. Princip vykreslování bude vysvětlen na jednoduchém příkladu zobrazení přímky. Pozice přímky nám určuje které body obrazovky mají být rozsvícené určitou barvou. Jednoduchými podmínkami jsme schopni rozhodnout zdali na aktuální pozici bodu rastru X, Y leží bod přímky. Pokud ano vykreslím bod přímky určitou barvou. Pokud jeden bod obrazovky pokrývá více objektů (např. přímek) je výsledná barva bodu dána logický součtem všech barev objektů zasahujících do tohoto bodu. Tímto způsobem jsou vykreslovány jednoduché objekty na obrazovku, u složitějších objektů jsou pak složitější podmínky určující oblast bodů objektu.

U vykreslování snímaných průběhů je situace obtížnější. Kdyby byl vykreslen pouze bod aktuálního vzorku, byl by celý průběh na obrazovce zobrazen pouze 1024 body a křivka by byla nespojitá. Aby vedla ze sousedních dvou bodů úsečka, je použita jednoduchá interpolace. Tím je výsledná křivka pěkně spojitá.

Blok *GRAPHIC DISPLAY* tedy každý hodinový takt určí jaké objekty zasahují do bodu na aktuální souřadnici obrazu. Výsledky podmínek jsou předány bloku *PIXEL COLOR*, který na základě těchto výsledků vypočte výslednou barvu bodu (pixelu) jako logický součet všech barev zasahujících objektů. Celý tento proces trvá pouze jediný takt čipu FPGA, takže vygenerování obrazu s rozlišením 1024x768 bodů trvá 1024 * 768 = 786432 hodinový taktů což při kmitočtu 64MHz je pouhých 12.3ms. Procedura realizující blok *GRAPHIC DISPLAY* je v programovém kódu uvedena jako *macro proc DisplayData (RAM0, RAM1, ATARAM, VideoOut, ClockRate);* a blok *PIXEL COLOR* jako procedura s názvem *macro proc SetPixelToWrite (TraceADC, TraceATA, Grid1, Grid2, MenuAdc0Pixel…);*.

4.3.6. Interaktivní menu, jeho funkce a ovládání

Pro snadné ovládání zařízení je měřící hardware data-loggeru vybaven interaktivním menu, ve kterém lze vybírat příslušné položky (parametry) a ty měnit. Menu je rozděleno na tři části, každá v jednom řádku. První resp. druhý řádek menu v horní části obrazovky slouží pro ovládání prvního resp. druhého analogového vstupu (ADC0, ADC1). Třetí řádek menu umístěný v dolní části obrazovky slouží pro ovládání ATA rozhraní. Podrobný význam jednotlivých položek interaktivního uživatelského menu viz. ***Tabulka 4***.

Pohyb v menu je umožněn pěti směrovým ovladačem. Pohybem doleva a do prava se postupně prochází celé menu. Pohybem ovladače nahoru a dolu se mění hodnoty aktuálně vybrané položky z menu, která je zvýrazněna barevným pruhem. Procedura obsluhující ovladač se v programovém kódu jmenuje *macro proc JoyEvents(ClockRate);*. Periodicky sleduje pohyb ovladače a podle toho vhodně mění obsah globálních proměnných. Některé procedury (ve schématech bloky) sledují určité globální proměnné a podle jejich hodnot je řízena jejich činnost.

Název ovládacího prvku	Význam prvku
CHx ↕	Posun průběhu signálu po svislé ose
CHx ↔	Posun průběhu signálu po vodorovné ose
TB xxx µs	Časová základna snímaného průběhu signálu v µs na dílek rastru
AMP xxxx mV	Amplituda snímaného průběhu signálu v mV na dílek rastru
RUN / STOP	Spuštění a zastavení snímání signálu
SIG / GND	Zobrazení snímaného průběhu signálu, nebo vodorovné přímky nulového napětí (Ground)
RESTART	Restartuje měřící hardware
HIST. EXPLORER	Nakonfiguruje čip hardwarem umožňujícím prohlížení uložených událostí z paměti FLASH
CONSOLE MENU	Nakonfiguruje čip hardwarem konzolového menu

Tabulka 4 *Položky interaktivního uživatelského menu měřícího hardware data-loggeru*

4.3.7. Princip menu a jeho zobrazování na VGA výstup

Interaktivní uživatelské menu navrhnutých HW je rozděleno do třech řádků. Každý řádek menu je v rozlišení obrazovky 1024x768 bodů tvořen právě 1024 body. Paměť RAM je organizována do bloků 18kb (18b x 1024). Jeden řádek menu zabírá jeden paměťový blok RAM, celkem celé uživatelské menu zabírá 3 paměťové bloky, tedy 3 x 18kb. Jeden řádek menu i s rozměry je naznačen viz. ***Obrázek 23***. Celý řádek je rozdělen na osm stejných prvků menu o délce 128b s výškou 18b.

Obrázek 23 *Prvky jednoho řádku interaktivního menu a jejich rozměry*

Bloky pamětí RAM určené pro realizaci menu jsou inicializovány již při kompilaci. Obsah části paměti je naznačen viz. ***Obrázek 24***. Poslední tj. 18bit slouží pro zvýrazňování aktuálně vybrané položky menu. Pro pohyb v menu jsou zavedeny globální proměnné , ukazující na vybraný prvek celého menu. To zajišťuje již zmíněná procedura *macro proc JoyEvents(ClockRate);*. Vybraný prvek má nastaveny všechny nejvyšší bity nastaveny na

log1. Procedura *macro proc DisplayData (RAM0, RAM1, ATARAM, VideoOut, ClockRate);* pak definuje prostor právě vybraného prvku a ten je při vykreslování vybarven.

