Slovenská Technická Univerzita
Fakulta informatiky a informačných technológií
Ilkovičová 3, 812 19 Bratislava 4

Návrh a realizácia experimentálnych mikropočítačov

(Tímový projekt)

Ročník : 1
Akademický rok : 2007/2008
Študijný program : Počítačové systémy a siete
A-team: Bc. Róbert Komarómy, Bc. Tomáš Krajčo, Bc. Marek Sobolič, Bc. Michal Tölgyessy, Bc. Martin Viceník
Vedúci projektu: doc. Ing. Tibor Krajčovič, PhD.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam skratiek

AIC Advanced Interrupt Controller

ADC Analog Digital Converter

AD prevodník Analógov-digitálny prevodník

ALU Arithmetic Logic Unit

ASB ARM System Bus

CPU Central processing unit

CG Clock Generator

CISC Complex instruction set computer

CSPR Current Program Status Register

DTE Data circuit-terminating equipment

DCE Data terminal equipment

DBGU Debug Unit

DA prevodník Digitálno-analógový prevodník

EEPROM Electrically Erasable Programmable ROM

EFC Embedded Flash Controller

FFPI Fast Flash Programming Interface

nFIQ Fast Interrupt Request

FIFO First In First Out

PS/2 IBM Personal System/2

ISP In-System Programming

JTAG Joint Test Action Group

LCD Liquid crystal display

MC Memory Controller

NMI Non-Maskable Interrupt

PIO Paralell Input Output Controller

PIT Periodic Interval Timer

PDC Peripheral DMA Controller

PLL Phase Locked Loop

PMC Power Management Controller

PWM Pulse Width Modulation

RAM Random Access Memory

ROM Read-Only Memory

RTT Real Time Timer

RS232 Recommended Standard 232/Serial port

RISC Reduced Instruction Set Computer

SD/MMC Secure Digital/MultiMediaCard

SPI Serial Peripheral Interface

SPI Serial Peripheral Interface

SP Stack Pointer

nIRQ Standard Interrupt Request

SRAM Static RAM

SDRAM Synchronous Dynamic RAM

SSC Synchronous Serial Controller

TC Timer/Counter

TTL Transistor–Transistor Logic

TWI Two Wire Bus

I2C Typ zbernice

UART Universal Asynchronous Receiver/Transmitter

USB Universal Serial Bus

USART Universal Synchronous/Asynchronous Receiver/Transmitter

USB UDP USB device port

VCP Virtual COM Port

Kapitola 1

Úvod

Už niekoľko dekád podlieha celá naša planéta konštantnému náporu digitalizácie. Kedysi analógové signály sú stále viac a viac nahrádzané spoľahlivejšími digitálnymi. Pojem digitálny sa pomaly stáva synonymom pre akúkoľvek elektroniku, s ktorou sa západný človek denno-denne stretáva. Je to preto, že čokoľvek, čo vyžaduje nejaké riadenie, priam volá po implementácii vnoreného systému. Pamäte, periférie a uprostred všetkého procesor. To je stručná schéma, ktorá dnes takpovediac vládne svetom. Pomáha vojakom v obrnenom vozidle rýchlo reagovať na vonkajšie podmienky a ovládať paľbu či navigáciu. Umožňuje existenciu Internetu, kde smerovače a prepínače, ktoré sú základom všetkých spojení, sú takisto vnorenými systémami. Riadenie výrobných a technologický procesov, elektronika v automobiloch či lietadlách, mikrovlnné rúry, videá, mp3 prehrávače - toto všetko sú zdanlivo úplne odlišné zariadenia, avšak postavené na spoločnej báze. Vo všetkých z nich obrovskou rýchlosťou prúdia reťazce jednotiek a núl zabezpečujúce ich plynulý chod.

Cieľom nášho tímu bude zhotoviť dva funkčné mikropočítače. Úspešné zvládnutie tohto projektu nám v rámci akademických podmienok umožní nahliadnuť do sveta vývoja a realizácie akýchkoľvek digitálnych systémov. Vďaka poznatkom, ktoré získame, budeme môcť lepšie porozumieť digitálnemu svetu okolo nás.

Kapitola 2

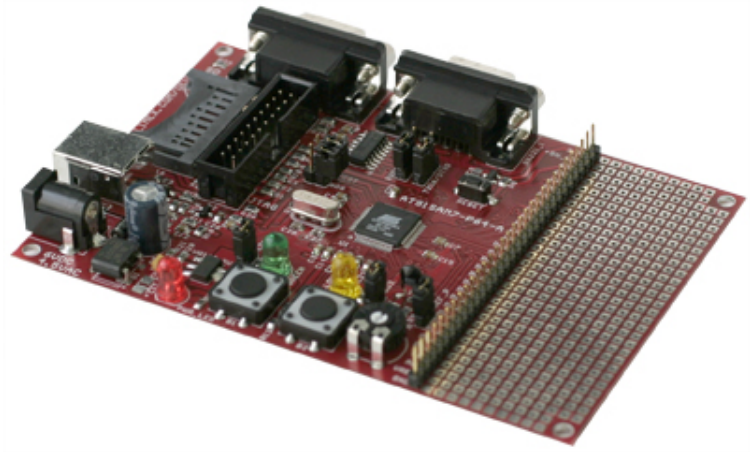
Analýza

2.1 Analýza existujúcich riešení

Táto kapitola opisuje existujúce riešenia v oblasti experimentálnych mikropočítačov.

2.1.1 SAM7-P64 development board

SAM7-P64 development board je komerčný produkt bulharskej firmy Olimex. Základ mikropočítača tvorí 16/32-bitový mikroprocesor AT91SAM7S64 s jadrom ARM7TDMI, ktorý obsahuje 64 kB flash pamäte ROM a 16 kB pamäte RAM, port USB 2.0, 10-bitový AD prevodník, 2 UART sériové porty, zbernicu I2C a rozhranie SPI. Všetky porty mikroprocesora sú vyvedené, čo umožňuje pripojenie rozširovacích modulov. Mikropočítač obsahuje JTAG konektor na programovanie a odladťovanie s využitím ARM-JTAG kábla z produkcie tejto firmy. Podporuje SD/MMC pamäťové karty.



Obr. 2.1: SAM7-P64 development board

Mikropočítač je realizovaný na 4-vrstvovej doske plošných spojov malých rozmerov. Viac o tomto mikropočítači sa možno dozvedieť v literatúre [1].

2.1.2 AT91SAM9260 development board

Ďalším zaujímavým produktom firmy Olimex je mikropočítač s procesorom AT91SAM9260 Základ mikropočítača tvorí 16/32-bitový mikroprocesor AT91SAM9260 s jadrom ARM9 a taktovacou frekvenciou 180 MHz. Obsahuje 512 MB NAND Flash pamäte ROM a 64 MB pamäte SDRAM. Obsahuje dva USB porty, jeden slúži na pripojenie ďalších zariadení a druhý na pripojenie samotného mikropočítača k nadriadenému systému. Vstavaný sieťový adaptér je Ethernet 100 Mb. Mikropočítač tiež obsahuje sériový port a podporuje SD/MMC pamäťové karty. Hodiny reálneho času sú napájane baterkou.

Jadro procesora ARM9 a veľká RAM a ROM pamäť umožňuje prevádzkovať OS Linux 2.6 a Windows CE. Viac o tomto mikropočítači sa možno dozvedieť v literatúre [2].

2.1.3 Mikropočítač na báze ATmega8515

Tento mikropočítač založený na mikroprocesore ATmega8515 navrhol Viktor Tlacháč a kol. v rámci tímového projektu. Procesor poskytuje internú flash pamäť programu, ktorá sa ľahko programuje vďaka podpore SPI rozhrania. Programovanie prebieha prostredníctvom jednoduchého ISP programátora, ktorý využíva mikroprocesor ATtiny2313P, komunikujúci prostredníctvom protokolu AVR910. Externá pamäť programu nie je podporovaná. Mikroprocesor tiež poskytuje PWM a analógový komparátor a internú pamäť RAM, ktorá je doplnená externou. Mikropočítač ďalej disponuje sériovým rozhraním pre pripojenie PC alebo mobilného telefónu, rozhraním USB pre pripojenie PC, tlačidlami prerušení, 6-miestnym 7-segmentovým displejom a 2 riadkovým LCD displejom. Dokumentácia k projektu je prehľadná a dobre napísaná. Mikropočítač je realizovaný na veľmi vysokej technickej aj koncepčnej úrovni, pôsobí veľmi dobrým dojmom. Viac o tomto mikropočítači sa možno dozvedieť v literatúre [3].

2.1.4 Mikropočítač na báze mikroprocesora 8086

Tento mikropočítač založený na mikroprocesore 8086 navrhol Michal Ondrovič a kol. v rámci tímového projektu. Dva obvody 27512 poskytujú spolu 128 kB EPROM pamäte programu. Pamäť programu s veľkosťou 64 kB pozostáva z dvojice obvodov 62256. Mikropočítač obsahuje dva vstupno-výstupné porty, jeden je RS232 a druhý je infračervený, vstupný port so spínačmi, výstupný port s indikáciou, tlačidlá prerušení a 6-miestny 7-segmentový displej. Projektová dokumentácia je prehľadná a projekt je na dobrej úrovni. Viac o tomto mikropočítači sa možno dozvedieť v literatúre [4].

2.1.5 Experimentálny mikropočítač

V roku 2005 Attila Štrba v rámci diplomovej práce navrhol a zrealizoval experimentálny mikropočítač DELUXIA-M1 s procesorom AT89S52, 64 kB pamäte programu a 32 kB pamäte údajov.

Tento mikropočítač ďalej poskytuje režim kompatibility s MPS51, rozhranie RS232C, rozhranie USB 2.0 a rozhranie PS/2 s vlastným procesorom. Prerušovací podsystém je realizovaný obvodom 8259A, mikropočítač tiež obsahuje dvojriadkový LCD displej, 6 miestny 7-segmentový displej a rozhranie pre TV prijímač.

Obr. 2.2: DELUXIA-M1

Mikropočítač pozostáva z viacerých nezávislých častí - modulov. Každý modul obsahuje svoju vlastnú riadiacu jednotku, ktorá pracuje nezávisle od hlavného mikroprocesora. Je navrhnutý s dôrazom na jednotnú výmenu informácií medzi modulmi a jadrom (univerzálnosť komunikácie), hardvérovú otvorenosť systému a jednoduché programovanie.

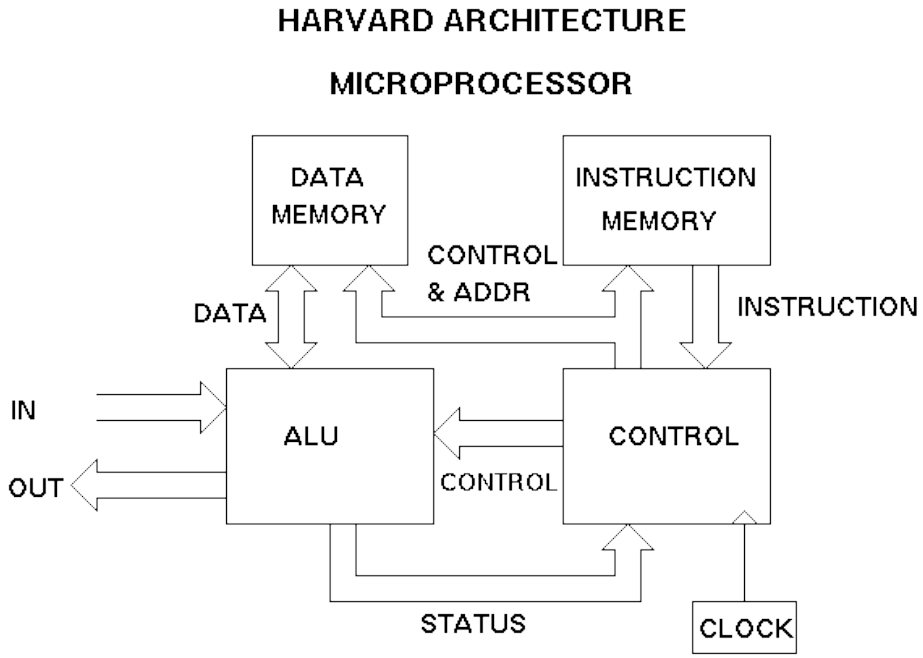
Ak by sme teda zhrnuli všetky faktory, musíme označiť tento mikropočítač za výborne zvládnutý. Je na ňom vidieť tvrdú prácu a snahu autora. Viac o tomto mikropočítači sa možno dozvedieť v použitej literatúre [5].

2.1.6 Zhrnutie analýzy existujúcich riešení

Boli opísané systémy, ktoré sa svojou funkcionalitou alebo charakterom podobajú mikropočítačom, ktorého zostrojenie je cieľom tohto projektu, pri čom sme získal veľkú dávku inšpirácie a aj sme sa veľa naučili. Niektoré systémy boli veľmi jednoduché a vhodné hlavne na výučbu, iné boli spracované skutočne profesionálne. V nasledujúcej kapitole sú opísane architektúry procesorov niektorých analyzovaných riešení.

2.2 Architektúra mikroprocesorov 8051

Architektúra 8051 bola vytvorená firmou Intel v roku 1980 a definuje mikroprocesor Harvardskej architektúry. Počítač podľa Harvardskej architektúry (na rozdiel od počítača podľa Von Neumannovej architektúry) obsahuje samostatné pamäte pre uchovávanie inštrukcií (programu) a dát (údajov).



Obr. 2.3: Harvardská architektúra

Najväčšiu popularitu dosiahla v 80-tych a počiatkom 90-tych rokov, v súčasnosti je nahradzovaná veľkým počtom iných procesorov, ktoré sú s ňou zároveň kompatibilné a rozširujú ju. Oficiálne označenie tejto architektúry je MCS 51.

Typický mikroprocesor architektúry 8051 obsahuje:

- CPU
- rozšírená sada inštrukcií CISC
- štyri obojsmerné vstupno-výstupné porty
- 8-bitovú dátovú zbernicu (preto sú tieto procesory označované ako 8-bitové)
- 16-bitovú adresnú zbernicu (umožňuje adresovať 216 bajtov = 64 kb ROM aj RAM pamäte)
- 128 bajtov RAM pamäte
- 4 kb ROM pamäte

- sériový port UART
- dve 16-bitové počítadlá
- režim úspory elektrickej energie

Veľkou výhodou architektúry 8051 je možnosť vykonávať logické operácie nad registrami a pamäťou RAM na bitovej úrovni. Vďaka tomu sa s obľubou využíva v priemyselnom riadení. Ďalšou výhodou sú štyri oddelené sady registrov. To výrazne znižuje oneskorenie obsluhy prerušenia, je to časovo menej náročné ako odkladanie registrov do zásobníka.

V jadre pôvodného procesora 8051 trvá jeden strojový cyklus 12 hodinových impulzov. Preto pri hodinovej frekvencii 12 MHz je procesor schopný vykonať 1 milión jednocyklových inštrukcií za sekundu. Každý strojový cyklus má dve časti, adresnú a dátovú. Počas adresnej časti cyklu sa na zbernici nachádza adresa pamäte, počas dátovej časti cyklu dáta, ktoré sa buď do pamäte zapisujú, alebo sa z nej čítajú.

Štvorica obojsmerných vstupno-výstupných portov umožňuje jednoduché pripojenie rôznych budičov alebo akčných členov. Dva porty môžu plniť funkciu 8-bitovej dátovej a 16-bitovej adresnej zbernice.

Moderný mikroprocesor architektúry 8051 podporuje ISP prostredníctvom protokolu JTAG a obsahuje 64 kb Flash EEPROM pamäte, prípadne môže mať aj vstavaný port USB, MP3 dekodér, radič IDE [1] či MultiMediaCard [2] a DA prevodník.

[1] Rozhranie IDE býva tiež označované ako ATA a umožňuje pripojenie mechaniky na čítanie optických diskov a diskových zariadení

[2] MultiMediaCard (MMC) je štandard opisujúci pamäťovú kartu malých rozmerov a veľkej kapacity, ktorá obsahuje pamäť typu Flash

2.3 Architektúra mikroprocesora 8086

8086 je 16-bitový mikroprocesor navrhnutý firmou Intel v roku 1978 a bol to prvý mikroprocesor architektúry x86, ktorá sa využíva dodnes v oblasti PC (od kancelárskych počítačov po výkonné pracovné stanice a servery).

Základné vlastnosti procesora 8086 sú:

- frekvencia procesora od 4,77 do 10 MHz
- rozšírená sada inštrukcií CISC
- všetky registre a dátové zbernice sú 16-bitové
- 20-bitová externá adresná zbernica umožňujúca prostredníctvom segmentácie adresovať 1 MB
- 16-bitová adresná zbernica pre vstupno-výstupné zariadenia
- dátová zbernica je multiplexovaná s adresnou (aby sa procesor vošiel do DIL-40 púzdra)
- podpora matematického koprocesora Intel 8087 na výpočty s pohyblivou desatinnou čiarkou
- možnosť spolupráce s ďalšími procesormi (viacprocesorové systémy)
- podpora nemaskovateľného prerušenia (NMI)

Maximálna veľkosť lineárne adresovateľnej pamäti je 64 kB, keďže registre majú veľkosť 16 bitov. Adresovať 1 MB pamäte je možné vďaka segmentácii pamäte. Ku konkrétnemu segmentu sa pristupuje cez segment registre.

Dĺžka inštrukcií je od 1 do 6 bajtov, preto výber a vykonanie inštrukcií je súbežné. Výber inštrukcie prebieha do 6 bajtovej fronty, čiže kratšie inštrukcie (práca s registrami procesora) sa vykonávajú rýchlejšie ako dlhšie inštrukcie (práca s pamäťou).

2.4 Mikroprocesory rady AVR

Táto rodina mikroprocesorov bola vyvinutá spoločnosťou Atmel v roku 1996. Jedná sa o jednoduchú 8-bitovú architektúru s inštrukčnou sadou typu RISC. Organizácia pamäte je založená na Harvardskej architektúre. Pamäť údajov a pamäť programu je teda fyzicky oddelená. Bola jednou z prvých rodín procesorov z prepisovateľnou Flash pamäťou priamo na čipe procesora.

Dá sa povedať, že AVR je modernou náhradou staršej rady procesorov typu 8051. Niektoré z prvých AVR procesorov mali dokonca rovnaké rozostavenie pinov ako 8051. V praxi sa dodnes používajú oba typy, my sme sa však v projekte rozhodli zamerať sa na AVR. Je to modernejší a bežnejší procesor. V rámci akademických podmienok je využívaný a menej preskúmaný ako 8051. Pre náš tím je preto väčšou výzvou a bude podľa nás mať väčší prínos pre študentov, pretože sa s ním majú možnosť málokde stretnúť.

2.4.1 Rozdelenie procesorov AVR

Procesory rodiny AVR sa delia na tri základné skupiny:

- TinyAVR
  - 1-8 kB pamäte programu
  - 8-20 pinové puzdro
  - obmedzený počet periférií
- megaAVR
  - 4-256 kB pamäte programu
  - 28-100 pinové puzdro
  - rozšírená inštrukčná sada - inštrukcie pre zaobchádzanie s pamäťami väčšieho rozsahu
  - rozšírený počet periférií
- AVR so špeciálnym aplikačným zameraním
  - procesory s vlastnosťami, ktoré nie sú v obsiahnuté v ostatných radách - napr. LCD radič, USB radič

2.4.2 Základná architektúra AVR

Pamäť údajov - RAM

Pamäť typu SRAM je integrovaná priamo na čipe. Takže nie je potrebná externá pamäť.

Pamäť programu - Flash

Mazateľná Flash pamäť je tak isto integrovaná priamo na čipe. Napriek tomu, že ide o 8-bitovú architektúru, každá inštrukcia v pamäti zaberie jedno alebo dve 16-bitové slová.

Interné registre

Pracovné registre procesora sú mapované ako prvých 32 pamäťových adries (0000 - 001Fh), ďalej nasleduje 64 vstupno-výstupných I/O registrov (0020 - 005Fh). Samotná pamäť SRAM začína po sekcii registrov, teda od 0060h.

Napriek tomu, že pracovné a I/O registre majú špecifické adresné schémy a optimalizované inštrukcie pre prístup k nim, môžu byť adresované ako súčasť pamäte SRAM. Programátor má teda k dispozícii oba tieto prístupy.

EEPROM

Niektoré typy procesorov majú internú pamäť typu EEPROM. Nie je však mapovaná v rámci adresovateľného pamäťového priestoru. Prístup k nej je možný rovnako ako k externým periférnym zariadeniam vďaka špeciálnym registrom a inštrukciám zápisu alebo čítania. Je preto omnoho pomalšia ako pamäť SRAM. Počet možných zápisov do EEPROM pamäte je obmedzený na 100 000.

Vykonávanie inštrukcií

Procesory AVR majú implementované jednoduché prúdové spracovanie inštrukcií. Znamená to, že zatiaľ, čo je jedna inštrukcia vyberaná, druhá sa vykonáva. Rodina AVR bola navrhnutá pre efektívne vykonávanie kompilovaného kódu jazyka C.

Rýchlosť MCU

Bežne je možný frekvenčný rozsah 0-16 MHz, niektoré vyššie rady podporujú až 20 MHz rýchlosti procesora. Všetky AVR procesory majú zabudovaný interný oscilátor, čiže nie je potrebný externý kryštál ani obvody s touto funkciou. Keďže väčšina inštrukcií je jednotaktových, dá sa povedať že AVR dosahuje 1 MIPS na 1 MHz.

2.4.3 ATmega32

Ide o predstaviteľa rady procesorov AVR. Vybrali sme si ho, pretože poskytuje postačujúcu internú Flash pamäť programu 32 kb a internú pamäť údajov SRAM 2 kb a 1024 B EEPROM pamäte. Nebude teda potrebné zapájať externé pamäte, čo je veľkou výhodou pri oživovaní

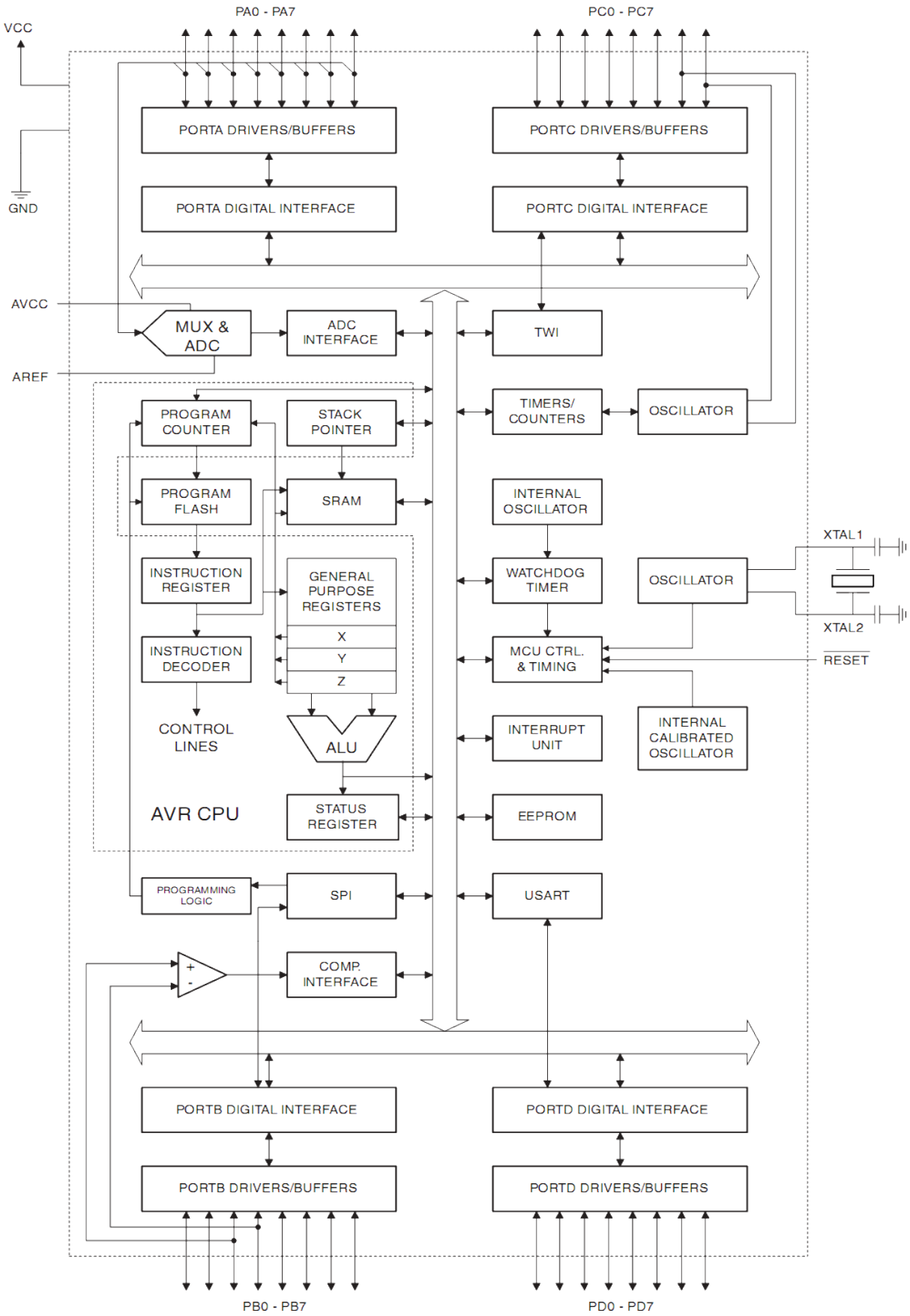
mikropočítača. Budeme sa teda môcť lepšie zamerať na funkčnosť a korektnú činnosť periférií. Procesor je schopný pracovať na frekvencii do 16 MHz. Ďalej poskytuje:

- JTAG rozhranie
- Dve 8 a jedno 16-bitové počítadlo
- 4 PWM kanály
- bitovo orientované sériové rozhranie
- programovateľný Serial USART
- možnosť ISP programovania
- Master/Slave SPI rozhranie pre programovanie Flash a EEPROM
- programovateľný Watchdog timer so separátnym oscilátorom priamo na čipe
- možnosť interných a externých prerušení
- 6 režimov spánku:
  - Idle
  - ADC Noise Reduction
  - Power-save
  - Power-down
  - Standby
  - Extended Standby

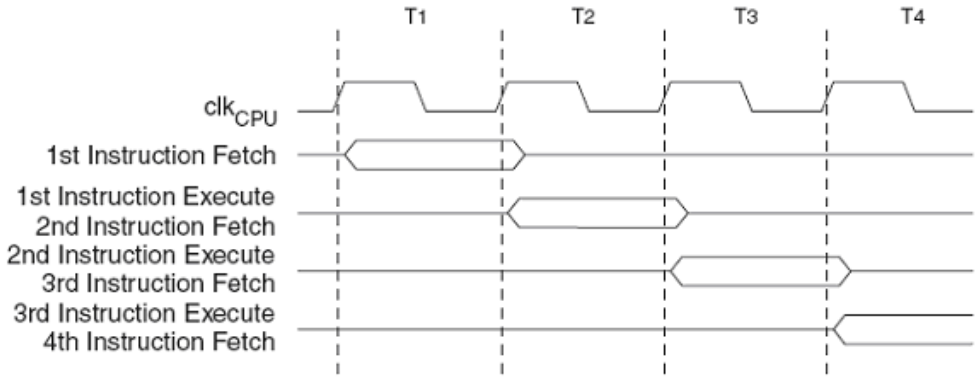
Všetky súčasti procesora sú graficky zobrazené na blokovej schéme na obrázku 2.4

Najväčšou výhodou RISC architektúry, na ktorej ja tento procesor postavený, je schopnosť vyberať a vykonávať inštrukcie veľmi rýchlo. Takmer vo všetkých prípadoch je schopný vybrať či vykonať inštrukciu počas jedného hodinového taktu.