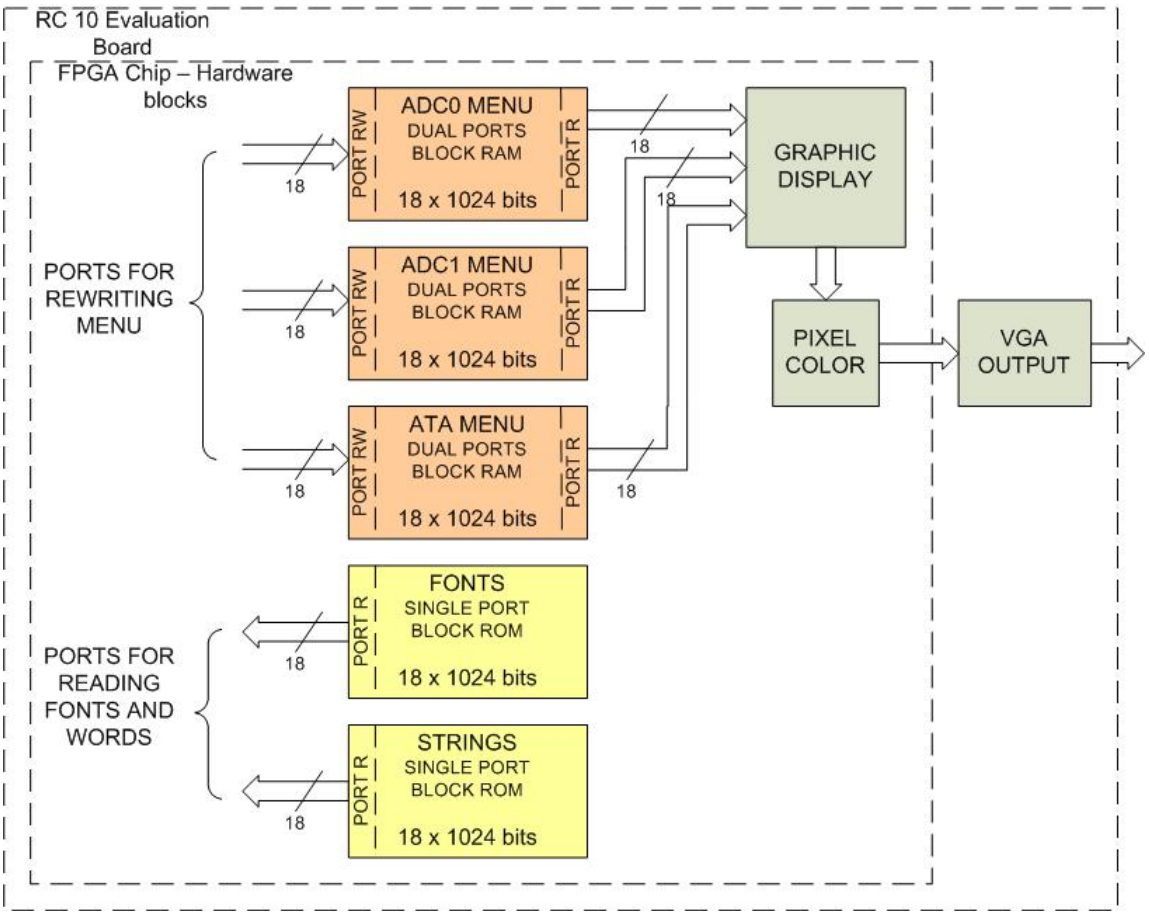
0 ◄——Délka menu 1024b——► 1023

```
1  0000000011111000000000000111111100000000000
2  0000000111111000000000001110001110000000000
3  0000001110111000000000011100000111000000000
4  0000011100111000000000111000000011100000000
5  0000111000111000000001110000000111000000000
6  0001110000111000000011100000001110000000000
7  0011100000111000000111000000011100000000000
8  0000000000111000000000000001110000000000000
9  0000000000111000000000000011100000000000000
10 0000000000111000000000000111000000000000000
11 0000000000111000000000001110000000000000000
12 0000000000111000000000011100000000000000000
13 0000000000111000000000111000000000000000000
14 0000000000111000000001110000000000000000000
15 0000000000111000000011100000000000000000000
16 0000000000111000000111111111111111000000000
17 0000000000111000001111111111111111000000000
18 0000000000000000000000000000000000000000000
```

Obrázek 24 *Blok paměti RAM realizující interaktivní menu, nebo banku znaků*

Princip vykreslování interaktivního menu je přiblížen na blokovém schématu viz. ***Obrázek 25***. Tři řádky menu jsou realizovány třemi paměťovými bloky RAM. V blokovém schématu jsou označeny jako *ADC0 MENU*, *ADC1 MENU* a *ATA MENU*. Portem *R* jsou data z pamětí neustále čtena a zpracovávána blokem *GRAPHIC DISPLAY* k vykreslení na VGA výstup. Port *RW* slouží pro přepisování položek v menu. Při změně parametru uživatelem, se z banky znaků nebo slov přečtou potřebné znaky a ty se portem RW zapíšou do správného řádku menu na potřebnou pozici.

Banky obsahující používané znaky nebo slova jsou realizovány jedno bránovými paměťovými bloky ROM viz. ***Obrázek 25***. Tyto paměti jsou dvě a reprezentují je bloky *FONTS* a *STRINGS* v blokovém schématu. Jediným portem *R* jsou tedy z těchto pamětí čteny potřebné znaky nebo slova.