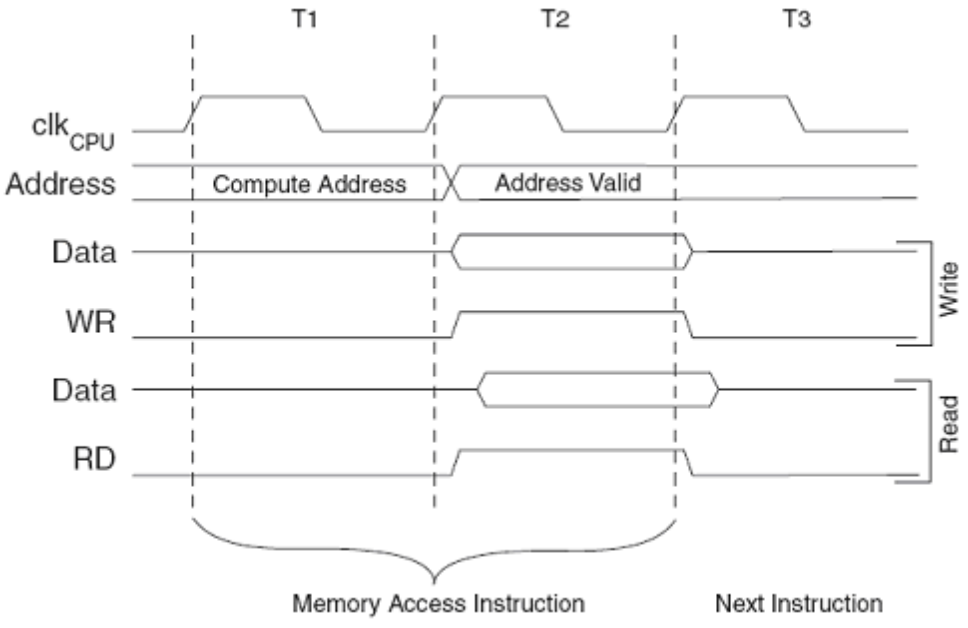
Prístup do internej pamäte údajov SRAM trvá 2 hodinové takty.



Obr. 2.4: Bloková schéma procesora ATmega32



Obr. 2.5: Časový diagram výberu a vykonania inštrukcie



Obr. 2.6: Časový diagram práce s pamäťou

2.5 Mikroprocesory rady AT91SAM7

Tieto mikroprocesory vyrába firma Atmel. Na čipe obsahujú pamäť RAM, FLASH, ROM. Najjednoduchšie modely tejto rady tvoria premostenie medzi 8-bitovými a 32-bitovými mikroprocesormi. Sú zamerané na aplikácie riadenia v reálnom čase, preto v sebe inkorporujú štandardné rozhrania ako 8-bitové mikroprocesory, ale poskytujú zvýšený výkon a 32-bitový adresový priestor.

Rodina AT91SAM7S je rozšírená rodinou AT91SAM7SE, ktorá pridáva externú zbernicu, ktorá umožňuje pripojiť viac ako gigabajt externej pamäti SRAM alebo FLASH.

Aplikácie, ktoré vyžadujú pripojenie typu Ethernet, USB a CAN sú podporované rodinou AT91SAM7X. Rodina AT91SAM7XC ďalej pridáva možnosť šifrovania šifrou AES a Tripple DES.

Ide o mikroprocesory založené na 32-bitovom jadre ARM7TDMI. Táto architektúra je typu RISC. Inštrukcie architektúry RISC sú jednoduchšie na dekódovanie ako inštrukcie architektúry CISC, čo umožňuje:

- vysokú priepustnosť inštrukcií
- vysokú rýchlosť odozvy prerušení v reálnom čase
- malú makrobunku procesora

Na zvýšenie rýchlosti vykonávania inštrukcií, jadro ARM7TDMI implementuje ich prúdové vykonávanie. Jednotka vykonávania inštrukcií je rozdelená na tri časti, čo umožňuje simultánne vykonávanie jednotlivých činností pri spracúvaní inštrukcií. Konkrétne ide o tieto činnosti:

- výber inštrukcie (fetch)
- dekódovanie inštrukcie (decode)
- vykonanie inštrukcie (execute)

Toto jadro je založené na architektúre Von Neumann. Používa spoločnú 32-bitovú zbernicu pre inštrukcie a dáta. Údaje môžu byť dlhé:

- 8-bitov (byte)
- 16-bitov (half-word)
- 32-bitov (word)

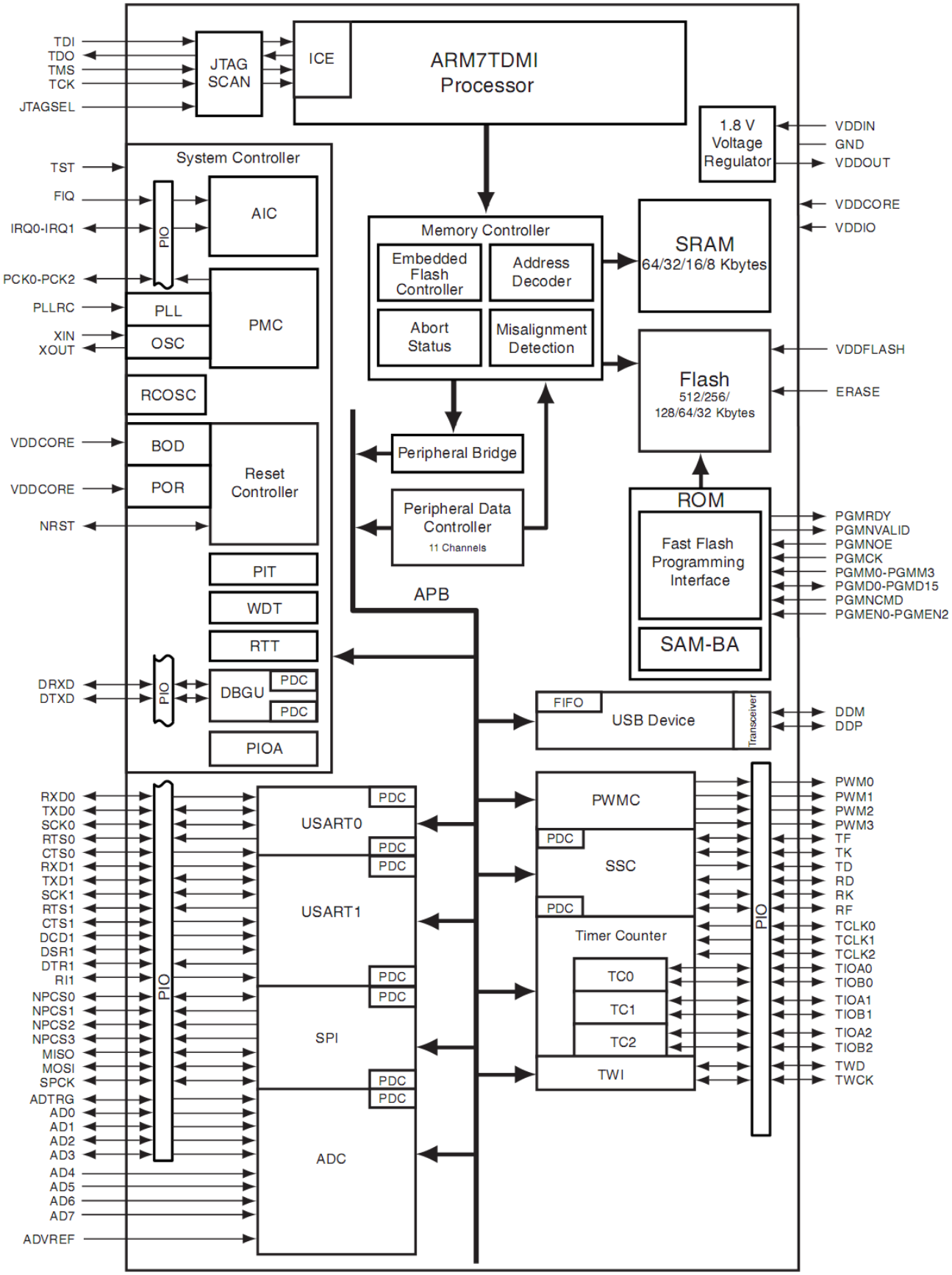
32-bitové slová musia byť zarovnané na hranicu 4 bajtov, 16-bitové na hranicu 2 bajtov. ARM7TDMI vie spracovávať dve inštrukčné sady:

- 32-bitovú ARM inštrukčnú sadu
- 16-bitovú THUMB inštrukčnú sadu

32-bitová sada poskytuje plný výkon procesora. 16-bitová poskytuje vyššiu hustotu uloženia inštrukcií t.j. program v THUMB inštrukčnej sade je menší a zaberá menšie množstvo pamäti. 16-bitová inštrukčná sada THUMB je podmnožina najčastejšie používaných 32-bitových inštrukcií.

Pre mikroprocesor založenom na jadre ARM sme sa rozhodli kvôli tomu, že v dnešnej dobe je tento mikroprocesor veľmi často používaný vo vnorených systémoch, ktoré vyžadujú vyšší výkon. Avšak tieto mikroprocesory sú určené pre moderné aplikácie, kde sa dá stretnúť s povrchovou montážou a nezriedka treba použiť aj viac ako 2 vrstvovú dosku plošných spojov. A takáto doska sa v amatérskych podmienkach nedá zhotoviť. Preto aj firmy, ktoré sa zaoberajú predajom elektronických súčiastok s týmto počítajú a predávajú len jednoduchšie modely týchto mikroprocesorov. Zložitejšie typy sú len na objednávku vo väčších množstvách. Procesor, dostupný pre náš projekt, je AT91SAM7S64.

Tento mikroprocesor obsahuje 64 kb pamäti FLASH, 16 kb pamäti SRAM, rozhranie USB, USART. Jadro mikroprocesora dokáže bežať až na frekvencii 55 Mhz. Interná pamäť FLASH je programovateľná pomocou ISP (in system programming) a to cez usb port, alebo cez debug rozhranie. Ďalšia možnosť ako programovať tento mikroprocesor je cez FFPI, a to buď paralelne alebo sériovo. Pri paralelnom programovaní musí byť mikroprocesor nastavený v špeciálnom test móde a pripojený do programátora cez vyhradené kontakty a ostatné musia byť nepripojené. Pri sériovom programovaní je obsah pamäti FLASH modifikovaný cez JTAG. Oba tieto módy (sériový aj paralelný) sú určené na programovanie pri veľkovýrobe. Na výrobu v malých množstvách je určené programovanie cez ISP.



Obr. 2.7: Bloková schéma AT91SAM7S64

2.6 Procesor ARM7TDMI

Ide o procesor architektúry ARM a verzie v4T. Tento procesor podporuje 7 módov vykonávania programu:

Používateľský mód : normálny mód ARM na vykonávanie programu

FIQ mód : navrhnutý na vysoko-rýchlostný dátový prenos

IRQ mód : používa sa pri ošetrovaní prerušení

Mód operačného systému : chránený režim operačného systému

Mód virtuálnej pamäte : implementuje virtuálnu pamäť a ochranu pamäti

Systémový mód : privilegovaný používateľský mód operačného systému

Nedefinovaný mód : umožňuje softvérovú emuláciu hardvérových koprocesorov

Tento procesor obsahuje spolu 37 registrov a to:

- 31 registrov na všeobecné použitie
- 6 stavových registrov

Týchto 31 registrov nie je prístupných naraz, ale sú prístupné podľa toho, v akom móde sa procesor nachádza. V každom okamihu je prístupných 16 registrov a ostatné sú synonymné registre, ktoré urýchľujú spracovávanie výnimiek. Register 15 je počítadlo programu (PC), ktorý sa môže použiť vo všetkých inštrukciách, ktoré odkazujú na dáta relatívne k aktuálnej inštrukcii. Register 14 obsahuje návratovú adresu po volaní podprogramu. Register 13 je podľa softvérovej konvencie ukazovateľ na vrch zásobníka (SP).

Stavový register programu CPSR obsahuje:

- štyri príznaky ALU negatívny, pretečenie, prenos, nula
- dva bity na zakázanie prerušení
- jeden bit na indikáciu inštrukcií ARM alebo Thumb
- päť bitov na zakódovanie aktuálneho módu procesora

Jadro ARM7TDMI podporuje päť typov výnimiek a pre každý typ má vlastný mód:

- rýchle prerušenie
- normálne prerušenie

- pokus o vykonanie nedefinovanej inštrukcie
- pamäťové prerušenia (na implementovanie virtuálnej pamäti)
- softvérové prerušenia

Reset kontrolér

Kontrolér, ktorý spracováva všetky resety systému bez externých komponentov. Riadi externý reset, reset procesora a periférií.

Časovač reálneho času RTT

Tento časovač má 32-bitov a je určený na počítanie sekúnd. Dokáže generovať periodické prerušenie a takisto dokáže generovať prerušenie vo vopred určenom čase.

Paralelný vstupno-výstupný kontrolér PIO

Tento kontrolér riadi 32 vstupno-výstupných liniek, ktoré môžu byť nakonfigurované ako vstupné alebo výstupné, alebo môžu byť pripojené k periférii. V tom prípade stav týchto liniek určuje periféria. Ak je linka nakonfigurovaná ako vstupno-výstupná, tak má nasledovné vlastnosti:

- dokáže generovať prerušenie pri zmene logického stavu
- má filter, ktorý filtruje impulzy kratšie ako polovica hodinového cyklu

Časovač periódy PIT

Poskytuje zdroj periodického prerušenia, čo možno využiť pre plánovač operačného systému.

Watchdog Timer

Môže byť použitý ako prevencia proti tomu, aby sa systém dostal do zacykleného stavu. Ide o 12-bitové počítadlo, ktoré sa dekrementuje a tým sa dá dosiahnuť čas na obnovu počítadla až 16 sekúnd, ak pomalé hodiny mikrokontroléra sú taktované na 32Khz.

Pamäťový kontrolér MC

Pamäťový kontrolér manažuje prístup na ASB a kontroluje prístup procesora ARM7TDMI a kontroléra PDC. Obsahuje arbiter zbernice, adresový dekodér a vnorený flash kontrolér EFC.

Vnorený flash kontrolér EFC

Je súčasťou pamäťového kontroléra a poskytuje rozhranie medzi pamäťou FLASH a internou 32-bitovou zbernicou.

Rozhranie rýchleho programovania FLASH FFPI

Poskytuje paralelné a sériové rozhranie na programovanie. Je určené na programovanie vo veľkovýrobe. Pri paralelnom móde sa rozhranie považuje za pamäť typu flash, pri sériovom programovaní sa programuje na základe štandardu JTAG 1149.1.

Dma kontrolér PDC

Prenáša údaje medzi sériovými perifériami na čipe ako sú UART, USART, SSC, SPI a pamäťou. Pri použití PDC na dátové prenosy sa vyhýbame použitiu mikroprocesora a tým aj réžie na správu prerušení čím celkový prenos prenos vyžaduje menej hodinových cyklov. Týmto spôsobom sa znižuje aj spotreba mikroprocesora. Kanály PDC sú implementované v pároch, kde jeden kanál slúži na príjem a druhý na vysielanie dát z periférií.

Kontrolér prerušení AIC

Dokáže spravovať 32 zdrojov prerušení. Rozpoznáva 8 úrovní priorít. Každý zdroj prerušenia je individuálne maskovatelný. Ovláda vstupy nIRQ a nFIQ procesora ARM. Vstupy AIC sú buď z interných periférií alebo sú externé.

Generátor hodinového signálu

Skladá sa z troch častí PLL, hlavného oscilátora a RC oscilátora a poskytuje 3 zdroje hodinových signálov:

- SLCK (slow clock) jediné permanentný zdroj hodinového signálu v systéme
- MAINCK (main clock) je výstup z hlavného oscilátora
- PLLCK je výstupom z deličky a PLL bloku

Kontrolér správy napájania PMC

Optimalizuje spotrebu energie takým spôsobom, že riadi hodinový signál jednotlivých periférií. Riadi hodinový signál mnohých periférií na mikrokontroléri a takisto aj samotného procesora ARM7TDMI.

Jednotka ladenia programu DBGU

Poskytuje rozhranie na sprístupnenie ladiacich možností Atmel ARM systémov. Pozostáva z 2 pinového UART portu.

Sériové rozhranie periférií SPI

Ide o sériové rozhranie, ktoré poskytuje komunikáciu s externými zariadeniami v móde Master a Slave. Takisto umožňuje komunikáciu s externým procesorom. Vo svojej podstate ide o posuvný register, ktorý vysiela jednotlivé bity inému zariadeniu SPI. Na tejto zbernici je jeden master, ktorý môže vysielať dáta jednému alebo viacerým podriadeným slave zariadeniam. V danom časovom okamihu môže vysielať len jedno podriadené slave zariadenie. Toto rozhranie pozostáva z dvoch dátových liniek a z dvoch kontrolných liniek:

- MOSI (master out slave in) ide o prenos z posuvného registra nadriadeného zariadenia do posuvného registra podriadeného zariadenia
- MISO (master in slave out) ide o prenos z posuvného registra podriadeného zariadenia do posuvného registra nadradeného zariadenia.
- SPCK (serial clock) tento signál je riadený nadriadeným zariadením a kontroluje tok dát
- NSS (slave select) ide o výberový signál rozhrania, ktorým sa vyberá podriadené zariadenie

Dvojvodičové rozhranie TWI

Toto rozhranie pozostáva z dvoch vodičov, z ktorých je jeden dátový a druhý je zdroj hodinového signálu. Toto rozhranie dovoľuje prenosové rýchlosti až do 400 Kbps. Umožňuje pripojiť sériové pamäte EEPROM (Electrically Erasable Programmable Read Only Memory). Je možné konfigurovať prenosovú rýchlosť, čím sa dá prispôsobiť k prenosovým rýchlostiam rôznych zariadení.

Univerzálny synchrónny asynchrónny prijímač vysielač USART

Ide o plne duplexnú synchrónnu asynchrónnu sériovú dátovú linku. Formát rámca je programovateľný (počet dátových bitov, počet stop bitov, parita). USART podporuje spojenie s PDC, čo dovoľuje dátové prenosy z prijímača a do vysielača z hlavnej pamäte.

Synchrónny sériový kontrolér SSC

Poskytuje synchrónnu sériovú linku na komunikáciu s externými zariadeniami. SSC podporuje mnoho sériových štandardov na komunikáciu používaných v audio a telekomunikačných sektoroch.

Časovač-Počítadlo TC

Integruje tri 16-bitové kanály. Každý z nich môže byť naprogramovaný na plnenie rôznych funkcií:

- meranie frekvencie
- počítanie udalostí
- meranie intervalov
- generovanie impulzov
- modulácia šírkou impulzu

Kontrolér modulácie šírkou impulzu PWM Makrobunka PWM kontroluje niekoľko kanálov. Každý kanál dokáže modulovať jeden výstupný signál. Charakteristika výstupného signálu ako je perióda, polarita a pomer dĺžky trvania úrovne 1 k úrovni 0 sú programovateľné.

Rozhranie USB UDP

Je kompatibilné so štandardom USB v2.0. AT91SAM7S64 obsahuje 4 koncové body a každý podporuje niekoľko USB prenosových módov.

Koncový bod	Typ prenosového módu
0	Control-Bulk-Interrupt
1	Bulk-Iso-Interrupt
2	Bulk-Iso-Interrupt
3	Control-Bulk-Interrupt

Tabuľka 2.1: Typy prenosových módov USB

Analógovo-digitálny prevodník ADC

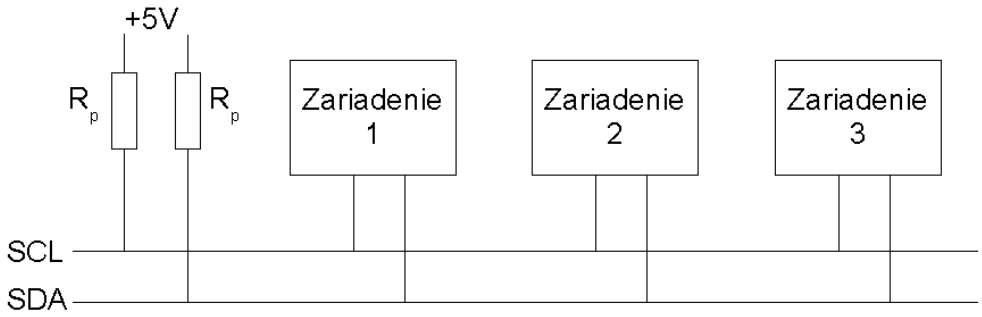
Tento prevodník je založený na postupnej aproximácii. Obsahuje analógový multiplexor, ktorí multiplexuje 8 vstupných liniek. Ide o 8 alebo 10-bitový programovateľný prevodník. Výsledky konverzie sú dostupné v spoločnom registri ako aj individuálnom registri pre danú linku.

2.7 Komunikačné rozhrania

2.7.1 Zbernica I2C [9]

I2C je zbernica používaná v mikroprocesorových systémoch a vnorených systémoch. Využíva sa hlavne v tých oblastiach, kde nie je dôležitá vysoká priepustnosť zbernice a na pripojenie malých sériových EEPROM pamätí, ktoré slúžia na ukladanie napr. konfigurácie, alebo displejov.

Zbernica pozostáva z dvoch vodičov označených ako SCL a SDA. SCL slúži na taktovanie, synchronizuje dátový prenos cez I2C zbernicu. SDA prenáša dáta. Tieto dva vodiče sú pripojené na všetky zariadenia na I2C zbernici. Všetky zariadenia zároveň využívajú ešte jeden vodič, ktorý slúži ako signálová zem. Vývody SDA a SCL procesora sú zväčša realizované jednoduchým budičom vo vnútri mikroprocesora, ktorý má otvorený kolektor. To znamená, že budič nie je schopný nastaviť tieto vývody na úroveň logickej jednotky a preto je nutné použiť pull-up rezistory.