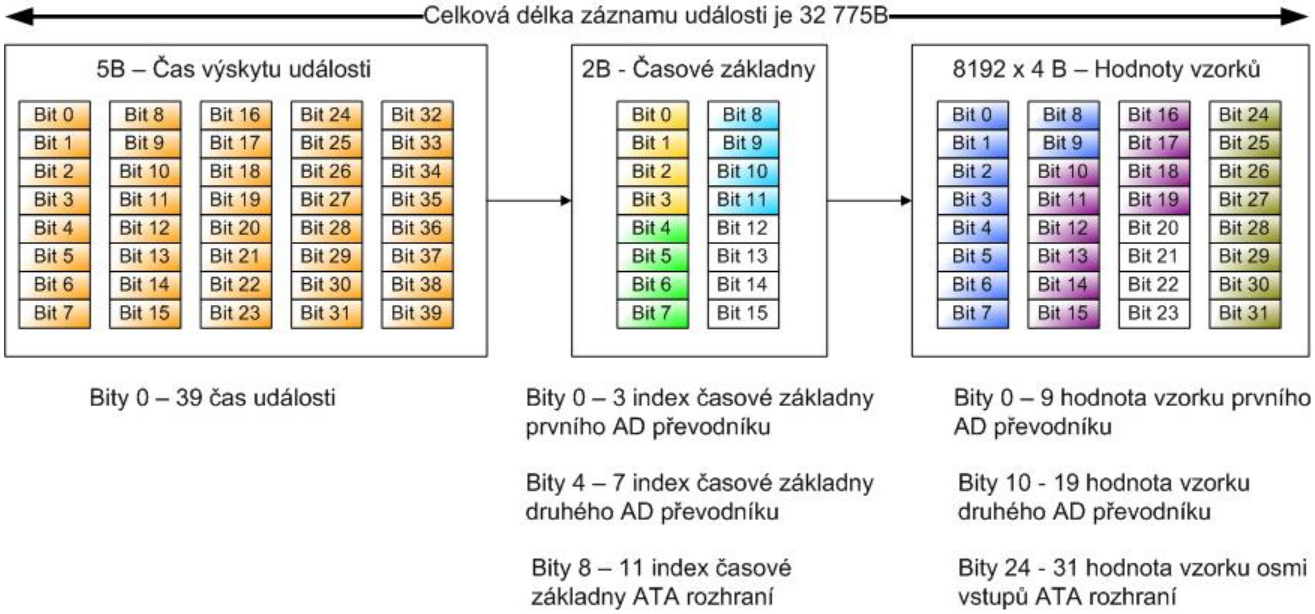


Obrázek 25 *Blokové schéma hardware realizující interaktivní menu s jeho vykreslováním na výstup VGA a banku znaků a slov*

4.3.8. Ukládání vzorků do FLASH paměti

Při každém výskytu události, popsané v kapitole 4.3.4 se vzorky nabrané v pamětích RAM všech vstupů uloží do paměti FLASH. Protože se do této paměti zapisují jednotlivé bajty, byla vytvořena struktura, podle které se jednotlivé bity hodnot vzorků vhodně rozdělují a spojují do jednotlivých bajtů ukládaných po sobě do paměti. Jeden vzorek obou analogových vstupů (ADC0 a ADC1) a vzorek osmi vstupů ATA rozhraní tedy zabere 4B viz. ***Obrázek 26*** první obdélník zprava. Protože je vzorků celkem 8192 je uloženo 8192 x 4 = 32768B v paměti. Aby byl záznam úplný je nutné uložit hodnotu času, kdy událost nastala od spuštění přístroje a hodnoty časových základen jednotlivých vstupů. Čas události reprezentuje 5B v prvním obdélníku a časové základny 2B v druhém obdélníku zleva viz. ***Obrázek 26***. Celkem zabírá jeden celý záznam události v paměti 32777B. Každý záznam události je v paměti FLASH uložen do samostatného souboru, kterých je vyhrazen prostor adres od 111 do 249.

V zdrojovém kódu měřícího hardware je procedura vykonávající tuto úlohu nazvána *macro proc StoreSamplesToFlash (FlashIndex, FlashIndexOffset);*. Parametrem *FlashIndex* je dána pozice, na kterou je záznam události v paměti FLASH uložen.