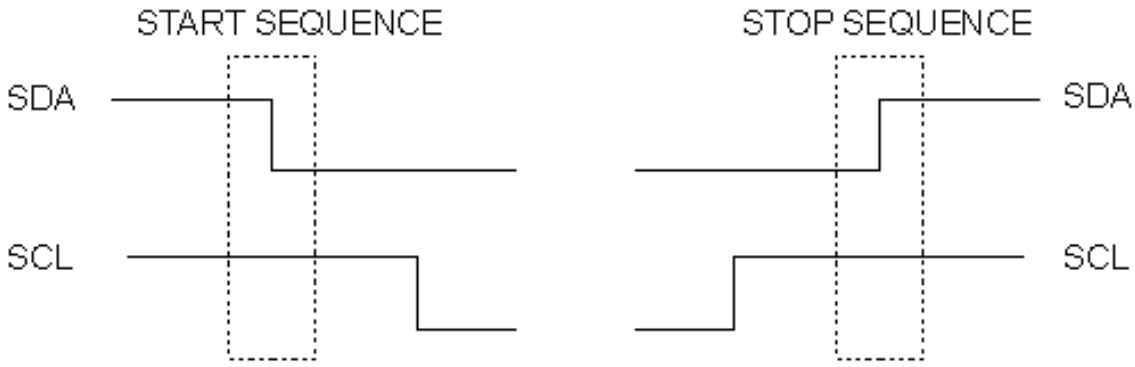


Obr. 2.8: Zapojenie I2C zbernice

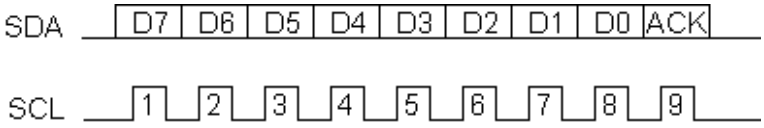
Pri komunikácii cez I2C zbernicu sa rozlišujú zariadenia na master a slave. Master je to jedno zariadenie, ktoré ovláda SCL (existuje aj možnosť viacerých Master zariadení, ale v tomto prehľade nebude opísaná kvôli svojej zložitosti a redundancii v rámci tohto projektu). Slave sú ostatné zariadenia a na rozdiel od zariadenia typu Master nedokážu zahájiť komunikáciu.

Dôležitou vlastnosťou I2C zbernice je, že SDA sa môže meniť len keď SCL má hodnotu logickej nuly. Len počas "start sequence" a "stop sequence" sa SDA môže zmeniť, zatiaľ čo SCL má hodnotu logickej jednotky. Tieto sekvenciu určujú začiatok a koniec komunikácie. Štandardná rýchlosť taktovania SCL je do 100kHz. Ak zariadenie Slave ešte nemá nachystané potrebné dáta, môže SCL podržať na úrovni logickej 0 a tým predĺžiť takt ("clock stretching").

Dáta sú prenášané ako séria ôsmych bitov. Bity sú vysielané na SDA v poradí od najvyššieho významového bitu. Po každých ôsmych bitoch prijímač vyšle späť vysielaču potvrdzovací bit, takže ôsmym preneseným bitom zodpovedá 9 pulzov SCL. Ak má potvrdzovací bit hodnotu 0, tak je prijímač pripravený prijímať ďalšie dáta. Ak má tento bit hodnotu 1, zariadenie nie je schopné prijať ďalšie dáta a zariadenie Master by malo ukončiť spojenie prostredníctvom "stop sequence".

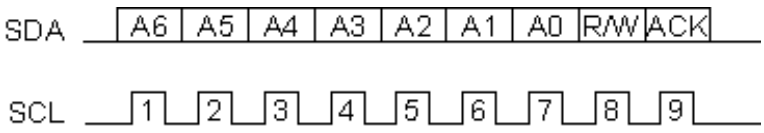


Obr. 2.9: Start sequence a stop sequence



Obr. 2.10: Prenos jedného bajtu

Zbernica I2C tiež podporuje adresovanie zariadení, pričom adresy majú 7 alebo 10-bitov (výskyt 10-bitových adries je zriedkavý). Pri odosielaní tejto adresy sa využije 8-bitový "rámec" komunikácie cez I2C, pričom prvých 7-bitov tvorí adresa a najnižší významový bit určuje, či sa zahajuje čítanie (hodnota 1) alebo zápis (hodnota 0) z adresovaného zariadenia.



Obr. 2.11: Odoslanie adresy a kódu operácie

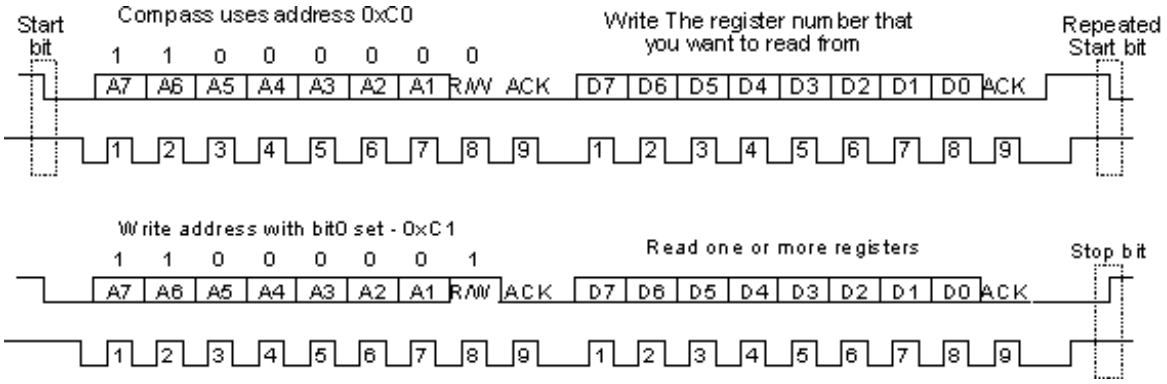
Priebeh komunikácie pri zapisovaní na zariadenie:

1. Zariadenie Master odošle "start sequence", čo upozorní všetky zariadenia Slave na začiatok komunikácie
2. Zariadenie Master odošle adresu zariadenia a typ operácie zápis. Zariadenie Slave, ktoré tejto adrese zodpovedá, pokračuje v prijíme dát, ostatné zariadenia komunikáciu ignorujú
3. Master odošle číslo registra zariadenia Slave, do ktorého chce zapisovať
4. Master odošle bajt, ktoré sa majú zapísať
5. Master pokračuje odosielaním ďalších bajtov, ktoré sa v zariadení Slave automaticky zapíšu do ďalších registrov, keďže zariadenie Slave si automaticky inkrementuje adresu registra po každom bajte

6. Odoslanie "stop sequence" zariadením Master

Priebeh komunikácie pri čítaní zo zariadenia:

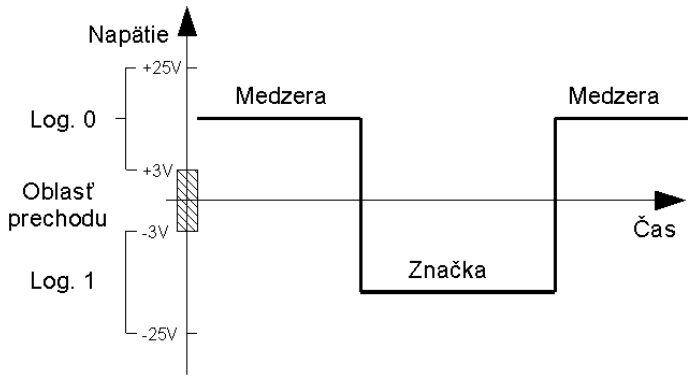
1. Zariadenie Master odošle "start sequence"
2. Zariadenie Master odošle adresu zariadenia a typ operácie zápis
3. Master odošle číslo registra zariadenia Slave, z ktorého budú dáta čítané
4. Zariadenie Master znova odošle "start sequence"
5. Zariadenie Master odošle adresu zariadenia a typ operácie čítanie
6. Master načíta bajt zo zariadenia
7. Odoslanie "stop sequence" zariadením Master



Obr. 2.12: Postupnosť bitov pri čítaní zo zariadenia

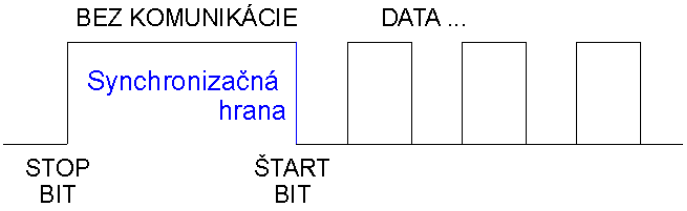
2.7.2 Rozhranie RS232 [10]

RS232 je rozhranie na prenos informácii, ktoré bolo pôvodne vytvorené na komunikáciu dvoch zariadení do vzdialenosti 20 m. Z dôvodu väčšej odolnosti voči vonkajším rušeniam rozhranie využíva pre logické úrovne napäťové úrovne, uvedené na obrázku 2.13.



Obr. 2.13: Napäťové úrovne RS232

Logická 1 je indikovaná zápornou úrovňou, zatiaľ čo logická 0 je prenášaná kladnou úrovňou. Sériové port obsiahnutý v analyzovaných mikroprocesoroch pracujú s logickými úrovňami TTL, preto bude potrebné využiť budič sériového portu. Výber konkrétneho modelu je obsiahnutý v kapitole 4 (Návrh).



Obr. 2.14: Synchronizácia zostupnou hranou štartovacieho impulzu

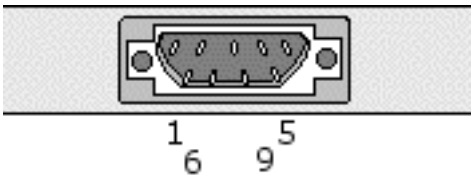
Prenos informácii prebieha asynchrónne, pomocou pevne nastavenej prenosovej rýchlosti a synchronizácie zostupnou hranou štartovacieho impulzu

Konektory a káble

V mikropočítačoch sa na rozhranie RS232 používajú predovšetkým konektory Cannon 9, ktorých zapojenie a význam vývodov je nasledovný.

Signály DTR, DSR, RTS, CTS plnili fukciu handshakingu pri poloduplexných zariadeniach [3]. V dnešnej dobe plneduplexných zariadení ale už praktický význam nemajú. V mikroprocesorových systémoch sa prakticky využívajú len signály RXD, TXD a SGND. Rozhranie DTE - DCE

[3]Poloduplexné zariadenie na rozdiel od plneduplexného zariadenia umožňuje v jednom čase komunikáciu len jedným smerom, čiže buď príjem, alebo odosielanie



Obr. 2.15: Konektor Cannon 9

1	DCD - Data Carrier Detect	Detekcia nosnej frekvencie. Modem oznamuje terminálu, že na telefónnej linke detekoval nosnú frekvenciu.
2	RXD - Receive Data	Tok dát z modemu (DCE) do terminálu (DTE).
3	TXD - Transmit Data	Tok dát z terminálu (DTE) do modemu (DCE).
4	DTR - Data Terminal Ready	Terminál týmto signálom oznamuje modemu, že je pripravený komunikovať.
5	SGND - Signal Ground	Signálová zem.
6	DSR - Data Set Ready	Modem týmto signálom oznamuje terminálu, že je pripravený komunikovať.
7	RTS - Request to Send	Terminál týmto signálom oznamuje modemu, že komunikačná cesta je voľná.
8	CTS - Clear to Send	Modem týmto signálom oznamuje terminálu, že komunikačná cesta je voľná.
9	RI - Ring Indicator	Indikátor zvonenia. Modem oznamuje terminálu, že na telefónnej linke detekoval signál zvonenia.

Tabuľka 2.2: Význam vývodov portu RS232 pre typ konektora Cannon 9

využíva viacero typov káblov. Všetky DTE - DCE káble sú priame a vývody sú prepojené 1:1. DTE - DTE a DCE - DCE káble patria medzi krížené. Kábel DTE - DCE sa nazýva Straight Cable (priamy), DTE - DTE sa nazýva Null - Modem a DCE - DCE sa nazýva Tail Circuit.

Parametre prenosu

Na starých termináloch IBM, ktoré sa používali len ako textové konzoly, ušetrili ich návrhári jeden bit prenosu a používali len 7-bitový prenos, ktorý umožňoval 128 kombinácií. Dnes sa v praxi nepoužíva, ale stal sa štandardom. Využíva sa prenos 8 bitov, za ktorými nasleduje stop bit.

Stop bit definuje ukončenie rámca. Zároveň poskytuje prijímaču dostatok času spracovať prijaté dáta.

Typická rýchlosť rozhrania je 2400, 4800, 9600, 19200, 38400 alebo 56700 bps. Možné sú aj vyššie rýchlosti, ale tieto pre účely mikropočítačov postačujú.

Zabezpečenie prenosu

Parita je najjednoduchší spôsob ako bez nárokov na výpočtový výkon zabezpečiť prenos dát. Vo vysielacom zariadení sa sčíta počet jednotkových bitov a doplní sa paritným bitom tak, aby bola zachovaná vopred dohodnutá podmienka nepárneho nebo párneho počtu jednotkových bitov. Existuje tiež space parity (tzv. nulová parita). Paritný bit má vtedy hodnotu logickej nuly. Využíva sa napríklad pri komunikácii 7-bitového zariadenia s 8-bitovým, kde paritný bit nahradzuje logickou nulou posledný bit v byte, tím je zachovaná kompatibilita s 8-bitovým prenosom.

Handshaking predstavuje mechanizmus riadenia toku dát a predstavuje potvrdenie prijímu dát alebo pripravenosti k prenosu a jeho zahájeniu na úrovni hardvérového alebo softvérového rozhrania. Prebieha na úrovni komunikačných protokolov (ZMODEM, KERMIT...), pomocou bežného dátového kanálu prijímač vysielaču oznámi, či je schopná dáta prijímať a spracovávať.

2.7.3 USB - Universal Serial Bus [11]

Základné vlastnosti USB :

- jednoduchosť použitia pre koncových používateľov
- jednoduchý model pre kabeláž a konektory
- koncový užívateľ je izolovaný od elektrický detailov zbernice
- samoidentifikujúce periférne zariadenia, automatické mapovanie funkcie na ovládač a konfigurácia
- dynamicky pripojiteľné a rekonfigurovateľné periférne zariadenia

Široké spektrum aplikácií:

- vhodné pre zariadenia s prenosovými pásmami od niekoľkých Kb/s až po Mb/s
- podporuje isochrónne a asynchrónne prenosy cez jednu množinu káblov
- podporuje súčasné spracovanie dát viacerých pripojených zariadení
- podporuje súčasné pripojenie až 127 fyzických zariadení
- podporuje viacnásobný prenos dátových tokov a tokov správ medzi host-om a zariadením
- dovoľuje zložené zariadenia
- nízke režijné náklady na protokol umožňujúce vyššie zaťaženie zbernice

Isochrónne prenosové pásmo:

- garantované prenosové pásmo a nízke oneskorenia vhodné pre telefón, audio, at .
- isochrónna komunikácia môže používa celé prenosové pásmo zbernice

Flexibilnosť:

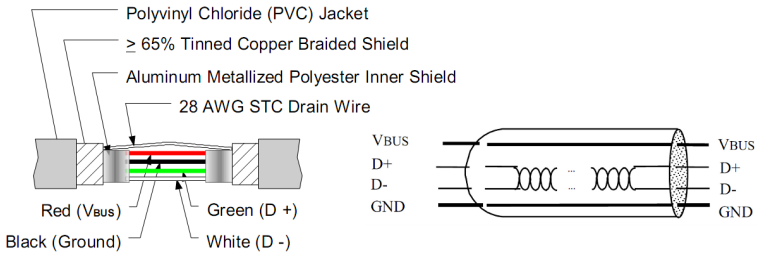
- podpora rôzne veľkosti paketov, čo umožňuje zariadeniam mať rôzne veľkosti vyrovnávacích bufferov
- dovoľuje rôzne dátové rýchlosti pre zariadenia prispôsobením veľkosti bufferov pre pakety a oneskorení

Prehľad architektúry

Fyzická topológia USB pozostáva zo spojenia downstream portu hubu k upstream portu iného hubu alebo zariadenia. USB môže pracovať v troch rýchlostiach. High-speed 480Mb/s, Full-speed 12Mb/s, požaduje použitie tieneného kábla s dvomi napájacími vodičmi a s dvomi krútenými signálovými vodičmi(twisted pair). Low-speed 1.5Mb/s, zvoľňuje požiadavky na kabeláž. Low-speed káble nevyžadujú tienenie alebo krútené páry signálových vodičov. Konektory sú navrhnuté na pripájanie za chodu. Low-speed mód nevyžaduje dokonalú EMI ochranu. Všetky módy môžu byť podporované na istej USB zbernici automatickým dynamickým prepínaním módov medzi prenosmi. Low-speed mód je definovaný pre podporu obmedzeného počtu nízkoprenosových zariadení (napr. myš), pretože ich všeobecné použitie by degradovalo využitie zbernice. Hodinový signál je prenášaný v kódovanom tvare spolu s diferenčným signálom dát. Kódovacia schéma hodín je NRZI s bit stuffingom na zabezpečenie adekvátnych prechodov. SYNC pole predchádza každý paket, aby umožnil príjemcom synchronizáciu hodín.

Konštrukcia kábla

Kábel nesie okrem dátových vodičov aj V_{BUS} a GND vodiče na zabezpečenie napájania pre zariadenia. Nominálna hodnota V_{BUS} je $+5V$ pri zdroji. USB umožňuje kabeláž premennej dĺžky, až do niekoľkých metrov. Na zabezpečenie garantovaných vstupných napäťových úrovní a správnej ukončovacej impedancie, sú použité BIASED (s predpätím) ukončenia na každom konci kábla. Tieto ukončenia taktiež umožňujú detekciu pripojenia a odpojenia zariadenia od portu a vykonávajú rozlišovanie medzi full-speed a low-speed zariadeniami.



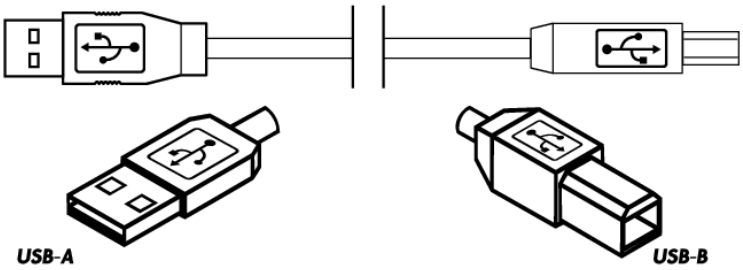
Obr. 2.16: Štruktúra kábla USB

Pin	Označenie	Farba vodiča	Popis
1	V_{BUS}	Červená	+5V
2	$D-$	Biela	Data -
3	$D+$	Zelená	Data +
4	GND	Čierna	Ground

Tabuľka 2.3: Vodiče v kábli USB

Koncovky

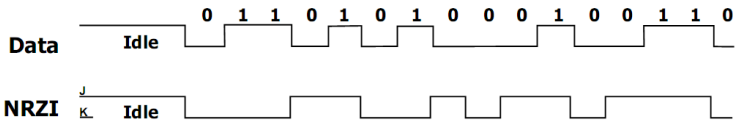
Na minimalizáciu problémov s ukončením káblov, USB používa tzv. "keyed connector" protokol. Fyzický rozdiel v konektore typu A a B zabezpečuje správne pripájanie zariadení k USB. A konektor sa používa na pripájanie zariadení. B konektor umožňuje výrobcom zariadení poskytovať štandardný odpojiteľný kábel.



Obr. 2.17: Typy koncoviek USB

Prenos signálu

Kódovanie a dekódovanie dát USB používa NRZI kódovanie dát pri prenose paketov. V RZI kódovaní, 1 je reprezentovaná žiadnou zmenou v úrovni a 0 je reprezentovaná zmenou v úrovni. Obrázok ukazuje dátový tok a jeho NRZI ekvivalent. Vysoká úroveň reprezentuje J stav na dátových vodičoch a nasledujúce obrázky ukazujú NRZI kódovanie. Reťazec núl spôsobí zmenu NRZI dát v každom bitovom cykle. Reťazec jednotiek spôsobí dlhé periódy bez zmien v dátach.



Obr. 2.18: NRZI kódovanie

Bit stuffing

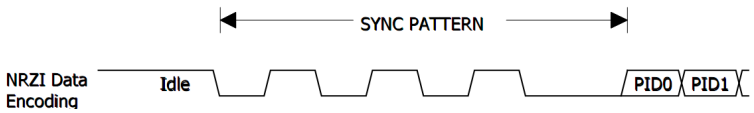
Z dôvodu zabezpečenia adekvátnych prechodov signálu, je použitý bit stuffing vysielajúcim zariadením keď posiela paket na USB. Nula je vždy vložená po šiestich po sebe idúcich jednotkách v dátovom toku predtým než je NRZI kódovaný. Slúži to na posilnenie prechodov v NRZI dátovom toku.

To umožňuje prijímačovej logike dátový prechod aspoň raz za každých sedem bitov na garantovanie, že komunikácia "beží". Bit stuffing je inicializovaný SYNC vzorkou a existuje počas celého prenosu. Bit stuffing je vždy realizovaný u vysielača, bez výnimky. Ak to vyžadujú bit stuffing pravidlá, nulový bit bude vložený dokonca aj keď je to posledný bit pred end-of-packet (EOP) signálom. Prijímač musí dekódovať NRZI dáta, rozlíšiť stuffed bity a zrušiť ich.

Ak prijímač prijme sedem po sebe idúcich jednotiek hocikde v pakete, potom nastala bit stuffing chyba a paket by mal byť ignorovaný.

SYNC vzorka

Pred každý paket je umiestená synchronizačná sekvencia, ktorá slúži na synchronizáciu hodín vysielača a prijímača.



Obr. 2.19: SYNC vozrka

Protokol USB

USB je centrálne riadená zbernica (polled bus). Host Controller iniciuje všetky dátové prenosy.

Všetky zbernicové transakcie zahŕňajú prenos do troch paketov. Každá transakcia začína keď Host Controller, na báze plánovania, pošle USB paket popisujúci typ a smer transakcie, adresu USB zariadenia a koncové číslo (endpoint number). Tento paket sa nazýva "token packet". USB zariadenie, ktoré je adresované vyberie seba samého dekódovaním príslušných adresných polí paketu. V danej transakcii sú dáta prenášané buď od Host-a k zariadeniu alebo od zariadenia k Host-ovi. Smer toku dát je určený v "token pakete".

Zdroj dát potom pošle dátový paket alebo indikuje, že už nie je čo prenášať. Cieľ, vo všeobecnosti, odpovedá handshake paketom, ktorý indikuje či bol prenos úspešný. Model dátového prenosu USB medzi zdrojom a cieľom na Host-ovi a koncovom bode zariadenia sa nazýva rúra (pipe). Sú dva typy rúr :

- prúdová rúra
- rúra správ

Prúdové dáta nemajú definovanú žiadnu štruktúru, na rozdiel od správ, ktoré majú definovanú štruktúru. Väčšina rúr je vytváraných pri konfigurácii USB zariadenia. Jedna rúra správ, tzv. "Default Control Pipe", existuje vždy pri zapnutí zariadenia, aby bol umožnený prístup ku konfigurácii, stavu a riadiacej informácii USB zariadenia. Plánovanie transakcií dovoľuje riadenie toku niektorých prúdových rúr.

Na hardvérovej úrovni sa dá predchádzať podtečeniu a pretečeniu bufferov použitím NAK paketu. Keď je poslaný NAK, pokus o obnovenie transakcie nastáva po pridelení času na zbernici. Mechanizmus riadenia toku dovoľuje konštrukciu flexibilných plánovaní, ktoré vyhovujú súčasnému spracovaniu heterogénneho mixu rúr. Z toho vyplýva, že viacnásobné rúry môžu byť obslúžené v rozdielnych intervaloch a s rôznymi dĺžkami paketov.

Zaujímavé obvody pre USB

Pomocou integrovaných obvodov firmy FTDI je možné cez USB jednoducho pripojiť do širokého spektra aplikácií. Základné použitie obvodov zbernice USB je skryté. Z hľadiska PC sa tvári ako štandardný (oveľa rýchlejší) sériový port, z hľadiska zariadenia sa tvári buď ako sériový port (RS232) alebo ako osembitová obojsmerná zbernica (RS245).

- FT8U232AM - konvertor USB - UART s prenosovou rýchlosťou 300 Bd až 920 kBd pre RS232 a až 2 MBd pre sbernicu RS485
- FT8U245AM - konvertor USB - FIFO (osembitová obojsmerná zbernica) s maximálnou prenosovou rýchlosťou až 1MByte/s
- FT8U100AX - multifunkčný USB hub kontroler so zabudovaným mikrokontrolérom, portami RS232, PS/2 (klávesnice a myš), IR a užívateľsky definovanými I/O pinami

FT8U232AM

Konvertor USB - UART s prenosovou rýchlosťou 300 Bd až 920 kBd pre RS323 a až 2 MBd pre zbernicu RS485. K dispozícii je i plné hardvérové riadenie prenosu - signály RTS, CTS. DTR. DSR. DCD a RI, a singnál TXDEN pre spoluprácu s konvertormi úrovne RS485. V obvode je zabudovaná dvojportová vyrovnávacia pamäť s veľkosťou 384 B pre smer od PC k aplikácii a 128 B pre smer k PC.

Obr. 2.20: Konvertor USB - UART

FT8U245AM

Konvertuje USB - FIFO (8bit) s maximálnou prenosovou rýchlosťou až 1 MByte/s. Prenos sa riadi veľmi jednoducho vstupnými signálmi RD a WR, stav vnútornej vyrovnávacej pamäte (384 B v smere od PC k aplikácii a 128 B pre smer k PC) je indikovaný signálmi TXE a RXF. Pokiaľ TXE=log.1, je vnútorná vyrovnávacia pamäť plná a nie je možné prijímať z periférnych zariadení ďalšie dáta. RFX=log.1 signalizuje aplikácii prítomnosť platných dát vo výstupnej vyrovnávacej pamäti, z ktorej ich môže koncové zariadenie čítať dovtedy, kým RXF=log.0.