***Obrázek 26**. Struktura uložených dat jedné události v paměti FLASH na desce RC10*

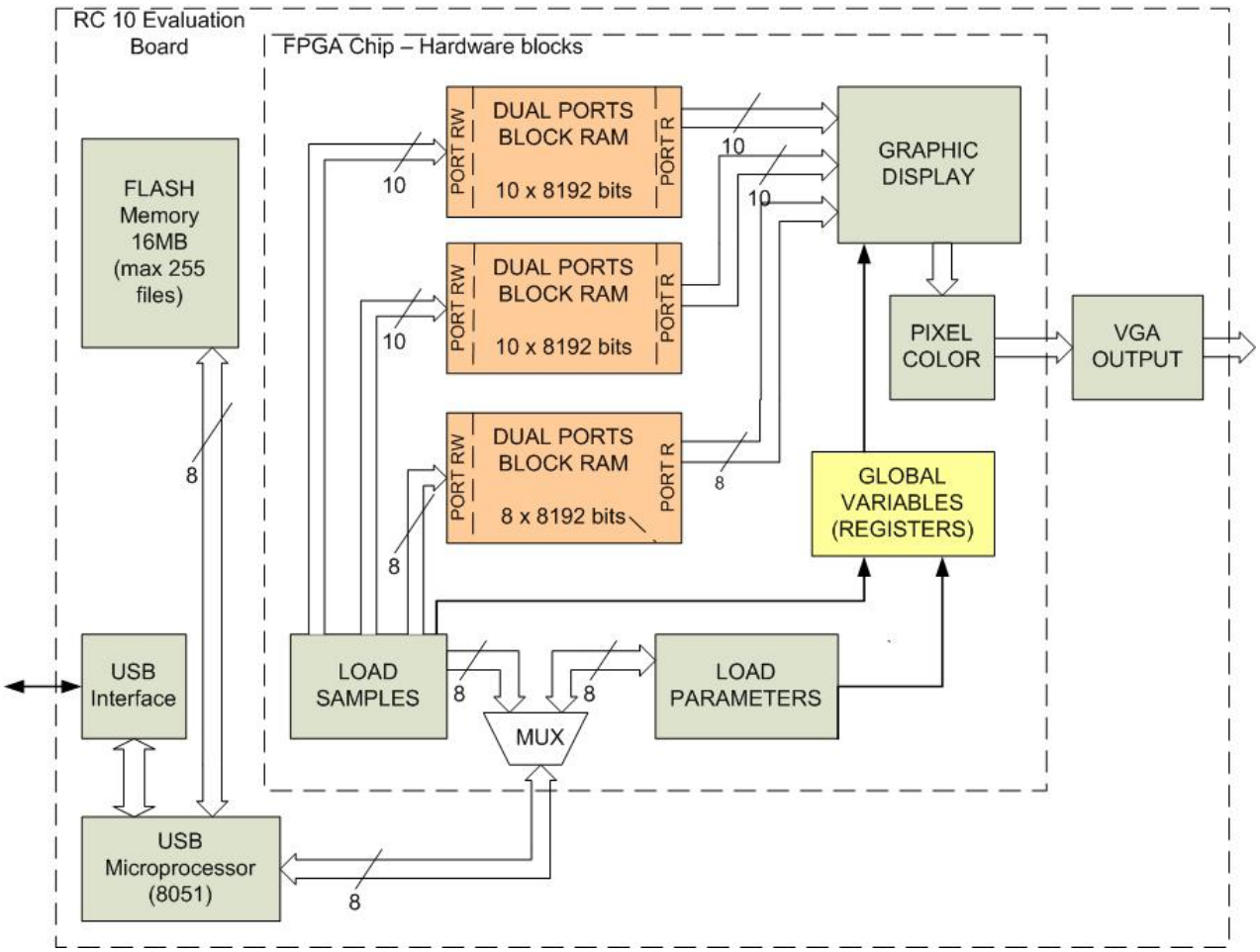
4.4. Prohlížeč uložených událostí

4.4.1. Hardware umožňující prohlížení uložených událostí

Tento hardware je odvozen od již navrhnutého měřícího hardware data-loggeru. Podstatná změna je provedena v načtení vzorků naměřených průběhů do pamětí RAM. Místo vzorkování vstupních signálů a ukládání hodnot do pamětí RAM jako tomu bylo u měřícího hardware, jsou zde vzorky signálů načteny z paměti FLASH. Blokové schéma tohoto hardware je uvedeno viz ***Obrázek 27***, kde blok nazvaný *LOAD SAMPLES* čte data událostí z paměti FLASH přes blok *USB MIKROPROCESOR* a zapisuje je do dvou bránových pamětí RAM portem *RW*. Druhý port pamětí RAM je využit pro zobrazování dat na grafický výstup VGA stejně jako tomu je u měřícího hardware data-loggeru.

Procedura reprezentující blok *LOAD SAMPLES* je v programovém kódu nazvána *macro proc LoadSamplesFromFlash (FlashIndex, FlashIndexOffset)*;.

Spolupráce jednotlivých bloků je opět řízena pomocí globálních proměnných (registrů), které v blokovém schématu naznačuje blok *GLOBAL VARIABLES*.



Obrázek 27 *Blokové schéma hardware umožňujícího zobrazení průběhů uložených událostí z paměti FLASH*

4.4.2. Interaktivní menu a jeho funkce

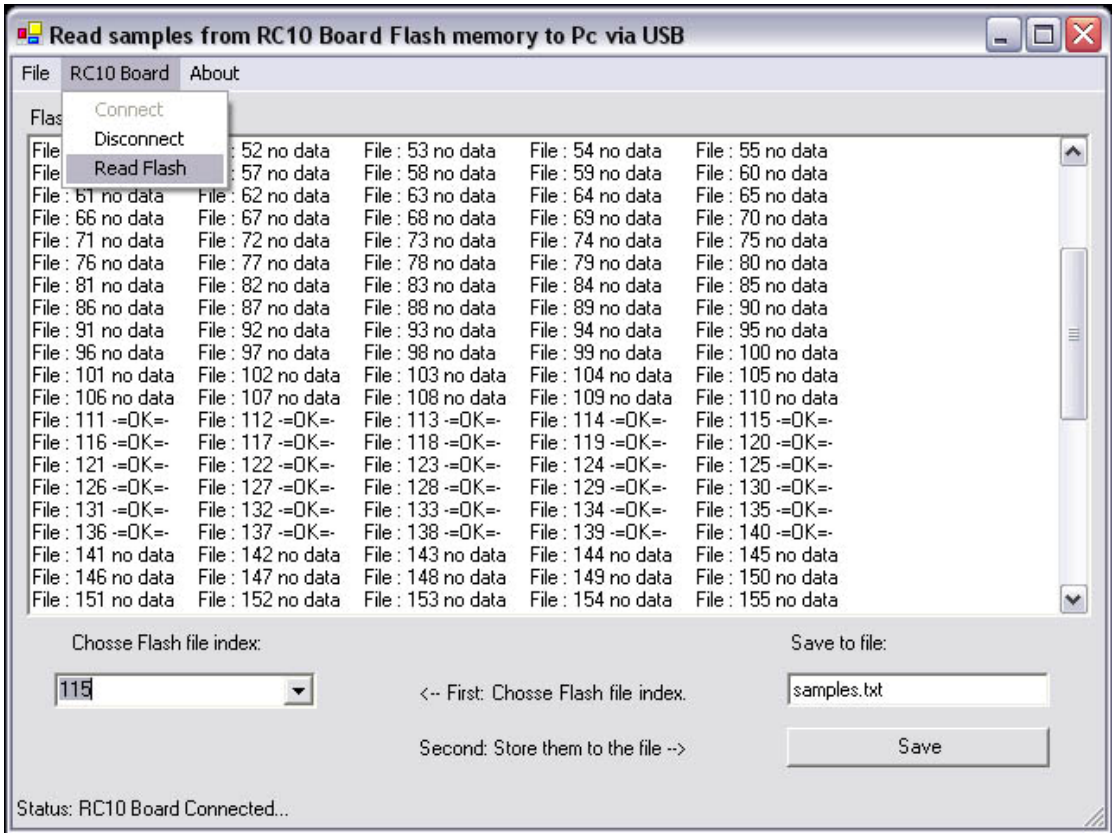
Menu tohoto hardware je velmi podobné menu měřícího hardware. Několik prvků menu bylo odstraněno, nebo pozměněno a některé prvky dostali význam pouze informativní. Prvky menu pro posun signálů po osách X, Y a změnu amplitudy signálů zůstali beze změn. Prvek s časovou základnou je nyní pouze informativní a ukazuje hodnotu časové základny použité při měření daného signálu. Prvek s popiskem *EVENT* ukazuje číslo právě načteného souboru s událostí z paměti FLASH. Jeho změnou je možno procházet celou paměť s událostmi. Pokud soubor v paměti FLASH má jinou velikost než tento hardware očekává, nebo vůbec neexistuje, průběhy signálů události se nezobrazí a vypíše se hláška *NO DATA*. V pravém horním rohu je zobrazena časová informace, kdy načtená událost nastala. Položka má popisek *EVENT TIME*, za kterým je čas uváděný v μs.

5. NÁVRH DOPLŇUJÍCÍHO SOFTWARE PRO PC

Velmi podstatnou část celého měřícího zařízení tvoří aplikační software na straně osobního počítače PC. Cílem bylo vytvořit software který přečte uložená data z paměti FLASH vývojové desky RC10 a uložit je do textového souboru pro jednoduché použití a další zpracování. Aplikaci jsem vytvořil v programovacím jazyku C++ pod vývojovým prostředím Microsoft Visual Studio .NET 2003. Výstupem aplikace je textový soubor, který obsahuje hodnoty naměřených průběhů. Pro zpracování těchto souborů jsem vytvořil skript v prostředí Matlab, který přečte zadaný soubor a vykreslí grafy uložených průběhů. Soubor je vygenerován v takovém formátu, že ho lze importovat do programu Microsoft Excel, v němž je také možné vygenerovat grafy uložených průběhů.