Obr. 2.21: Konvertor USB - FIFO

Spoločnými vlastnosťami obidvoch konvertorov je podpora protokolu USB 1.1, možnosť pripojenia externej EEPROM obsahujúcej užívateľské sériové číslo alebo identifikačný reťazec, možnosť napájania 4,4 V až 5,25 V priamo z USB (zabudovaný 3,3 V regulátor), integrovaný násobič kmitočtu 6 MHz - 48 MHz pre časovanie USB operácií. Obidva obvody sa vyrábajú kompaktnom puzdre MQFP (7x7mm) s 32 vývodmi (rozpätie 0.8 mm). Ku koncovému portu UART alebo FIFO sa dá pristúpiť prostredníctvom ovládačov VCP (Virtual COM Port) z väčšiny operačných systémov, ako aj sú dostupné knižnice pre programovacie jazyky (C++, Delphi).

Kapitola 3

Špecifikácia požiadaviek

Kapitola uvádza požiadavky na funkčnosť navrhovaných systémov, ich hardvérové vybavenie a firmvér a tiež požiadavky na program pre hostiteľský počítač.

3.1 Požiadavky na funkčnosť mikropočítačov

Riešenie každého mikropočítača musí spĺňať nasledovné požiadavky:

- dve sériové linky
- ďalšie definované vstupné a výstupné zariadenia a indikačné prvky
- základné programové vybavenie (monitor), umožňujúce demonštrovať funkčnosť mikropočítača a ladenie aplikačných programov
- jednoduché programovanie prostredníctvom USB portu

Požiadavky na monitor:

- znakovo orientovaná komunikácia s hostiteľským počítačom
- otestovanie funkčnosti mikropočítača
- práca s registrami
- práca s pamäťou
- načítanie vykonateľného programu v definovanom formáte
- nastavenie/zrušenie bodov prerušenia
- spustenie/zastavenie vykonávania programu
- ďalšie špecifické funkcie pre daný typ mikroprocesora

Požiadavky na program pre hostiteľský počítač:

- grafické používateľské rozhranie pre OS Windows alebo Linux
- univerzálny program pre prácu s obidvomi mikropočítačmi
- výber rozhrania, cez ktoré bude komunikácia realizovaná
- terminál na znakovo orientovanú komunikáciu s pripojeným mikropočítačom
- kompilácia zdrojového programu
- programovanie mikropočítačov prostredníctvom USB portu

3.2 Požiadavky na hardvér mikropočítačov

Nasledujúce požiadavky boli stanovené po konzultáciách s vedúcim projektu.

3.2.1 Mikropočítač na báze ATmega32

- procesor ATmega32 v puzdre DIL40
- 32 kb internej Flash pamäte programu
- 2 kb internej SRAM pamäte údajov
- port USB 2.0
- sériový port RS232
- generovanie externých prerušení tlačidlami
- 6 miestny numerický displej
- 2 riadkový LCD displej

3.2.2 Mikropočítač na báze AT91SAM7S64

- procesor AT91SAM7S64
- 64 kb Flash EEPROM pamäte
- 16 kb pamäte SRAM
- port USB 2.0

- dva sériové porty RS232
- 10-bitový AD prevodník
- podpora protokolu JTAG
- modulárna architektúra

K mikropočítaču budú zhotovené nasledujúce moduly:

- jednoduchá klávesnica (tlačidlá prerušení)
- dvojriadkový LCD displej

Kapitola 4

Návrh

Nasledujúca kapitola sa venuje návrhu konkrétnych prototypov a obslužného softvéru pre hostiteľský počítač. V prvej časti je konceptuálny návrh a bloková schéma, druhá časť sa bude zaoberať návrhom podrobnejšie.

4.1 Mikropočítač na báze ATmega32

Mikropočítač bude postavený na procesore ATmega32, návrh počítača znázorňuje schéma na obrázku 4.1

4.1.1 Procesor

Procesor ATmega32 je vysoko-výkonný, nízko-energetický 8-bitový mikroprocesor založený na architektúre RISC. Poskytuje 32 kB internej Flash pamäte programu a 2 kB SRAM. Tieto interné bloky budú použité pre realizáciu všetkých pamäťových požiadaviek procesora. Pamäť programu sa bude programovať prostredníctvom sériového rozhrania Master/Slave SPI, ktorý vyvedieme od procesora, kvôli lepšej manipulácii. Ide o signály SCK, MIS0, MIS1 a SS.

Ku procesoru bude štandardne zapojené resetovacie tlačítko, ktoré po stlačení vyšle na pin procesora log. 0. Kryštál bude mať frekvenciu 8 MHz.

4.1.2 Zbernica

Procesor ATmega32 nemá možnosť vyvedenia externej multiplexovanej zbernice. Preto je potrebné realizovať ju vlastným riešením použitím voľných portov procesora. Port *A* slúži ako AD prevodník. Keďže požiadavka na takúto funkciu nie je v projekte zahrnutá, je možné tento port použiť na realizáciu dátovej zbernice. Všetky externé obvody sú dimenzované pre 8-bitový systém, preto je 8-bitový port postačujúci pre implementáciu všetkých špecifikovaných funkcií experimentálneho mikropočítača.

Funkcia adresnej zbernice bude implementovaná pomocou jednoduchého dekodéra a niekoľkých externých riadiacich signálov. Na realizáciu sa využije port C procesora, ktorého funkcie sú v rámci projektu tiež postrádateľné. Systém obsahuje celkovo 8 záchytných registrov, ktoré je nutné adresovať. Preto postačuje štandardný 1 z 8 74LS137N dekodér. Na jeho výstup budú pripojené záchytné registre pre 7-segmentové displeje a LCD displej. Dekodér bude aktivovaný externým signálom, aby bolo možné dosiahnuť stav, kedy žiadne z adresovaných zariadení nie je aktívne. Na vstup dekodéra budú privedené 3 signály z portu C, pomocou ktorých bude programátor manuálne adresovať zvolené zariadenia.

4.1.3 Segmentové displeje

Segmentový diplej zobrazuje číslicu rozsvietením daných segmentov. Toto rozsvietenie zabezpečujú diódy, ktoré sa nachádzajú pod danými segmentami. Vzhľadom na ich polaritu, rozlišujeme zapojenie zo spoločnou anódou alebo katódou. Pri zapojení so spoločnou anódou je potrebné zapojenie ku zdroju napätia a na aktiváciu jednotlivých častí displeja sa použijú negované signály.

Informácie vysielané na tieto displeje budú najskôr uložené do záchytného registra 74HC573N, potom sa až vyšlú do displeja. Na takéto zapojenie je potrebný odpor o veľkosti 1 kΩ. Záchytné registre sú aktivované pomocou výstupov z dekódera (bude popísaný neskôr).

Pri navrhovaní sme si definovali, že použijeme celkovo 6-ciferný diplej. Budeme ho realizovať ako tri 2-ciferné displeje DA56-11GWA od firmy KingBright. Jedná sa o 14.22 mm vysoké 2-ciferné displeje so spoločnou anódou.

4.1.4 LCD displej

2-riadkový LCD displej bude realizovaný použitím štandardného Hitachi HD44780 radiča. Pre komunikáciu s ním budú využité 2 záchytné registre pripojené na dátovú zbernicu procesora. Registre budú adresované centrálnym dekodérom. Jeden z nich slúži na uchovávanie 8 bitov dát, ktoré LCD modul spracúva paralelne. Druhý bude slúžiť na ovládanie troch riadiacich signálov prenosu.

4.1.5 Sériové rozhranie

Sériové rozhranie bude realizované s využitím pinou procesora (TX)PD1 a (RX)PD0. Tieto budú privedené na vstupné piny obvodu MAX232, ktorý sa štandardne používa na komunikáciu cez sériové rozhranie. Obvod slúži na prevod úrovní z RS232 na TTL a naopak. MAX232 obsahuje väčšinou 2 prevodníky z RS232 napätia +12 V a -12 V na TTL napätie 5 V a 0 V. Výhodou je, že mu stačí zdroj napätia iba 5 V, nie ostatné. Konektor bude štandardný CANNON 9.

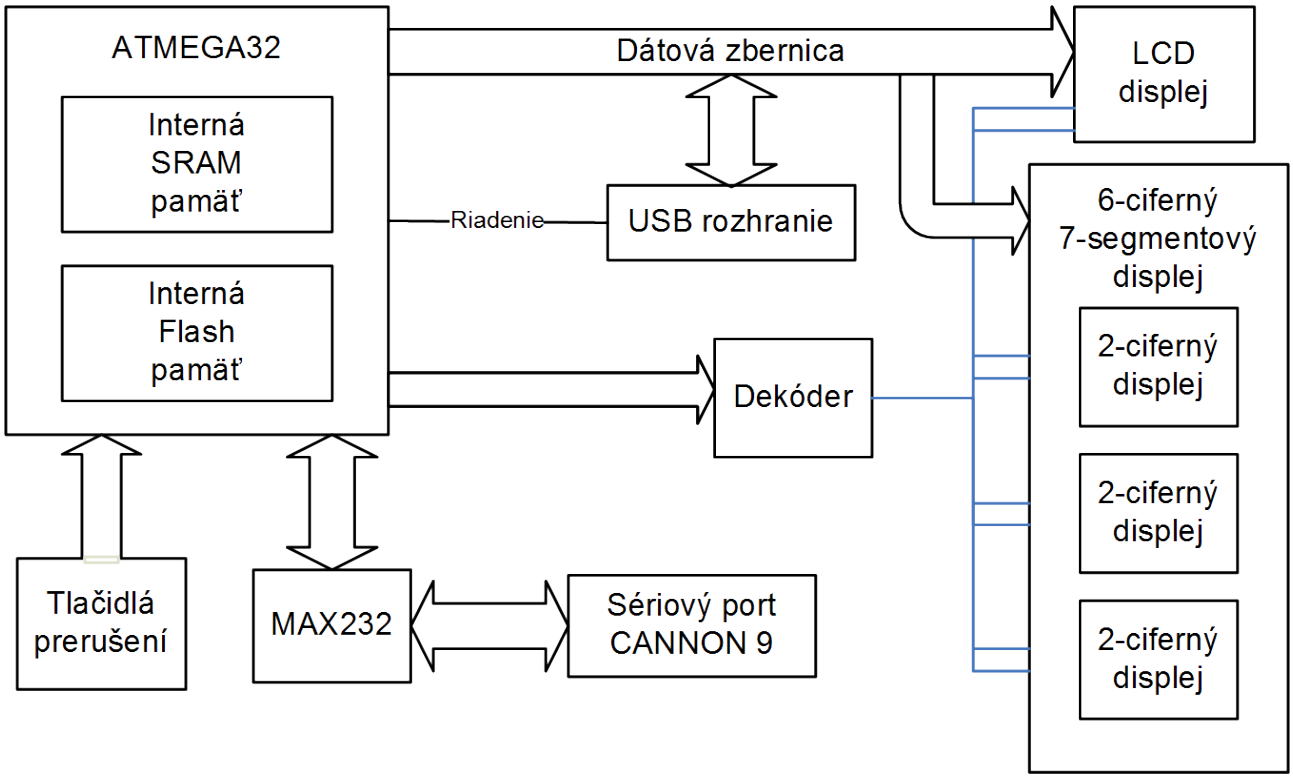
4.1.6 USB rozhranie

USB port bude realizovaný pomocou externého radiča FT245BM, ktorý bude pripojený na dátovú zbernicu. Dáta budú do obvodu vstupovať paralelne po bajtoch. Obvod nie je pripojený na centrálny dekodér, pretože používa až 4 riadiace signály. Keďže interný sériový port procesora bude obsadený štandardným sériovým portom, je nutné implementovať riadenie USB obvodu samostatne. Na toto budú slúžiť 3 signály portu C a jeden signál externého prerušenia. Použitie externého prerušenia je aplikované pre realizáciu signalizácie naplnenia buffera radiča USB obvodu. Takto je možné vyhnúť sa komplikovanému testovaniu pri prijímaní dát.

Experimentálny mikropočítač bude v komunikácii vystupovať ako zariadenie, nie ako host. Radič bude zapojený v konfigurácii samostatného napájania. Mikropočítač teda nebude využívať možnosť napájania cez USB port, ale štandardný zdroj jednosmerného napätia s požadovanými vlastnosťami.

4.1.7 Tlačidlá prerušení

Tlačidlá prerušení budú pripojené na piny externých prerušení procesora, ktorý umožňuje pripojiť až 3 rôzne zdroje.



Obr. 4.1: Bloková schéma mikropočítača na báze ATmega32

4.2 Mikropočítač na báze AT91SAM7S64

Kapitola obsahuje návrh jednotlivých funkčných blokov a blokovú schému navrhovaného systému. Obvodová schéma je v prílohe B na konci dokumentu.

4.2.1 Procesor

Procesor AT91SAM7S64 obsahuje 16 kB pamäte SRAM a 64 kB pamäte Flash EEPROM. Obsahuje vstavaný port USB 2.0, dva vstavané porty UART a 10-bitový AD prevodník. Vlastnosti PIO budú využité aj pri splnení požiadavky - modularity architektúry navrhovaného mikropočítača.

Procesor obsahuje viacero napájacích vstupov (VDDIN, VDDFLASH, VDD), ktoré požadujú napätie 3 až 3,6 V, a vlastný generátor napätia (vývod VDDOUT) 1,85 V, ktorý je možné nepoužiť, ak sa toto napätie stabilizuje externe. Týmto či už externe alebo interne stabilizovaným napätím sa napája jadro procesora a PLL (VDDCORE, VDDPLL).

Všetky tieto napájacie vstupy a výstupy (VDDOUT je výstup) je potrebné ošetriť odrušovacími kondenzátormi, ako je uvedené v [15]. Napájacie napätie bude 3,3 V a bude sa vytvárať z 5V prostredníctvom stabilizátora.

Rozhranie JTAG bolo navrhnuté podľa [16]. Signály TMS, TDI a TCK je nutné ošetriť externými zdvíhacími rezistormi, pretože ich nemajú integrované.

Jumper TST slúži spolu s jumpermi SAMBABR0 až 2 na spustenie SAM-BA Boot Recovery (špeciálna funkcia na obnovenie obsahu prvých dvoch sektorov Flash EEPROM). Obsahuje interný pull-down rezistor, takže jumper možno nechať nezapojený pre normálnu funkciu procesora.

Všetky výstupy z PIO sú vyvedené spolu s napájaním a zemou na lámací konektor 2x17 pre zabezpečenie rozšíriteľnosti. Prostredníctvom konfiguračných prepínačov JP2, JP3, JP4, JP5 a JP6 možno odpojiť väčšinu funkcií z PIO a tak uvoľniť tieto vývody pre iné použite.

4.2.2 Bootloader

Využije sa vstavaný bootloader SAM-BA (SAM Boot Assistant), ktorý je uložený v internej pamäti procesora typu ROM. Umožňuje programovanie vstavanej Flash EEPROM pamäte prostredníctvom USB rozhrania, ktoré je súčasťou procesora, alebo pomocou Debug Unit procesora, ktorá s hostiteľským počítačom komunikuje prostredníctvom UART portov.

Pre správnu funkciu USB v režime SAM-BA boot je nutné použiť kryštál s frekvenciou 18,432 MHz. Rovnako je potrebné zabezpečiť správnu frekvenciu PLL RC filtra druhého rádu. Táto frekvencia musí byť 48 MHz, 96 MHz, alebo 192 MHz s presnosťou na 0,25 %. Potrebné hodnoty možno vypočítať pomocou nástrojov, ktoré sa nachádzajú na stránke firmy Atmel v sekcii AT91.

4.2.3 Monitor

Keďže sa využije bootloader vstavaný v procesor, funkcie monitora budú vykonávané zvláštnym používateľským programom, ktorý bude podporovať potrebný komunikačný protokol. Budú podporované nasledujúce testy:

- Test displeja.
- Test pamäte RAM.
- Test tlačidiel.
- Test sériového portu.
- Test portu USB.

Je potrebné zabezpečiť ich podporu aj na strane programu pre hostiteľský počítač.

4.2.4 Pamäť programu a pamäť údajov

Procesor obsahuje dostatočné množstvo ľahko programovateľnej pamäte programu a rovnako dostatočné množstvo pamäte údajov. Zároveň podpora externých pamätí nie je priamo podporovaná, čiže navrhovaný prototyp nebude obsahovať žiadne externé pamäte.

4.2.5 USB port

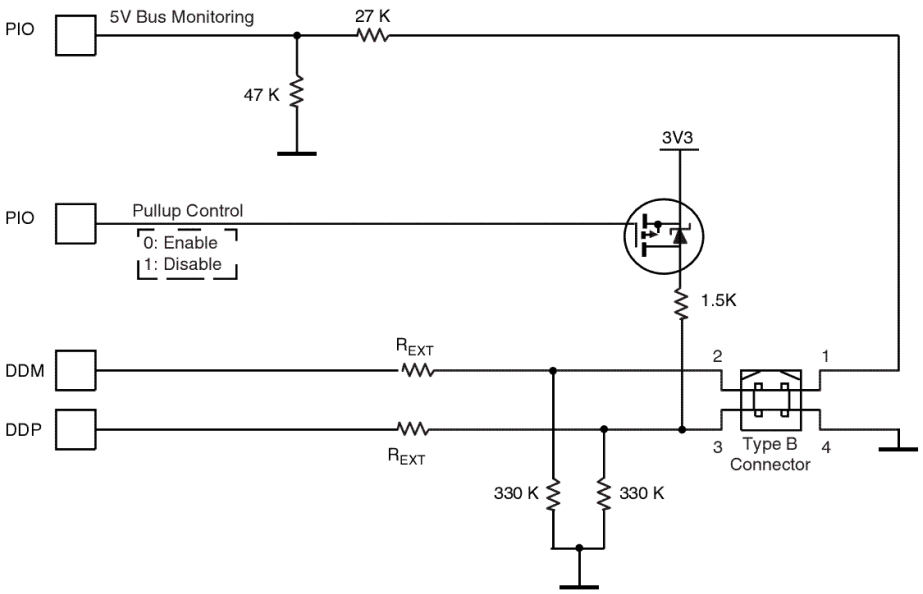
Keďže vybraný mikroprocesor má vstavaný port USB 2.0, realizácia tohto portu pozostáva v pripojení niekoľkých súčiastok a konektora. Pre správnu funkciu rozhrania USB bude tiež potrebné vyhradiť 2 vstupy PIO na Bus Monitoring a Pullup Control [lietartura datasheet 600]. Vstavaný port bude umožňovať mikropočítaču fungovať ako USB zariadenie, nie ako USB host, čiže nebude podporovať pripájanie externých zariadení cez tento port.

Port USB je zapojený podľa katalógového listu procesora.

Prostredníctvom jumperov JP5 a JP6 možno odpojiť pomocné vývody PIO pre "5V Bus Monitoring" a "Pullup Control" a tak použitie USB zablokovať.

4.2.6 Sériové porty

Dva v procesore vstavané UART porty umožnia jednoduchú realizáciu dvoch RS232 portov pripojením príslušných budičov (MAX232), keďže rozhranie RS232 pracuje s inými napäťovými úrovňami ako procesor. Porty UART musia byť na výstupné piny namapované prostredníctvom PIO.



Obr. 4.2: Zapojenie portu USB podľa katalógového listu AT91SAM7S64

Na prispôsobenie UART výstupov procesora požiadavkám RS-232 bol použitý obvod z rodiny MAX232, konkrétne MAX3232, ktorý pracuje s napájacím napätím 3,3 V. Obsahuje dva kanály a preto je použitý na oba porty. Jeho zapojenie zodpovedá odporúčanému zapojeniu podľa katalógového listu [17]. Prepínače JP2 a JP3 spolu slúžia na nastavenie prvého portu RS232 pre Debug režim procesora. Ich úplne odobratie zablokuje tento sériový port.

Procesor obsahuje tiež sériové rozhrania SPI (Serial Programming Interface) a TWI (Two-Wire Interface, tiež známe ako I2C), ktoré sú vyvedené na samostatné konektory. Dané vývody procesora je však nutné najskôr namapovať prostredníctvom PIO kontroléra.

4.2.7 AD prevodník

V procesore vstavanému 10-bitovému AD prevodníku stačí namapovať konkrétne vstupné piny procesora prostredníctvom PIO, pričom 4 analógové vstupy sú priradené daným vývodom napevo (AD4 až AD7). Prostredníctvom integrovaného multiplexora umožňuje pripojenie až štyroch ďalších analógových signálov. Funkciu AD prevodníka bude možné demonštrovať napríklad potenciometrom.

AD prevodník využíva referenčné napätie pripojené na ADVREF, teda 3,3 V. Na vstup AD prevodníka AD4 je pripojený potenciometer, ktorý reguluje jeho napätie v rozsahu 0 až 3,3 V. Vstupy AD5 až AD7 nie sú využité a tak sú aj spolu s AD4 vyvedené na konektor.

4.2.8 Dvojriadkový LCD displej

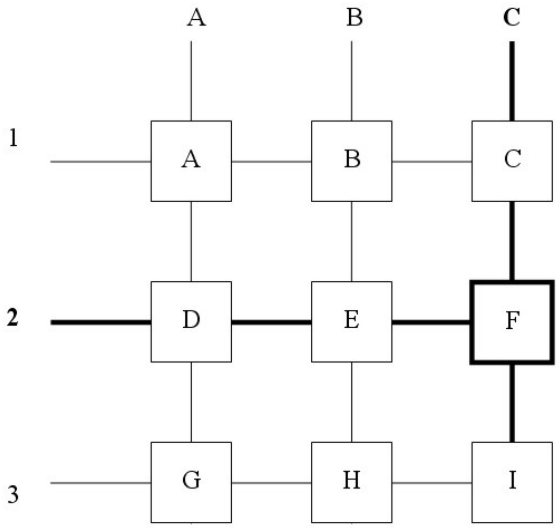
Modul displeja sa bude pripájať prostredníctvom kábla na vyvedený PIO z procesora a bude vyžadovať správne nastavenie registrov ovládajúcich PIO v rámci užívateľského programu. Realizácia bude prostredníctvom displeja z rodiny MC1602E a dvoch záchytných registrov. Toto riešenie opísal S. Angelovič a kol. v rámci svojho tímového projektu [3].

Mikropočítač vyžaduje ale obvod, ktorý pracuje aj s napätím 3,3V a preto bude použitý displej BC1602A, ktorý tejto podmienke vyhovuje. [18] Obsahuje kontrolér KS0066. Použité zapojenie bude používať 4-bitový dátový režim. [19] Pre jednoduchosť bude displej osadený na doske mikropočítača a nebude tvorený modulom.

Displej je možné zablokovať prostredníctvom JP4, aby nezaberal žiadne vývody PIO.

4.2.9 Tlačidlá prerušení

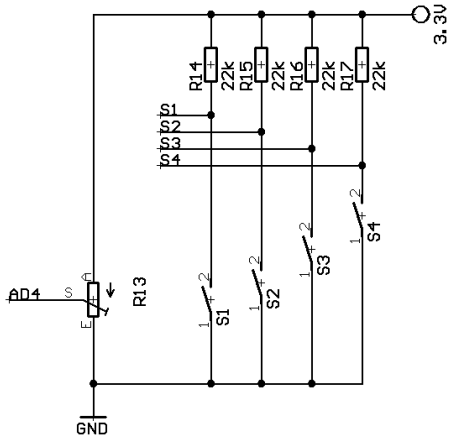
Tlačidlá sa rovnako ako displej budú pripájať na vyvedený PIO procesora v podobe modulu. Stlačenie ľubovoľného tlačidla vyvolá externé prerušenie (vstup pre externé prerušenie možno namapovať na PIO) a na vybrané vstupy PIO bude odoslaný kód tlačidla. Tlačidlá budú pravdepodobne pripojené prostredníctvom matice vodičov, čiže každé tlačidlo bude identifikované dvojicou aktívnych vstupov PIO. Takéto riešenie je efektívne a šetrí vstupy PIO pri počte tlačidiel väčšom ako 4.



Obr. 4.3: Maticová organizácia tlačidiel

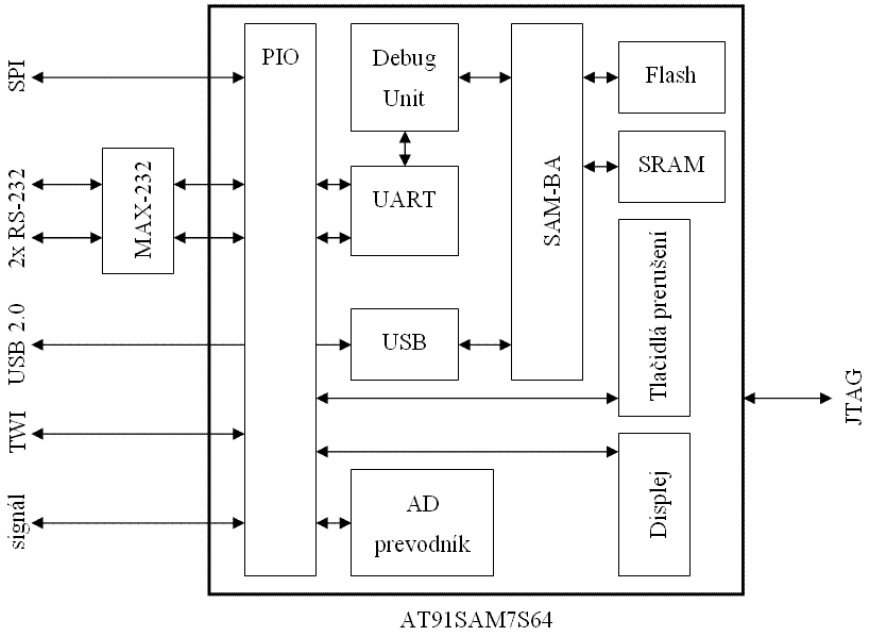
Od tohto konceptu bolo upustené a mikropočítač teda využíva štvoricu jednoduchých 2-pólových tlačidiel (S1 až S4) so zdvíhacím rezistorom, ktoré sú pripojené na vývody procesora PA19, PA20, PA23 a PA24. Tieto vývody samozrejme teba naprogramovať pre potreby týchto tlačidiel prostredníctvom PIO. Tlačidlá sú zapojené spolu so sériovými 22kOhm rezistormi, aby

pri ich stlačení nebol zdroj skratovaný. Procesor automaticky filtruje všetky impulzy kratšie ako pol taktu a ďalšie ošetrenie proti zákmitu bude realizované softvérovo.