5.1. Aplikace pro čtení dat z Flash paměti vývojové desky RC10 do PC

Jak už jsem uvedl, aplikace je napsaná v jazyce C++, a odladěná na PC pod operačním systémem Microsoft Windows XP. Pro bezproblémový chod aplikace je nutné mít nainstalovaný ovladač pro vývojovou desku RC10 (RC10 USB Driver).



Obrázek 28. *Okno aplikace vyvinuté pro čtení naměřených dat z paměti FLASH vývojové desky RC10*

Po spuštění aplikace se zobrazí hlavní okno viz. ***Obrázek 28***. V nabídce *RC10 Board* v horní liště se vybere položka *Connect*. Pokud nebyla vývojová deska RC10 připojena k PC přes rozhraní USB, vypíše se v dolní liště chybová hláška. Po připojení desky opakujeme připojení *Connect*. Deska je nyní připojena k aplikaci a v nabídce *RC10 Board* se zpřístupnila položka *Read Flash.* Jejím vybráním se přečte celá paměť FLASH v desce RC10 a informace o jejím obsahu se vypíšou do středového okna aplikace. U indexů (1 až 254) je vypsána informativní hláška *„no data“* pokud soubor v paměti neobsahuje potřebná data. Pokud je délka souboru správná je vypsáno u příslušného indexu *„-= OK =-“*. Čísla indexů které vyhovují svou délkou záznamu jsou vypsány do rozbalovacího seznamu níže. Vybráním čísla indexu souboru z paměti FLASH vybereme data, která mají být zpracována a uložena do textového souboru. Kliknutím na tlačítko *Save* se uloží data do souboru pod názvem uvedeným nad tlačítkem. Aplikaci je možné ukončit křížkem v horním pravém rohu okna.

5.1.1. Výchozí příklad

Při návrhu a tvorbě aplikace jsem vycházel z příkladu *FlashUSB*, který je součástí balíčku PDK. Tento ukázkový příklad provádí několik jednoduchých testů pro ověření funkčnosti vývojové desky RC10 a její součástí. Příklad demonstruje přenos dat mezi PC a deskou RC10 rozhraním USB, zápis a čtení z paměti FLASH, konfiguraci FPGA čipu z zkompilovaného souboru uloženého na PC i v paměti FLASH na desce RC10. Program je napsán jako konzolová aplikace Win32 pod vývojovým prostředím Microsoft Visual Studio .NET 2003.

5.1.2. Knihovna pro práci s vývojovou deskou RC10 v C++

V uvedené aplikaci je využita knihovna *rc.lib* a její hlavičkový soubor (header file) *rc.h*, kde jsou funkce a metody pro snadnější práci z vývojovou deskou RC10. Přímo se jedná o podporu práce z pamětí FLASH a komunikací přes rozhraní USB. Knihovna je součástí vývojového prostředí DK / PDK a jsou z ní využity následující funkce:

- `RCStatus RCBoardOpen (int Board, RCBoard *BoardPtr);`
  Funkce pro navázání spojení s vývojovou deskou RC10 přes rozhraní USB.

- `RCStatus RCBoardClose (RCBoard Board);`
  Funkce pro ukončení spojení s vývojovou deskou RC10.

- `RCStatus RCDeviceClear (RCBoard Board, int Device);`
  Funkce vyvolá reset desky RC10 a vyčistí se obsah FPGA.

- `RCStatus RCFlashIndexGetLength (RCBoard Board, int Flash, int Index, int *BytesPtr);`
  Funkce přečte soubor z FLASH paměti a vrátí jeho délku (počet bajtů) do parametru BytesPtr.

- `RCStatus RCFlashIndexRead (RCBoard Board, int Flash, int Index, int Start, int Bytes, char *Buffer, int *BytesRead);`
  Funkce načte požadovaný soubor z paměti FLASH do bajtového pole Buffer.

5.1.3. Čtení dat z Flash paměti do PC

Přečtení zvoleného záznamu (souboru) z FLASH paměti se provádí knihovní funkcí *RCFlashIndexRead* , která naplní bajtové pole bajty z FLASH paměti. Struktura uložených dat ve FLASH paměti na vývojové desce RC10 byla podrobně vysvětlena v kapitole 4.3.8.

Prvních 5 bajtů záznamu tvoří 40 bitovou hodnotu času události v mikrosekundách. To je čas, kdy nastala požadovaná událost od spuštění zařízení, a proběhlo ovzorkování signálů a jejich uložení do FLASH paměti. Další 2 bajty nesou informace o časových základnách obou AD převodníků a ATA rozhraní. 4 nižší bity prvního bajtu tvoří index do pole konstant časové základny prvního AD převodníku. Obdobně 4 vyšší bity pro druhý AD převodník. Index časové základny pro ATA rozhraní je tvořen 4 nižšími bity druhého bajtu. Hodnoty konstant časové základny v mikrosekundách jsou následující: 0.5, 1, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100.

Dále v záznamu následují čtveřice bajtů obsahující jednotlivé vzorky obou AD převodníků a ATA rozhraní. Těchto čtveřic je celkem 8192 a každá obsahuje dvě 10 bitové hodnoty vzorků zaznamenané z obou AD převodníků, a jednu 8 bitovou hodnotu reprezentující logické úrovně všech 8 vstupů ATA rozhraní. Celková délka jednoho záznamu v paměti FLASH je tedy určena následujícím výpočtem $(8192 * 4) + 7 = 32775$ [Bytes].

5.1.4. Úprava dat a formát výstupního textového souboru

Načtený záznam z paměti FLASH je ve formátu bajtového pole, to je potřeba upravit a převést na reálné hodnoty vzorků, a doplnit k nim hodnotu časové informace podle hodnoty časové základny.

Nejdříve je potřeba z bajtového pole přečíst prvních 5 bajtů, spojit je ve správné pořadí. Tím je získaná 40 bitová hodnota, kterou se převede na reálné číslo reprezentující čas, kdy nastala očekávaná událost viz. ***Obrázek 26***. Převod jednoho bajtu do 8 bitového binárního čísla zajišťuje funkce *void convertToBin (unsigned short int byteValue,unsigned short int *binValue)*. Funkce dostane v prvním parametru *byteValue* hodnotu reprezentující jeden bajt, převede ho a uloží na adresu osmi prvkového pole *binValue*. Každý prvek tohoto osmi

prvkového pole představuje jeden bit binárního čísla, tedy 0 nebo 1. Spojená 40 bitová binární hodnota se převede na reálné číslo pomocí další funkce *unsigned short int convertToInt (unsigned short int *binValue, int bitsCount)*. Prvním parametrem této funkce je adresa na n-prvkové pole *binValue*, kde počet prvků uvádí druhý parametr *bitsCount*. Funkcí je tedy možné převést libovolně dlouhé prvkové pole obsahující 0 a 1 na celé číslo. Funkce vrátí výsledek v podobě celého čísla.

Další dva bajty (tedy 6. a 7.) viz. ***Obrázek 26*** se upraví obdobně s využitím obou funkcí. Získáme tak informaci o časových základnách prvního a druhého AD převodníku a ATA rozhraní. Časové základny jsou ve formátu mikrosekundy / dílek. Jeden dílek je 32 bodů (pixels) na obrazovce což odpovídá 32 vzorků signálu. Maximální počet uložených vzorků je 8192 pro každý AD převodní i pro ATA rozhraní. Pomocí jednoduchého výpočtu podle ***Rovnice 1.*** stanovíme hodnoty všech tří časových os, kde *act_time* je aktuální čas, *n* je číslo vzorku (0 až 8192), *time_base* je hodnota časové základny a konstanta 32 je počet pixelů na jeden dílek rastru na obrazovce data-loggeru.

Rovnice 1.

$$act\_time = n * \frac{time\_base}{32} \quad [\mu s]$$

Rovnice 2.

$$voltage = (sample\_value - 512) * \frac{voltage\_range}{1024} \quad [V]$$

Dále záznam obsahuje 8192 čtveřic bajtů, kde každá čtveřice bajtů obsahuje 10 bitovou hodnotu aktuální amplitudy vzorku prvního i druhého AD převodníku a 8 bitovou hodnotu aktuálního logického stavu 8 vstupů ATA rozhraní. Tyto bajty je potřeba upravit analogickou cestou jako bylo již uvedeno v minulém odstavci. Struktura uložených bajtů viz. ***Obrázek 26***. Získanou aktuální hodnotu vzorku tedy představuje hodnota v rozmezí 0 až 1023. Ta odpovídá rozsahu napětí ±1V při nastaveném rozsahu data-loggeru 2V špička-špička (peak to peak). Převod hodnoty vzorku na napětí provedeme výpočtem podle ***Rovnice 2.***, kde *voltage* je výsledná hodnota napětí, *sample_value* je hodnota amplitudy vzorku (0 až 1023), *voltage_range* je rozsah napětí AD převodníku. Konstantu 512 odečteme, aby se střed rozsahu posunul na nulovou úroveň, protože nulová hodnota napětí odpovídá hodnotě 512, což je polovina rozsahu. Konstanta 1024 přestavuje rozsah amplitudy odebraného vzorku napětí.

Struktura výstupního textového souboru je zvolena tak, aby bylo možné soubor zpracovat programovacím jazykem Matlab, i jej importovat do programu Microsoft Excel. Soubor tedy obsahuje 4096 řádků, kde každý řádek je tvořen číslem řádku, hodnotou časové osy prvního AD převodníku, amplitudou vzorku prvního AD převodníku, dále tytéž dvě

hodnoty pro druhý AD převodník, pak hodnota časové osy ATA rozhraní a osm jedno bitových údajů reprezentující aktuální logický stav osmi vstupů ATA rozhraní. Všechny tyto údaje jsou odděleny čárkou. Příklad uložených dat v souboru viz. ***Obrázek 29***.

samples.txt - Poznámkový blok

Soubor Úpravy Formát Zobrazení Nápověda

```
213, 66.562500, 0.14453125, 3.328125, 0.19921875, 33.281250, 1, 0, 0, 0, 0, 0, 0, 0
214, 66.875000, 0.18945313, 3.343750, 0.21289063, 33.437500, 1, 0, 0, 0, 0, 0, 0, 0
215, 67.187500, 0.13476563, 3.359375, 0.20507813, 33.593750, 1, 0, 0, 0, 0, 0, 0, 0
216, 67.500000, 0.03906250, 3.375000, 0.20703125, 33.750000, 1, 0, 0, 0, 0, 0, 0, 0
217, 67.812500, 0.48632813, 3.390625, 0.19531250, 33.906250, 1, 0, 0, 0, 0, 0, 0, 0
218, 68.125000, -0.02539063, 3.406250, 0.19531250, 34.062500, 1, 0, 0, 0, 0, 0, 0, 0
219, 68.437500, 0.24414063, 3.421875, 0.18750000, 34.218750, 1, 0, 0, 0, 0, 0, 0, 0
220, 68.750000, 0.19726563, 3.437500, 0.17968750, 34.375000, 1, 0, 0, 0, 0, 0, 0, 0
221, 69.062500, 0.17968750, 3.453125, 0.17968750, 34.531250, 1, 0, 0, 0, 0, 0, 0, 0
222, 69.375000, 0.18164063, 3.468750, 0.17968750, 34.687500, 1, 0, 0, 0, 0, 0, 0, 0
223, 69.687500, 0.18554688, 3.484375, 0.18164063, 34.843750, 1, 0, 0, 0, 0, 0, 0, 0
224, 70.000000, 0.18554688, 3.500000, 0.17773438, 35.000000, 1, 0, 1, 0, 0, 0, 0, 0
225, 70.312500, 0.17382813, 3.515625, 0.18164063, 35.156250, 1, 0, 1, 0, 0, 0, 0, 0
226, 70.625000, 0.18359375, 3.531250, 0.18750000, 35.312500, 1, 0, 1, 0, 0, 0, 0, 0
227, 70.937500, 0.19140625, 3.546875, 0.18945313, 35.468750, 1, 0, 1, 0, 0, 0, 0, 0
```