Obr. 4.4: Zapojenie tlačidiel a potenciometra

Bloková schéma mikropočítača na báze AT91SAM7S64



Obr. 4.5: Bloková schéma mikropočítača na báze AT91SAM7S64

4.3 Návrh programu pre hostiteľský počítač

Aplikácia na hostiteľskom PC má spĺňať požiadavky definované v špecifikácii. Pomocou programu sa musí dať ovládať a monitorovať činnosť mikropočítačov, teda program musí byť schopný komunikovať s mikropočítačmi. Na to je potrebné zabezpečiť komunikáciu s mikropočítačmi. S mikropočítačmi sa komunikuje cez sériové rozhranie, čiže s použitím štandardu RS232. Komunikácia pomocou RS232 je realizovaná posielaním sekvencie bitov, rámec má štandardne 7-8 bitov +1 stopbit. Aplikácia musí vedieť prijímať a posielať rôzne príkazy pre mikropočítače, preto je potrebné vytvoriť komunikačný protokol. Protokol by mal byť podľa možnosti jednoduchý, rovnaký pre obidva mikropočítače, a každý príkaz v ňom musí byť jednoznačne zadefinovaný.

Kompilácia programu

V aplikácii sa musí dať skompilovať zdrojový program pre oba mikropočítače, čiže musí byť možnosť zvolenia pre ktorý mikropočítač má byť program skompilovaný. Programy budú písané v C jazyku, preto aplikácia musí obsahovať kompilátory jazyka C pre obidva mikropočítače.

Ďalšou funkcionalitou aplikácie, by mala byť možnosť "debugovania" programu, teda možnosť zastavenia programu na určitom mieste a následného výpisu registrov.

Z používateľského hľadiska sa jedná o možnosť vkladania breakpointov do programu. Tu musí byť vymyslený spôsob reprezentácie a obsluhy breakpointov.

Nahranie programu do mikropočítača

Aplikácia musí umožňovať aj nahranie programu do mikropočítačov, na to musí byť v komunikačnom protokole zadefinované kam a koľko bytov sa má uložiť.

Používateľské prostredie

Používateľské prostredie má byť čo najjednoduchšie, odolné voči chybnému zaobchádzaniu a práca s ním by mala byť intuitívna. Pôvodne sa uvažovalo, že aplikácia bude fungovať pod OS Windows, ale existuje viacero prostriedkov, ktorými sa dá dosiahnuť multiplatformovosť (napr. programovací jazyk Java).

Komunikácia cez sériový port z programátorského hľadiska

V **OS Windows** sa pri tvorbe aplikácií k portom nepristupuje priamo, na prístup k portom sa využíva aplikačné programové rozhranie(API). API vytvára štandardné rozhranie pre prístup k hardvéru. Najprv sa získa "handler" na dané zariadenie volaním CreateFile() s parametrom identifikátora zariadenia("Com1","Com2",..). K sériovým portom sa potom pristupuje ako k súborom a používajú sa funkcie ReadFile(), WriteFile(), CloseHandle().

OS Linux tiež neposkytuje priamy prístup k sériovým portom, ale má ovládače na úrovni jadra, ktoré mapujú zariadenia do súborového systému a v aplikácii sa k nim tiež pristupuje ako k súborom. Na zjednodušenie práce programátora sú vytvorené knižnice. Prístup k sériovým portom potom môže vyzerať takto:

```
#include <SerialStream.h>

// otvorenie sériového portu
SerialStream serial_port( "/dev/ttyS0" ) ;

// prečítanie znaku zo sériového portu.
char next_char ;
serial_port >> next_char ;

// zapísanie znaku na sériový port.
serial_port << next_char ;
```

Príkazy komunikačného protokolu

Budú mať rovnakú štruktúru ako sú príkazy podporované bootloaderom. Posielaný príkaz v komunikačnom protokole bude definovaný ako 1bajt, ktorý vyjadruje typ príkazu a jeho súčasťou budú tiež jeho argumenty. Pri príkazoch, kde sa posielajú aj dáta, budú za typom príkazu a argumentami posielané ďalšie bajty. Napr. pri nahrávaní programu do pamäte sa za typom príkazu pošle aj dĺžka programu v bajtoch a adresa od ktorej sa má začať zapisovať do pamäte. Nasledovať bude n bajtov programu, kde n je definované v poli length daného príkazu.

Pri spustení testu častí systému sa ako odpoveď očakáva reťazec bitov, kde každý bit prislúcha danému zariadeniu, a hovorí či zariadenie pracuje správne(1) alebo nie (0).

4.3.1 Ladenie programu

V aplikácii bude možné vkladať do programu breakpointy na ľubovolné miesta. Tieto breakpointy sa pri kompilácii nahradia za inštrukcie interného prerušenia. Pri prerušení sa vykoná rutina, ktorá pošle na sériový port výpis registrov. Pre registre oboch mikropočítačov sa stanový presné poradie, aby sa s výpisom registrov nemusel posielať aj názov registrov. Registre s hodnotami budú následne zobrazené v tabuľke.

Procesor AT91SAM7S64 s jadrom ARM7 podporuje interné softvérové prerušenie na úrovni Software Interrupt Instruction (SWI).

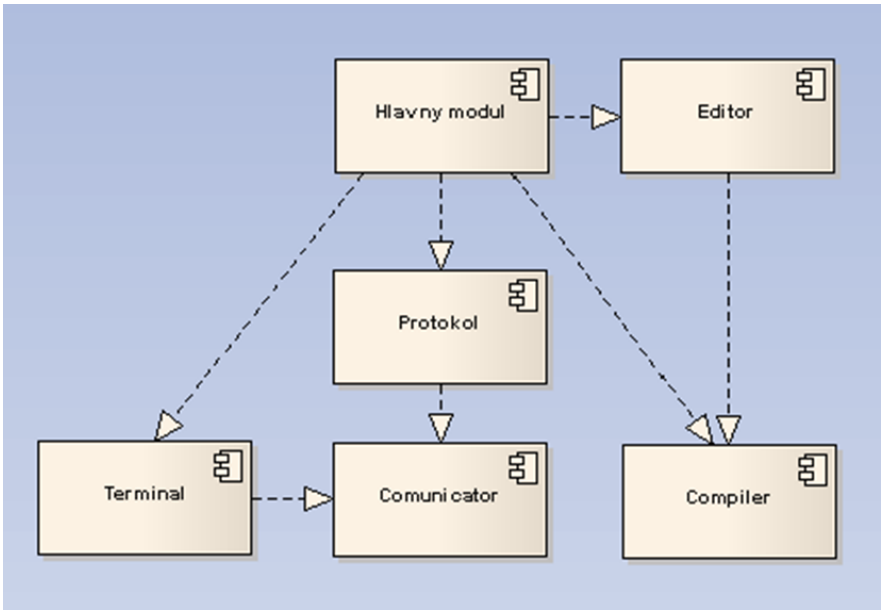
Procesor ATmega32 interné softvérové prerušenie nepodporuje a preto sa do kompilovaného programu bude automaticky vkladať ladiaci podprogram, ktorý bude uskutočňovaný na jednotlivých breakpointoch.

4.3.2 Používateľské prostredie

Pre skúsenosti programátora s programovaním v jazyku C/C++ bude aplikácia písaná v tomto jazyku. Na vytvorenie grafického používateľského prostredia bude použitá knižnica QT. Program napísaný s touto knižnicou je skompilovateľný aj pod OS Linux aj pod OS Windows. Takto ten istý zdrojový kód môže byť použiteľný pod oboma operačnými systémami. Vytvorená aplikácia potrebuje QT knižnicu pre daný operačný systém.

4.3.3 Štruktúra programu

Kvôli prehľadnosti a jednoduchosti upravovania bude mať program modulárnu štruktúru. Navrhnutých je 6 základných modulov programu:



Obr. 4.6: 6 základných modulov programu

Hlavný modul

Obsahuje hlavné okno používateľského rozhrania a volajú sa z neho všetky základné funkcie aplikácie, a spúšťajú sa z neho ostatné moduly.

Terminal

Slúži na znakovo orientovanú komunikáciu s pripojeným mikropočítačom, posiela bajty cez comunicator na sériový port a zobrazuje prijaté bajty.

Protokol

Obsahuje komunikačný protokol, ktorý je definovaný v kapitole 4.3.1. Protokol kóduje príkazy zadané v používateľskom prostredí a systému na reťazec bytov definovaný protokolom, ktorý následne posiela na comunicator. Od comunicatora tiež prijíma bajty, ktoré dekóduje a vykoná príslušnú operáciu(napr. zavolá zobrazenie výpisu registrov)

Comunicator

Jeho úlohou je zabezpečiť komunikáciu po sériovej linke oboma smermi, teda má na starosti posielanie a prijímanie bajtov. Comunicator dostáva reťazce bajtov od Terminálu a Protokolu, ktoré následne posiela na sériový port, naopak prijaté bajty posiela naspäť na terminál/protokol. Comunicator obsahuje buffer pre prichádzajúce aj odchádzajúce bajty, aby nedošlo k strate dát. Comunicator vždy naraz spracúva len jednu požiadavku. Ak po odoslaní žiadosti sa čaká na odpoveď, Comunicator ignoruje ďalšie žiadosti, kým nedostane požadovanú odpoveď. Comunicator bude kvôli rozdielnemu spôsobu programovania komunikácie cez sériový port, zvlášť naprogramovaný pre OS Linux a OS Windows.

Editor

Slúži na zobrazenie zdrojového kódu. Taktiež poskytuje možnosť vkladania a odoberania breakpointov. Breakpointy si zapamätá ako čísla riadkov, na ktoré boli vložené, ktoré pri kompilácii pošle compileru.

Compiler

Compiler má na starosti kompiláciu programov pre oba mikropočítače. Pri kompilácii sa na miesta breakpointov vloží inštrukcia interného prerušenia.

4.4 Komunikácia firmvéru a hostiteľského počítača

Hostiteľský počítač komunikuje s firmwarom cez sériový port pomocou definovaného protokolu. Protokol podporuje dva režimy komunikácie, a to režim neštrukturovaných správ resp. konzolový režim, ktorý sa využije pri znakovo orientovanej komunikácii s mikropočítačom. Tento režim nemá definovanú štruktúru správ. Pri tomto móde sa len echujú znaky prijaté cez port. Druhým režimom je režim štrukturovaných správ resp. režim monitora, ktorý sa využíva pri programovom komunikovaní s mikropočítačom. Tento režim má definovanú štruktúru správ. Medzi týmito dvoma režimami sa prechádza príkazmi START_STRUCT_MODE a END_STRUCT_MODE. Pri príkaze START_STRUCT_MODE ide o sekvenciu 3 bajtov, o vhodnej hodnote. Pri príkaze END_STRUCT_MODE ide o špeciálnu štruktúrovanú správu. V štruktúrovanom móde sú vymieňané nasledujúce štruktúrované správy:

- DUMP_REG - táto správa spôsobí získanie obsahu daného registra
- LOAD_REG - táto správa spôsobí nahratie hodnoty do registra
- DUMP_MEM - táto správa spôsobí získanie obsahu zadanej pamäťovej oblasti
- LOAD_MEM - spôsobí uloženie nahratého programu na danú adresu v pamäti
- JUMP - spôsobí skočenie na zadanú adresu a začatie vykonávania programu
- END_STRUCT_MODE - koniec štruktúrovaného módu

Keďže ide o štruktúrované správy, môžme správy reprezentovať štruktúrami v jazyku C. Takáto reprezentácia je vhodná z hľadiska programovania ako firmwaru, tak aj programu pre hostiteľský počítač. Všetky štruktúry začínajú tým istým a to typom správy. Nasledujúce polia sú špecifické pre danú správu. Takýto protokol sa jednoducho rozširuje o nové správy, stačí len pridať novú štruktúru. Treba dať však pozor na to, aby polia na strane hostiteľského počítača boli rovnako veľké, ako polia na strane mikropočítača.

Ak použijeme pre typ správy 1 bajt môžeme podporovať až 256 správ. Toto číslo je viac než postačujúce pre našu aplikáciu. Teraz bližšie opíšeme štruktúru jednotlivých správ.

Takže správa má nasledovný formát:

1B	1B	nB
Typ správy	Dáta správy	Dátová časť

Typ správy je spomínané jednobajtové pole typu správy. Dáta správy sú dáta, ktoré vyžaduje správa. Dátová časť sú údaje, ktoré sa prenášajú pri správach ako je výpis pamäti alebo nahrávanie programu do pamäti.

Iniciátorom komunikácie je hostiteľský počítač, t.j. v komunikácii vystupuje ako master. Podriadeným v komunikácii je mikropočítač, t.j. vystupuje ako slave. Toto umožňuje rozdeliť

komunikáciu na požiadavky (requests) a odpovede (response). Požiadavky posiela hostiteľský počítač a odpovede posiela podriadený mikropočítač.

Teraz bližšie rozoberieme jednotlivé správy (stĺpec smer vyjadruje, ktorým smerom sa správa posiela vzhľadom k hostiteľskému počítaču t.j. smer OUT znamená, že sa posiela smerom z hostiteľského počítača do mikropočítača, smer IN znamená, že sa posiela smerom z mikropočítača do hostiteľského počítača):

Správa	Typ	Smer
DUMP_REG	1	OUT
DUMP_REG_RES	2	IN
LOAD_REG	3	OUT
LOAD_REG_RES	4	IN
DUMP_MEM	5	OUT
DUMP_MEM_RES	6	IN
LOAD_MEM	7	OUT
LOAD_MEM_RES	8	IN
JUMP	9	OUT
JUMP_RES	10	IN
END_STRUCT_MODE	11	OUT
ERROR	12	IN
BREAKPOINT	13	IN
CONTINUE	14	OUT

Tabuľka 4.1: Správy a im prislúchajúce typy

V nasledujúcich odstavcoch upresníme štruktúru jednotlivých správ.

4.4.1 Správa DUMP_REG

Túto správu posiela hostiteľský počítač podriadenému mikropočítaču, keď chce získať hodnotu registra. Ide o správu typu request, požiadavka. Odpoveďou na túto správu môže byť správa DUMP_REG_RES, alebo ERROR a to v prípade, že číslo registra špecifikované v požiadavke je neplatné.

4.4.2 Správa DUMP_REG_RES

Túto správu posiela mikropočítač nadriadenému počítaču ako odpoveď na správu DUMP_REG. Ide o správu typu response, odpoveď. Obsahuje hodnotu žiadaného registra. Má nasledovný formát. Hodnota registra je 4 bajtové pole. Pri jeho vysielaní sa vysiela od najnižšieho bajtu

1B	1B
Typ=1	REG=<0-15>

1B	4B
Typ	Hodnota registra

Tabuľka 4.2: Správa DUMP_REG a DUMP_REG_RES

smerom k vyšším.

4.4.3 Správa LOAD_REG

Túto správu posieľa nadriadený počítač podriadenému mikropočítaču. Ide o správu typu request, požiadavka. Obsahuje hodnotu registra, ktorý sa má nahrať. Hodnota registra je 4 bajtová. Pri vysielaní sa vysiela od najnižších bajtov smerom k vyšším. Odpoveďou na túto správu je správa LOAD_REG_RES alebo ERROR a to v prípade, že číslo registra špecifikované v požiadavke je neplatné.

4.4.4 Správa LOAD_REG_RES

Túto správu posiela podriadený mikropočítač nadriadenému počítaču a to v prípade, že sa vyhovelo požiadavke LOAD_REG. Formát:

1B	1B	4B
Typ	Číslo registra	Hodnota registra

1B
Typ

Tabuľka 4.3: Správa LOAD_REG a LOAD_REG_RES

4.4.5 Správa DUMP_MEM

Túto správu posiela nadriadený počítač podriadenému mikropočítaču. Ide o správu typu request, požiadavka. Obsahuje začiatočnú adresu pamäťovej oblasti a jej veľkosť, ktorú si želáme vypísať. Odpoveďou na túto správu je správa DUMP_MEM_RES a to v prípade, že požiadavka obsahuje správnu začiatočnú adresu a správnu dĺžku t.j. nepristupuje sa k pamäťovej oblasti, v ktorej nie je namapovaná žiadna pamäť. Ak sa pristupuje k oblasti v ktorej nie je namapovaná pamäť tak sa odpovie správou ERROR.

4.4.6 Správa DUMP_MEM_RES

Táto správa je odpoveď na správu DUMP_MEM. Posiela ju podriadený mikropočítač nadradenému počítaču. Pole dáta je premenlivej veľkosti v závislosti od toho aká veľká pamäťová oblasť sa žiadala vypísať.

1B	4B	2B
Typ	Začiatočná adresa	Dĺžka

1B	nB
Typ	Dáta

Tabuľka 4.4: Správa DUMP_MEM a DUMP_MEM_RES

4.4.7 Správa LOAD_MEM

Táto správa sa posiela z nadriadeného počítača podriadenému mikropočítaču. Ide o správu typu request, požiadavka. Odpoveďou na túto správu je správa LOAD_MEM_RES a to v prípade, že nahrávanie obsahuje správnu adresu a dĺžku. V prípade, že sú tieto polia nesprávne sa odpovie správou ERROR.

4.4.8 Správa LOAD_MEM_RES

Táto správa sa posiela v prípade, že predchádzajúca správa LOAD_MEM prebehla bez problémov. Formát je nasledovný:

1B	4B	2B	nB
Typ	Začiatočná adresa	Dĺžka	Dáta

1B
Typ

Tabuľka 4.5: Správa LOAD_MEM a LOAD_MEM_RES

4.4.9 Správa JUMP

Táto správa sa posiela nadriadeným počítačom mikropočítaču. Spôsobí začatie vykonávania programu od špecifikovanej adresy. Najskôr sa posiela najnižší bajt a potom vyššie. Odpoveďou na túto správu, môže byť správa JUMP_RES a to v prípade, že špecifikovaná adresa je správna. V prípade nesprávnej adresy sa pošle ERROR.

4.4.10 Správa JUMP_RES

Táto správa sa posiela nadradenému počítaču podriadeným mikropočítačom. Ide o správu typu response, odpoveď. Je to odpoveď na správnu požiadavku JUMP. Formát je nasledovný:

1B	4B
Typ	Adresa

1B
Typ

Tabuľka 4.6: Správa JUMP a JUMP_RES

4.4.11 Správa END_STRUCT_MODE

Táto správa sa posieľa keď sa chce prejsť z módu monitora (štruktúrovaného módu) do módu konzole.

4.4.12 Správa ERROR

Táto správa sa posiela pri chybových stavoch. Má rovnaký formát ako END_STRUCT_MODE.

4.4.13 Správa BREAKPOINT

Táto správa sa posiela nadradenému počítaču podriadeným mikropočítačom. Mikropočítač túto správu posiela ak pri vykonávaní programu nastane softvérové prerušenie, vykonávanie programu sa zastaví.

4.4.14 Správa CONTINUE

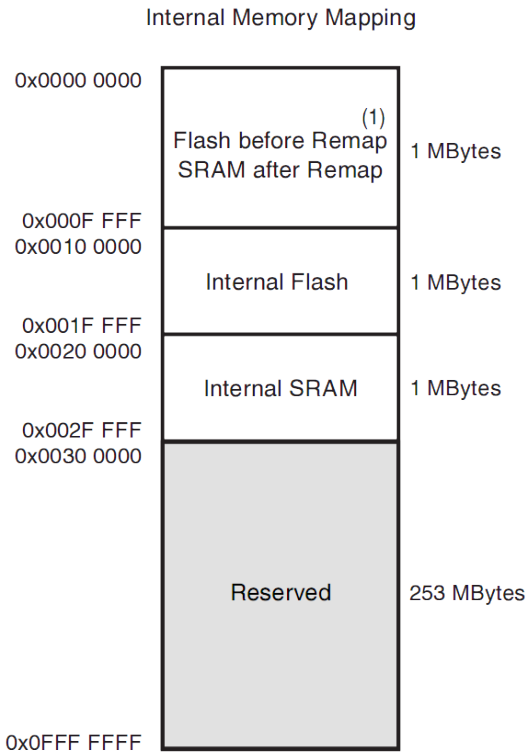
Táto správa sa posiela nadriadeným počítačom mikropočítaču. Posiela sa po prijatí správy BREAKPOINT o spôsobí pokračovanie vykonávania programu na adrese, na ktorej bol pozastavený.

1B
Typ

Tabuľka 4.7: Správa END_STRUCT_MODE ,ERROR, BREAKPOINT a CONTINUE

4.5 Firmware pre AT91SAM7S64

Úlohou firmwaru je ovládanie mikropočítača, zabezpečovanie požadovaných funkcií a komunikácia s hostiteľským počítačom. Rozoberme bližšie situáciu v akej sa nachádzame pri tvorbe firmwaru pre mikropočítač AT91SAM7S64. Tento mikropočítač obsahuje 64KB pamäte typu flash a 16KB pamäte typu SRAM. Mapovanie jednotlivých pamätí je zobrazené na obrázku.



Obr. 4.7: Pamäťová mapa AT91SAM7S64

Po resete procesora je mapa nasledovná: v spodnom 1MB pamäte je namapovaná pamäť flash, tá je namapovaná aj od 1MB. Od 2 MB je namapovaná interná pamäť SRAM. Spodný 1MB sa dá premapovať tak, aby tu bola namapovaná pamäť SRAM.

Na kompiláciu programov pre tento typ procesora sa dá použiť voľne dostupná verzia gcc (GNU C compiler). Konkrétne je potrebné použiť sadu utilít binutils, gcc a knižnicu newlib, ktorá slúži ako štandardná knižnica c volaní. Balík binutils obsahuje nástroje na zlinkovanie, assemblovanie a manipuláciu so súbormi, tak aby sa dosiahol požadovaný výstupný bin súbor, ktorý je možné prostredníctvom utilít na ISP(In-System programming) programovanie, od výrobcu mikroprocesora, použiť na naprogramovanie.Po kompilácii programu nasleduje fáza zlinkovania. Aby linker dodržal požadované mapovanie pamäte je ho nutné správne nakonfigurovať. Na to slúži konfiguračný súbor. Tu uvádzame len vybraté kľúčové časti konfiguračného súboru:

```
.text: {
```

```
_stext = .;
*(.text)
*(.rodata)
*(.rodata*)
. = ALIGN(4);
_etext = .;
}

.data : AT (ADDR (.text) + SIZEOF (.text) ) {
_sdata = .;
*(.vectors)
*(.data)
_edata = .;
}

.bss (NOLOAD) : {
. = ALIGN(4);
_sbss = .;
*(.bss)
_ebss = .;
}
```

Ako z konfigurácie vidno výstupný binárny súbor sa delí na sekcie (text, data, bss). Sekcia text obsahuje kód programu a read-only dáta. Druhá sekcia obsahuje read/write dáta, ktoré môžu byť modifikované. Tretia sekcia, bss (block started by symbol), obsahuje neinicializované premenné a z toho dôvodu ju nie je nutné vkladať do binárneho súboru. Pri nahratí programu sa časť kódu postará, aby oblasť pamäte vyhradená pre túto sekciu bola nainicializovaná na nulové hodnoty.

Uvedená konfigurácia hovorí, že sekcia text bude obsahovať sekcie text všetkých vstupných súborov (toto hovorí parameter *(.text)), ďalej bude obsahovať read-only dáta všetkých vstupných súborov (direktívy *(.rodata) a *(.rodata*). Definujú sa tu dve premenné _stext a _etext, ktoré obsahujú začiatočnú a koncovú adresu sekcie text. Hodnota konca sekcie text je zarovnaná na adresu deliteľnú 4.

Ďalšia sekcia je sekcia read/write dát. Obsahuje sekcie vectors zo všetkých vstupných súborov, a ďalej sekcie data zo všetkých vstupných súborov. Bude nahratá priamo za sekciu text t.j. bude v pamäti flash hneď za sekciou text (to zabezpečuje direktíva AT t.j. je nahratá na adresu load memory address t.j. adresa kde sa sekcia nachádza keď program nebeží). Štartovací kód musí túto sekciu prekopírovať na adresu v pamäti SRAM (na jej virtual memory address, čo je

adresa na, ktorej sa sekcia nachádza keď už program beží). Pre podporu štartovacieho kódu sú premenné _sdata a _edata, ktoré obsahujú začiatok a koniec sekcie data.