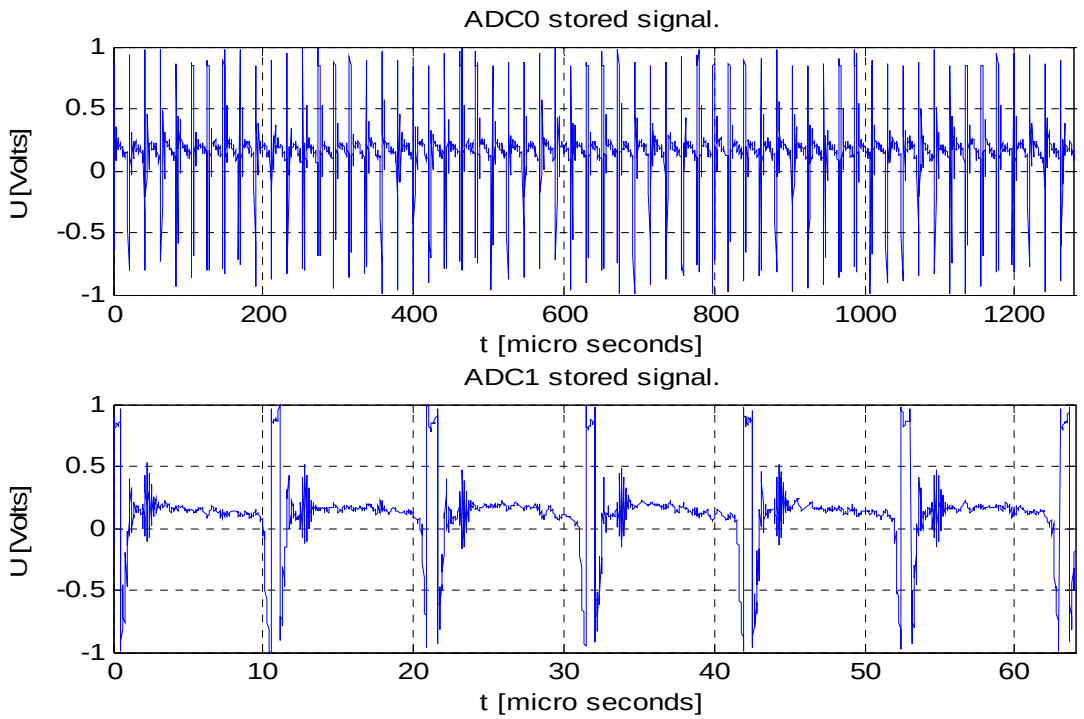
Obrázek 29. *Ukázka uložených dat ve výstupním souboru vyvinuté aplikace (otevřeno v poznámkovém bloku ve Windows XP)*

5.2. Funkce vykreslující grafy průběhů v programu Matlab

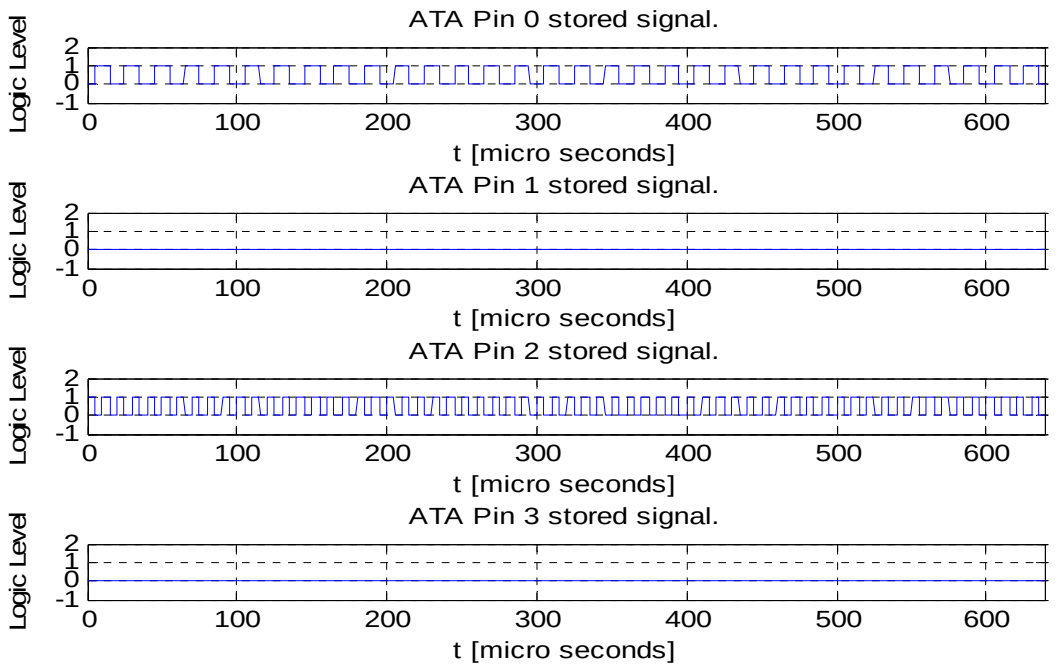
Funkce je napsaná v programovacím prostředí Matlab a jmenuje se *readSamples*. Pro spuštění stačí otevřít prostředí Matlab a nasměrovat pracovní adresář (Workspace) do složky se souborem popisované funkce. Spuštění funkce lze provést příkazem *readSamples('fileName')*, kde parametr *fileName* představuje jméno čteného souboru. Pokud se soubor nachází v jiné složce, než je samotná funkce je potřeba před jméno souboru uvést celou absolutní cestu. Příklad použití funkce je tedy: *readSamples('C:/dir1/dir2/samples.txt')*. Vykreslené grafy skriptu viz. ***Obrázek 30*** a ***Obrázek 31***.

Funkce přečte zadaný soubor a hodnoty jsou naplněny do matice o velikosti 8192 řádků x 14 sloupců. Jednotlivé sloupce matice jsou převedeny do vektorů čísel, reprezentující časové osy a amplitudy vzorků naměřených signálů. Z těchto vektorů je vytvořeno deset grafů naměřených signálů pro danou událost. První dva grafy zobrazují naměřený signál z obou AD převodníků. Amplituda je uvedena ve voltech a má rozsah 2V špička-špička (peak to peak). Časová osa je v mikrosekundách, kde čas nula odpovídá času události. Dalších osm grafů zobrazuje naměřené signály na osmi logických vstupech ATA rozhraní. Logická úroveň těchto signálu nabývá hodnot logická 0 nebo 1. Časová osa je řešena obdobně jako v předchozích grafech. Hodnoty časových os obou AD převodníků a ATA rozhraní se mohou lišit v závislosti na zvolené časové základně při měření.