Posledná sekcia je sekcia neinicializovaných premenných, ktorú netreba nahrávať do výsledného binárneho súboru (direktíva NOLOAD). Túto sekciu nainicializuje štartovací kód na samé nuly. Symbol _sbss a _ebss slúžia ako podporné premenné pre tento štartovací kód.

Linkeru treba ešte povedať, že sekcia text bude bežať na adrese začiatku flash t.j. adresa 0x100000 a adresa dátovej sekcie bude 0x200000. To sa dosiahne prepínačmi -Ttext 0x100000 a -Tdata 0x200000.

4.5.1 Štartovacia sekvencia AT91SAM7S64

Aby mohol program napísaný v jazyku C bežať, treba nainicializovať niekoľko vecí, časť je napísaná v jazyku assembler a časť v jazyku C, tá časť čo je napísaná v jazyku assembler slúži hlavne na inicializáciu prostredia pre jazyk C a tá časť čo je napísaná v jazyku C slúži na inicializáciu periférii. Takže treba nainicializovať nasledovné:

- poskytnúť vektory výnimiek
- inicializovať kritické periférie
- inicializovať zásobníky pre dané módy procesora, ktoré aplikácia využíva
- inicializovanie pamäťových sekcií

4.5.2 Vektory výnimiek

Keď sa vyskytne výnimka procesor skočí na jednu z 8 inštrukcií na adresách 0x00 po 0x1c. Výnimky, ktoré nemajú byť obsluhované stačí nastaviť ako nekonečný cyklus na tú istú adresu, na ktorej sa inštrukcia obsluhy nachádza. Napr. pre reset vektor možno použiť nasledovnú inštrukciu:

```
reset_vector:
ldr pc,=reset_handler
```

Pre irq prerušenie:

```
irq_vector
b irq_handler
```

4.5.3 IRQ handler

Hlavnou úlohou IRQ handlera je vybrať adresu obslužnej rutiny. Adresa sa uchováva v IVR (interrupt vector register) AIC (advanced interrupt controller) kontroléra. Kód ktorý vyberá adresu obslužnej rutiny je nasledovný:

```
ldr r14,=AT91C_BASE_AIC
ldr r0,[r14, #AIC_IVR]
bx r0
```

Pred volaním obsluhy prerušenia však treba uložiť registre na zásobník. Ide hlavne o registre r0-r3 a r12, pretože tieto sa používajú ARM C konvenciou. Toto sa dosiahne kódom: (r14 obsahuje návratovú adresu obslužnej rutiny zvýšenú o 4 preto treba dekrementovať o 4)

```
sub r14, r14, #4
stmfd sp!, {r0-r3, r12, r14}
ldr r14,=AT91C_BASE_AIC
ldr r0,[r14, #AIC_IVR]
bx r0
```

Tento kód najskôr dekrementuje hodnotu návratu o 4, následne uloží registre na zásobník, do r14 uloží začiatočnú adresu periférie AIC (advanced interrupt controller) a následne hodnotu registra IVR (interrupt vector register) uloží do r0. Následne sa vykoná skok na túto rutinu. Po tom ako sa vykoná obslužná rutina prerušenia, tak treba uložiť.

4.5.4 Inicializácia flash a hodín signálu

Ako prvé musíme nakonfigurovať počet čakacích stavov flash. Pre našu aplikáciu sme sa rozhodli pre 48Mhz, Táto frekvencia vyžaduje 1 čakací stav, je to kvôli tomu, aby keď procesor beží na plnej rýchlosti tak flash nestíha, a procesor musí čakať na flash.

Ďalej sa inicializuje hlavný oscilátor a PLL (phase locked loop), obidva sa konfigurujú v pmc (power management controller). Prvým krokom je inicializácia hlavného oscilátora a čakanie kým sa stabilizuje. Oscilator sa štartuje zapísaním štartovacieho času a nastavením bitu MOSCEN v Main Oscilator Registry PMC kontroléra.

Keď je nainicializovaný hlavný oscilátor tak môžme inicializovať PLL. PLL je vytvorené z dvoch blokov. Prvý blok delí vstupný hodinový signál a druhý ho násobí. Obidva faktory (deliaci aj násobiaci) sa zapisujú v PLL registry PMC kontroléra. V našom projekte máme 18.432Mhz kryštál a chceme výstupnú frekvenciu okolo 96Mhz.

```
f_in = 18.432Mhz
DIV = 14
```

```
MUL = (73 - 1) = 72
f_out = (18.432/14) * 73 = 96.109Mhz
```

Taktiež musíme čakať kým sa PLL stabilizuje. Potom ako je nakonfigurovaný PLL môžeme nastaviť hlavné hodiny (main clock). Najskôr nastavíme deliaci faktor, a potom vyberieme ako vstup zdrojového signálu PLL.

4.5.5 Inicializácia kritických periférií (LowLevelInit)

Inicializácia kritických periférií pozostáva z krokov uvedených vyššie (flash a hodinové signály) a ďalej inicializácie AIC (advanced interrupt controller) a watchdog (popísané ďalej). Tieto operácie sú zvyčajne napísané v C. Pretože C funkcie môžu vyžadovať prístup k zásobníku je ho potrebné inicializovať. Stack pointer je r13. Nasledovný kód nastaví zásobník a zavolá funkciu LowLevelInit, ktorá vykoná inicializáciu kritických periférií:

```
ldr sp,=STACK_ADDR
ldr r0,=AT91C_LowLevelInit
mov lr, pc
bx r0
```

Potom čo prebehne procedúra LowLevelInit, program môže skočiť do hlavnej aplikácie (samozrejme musíme ešte nastaviť zásobníky a dátové segmenty).

4.5.6 WatchDog

Po resete je watchdog povolený. V aplikáciách, ktoré ho nepotrebujú sa dá vypnúť príkazom:

```
AT91_BASE_WDTC->WDTC_WDMR=AT91C_WDTC_WDDIS;C
```

4.5.7 Inicializácia zásobníkov

Každý ARM mód má vlastný zásobník (register sp), preto každý mód používaný aplikáciou musí mať inicializovaný register sp (zasobnik). Zásobník rastie smerom dolu. Vhodné je uložiť prvý zásobník na vrch internej RAM. Pre každý používaný mód sa vyhradí zásobník o vhodnej veľkosti. Mód supervisor a user majú veľké zásobníky, IRQ a FIQ módy majú stredne veľké, a ostatne majú len niekoľko bajtov.

Inicializácia zásobníkov sa uskutoční vstupom do príslušného módu a nastavenie zásobníka na požadovanú adresu.

```
alias r13 alias sp
```

Zásobník sa nastavuje tak, že sa nastaví register na požadovanú adresu prvého zásobníka a po nastavení sa dekrementuje o veľkosť zásobníka, nová hodnota registra r0 bude hodnotou pre nový zásobník. Opísané kódom:

```
ldr r0, =IRAMEND
/* vstup do interrupt modu */
mov CPSR_c, #ARM_MODE_IRQ | I_BIT | F_BIT
mov r13, r0
sub r0,r0, #IRQ_STACK_SIZE
/* vstup do supervisor modu */
mov CPSR_c, #ARM_MODE_SVC | F_BIT
mov r13,r0
```

4.5.8 Inicializácia pamäťových segmentov

Binárny súbor je rozdelený na dva segmenty. Prvý obsahuje kód aplikácie a read-only dáta. Druhý obsahuje read-write dáta, t.j. dáta, ktoré môžu byť modifikované. Tieto dve sekcie sú text a data.

Premenné v dátovom segmente môžu byť inicializované alebo neinicializované.Neinicializované premenné sú v špeciálnej subsekcii BSS (block started by symbol). Na začiatku vykonávania programu musí byť inicializovaný dátový segment. Toto je dôležité lebo read/write premenné sú uložené v ram.

Inicializované dáta sú v binárnom súbore a sú nahrané do pamati. V našom firmwary sa inicializované dáta nachádzajú za text sekciou. Štartová a koncová adresa dát, ktoré sa kopírujú je uložená v kóde pomocou špeciálnych kompilerovskych direktiv (viď vyššie). Napríklad:

```
_lp_data:
.word _etext
.word _sdata
.word _edata
```

Kopírovacia operácia pozostáva z nahratia týchto hodnôt, a prechádzanie cez konkrétne dáta. Príklad:

```
_init_data:
ldr r2,=_lp_data
ldmia r2,{r1, r3, r4}

1:
cmp r3,r4
```

```
ldrcc r2,[r1],#4
strcc r2,[r3],#4
bcc 1b
```

Tento kód funguje nasledovne: najskôr sa nahrajú hodnoty začiatku dátového segmentu v binárnom súbore, začiatku dátového segmentu v pamäti RAM a koncová adresa v pamäti RAM. Následne sa kontroluje či sme neskopírovali celý segment. Ak nie tak sa kopíruje zo zdroja dát do cieľovej adresy.

4.5.9 Príkaz REMAP

Potom, ako sa vykoná tento príkaz, je interná SRAM dostupná cez adresu 0x0 (predtým tam bola flash). Remap príkaz je dostupný cez používateľské rozhranie kontroléra pamäti a to konkrétne registrom MC_RCR (Remap Control Register). Ak sa nastaví bit RCB na jednotku tak sa vykoná tento príkaz. Ak sa znovu vykoná tento príkaz (opätovným zápisom RCB), tak sa toto mapovanie zruší.

4.5.10 Inicializácia AIC (advanced interrupt controller)

AIC dovoľuje definovať handler pre každý zdroj prerušenia. AIC je budený hodinovým signálom stále, takže netreba povoľovať jeho hodiny v PMC. Pri inicializácii je dôležité zakázať prerušenia a ich vyčistenie:

```
// zakázanie prerušení
AT91C_BASE_AIC->AIC_IDCR=0xffffffff;
// vyčistenie prerušení
AT91C_BASE_AIC->AIC_ICCR=0xffffffff;
```

AIC má interný hardvérový zásobník, pre jeho vyčistenie je nutné zapísať 8 krát nejakú hodnotu do EOICR (End Of Interrupt Command Register).

4.5.11 Konfigurácia prerušenia

Vyžaduje uskutočniť 5 krokov:

- Zakázať prerušenie
- Konfigurácia Source Mode Registra
- Konfigurácia Source Vector Registra
- Povolenie prerušenia na úrovni periférie

- Povolenie prerušenia na úrovni AIC

Prvý krok je potrebný, keby sa vyskytlo prerušenie v čase, v ktorom sa čítajú mode a vector registre, pretože by mohlo dôjsť k nepredvídateľnému správaniu. Na zakázanie sa zapíše ID prerušenia, ktoré treba zamaskovať, do registra Interrupt Disable Command Register (IDCR).Register Source Mode slúži na konfiguráciu priority prerušenia a trigger módu.Priorita môže byť od 0 (najnižšia) po 7 (najvyššia). Trigger mód pre prerušenia prichádzajúce od interných periférií musí byť level-senzitívny, externé prerušenia sú závislé od toho ako sú pripojené k čipu.

Register Source Vector obsahuje adresu obslužnej rutiny prerušenia.

Napokon zdroj prerušenia môže byť povolený na úrovni periférie (zvyčajne v mode registry) a v registry Interrupt Enable Command Register (IECR) AIC kontrolera.

4.5.12 Všeobecné používanie periférií

Väčšina periférií sa inicializuje nasledovnými krokmi:

- povolenie hodín periférie v PMC (Power management controller)
- povolenie periférie na pio (parallel input/output) pinoch
- konfigurácia zdroja prerušenia periférie v AIC
- povolenie zdroja prerušenia na úrovni periférie

Väčšina periférií nie je implicitne riadená hodinovým signálom. Preto na spustenie periférie treba explicitne povoliť hodiny. Robí sa to cez PMC. Výnimkou je System Controler, ktorý je implicitne riadený hodinovým signálom. Pre periférie, ktoré vyžadujú piny chipu, tak sa musia najskôr nakonfigurovať v PIO (paralel input/output kontroler). A ak periféria využíva prerušenia, tak sa tieto musia nastaviť v AIC.

4.5.13 PIT (Periodic Interval Timer)

Generuje periodické prerušenia. Používa MCK(Master clock) vydelené 16 ako vstupný hodinový signál, a obsahuje 20 bitové počítadlo. Vždy keď počítadlo dosiahne predprogramovanú hodnotu, tak sa vygeneruje prerušenie a inkrementuje sa druhé počítadlo. PIT je súčasťou System Controllera takže je implicitne riadený hodinovým signálom, t.j. netreba povoľovať jeho hodiny v PMC(ani sa to nedá). Mode register obsahuje PIV (periodic interval value), ktorá hovorí PIT-u kedy má resetovať interné počítadlo. Keď chceme generovať 1ms prerušenia, tak musí byť táto hodnota nastavená na počet tikov v jednej milisekunde:

```
PIV=(MCK/16)*0.001
```

Toto sa dosiahne nasledovným kódom:

```
AT91C_BASE_PITC->PITC_PIMR = AT91B_MCK/(16*1000) - 1;
```

Pred tým ako sa spustí timer sa musí nastaviť vektor prerušenia v AIC. Potom sa môže povoliť prerušenie v periférii v registri PIT Mode Register nastavením bitu PITIEN a môže sa v tej istej operácii spustiť nastavením bitu PITEN. Keď príde interrupt od PIT tak jeho potvrdenie (v periférii) sa jednoducho spraví prečítaním PIT Value registra. Tento register obsahuje dve hodnoty:

- aktuálna hodnota interného počítadla
- PICNT (Periodic interval counter)

Obslužná rutina prerušenia PIT je jednoduchá. Najskôr sa prečíta PIVR a získa sa hodnota PICNT, globálna premenná sa inkrementuje počtom tikov, ktoré sa prečítali. Keďže PIT zdieľa prerušenie s ostatnými perifériami system controlera, tak sa musí kontrolovať či prerušenie bolo skutočne vygenerované PIT. Toto sa dosiahne čítaním Status registra PIT a kontrolou bitu PITS, ktorý je nastavený, keď prerušenie spôsobil PIT.

4.5.14 Inicializácia DEBUG UNIT

Táto jednotka poskytuje dvoj-pinový UART. Je súčasťou system controllera, takže nie je nutné povolovať hodiny. Treba nakonfigurovať jej dva piny (DTXD a DRXD) v pio controllery. Zápis ID oboch pinov do registra PDR (PIO disable register) PIO kontrolera dovoľuje, aby periféria kontrolovala dané piny. Niektoré piny sa zdieľajú medzi dvoma rôznymi perifériami. Na to, ktorá kontroluje daný pin je register Peripheral A Select Register (ASR) a Peripheral B Select Register (BSR).

Ďalšia akcia je zakázanie logiky príjmu a odosielania, ako aj zakázanie prerušení. Toto dovoľuje ľahkú a bezproblémovú rekonfiguráciu. Nastavenie bitov RSTRX a RSTTX v CR (Control register) DBGU resetuje a zakáže prijímač aj vysielač. Nastavenie všetkých bitov v Interrupt Disable Register IDR zakáže všetky prerušenia, ktoré prichádzajú od Debug Unit. Potom treba nastaviť baud rate. Hodiny, ktoré prichádzajú do DBGU sú MCK. Tieto sa delia predprogramovaným faktorom. Vzťah na výpočet deliteľa (clock divisor) je nasledovný:

```
CD = MCK/(16*baudrate)
```

Tak napríklad ak je MCK = 48 Mhz a chceme 115200 tak CD = 26. Mode register (MR) ma dve konfigurovateľné hodnoty. Prvá je channel mode. Táto, je dobré, keď sa nastaví na normal. Ak sa pole CHMODE nastaví na nuly tak sa vyberie normal mod. MR ďalej obsahuje nastavovanie paritného bitu. Môže byť even, odd, mark a space. Žiadna parita je 1xx t.j. na bitoch xx nezáleží.

Teraz je DBGU plne nakonfigurovaná, stačí povoliť vysielač . To sa spraví nastavením bitu TXEN v Control Registry.

Posielanie znaku DBGU je jednoduché. Stačí zapísať znak do THR (Transmit holding register), ale pred tým musí byť vysielač pripravený. Dva bity v DBGU status registry indikujújej stav, bit TXEMPTY indikuje, že vysielač je povolený a posiela znaky. Ak je nastavený tak sa neposiela žiaden znak. Ďalší bit je TXRDY. Ak je tento bit nastavený tak vysielač skončil kopírovanie hodnoty THR do vnútorného shift registra, ktorý používa na posielanie dát. V praxi to znamená, že do THR sa môže zapísať znak ak je TXRDY nastavený, bez ohľadu na to aká je hodnota v TXEMPTY. Keď je nastavený TXEMPTY tak skončil celý transfer.

4.5.15 Inicializácia USART

Inicializácia pozostáva zo zakázania kontrolovania pinov PIO-m, čím sa dosiahne to, aby dané piny kontrolovala periféria. Potom treba vybrať, či to bude periféria A alebo periféria B (keďže piny sú zdieľané medzi dvoma perifériami). USART je periféria A. Keďže USART nie je implicitne riadený hodinovým signálom, treba pre správnu funkčnosť t.j. pre zapnutie povoliť jeho hodinový signál. To sa robí zápisom ID periférie do registra PCER (Peripheral Clock Enable Register). Keď je USART zapnutý tak treba nakonfigurovať baud rate. Vzťah na výpočet je rovnaký ako pre DBGU:

```
CD=MCK/(baudrate*16)
```

Po nastavení prenosovej rýchlosti treba nastaviť mód USARTU. To sa dosiahne nastavením registra MR(mode register) USARTU. Tu sa dá vybrať prenosový mód USARTU, počet dátových bitov, výber zdroja hodinového signálu, výber parity a počet stop bitov. Keď je toto povolené tak potom stačí povoliť príjem a odosielanie znakov.

4.5.16 Príjem a vysielanie znakov prostredníctvom USART

Pre podporu príjmu znakov je v CSR (Channel Status Register) bit RXRDY, ktorý indikuje či v RHR (Receive Holding Register) je prijatý znak, ak áno tak je nastavený bit RXRDY na jednotku. Pre vysielanie znakov je v CSR bit TXRDY, ktorý indikuje, či je v THR (Transmitt Holding Register) nevyslaný znak ak áno tak je tento bit nastavený na 0, ak je THR prázdny tak je nastavený na 1.

4.5.17 LCD displej

Ovládanie LCD displeja

LCD displej je zapojený ako externá periféria cez piny PIO. Preto je nutné najskôr nakonfigurovať PIO kontrolér, tak aby mohla prebiehať komunikácia s displejom. PIO treba nakonfigurovať

nasledovne:

- PIO kontroluje piny, dátovej zbernice displeja t.j. piny PA31 až PA24, a piny riadiacej zbernice PA23, PA8 a PA7. Toto sa dosiahne zápisom do PIO registra PER (pio enable register).
- PIO piny súvisiace s displejom nastavíme ako výstupné. Toto sa dosiahne zápisom do registra OER (output enable register).
- piny dátovej zbernice displeja sú zapisované ako 8 bitová hodnota. Na toto treba PIO nastaviť tak, aby umožnil zápis naraz celej 8 bitovej hodnoty. Toto sa dosiahne zápisom do registra OWER (Output Write Enable Register).
- ešte zakážeme interné pull-up rezistory. Toto sa dosiahne zápisom do registra PUDR (Pull-Up Disable Register)

Komunikácia s LCD displejom

Použitý displej je z rady EA DOG-M od firmy Electronic Assembly. Má radič ST7036, ktorý je kompatibilný so štandardným radičom HD44780 od firmy HITACHI. V našom zapojení sa s radičom komunikuje cez 8 bitovú dátovú zbernicu a 3 bitovú riadiacu zbernicu. Riadiaca zbernica pozostáva z 3 signálov:

- RS (Register Select) výber inštrukčného (RS=0) alebo dátového (RS=1) registra
- RW (Read/Write) výber čítania (RW=1) alebo zápisu (RW=0)
- E (Enable) zostupnou hranou sa prenesú údaje do radiča t.j. slúži ako riadiaci kvázi-hodinový signál.

Pred samotným zápisom znakov na displej treba najskôr displej zapnúť. To sa udeje zapísaním inštrukcie do inštrukčného registra. Konkrétne je to inštruckia Display On/Off. V tejto inštrukcii sa dá zapnúť aj kurzor, poprípade jeho blikanie. Zápis dát na displej potom prebieha nasledovne:

1. zapíše sa hodnota na dátovú zbernicu displeja,
2. nastaví sa register RS na dátový register alebo inštrukčný register (podľa požiadavky na zápis inštrukcie alebo zápis znakov)
3. nastaví sa RW na zápis
4. nastaví sa E na 1 potom následne na 0, čím zabezpečíme zostupnú hranu tohto signálu

Instruction	Instruction Code										Description	Instruction Execution Time		
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		OSC= 380kHz	OSC= 540kHz	OSC= 700kHz
Clear Display	0	0	0	0	0	0	0	0	0	1	Write "20H" to DDRAM. and set DDRAM address to "00H" from AC	1.08 ms	0.76 ms	0.59 ms
Return Home	0	0	0	0	0	0	0	0	1	x	Set DDRAM address to "00H" from AC and return cursor to its original position if shifted. The contents of DDRAM are not changed.	1.08 ms	0.76 ms	0.59 ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies display shift. These operations are performed during data write and read.	26.3 µs	18.5 µs	14.3 µs
Display ON/OFF	0	0	0	0	0	0	1	D	C	B	D=1:entire display on C=1:cursor on B=1:cursor position on	26.3 µs	18.5 µs	14.3 µs
Function Set	0	0	0	0	1	DL	N	DH	IS2	IS1	DL: interface data is 8/4 bits N: number of line is 2/1 DH: double height font IS[2:1]: instruction table select	26.3 µs	18.5 µs	14.3 µs
Set DDRAM Address	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set DDRAM address in address counter	26.3 µs	18.5 µs	14.3 µs
Read Busy Flag and Address	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Whether during internal operation or not can be known by reading BF. The contents of address counter can also be read.	0	0	0
Write Data to RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Write data into internal RAM (DDRAM/CGRAM/ICONRAM)	26.3 µs	18.5 µs	14.3 µs
Read Data from RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Read data from internal RAM (DDRAM/CGRAM/ICONRAM)	26.3 µs	18.5 µs	14.3 µs

Obr. 4.8: Podporované inštrukcie radiča ST

4.5.18 Ošetrenie stavu tlačidiel

Tlačidlá sú pripojené na vstupné porty PIO, t.j. piny, na ktoré sú napojené tlačidlá, sú nakonfigurované ako vstupné a kontrolu nad nimi má PIO. Ďalej sú tieto nastavené tak, aby generovali prerušenie pri zmene stavu na danom pine. Toto sa dá prečítať z registra ISR (Interrupt Status Register) PIO kontroléra. Stav tlačidiel t.j. či sú stlačené alebo uvoľnené, sa dá prečítať z registra PDSR (Pin Data Status Register). Obslužná rutina číta hodnoty týchto registrov a nastavuje vhodné premenné, ktoré zodpovedajú stlačenému tlačidlu.

4.5.19 Test pamäti

Pamäť testujeme algoritmom “chodiacia jednotka”. Na danú adresu sa zapíše hodnota (hexa) 0x00000001, táto hodnota sa potom prečíta a ak je rovnaká na sa táto hodnota posunie o jeden bit doľava, a znova sa zapíše na adresu v pamäti, znovu sa prečíta, a vykoná sa obdobná kontrola a posun ako v predchádzajúcom kroku. V prípade, že sa neprečíta správna hodnota tak to signalizuje chybu, a spôsobí výpis chybového hlásenia na displeji.

4.5.20 Body prerušenia - breakpointy

Prerušenie v programe, sme vyriešili nasledovne. V používateľskom programe sa na žiadané miesto v programe vloží špeciálna inštrukcia, ktorá spôsobí zavolanie obslužnej rutiny preruše-

nia. Táto rutina spôsobí prerušenie práve vykonávaného programu a prejdenie do štruktúrovaného módu, v ktorom je umožnená komunikácia s nadriadeným počítačom. Keď používateľský program prijme správu JUMP_RES, ktorá znamená, že mikropočítač spustil vykonávanie nahratého programu, tak čaká na správu BREAKPOINT, ktorá znamená, že program dosiahol bod prerušenia.

Na riešenie bodu prerušenia existujú dve možnosti. A to buď použitie inštrukcie swi (Software Interrupt), ktorá spôsobí zavolanie obslužnej rutiny prerušenia. Druhou možnosťou je vyhradiť špeciálnu adresu v pamäti na uloženie začiatočnej adresy obslužnej rutiny. Výhodou prvého riešenia je to, že netreba vyhradzovať špecálnu adresu. Pre náš projekt sme sa rozhodli pre inštrukciu swi, ktorá je univerzálnejšia. Takže vývojové prostredie vloží inštrukciu swi na požadovanú adresu. V zápise C programu to znamená:

```
asm("swi");
```

4.5.21 Plánovač firmwaru

Na podporu funkcií firmwaru (komunikácia, práca s registrami, práca s registrami, práca s pamäťou, načítanie programu v zadefinovanom formáte, práca s breakpointami, spustenie/zastavenie vykonávania, programu) bude naimplementovaný plánovač. Tento plánovač bude prepínať medzi jednotlivými vláknami, a tým im priraďovať procesorový čas. Čas do plánovača bude generovaný PITC (Periodic Interval Timer). Vlákna budú bežať v časových kvantách. Pridelovanie časových kvánt jednotlivým vláknam bude zabezpečovať funckia schedule, ktorá vyberie z listu procesov pripravených na bežanie, jedno vlákno. Toto vlákno potom využije celé alebo časť svojho časového kvanta.

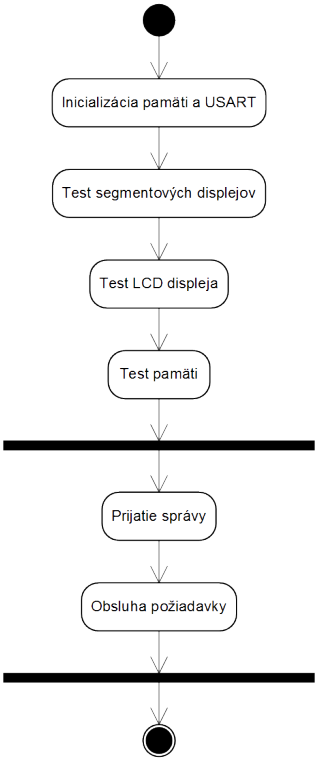
Plánovač pre projekt nie je nutný, avšak ak ho nepoužijeme nebude sa dať použiť funkcia zastavenia vykonávania programu. Je to z toho dôvodu, že ak je spustený ladený program, nie je spôsob ako ho explicitne ukončiť, pretože program si nemusí byť vedomý toho, že ho niekto chce ukončiť. Mohlo by sa to dosiahnuť, ale tak, že program kontroluje nejakú stavovú premennú a podľa nej sa rozhoduje či sa ukončí alebo nie. Ukončenie by znamenalo návrat riadenia do firmwaru. A tým obnovenie komunikácie s nadradeným počítačom. Ale toto by znamenalo pridávanie kontrolného kódu do programu a nevieme, na ktoré miesto ho pridať, bez analýzi kódu. Pretože by sa mohol priadať do miesta kde nie je vôbec tento kód volaný a to by znamenalo, že sa tento kód vôbec nezavolá.

4.6 Firmvér pre ATMega32

Výrobca procesorov rady AVR, firma Atmel, poskytuje program AVR Studio 4, ktorý slúži na písanie obslužného softvéru pre mikropočítače s týmito procesormi. Softvér má grafické rozhranie, podporuje písanie programov v assembleri, ale aj v jazyku C. Pre písanie v jazyku C je potrebný kompilátor WinAVR, ktorý je tiež voľne dostupný. Programy je možné kompilovať, krokovať, simulovať a odladďovať.

Základom firmvéru mikopočítača budú 2 základné funkcie:

- Test obvodov a funkcií mikropočítača (POST [1])
- Program, ktorý bude spolupracovať so softvérom na PC



Obr. 4.9: Návrh firmvéru

4.6.1 POST testovanie

Testovanie je dôležité, aby sme vedeli, či mikropočítač pracuje správne. Po štarte sa bude vykonávať test:

- 7-segmentových LED displejov

[1] POST - Power On Self Test

- LCD displeja
- Pamäte

Pri 7-segmentových displejoch bude treba napísať program, ktorý overí každý segment displeja. Postupne sa budú vypisovať hodnoty, aby sa preverila funkčnosť každej číslice.

Pri teste LCD displeja je treba vypísať určitú hlášku. Počas úvodných testov sa ich výsledky budú zobrazovať na LCD.

Pre test pamäte je možné použit viacero techník. Každá robí zápis hodnoty na pozíciu a následné čítanie.

jednoduchý zápis = zápis rovnakej hodnoty na všetky miesta pamäti

náhodný zápis = zápis hodnoty na náhodné miesta pamäti

šachovnica = použitie striedajúcich hodnôt

parita = zápis hodnoty bitu parity adresy pamäťovej bunky

diagonála = zápis odlišnej hodnoty do riadka a stĺpca

postupujúca jednotka/nula = inicializujeme miesta v pamäti na rovnaku hodnotu, napr.: 0. Pohybujeme sa po všetkých miestach a pokúšame sa na dané miesto zapísať 1 a znova prečítať. Po ukončení testu sa hodnota daného miesta zmení späť na inicializačnú hodnotu.

pochodujúca jednotka/nula = podobne ako v predchádzajúcom prípade, ale hodnota daného miesta sa nemení

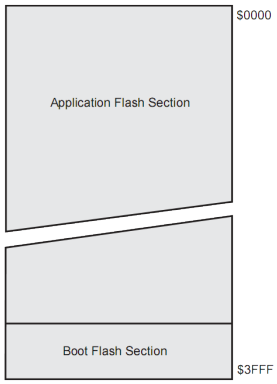
4.6.2 Pamäť

ATmega32 obsahuje 32 kB vnútornej programovateľnej flash pamäte na uloženie programu. Flash pamäť programu je kvôli softvérovej bezpečnosti rozdelená na dve sekcie, na Boot Program sekciu a Application Program sekciu (obrázok 4.10). Mala by zvládnuť viac ako 10 000 zapísaní a mazaní. Programové počítadlo má 14 bitov, ktorým je možné adresovať priestor o veľkosti 16 K.

Spodných 2144 dátových miest je vyhradených pre súbor registrov, pamäť pre vstupno-výstupné operácie a internú SRAM pamäť. Z toho prvých 96 pozícií adresuje súbor registrov(32) a pamäť pre vstupno-výstupné operácie(64), ďalších 2048 adresuje internú SRAM pamäť.

4.6.3 Inicializácia a používanie USART

Predtým ako chceme používať USART na akúkoľvek komunikáciu, je potrebné spraviť inicializáciu. Tá pozostáva z nastavenia rýchlosti, formátu rámca a povolenia vysielania a prijímania. Inicializáciu nám zabezpečí nasledujúca procedúra:



Obr. 4.10: Rozdelenie pamäte

```
void USART_Init( unsigned int baud )
{
  /* Set baud rate */
  UBRRH = (unsigned char)(baud>>8);
  UBRRL = (unsigned char)baud;

  /* Enable receiver and transmitter */
  UCSRB = (1<<RXEN)|(1<<TXEN);

  /* Set frame format: 8data, 2stop bit */
  UCSRC = (1<<URSEL)|(1<<USBS)|(3<<UCSZ0);
}
```

Komunikácia môže prebiehať v troch režimoch. V závislosti na použitom režime je potrebné nastaviť rýchlosť prenosu, ktorá závisí od frekvencie oscilátora. Prehľad o vypočítaní parametra rýchlosti poskytuje tabuľka

Režim	Rovnica pre výpočet hodnoty	Rovnica pre výpočet hodnoty UBRR
Asynchrónny režim normálny (U2X = 0)	$BAUD = \frac{f_{OSC}}{16(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{16(BAUD)} - 1$
Asynchrónny režim 2-násobný (U2X = 1)	$BAUD = \frac{f_{OSC}}{8(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{8(BAUD)} - 1$
Synchrónny režim Master	$BAUD = \frac{f_{OSC}}{2(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{2(BAUD)} - 1$

Tabuľka 4.8: Prehľad režimov a vzorce pre výpočet rýchlosti

Vysielanie dát cez USART sa skladá z dvoch krokov, prvý je cyklus čakania, pokiaľ sa vyprázdni vysielací buffer, pričom sa kontroluje Data Eegister Empty (UDRE) flag, druhým je vloženie samotných dát do buffra a odoslanie. Vysielanie dát nám zabezpečí nasledujúca procedúra:

```
void USART_Transmit( unsigned char data )
{
  /* Wait for empty transmit buffer */
  while ( !( UCSRA & (1<<UDRE)) );

  /* Put data into buffer, sends the data */
  UDR = data;
}
```

Prijímanie dát cez USART pozostáva z čakacieho cyklu, počas ktorého je kontrolovaný Receive Complete (RCX) flag. Prijímanie dát nám zabezpečí nasledujúca procedúra:

```
unsigned char USART_Receive( void )
{
  /* Wait for data to be received */
  while ( !(UCSRA & (1<<RXC)) );

  /* Get and return received data from buffer */
  return UDR;
}
```

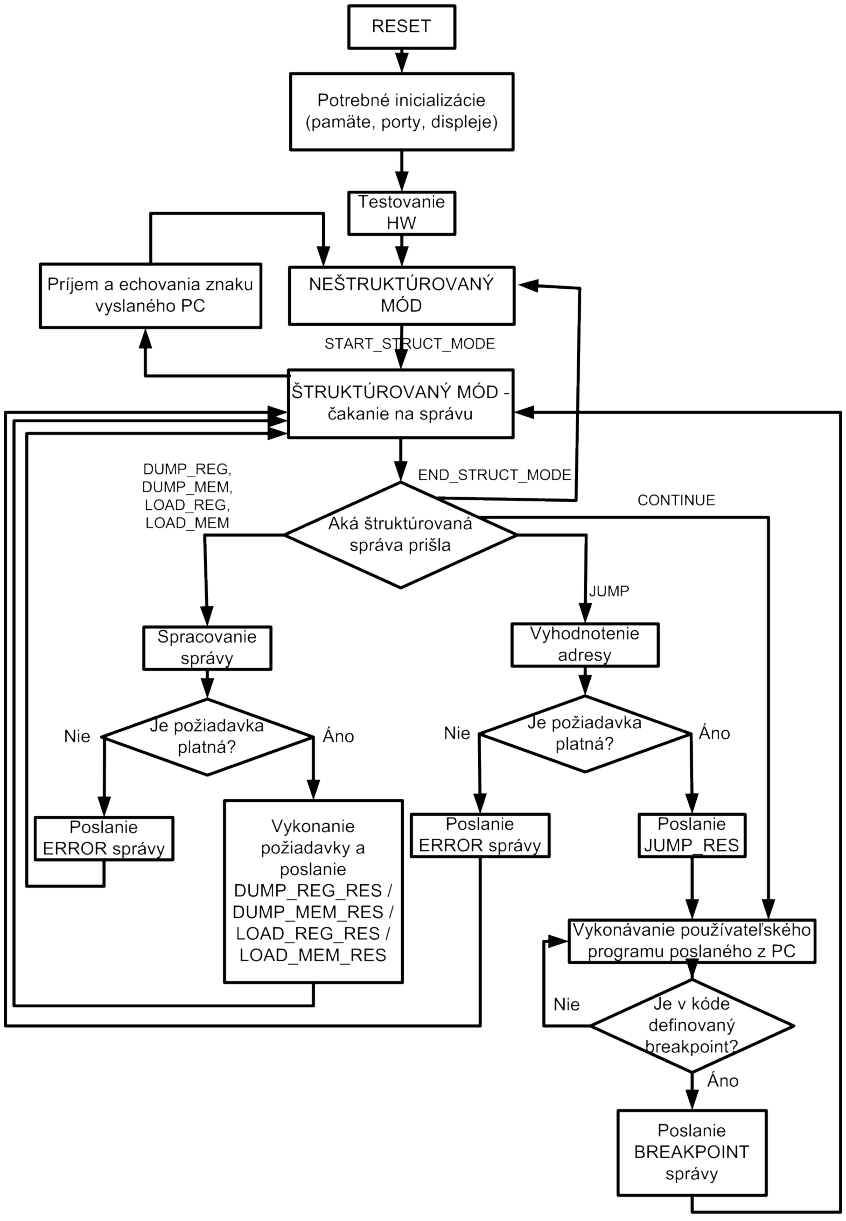
Procedúra pre vyprázdnenia buffra:

```
void USART_Flush( void )
{
  unsigned char dummy;
  while ( UCSRA & (1<<RXC) )
    dummy = UDR;
}
```

4.6.4 Podrobnejší popis firmvéru

Funkcionalitu firmvéru možno rozdeliť do dvoch základných celkov. Jedným z nich je inicializácia a potrebné otestovanie komponentov, druhým je komunikácia z hostiteľským PC s realizáciou nahrania externého programu a jeho následná inicializácia. Komunikácia je realizovaná prostredníctvom všeobecného komunikačného protokolu pre oba mikropočítače definovaného

v rámci tohto projektu. Konkrétna realizácia pre ATmega32 je schematicky znázornená na obrázku 4.11. Mikropočítač sa môže nachádzať v dvoch základných módoch. V prípade neštruktúrovaného módu len echuje znaky prijaté z PC cez USB rozhranie.



Obr. 4.11: Podrobnejší návrh firmvéru

Druhým módom je režim posielania štruktúrovaných správ. V rámci neho je možné zrealizovať nahranie externého programu a jeho následné spustenie. Ak užívateľ do programu vložil body prerušenia, tzv. breakpointy, je v prípade takéhoto prerušenia možné získať hodnoty z pamäte programu mikropočítača, registrov rovnako ako ich modifikácia. Správy sú podrobne definované v rámci tohto dokumentu. Po vykonaní príslušných požiadaviek môže používateľ vrátiť program mikropočítača opäť do chodu.

4.6.5 Realizácia používateľského programu z PC

Jednou zo základných vlastností firmvéru počítača je možnosť priameho programovania z PC. Skompilovaný program je pomocou definovaných správ prenášaný cez USB rozhranie a prijímaný do procesora po bajtoch. Procesor ATmega32 je navrhnutý s Harvardskou architektúrou, čo znamená, že má oddelenú pamäť programu a pamäť údajov. Navyše pamäť programu má organizovanú po 16 bitových slovách, keďže všetky jeho RISC inštrukcie majú 16 alebo 32 bitov. Pamäť údajov SRAM je však organizovaná po á bitových slovách, pretože sa jedná o 8-bitový procesor. Rovnako aj všetky pracovné registre sú 8-bitové. Zapisovanie používateľského programu do pamäte SRAM teda neprichádza do úvahy. Jedinou možnosťou je programovanie flash pamäte programu priamo za behu procesora. Na toto slúži špecifická inštrukcia SPM. Pracuje s dvomi 8-bitovými registrami, ktorých obsah zreťazí a zapíše na výsledné miesto v pamäti programu. Inštrukciu však nie je možné spustiť odkiaľkoľvek, ale len z tzv. Bootloader sekcie flash pamäte. Toto obmedzenie je implementované kvôli softvérovej bezpečnosti aplikácií. Po dokončení zápisu programu sa čaká na správu JUMP. Táto správa je impulzom pre skok na začiatok programu, a teda jeho následné spustenie. Po ukončení vykonávania používateľského programu je mikropočítač opäť v režime štruktúrovaných správ a očakáva ďalšie regulovanie z PC.

4.6.6 Realizácia bodov prerušenia - breakpointy

Dôležitou funkciou firmvéru je možnosť prerušenia chodu používateľského programu na určitú dobu. Na strane PC je toto realizované vložením definovaného reťazca do kódu za inštrukciu, po ktorej sa má chod programu zastaviť. Samotné zastavenie je v prípade ATmega32 možné realizovať dvomi spôsobmi.

Prvým je fiktívne softvérové prerušnie. Procesor totiž nemá možnosť skutočného softvérového prerušenia. Je však možné využiť pin externého prerušenia. Hodnotu pinu je totiž možné zmeniť priamo z procesora, čo vychádza z implementácie vstupno/výstupných portov. Procesor teda môže jednoduchou inštrukciou zmeniť jeho hodnotu, čo následne spustí obslužnú rutinu daného externého prerušenia. Tento spôsob však pre náš projekt nie je vhodný, pretože všetky tri piny externých prerušení sú použité na iný účel.

Druhým spôsobom je použitie absolútneho skoku. Obslužná rutina sa v rámci firmvéru nachádza na konkrétnej adrese v pamäti programu, ktorú je po kompilácii frimvéru možné jednoducho zistiť. Ak kompilátor používateľského programu na PC vloží na každý požadovaný bod prerušenia absolútny skok na spomínanú adresu, je obslúženie zastavenia programu vyriešené. Obslužná rutina prerušenia sa správa ako štruktúrovaný režim mikropočítača. Reaguje na podnety zo strany PC až pokým neprijme správu CONTINUE. Vtedy sa vráti k vykonávaniu programu v bode, kde bolo zavolané prerušenie a pokračuje v jeho vykonávaní.

Kapitola 5

Implementácia

5.1 Implementácia HW časti mikropočítača ARM

Táto kapitola opisuje praktickú realizáciu a oživovanie experimentálneho mikropočítača. Zaoberá sa aj problémami a chybami návrhu a ich odstraňovaním.

5.1.1 Zmeny v návrhu

Oproti návrhovej fáze projektu boli vykonané tu opísané zmeny. Namiesto displeja BC1602A [18], bol vybraný displej EA DOG-M 162 [20], lebo umožňuje aj 3,3V napájanie, čo predošlý typ neumožňoval. Tiež má dostupnú podrobnejšiu dokumentáciu a nepotrebuje žiadne montážne otvory. Vyrába sa vo viacerých farebných kombináciách. Jeho zapojenie zodpovedá katalógovému zapojeniu, využíva sa pri tom 8 údajových bitov. Návrh plošného spoja

Návrh bol uskutočnený pomocou programu Eagle [1] . Najskôr bolo potrebné do programu zakresliť schému zapojenia. Program má intuitívne ovládanie a rozsiahle knižnice, v ktorých sa nachádzajú schematické značky a puzdrá jednotlivých súčiastok, konektorov a obvodov. Niektoré obvody však v knižnici chýbali a tak bolo potrebné zakresliť ich schematické značky a puzdra. Šlo o obvod EA-DOGM162W-A, čiže displej, a puzdro procesora AT91SAM7S64, ktoré bolo zakreslené podľa katalógového listu.

Neskôr sa podarilo zistiť, že uvádzané údaje v rámci výkresov puzdra procesora "E1", "E2", "D1" a "D2" sú o 0,01" menšie, ako by mali podľa špecifikácie použitého puzdra LQFP64 byť. K rovnakému záveru vedie aj sčítanie medzier medzi vývodmi puzdra "e" a porovnanie s hodnotou "D2" alebo "E2". Tieto chybné údaje sa v čase vypracovávania tohto projektu nachádzali na 518. strane v [6]. Puzdro bolo potom opravené.

Plošný spoj bol navrhnutý dvojvrstvový, bez spájkovacej masky, so zreteľom na jednoduchú prístupnosť jednotlivých prvkov. Tiež bolo kontrolované, aby vodivé cesty vo vrchnej a spod-

[1] http://www.cadsoft.de/ - CadSoft Online: EAGLE Layout Editor

nej vrstve nešli ponad seba a nevytvárali slučky, čo zabraňuje rôznym parazitným vplyvom (indukcia, kapacita).

Keď bol plošný spoj hotový, bola zistená nasledujúce chyby návrhu. Pri oboch RS-232 portoch vymenené zapojenie "receive" a "transmit". Cesty boli preškriabané a nahradené správne zapojenými drôtovými prepojkami.

5.1.2 Oživovanie

Najskôr bol osadený napäťový stabilizátor a k nemu patriace kondenzátory. Po pripojení na laboratórny zdroj sa však hlásil skrat. To bolo tým, že puzdro stabilizátora TO-220 má na zadnej strane vyvedenú zem a táto zadná strana sa dotýkala vodivých ciest na plošnom spoji. Preto musela byť vložená izolácia.

V ďalšom kroku bol osadený procesor a príslušné kondenzátory. Na vývode procesora VDDOUT bolo namerané napätie okolo 1,8V, čo znamená, že v procesore integrovaný generátor tohto napätia bol funkčný.

Keďže procesor je v SMD puzdre LQFP-64, je napevno prispájkovaný na doske a tak neexistuje žiadny spôsob, ako ho programovať bez použitia buď SAM BA rozhrania, ktoré využíva USB alebo RS-232 porty alebo rozhrania JTAG. Preto bol osadený USB konektor a príslušné súčiastky. Po pripojení napájacieho napätia a pripojení mikropočítača USB káblom k počítaču by sa malo v operačnom systéme objaviť nové zariadenie, čo sa však nestalo.

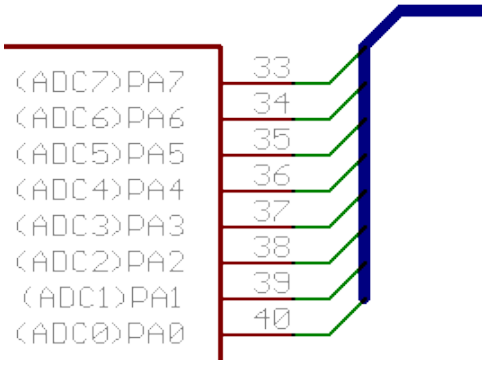
Podozrenie padlo na FET tranzistor slúžiaci na USB pull-up control, toto zapojenie je na obrázok 4.2. a kopíruje katalógové zapojenie. Potrebný tranzisotr BSS92 [21] nebol dostupný na sklade žiadnej firmy predávajúcej súčiastky a tak bol zvolený ekvivalent. IRF9520PBF [22]. Tento tranzistor síce je P-kanálový MOSFET, ale ide o výkonový tranzistor až do 50W. Nakoniec bol tranzistor odspájkovaný a nahradený prepojkou a bola zmenená hodnota rezistora R10 z 330K na 10K. Táto úprava vychádza z poznatkov získaných na fóre www.at91.com, ktoré ukázali, že v procesore integrovaný USB port dokáže fungovať aj bez problémového tranzistora.

Po tejto úprave sa už mikropočítač po pripojení k PC ohlásil, SAM-BA konzola bola funkčná, takže mohla začať fáza programovania. Prvý program mal za úlohu blikať LED diódou, ktorá bola dočasne pripojená namiesto tlačítka. Postupne boli osadené všetky súčiastky (budič portov RS-232, displej) a ich funkcie overené. Nevyskytli sa už žiadne vážne problémy a implementácia HW časti teda bola úspešná.

5.2 Implementácia HW časti mikropočítača ARM

5.2.1 Realizácia dátovej zbernice

Keďže procesor ATmega32 nepodporuje vyvedenie externej adresnej zbernice, bolo potrebné zrealizovať implementáciu zbernice manuálne. Na tento účel bol využitý vstupno-výstupný port A procesora. Zapojenie je znázornené na obrázku 5.1.



Obr. 5.1: Realizácia dátovej zbernice

Zbernica sa využíva na posielanie dát radiču USB rozhrania, do záchytných registrov pre 7-segmentové displeje a LCD displej. Pripojenie zariadení k zbernici je realizované prostredníctvom dekódera 74LS137N. Jeho vstupy A, B, C sú vyvedené na tri inak nevyužité piny procesora 22, 23 24, rovnako ako aktivačný vstup (pin 28). Výstupy dekódera sú negované, preto je za ne potrebné pripojiť invertory. Výstupy y0 až y5 aktivujú záchytné registre pre 3 dvojciferné displeje DA56-11GNA. Výstup y6 aktivuje záchytný register LCD displeja. Pin y7 je nevyužitý.

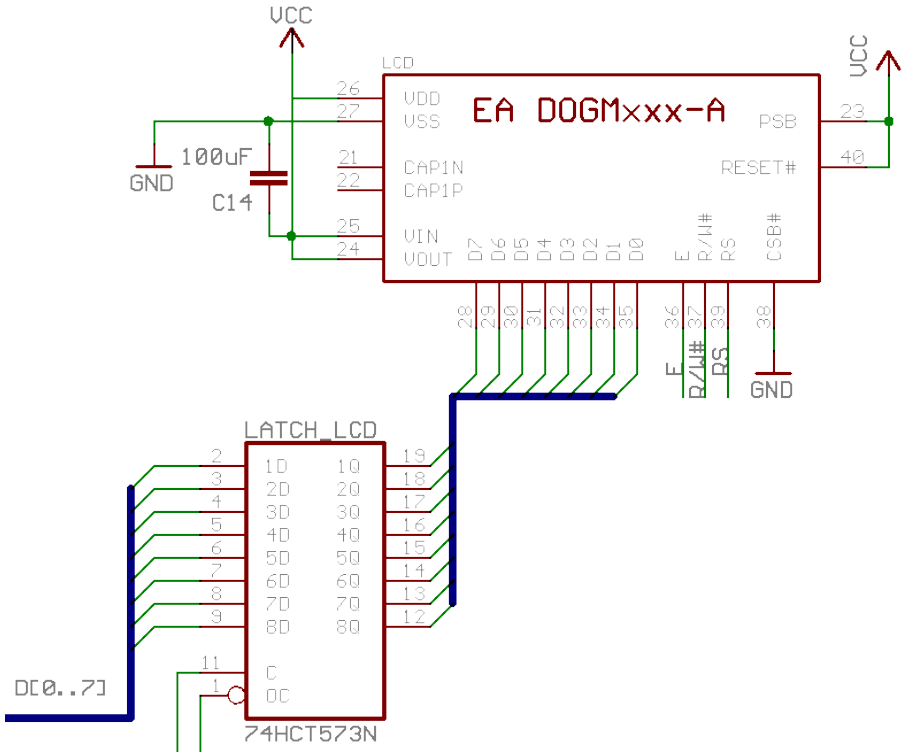
Pri akomkoľvek presune dát z alebo do procesora je potrebné všetky obvody aktivovať a ovládať manuálne prostredníctvom príslušných pinov procesora. Implementáciu zabezpečí firmvér mikropočítača.

Postup pri práci s dátami:

1. Aktivácia obvodu dekódera
2. Nastavenie vstupu dekódera na požadovanú hodnotu zodpovedajúcu zariadeniu, s ktorým si bude procesor vymienať dáta
3. V prípade príjmu dát procesor číta dáta z portu A, v prípade vysielania dát nastaví procesor aktívne dáta na port
4. Deaktivácia zariadenia zmenou vstupnej hodnoty dekódera

5.2.2 Zapojenie LCD displeja

Na realizáciu LCD displeja je použitý obvod MC1602E. Zapojenie je znázornené na obrázku 5.2.