Vygenerované grafy průběhů je možno uložit jako obrázky ve formátu jpg nebo jako metafile soubory.



Obrázek 30. *Ukázka naměřených průběhů obou AD převodníků.*
Graf vykreslen programem Matlab z dat výstupního souboru aplikace.



Obrázek 31 *Ukázka naměřených průběhů z ATA rozhraní.*
Graf vykreslen programem Matlab z dat výstupního souboru aplikace.

5.3. Import dat do programu Microsoft Excel

Výstupní soubor z aplikace popsané v kapitole 5.1 je možné importovat do programu Microsoft Excel. Jako oddělovací znak sloupců při importu souboru je nutné nastavit znak čárky “,”. Program importuje sloupce hodnot ze souboru do aktivního sešitu. První sloupec obsahuje pouze čísla řádků. Druhý sloupec obsahuje čas odebrání vzorku prvního AD převodníku, jehož hodnota amplitudy je ve sloupci následujícím, tj. třetím. Další dva sloupce, tedy čtvrtý a pátý jsou obdobou druhého a třetího sloupce, ale jsou to hodnoty pro druhý AD převodník. Následující šestý sloupec obsahuje čas odebrání vzorků osmi logických vstupů, jejichž logická úroveň je uvedena v následujících osmi sloupcích.

6. ZÁVĚR

Při návrhu tohoto zařízení jsem se seznámil s moderní mikroelektronikou řízenou čipem FPGA. Naučil jsem se základní principy návrhů logických obvodů (hardware) pro programovatelná hradlová pole FPGA. Osvojil jsem si popisovaný software pro návrh podobných aplikací.

Navrhnuté zařízení je použitelné v širokém spektru aplikací, ve kterých je potřeba zaznamenat průběhy připojených signálů o vysokém kmitočtu v okolí např. napěťové špičky, překročení definované napěťové úrovně a nebo výskytu definované logické úrovně na digitálních (logických) vstupech. Zařízení může být napájeno ze záložního bateriového zdroje, protože použitý čip FPGA Spartan 3 1500L se řadí mezi čipy s malou proudovou spotřebou. Pro provoz zařízení stačí pouze 5V napájení. Naměřená spotřeba jednotlivých hardwarových konfigurací je uvedena v následující tabulce:

Hardwarová konfigurace	**Spotřeba vývojové desky [mA]**
Konzolové menu	250
Měřící hardware	300
Prohlížeč uložených událostí	180

Mobilitu zařízení lze podtrhnout implementací malého barevného grafického displeje, na kterém by uživatel mohl sledovat měřené vstupní signály a volit režimy zařízení z interaktivních menu, podobně jako tomu je na nynějším grafickém výstupu VGA.

V aktuální navrženém stavu má zařízení následující parametry, zjištěné jednoduchém testováním a měřením:

Napěťový rozsah obou analogových vstupů	2V špička-špička
Maximální frekvence vstupních analogových signálů	32MHz (2 vzorky na T)
Napěťové úrovně digitálních (logických) vstupů	Log 0 ~ 0V a Log 1 ~ 3,3V
Maximální frekvence digitálních (logických) vstupů	32MHz (2 vzorky na T)
Počet záznamů událostí v paměti FLASH	138
Velikost jednoho záznamu události	32775B
Velikost všech záznamů v paměti FLASH	4,5MB (zaokrohleno)
Počet všech vzorků všech záznamu událostí	4521984 vzorků

Délka uložených vstupních signálů při události a čas potřebný k uložení jedné události je závislí na zvolené časové základně (vzorkovací frekvenci). Hodnoty časů jsou uvedeny v následující tabulce:

Časová základna	Vzorkovací kmitočet	Délka signálu uloženého při události	Minimální doba zpracování události
0.5 µs/dílek	64 MHz	128 µs	192 µs
1 µs/dílek	32 MHz	256 µs	256 µs
5 µs/dílek	6.4 MHz	1.28 ms	0.768 ms
10 µs/dílek	3.2 MHz	2.56 ms	1.408 ms
20 µs/dílek	1.6 MHz	5.12 ms	2.688 ms
30 µs/dílek	1.067 MHz	7.68 ms	3.968 ms
40 µs/dílek	0.8 MHz	10.24 ms	5.248 ms
50 µs/dílek	0.64 MHz	12.8 ms	6.528 ms
60 µs/dílek	0.532 MHz	15.36 ms	7.808 ms
70 µs/dílek	0.459 MHz	17.92 ms	9.088 ms
80 µs/dílek	0.4 MHz	20.48 ms	10.368 ms
90 µs/dílek	0.356 MHz	23.04 ms	11.648 ms
100 µs/dílek	0.32 MHz	25.6 ms	12.928 ms

Zařízení je navrhnuto poměrně obecně, proto je možno jednotlivé hardwarové konfigurace rozšiřovat a upravovat podle požadavků uživatele nebo budoucího koncového zákazníka.

7. SEZNAM POUŽITÉ LITERATURY

[1] P. Marwedel, *Embedded System Design,* Boston: Kluwer Academic Publisher, 2003.

[2] M.J.S. Smith, *Application-Specific Integrated Circuits,* Addison Wesley, 1997.

[3] Matthew Aubury, *RC10 Platform Development Manual,* Celoxica, UK, 2005

[4] Roger Gook, *Platform Development Kit Manual,* Celoxica, UK, 2005

[5] *User manual S1D15G14*, Seiko Epson Corporation, 2003

[6] Wikipedia – The Free Encyclopedia, *Field-programmable gate array,* 2006 [online]. URL: http://en.wikipedia.org/wiki/FPGA#Applications

[7] Automa – časopis pro automatizační techniku, *Programovatelná hradlová pole – FPGA,* 2006 [online]. URL: http://www.odbornecasopisy.cz/automa/2006/au020609.htm

[8] Xilinx, *Spartan-3 Complete Data Sheet (All four modules),* 2006 [online]. URL: http://direct.xilinx.com/bvdocs/publications/ds099.pdf

Seznam ilustrací

Seznam tabulek

Seznam rovnic