Obr. 5.2: Zapojenie LCD displeja

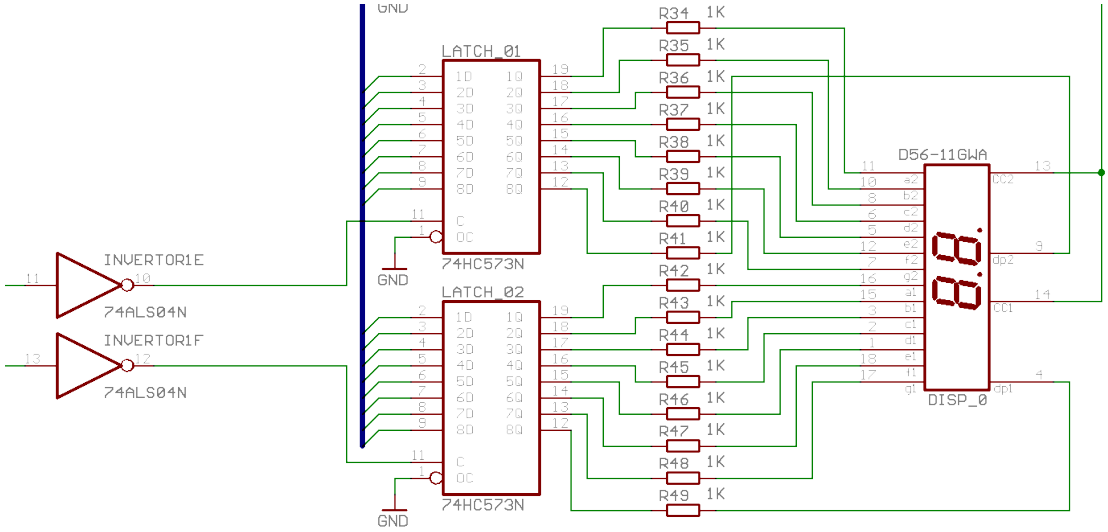
Dáta sú k obvodu privádzané pomocou dátovej zbernice. Kvôli zaručeniu bezproblémovej komunikácie je medzi zbernicou a obvodom zapojený záchytný register ovládaný centrálnym dekóderom. Riadiace vstupy displeja E, R/W# a RS sú vyvedené priamo na piny procesora 19-21. Riadenie displeja bude zabezpečené manuálne prostredníctvom týchto pinov.

5.2.3 Zapojenie 7-segmentových displejov

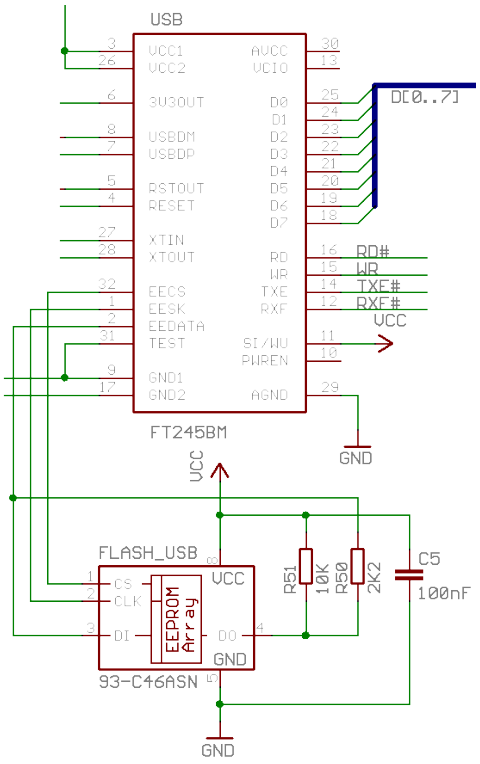
7-segmentové displeje sú realizované prostredníctvom záchytnych registrov 74HC573N. Toto riešenie umožňuje zobraziť akýkoľvek znak a nie je limitované znakovou sadou ako by to bolo v prípade využitia špeciálnych dekóderov. Ukážkové zapojenie je na obrázku 5.3.

5.2.4 Zapojenie USB rozhrania

USB rozhranie je realizované podľa návrhu. Zapojenie je realizované podľa technickej dokumentácie výrobcu obvodu FT245BM. Ukážka zapojenia dôležitých súčastí je na obrázku 5.4.



Obr. 5.3: Zapojenie 7-segmentových displejov



Obr. 5.4: Zapojenie USB rozhrania

Dátové rozhranie obvodu je pripojené k dátovej zbernici procesora. K obvodu je pripojená aj voliteľná flash pamäť EEPROM z dôvodu uchovávania určitých špecifických informácií, s ktorými je možné pracovať bez účasti procesora a jeho pamätí. Riadiace signály RD#, WR, TXF# sú vyvedené na voľné piny procesora 25-27. Riadiaci signál RXF# je vyvedený na pin externého prerušenia INT2. Toto zapojenie umožňuje jednoduchšiu implementáciu príjmu dát z obvodu.

5.2.5 Zapojenie sériového rozhrania

Sériové rozhranie je realizované pomocou obvodu MAX232 ako bolo definované v návrhu projektu. Obvod je zapojený podľa technickej dokumentácie výrobcu a pripojený na USART rozhranie procesora.

5.2.6 Oživovanie mikropočítača ATmega32

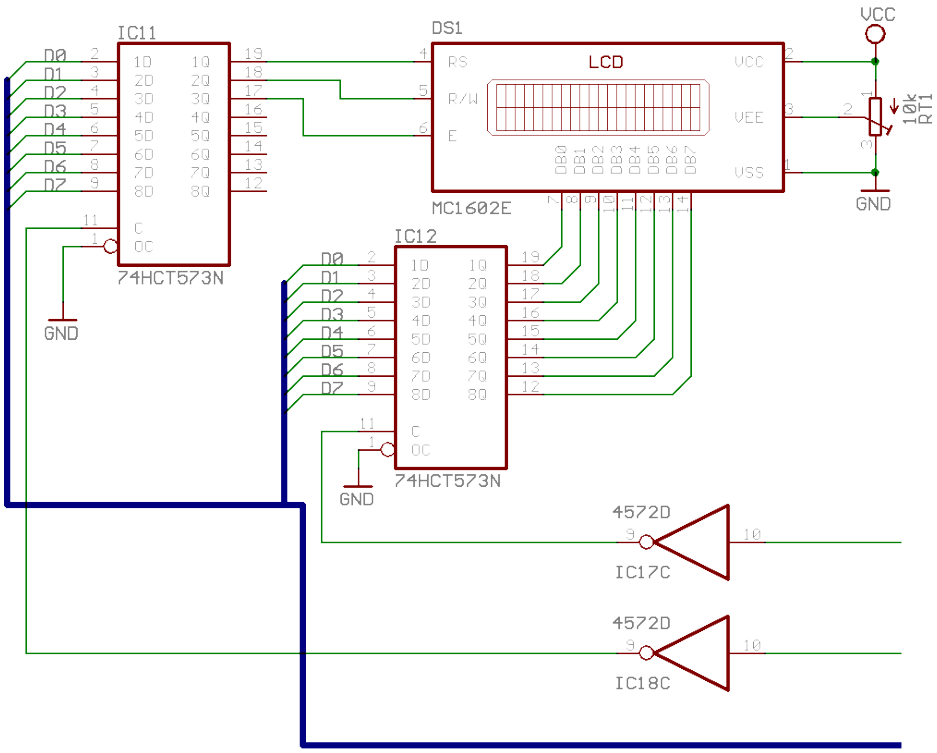
Doska plošného spoja bola osádzaná postupne počnúc 7-segemntovými displejmi a USB portom. V určitom štádiu oživovania sa na doske objavil skrat. USB sa ešte podarilo bez problémov otestovať, neskôr však neznámy skrat znemožnil akékoľvek ďalšie testovanie. Keďže naša doska plošného spoja má špeciálny nevodivý povlak, nemohol byť skrat spôsobený kovovými pilinami. Ďalšou možnosťou skratu bolo neúmyselné spojenie nejakých dvoch nožičiek niektorej zo súčiastok. Optickou kontrolou bola táto možnosť vylúčená. Následne sme vyskúšali, či chyba nemôže byť na strane tlačidiel externých prerušení. V prípade nesprávneho zapojenia by zem s napätím mohli byť prepojené práve tu. Po prerezaní ciest na oboch tlačidlách sa ukázalo, že zdroj skratu bol skutočne tu. Chyba nebola systematického charakteru v návrhu, ale v zle definovanej schematickej značke v rámci programu Eagle, ktorý bol použitý na realizáciu schémy.

Ďalším problémom bolo zle navrhnuté zapojenie sériového portu. Rovnaká chyba sa vyskytla aj v prípade mikropočítača ARM, riešenie je teda popísané v sekcii 5.1.1.

5.2.7 Zmeny v návrhu mikropočítača ATmega32

V schéme zapojenia mikropočítača rady AVR bolo potrebné vykonať niekoľko zmien. Skryté napájanie niektorých súčiastok nebolo externe vyvedené, drobné zmeny boli vykonané aj v rámci zapojenia USB rozhrania a sériového rozhrania. Najvýraznejšia zmena v návrhu bola vykonaná v realizácii LCD displeja. Pôvodný výňatok schémy je na obrázok 5.5.

Usúdili sme, že záchytný register IC11 je zbytočný. Slúžil na realizáciu riadenia vstupov displeja pomocou dátovej zbernice. Keďže však zostávalo viacero pinov procesora nevyužitých, rozhodli sme sa na riadenie použiť tieto piny. Tým sa znížili nároky na súčiastky aj na dátovú zbernicu a uvolnil sa jeden výstup dekódera. Reálna výsledná implementácia je zdokumentovaná v rámci kapitoly Implementácia HW časti.



Obr. 5.5: Pôvodná schéma LCD

Zmeny sa udiali aj v niektorých použitých súčiastkach, pretože nie všetky navrhované bolo možné kúpiť. Vo všetkých prípadoch boli pôvodné nahradené použiteľnými ekvivalentmi a sú uvedené v zozname súčiastok v rámci tohto dokumentu.

5.3 Implementácia programu pre PC

V tejto kapitole zdôvodníme výber programovacieho jazyka. Bližšie opíšeme objektový model programu, niektoré triedy a ich dôležité metódy a popíšeme grafické používateľské prostredie.

5.3.1 Výber programovacieho jazyka

Voči návrhu programu pre PC nastala zmena vo výbere programovacieho jazyka. Pôvodne sa uvažovalo, že program sa bude písať v jazyku C/C++ a grafické prostredie bude využívať sadu nástrojov vývojového prostredia Qt, najmä kvôli "multiplatformovosti". Programátor sa však rozhodol pre napísanie programu v jazyku C#.net z 2 dôvodov. Grafické prostredie sa C#.net vytvára veľmi jednoducho, rýchlo a intuitívne. Programátor sa preto mohol viacej zamerať na funkcionalitu programu. Druhým dôvodom je možnosť spustenia programu aj pod OS Linux. To je možné vďaka projektu Mono. Pomocou Mono je možné skompilovať kód napísaný v C# aj pod OS Linux. Nevýhodou programu napísaného v C#.net je jeho rýchlosť. Program využíva .net Framework, teda kód je manažovaný, na rozdiel od aplikácií vytvorených v Qt, ktoré sú na danej platforme preložené do natívneho kódu a preto sú rýchlejšie. Táto nevýhoda by však nemala ovplyvniť korektnosť správania programu.

5.3.2 Objektový model programu

V tejto časti opíšem niektoré triedy programu a ich metódy. Zameral som sa na opis len dôležitých tried a metód z hľadiska funkcionality, neopisoval som väčšinu tried vytvárajúcich grafické prostredie. Diagram tried možno vidieť na obrázku xy. Jednotlivé triedy boli navrhnuté a implementované tak, aby mohli byť použité aj v iných projektoch, príp. pri zmene špecifikácie ľahko modifikovateľné. Funkcionalita grafického prostredia je preto oddelená od tried vytvárajúcich funkcionalitu programu.

MainForm - Hlavná trieda programu. Pri inicializácii si vytvorí inštancie tried Comunicator, Protocol a Terminal, pomocou ktorých sa ďalej zabezpečuje komunikácia s mikroprocesormi cez sériový port. MainForm tiež zabezpečuje funkcionalitu hlavného okna grafického prostredia.

EditorForm - Každému otvorenému súboru prislúcha jedna inštancia triedy. EditorForm je jednoduchý editor textu. Slúži na editovanie programov napísaných pre mikroprocesory. Jeho najdôležitejšou funkcionalitou je možnosť vkladania breakpointov a kompilovanie programu.

Do načítaného programu môže používateľ vkladať breakpointy na dané riadky a takisto ich aj vymazávať pomocou tlačidiel na hornej lište. Pri stlačení vloženia breakpointu sa

na dané miesto v programe vloží riadok s inštrukciou softvérového prerušenia pre daný procesor.

Na samotnú kompiláciu programu s inštrukciami breakpointov sa zavolá externý kompilátor gcc s príslušnými parametrami.

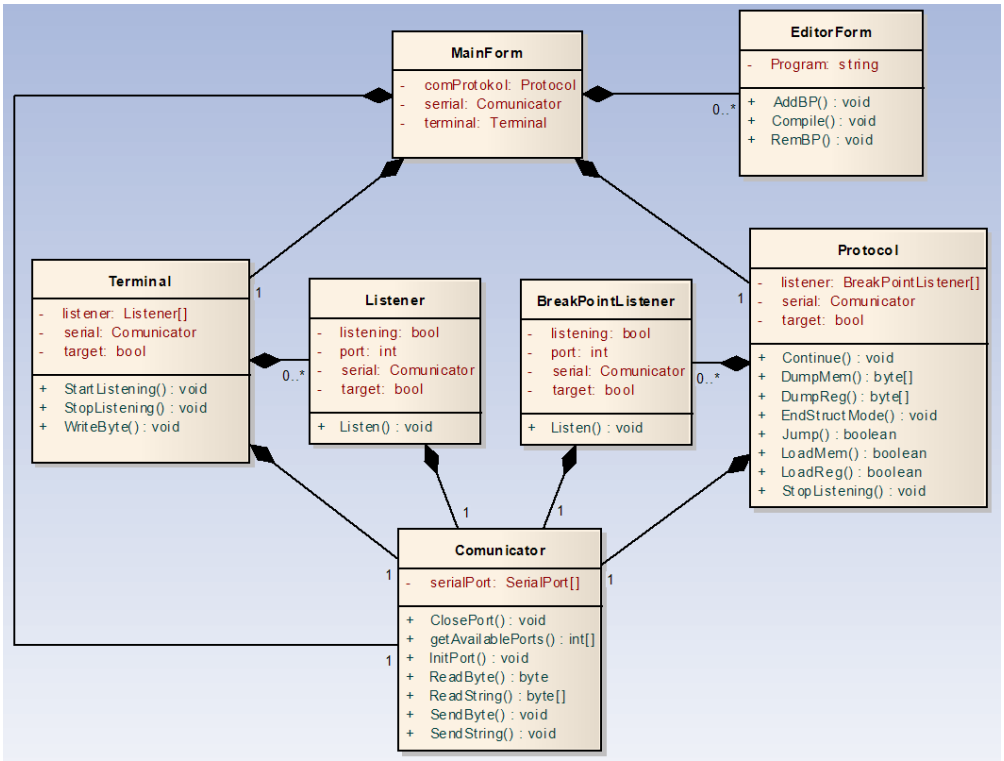
Terminal - Slúži na znakovo orientovanú komunikáciu s mikroprocesorom. Takýto typ komunikácie sa spustí pri zadaní požiadavky EndStructMode z GUI. Pri zadaní tejto požiadavky si vytvorí inštanciu triedy Listener a vo vlákne spustí počúvanie na príslušnom porte pre každú novú komunikáciu. Terminal prijíma požiadavky zaslania znakov, ktoré posiela ďalej Comunicatoru.

Protocol - Slúži na komunikáciu s mikroprocesormi pomocou protokolu definovaného v kapitole xy. Táto komunikácia je formou request-response, spustí sa pri zadaní požiadavky StartStructMode z GUI. Počítač posiela požiadavky, na ktoré mikroprocesor odpovedá. Požiadavky sú vyvolávané cez GUI, z GUI sa zavolá príslušná metóda Protocolu, Protocol k nim vytvorí prislúchajúcu správu definovanú protokolom, ktorú ďalej posiela Comunicatoru. Na správy očakáva príslušné odpovede, príp. chybové odpovede. Protocol má vytvorenú inštanciu triedy BreakPointListener pre všetky prebiehajúce komunikácie. Po poslaní požiadavky na spustenie programu, začne prislúchajúci BreakPointListener vo vlákne počúvať na príslušnom porte či neprišla správa BREAKPOINT, ktorú oznámi používateľovi.

Listener - Listener je vytváraný Terminalom. Je to trieda, v ktorej sa volá vo vlákne metóda Listen(). V tejto metóde sa neprestajne počúva na príslušnom sériovom porte a prijaté byty sa posielajú do vyšších vrstiev.

BreakPointListener - je vytváraný Protocolom. Je to trieda, v ktorej sa volá vo vlákne metóda Listen() ak bola zaslaná požiadavka na spustenie vykonávan ia programu. V tejto metóde sa neprestajne počúva na príslušnom sériovom porte a čaká sa na správu BREAKPOINT, ktorá sa oznámi užívateľovi.

Comunicator - Môže zabezpečovať komunikáciu cez 4 sériové porty naraz, preto každá metóda Comunicatoru sa volá na konkrétny sériový port. Comunicator je v podstate nadstavbou nad triedou System.IO.Ports.SerialPort, ktorú poskytuje C#.. Spravuje chyby vzniknuté pri sériovej komunikácii, posiela príslušné exception-y do vyšších vrstiev, umožňuje posielať dlhšie reťazce ako je veľkosť buffera triedy SerialPort. Keďže pri čítaní sa vždy čaká na konkrétny počet bytov, metódy ReadByte a ReadString vracajú hodnotu až po načítaní požadovaného počtu bytov.



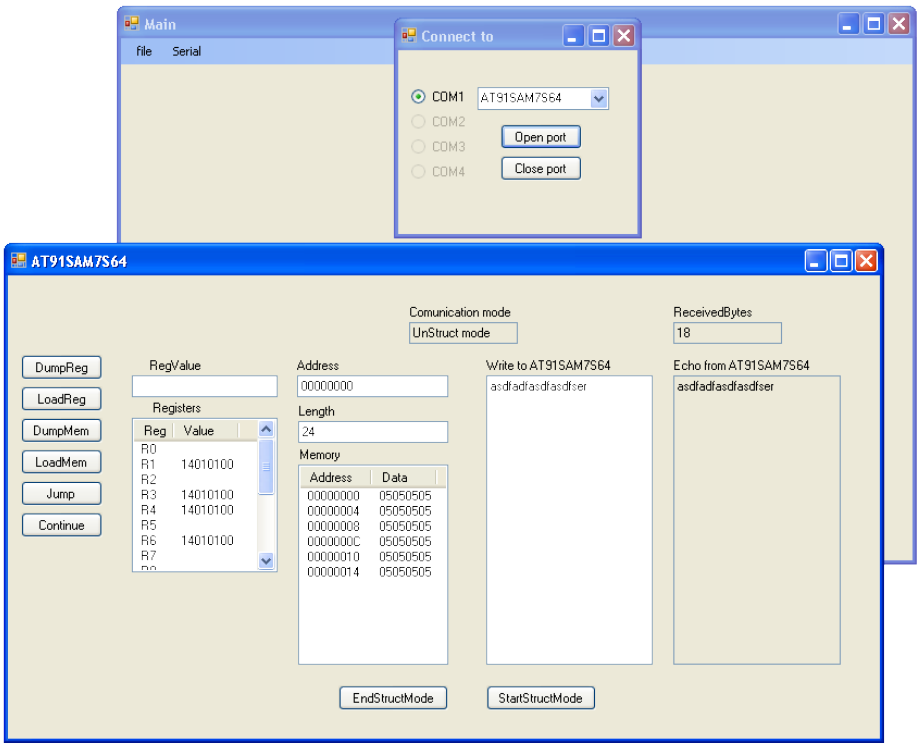
Obr. 5.6: Class diagram

5.3.3 Testovanie programu

V triedach Terminal, Protocol a Listener sa vytvorili “callbackove” volania. Na miesto volania metód Comunicatora sa napevno nastaví hodnota, ktorú má daná metóda vrátiť. Takto je možné jednoducho otestovať funkčnosť jednotlivých tried(okrem triedy Comunicator) a GUI . Program beží v testovacom móde ak atribúty “target” jednotlivých tried nie sú nastavené, ak sú nastavené program využíva metódy Comunicatora a snaží sa komunikovať s mikroprocesormi.

5.3.4 Grafické používateľské prostredie

Grafické prostredie je jednoduché a intuitívne. Na komunikáciu s mikroprocesorom slúži okno prislúchajúce danému mikroprocesoru. Naraz môže byť otvorených viacero komunikačných okien, teda môže sa naraz komunikovať s viacerými mikroprocesormi. Ovládanie je urobené pomocou tlačidiel, predstavujúcich jednotlivé príkazy poslania správ mikroprocesoru. Program upozorní na nesprávne zadané alebo nezadané parametre chybovými hláškami. Okno je rozdelené na 2 časti - komunikáciu cez Protocol a komunikáciu cez Terminal. Prepínanie medzi týmito spôsobmi komunikácie sa vykonáva pomocou tlačidiel EndStructMode a StartStructMode.



Obr. 5.7: Grafické používateľské prostredie

Kapitola 6

Zhodnotenie

Pri riešení tohto projektu sme sa znažili o zodpovedný a cieľavedomý prístup s jasne definovanými čiastkovými úlohami pre jednotlivých členov tímu. V zimnom semestri sme sa snažili stihnúť čo najlepšie pochopiť problematiku návrhu hardvérovej časti pre jednotlivé mikropočítače a tiež problematiku ich programovania. Výsledkom bol primerane rozsiahly dokument, ktorý mal nedostatky, ale riešené problémy opisoval v miere, ktorú hodnotíme ako nadpriemernú. Po odovzdaní tohto dokumentu na konci zimného semestra zostávalo napísať posudok pre dokument od konkurenčného tímu. Tento posudok bol spravený dôkladne, v úprimnej snahe pomôcť druhému tímu poukázaním na možné problémy, a tiež oceniť to, čo bolo na projektovej dokumentácii kvalitné.

Volný čas, ktorý vznikol po úspešnom absolvovaní písomných skúšok v zimnom semestri, sme sa snažili využiť na doladenie návrhu harvérovej časti mikropočítačov a návrh dosiek s plošnými spojmi. Vďaka tomu sa návrh jednej dosky podarilo dokončiť v prvom týždni letného semestra a doska mohla ísť rýchlo do výroby. Výroba druhej dosky bola poznačená tým, že v čase ukončenia jej návrhu mal Ustav Informatiky SAV, kde bola vyrobená prvá doska, nedostatok zamestnancov, čo by výrobu spomalilo. Preto bolo nutné vyhľadať alternatívneho výrobcu. Malá súkromná firma z Pieštan nám nakoniec hotový plošný spoj po niekoľkých telefonátoch zaslala a to už bol druhý mikropočítač skoro oživený.

Vývoj SW a FW časti prebiehal pomerne hladko a rýchlo, aj napriek množstvu iných závazkov voči ostatným predmetom. Veľkou prekážkou bola porucha programátora obvodov, ktorý šiel behom dvoch týždňov dva krát do opravy na východ Slovenska.

Na projekte sa podielali:

Bc. Martin Viceník - vedenie tímu, implementácia HW časti, denník stretnutí, nákupy a objednávky

Bc. Tomáš Krajčo - programovanie mikropočítačov, naštudovanie rozsiahlej problematiky programovania rodiny mikroprocesorov AT91SAM7

Bc. Michal Tölgyessy - implementácia HW časti, programovanie FW

Bc. Marek Sobolič - implementácia HW časti, programovanie FW, správa dokumentácie a webu

Bc. Róbert Komarómy - vývoj aplikácie na programovanie mikropočítačov pomocou PC

Literatúra

[1] *AT91SAM7S64 DEVELOPMENT PROTOTYPE BOARD*
`http://www.olimex.com/dev/sam7-p64.html`
Posledný prístup na stránku: 9.11.2007

[2] *AT91SAM9260 DEVELOPMENT PROTOTYPE BOARD*
`http://www.olimex.com/dev/sam7-p64.html`
Posledný prístup na stránku: 9.11.2007

[3] *S. ANGELOVIČ a kol.:* **Návrh a realizácia sady mikropočítačov**
Bratislava, FIIT STU, 2007, Tímový projekt

[4] *M. ONDROVIČ a kol.:* **Návrh a realizácia sady mikropočítačov**
Bratislava, FIIT STU, 2007, Tímový projekt

[5] *A., ŠTRBA:* **Experimentálny mikropočítač**
Bratislava, FIIT STU, 2005, Diplomová práca

[6] *AT91SAM7S Series Preliminary*
`http://atmel.com/dyn/resources/prod_documents/doc6175.pdf`
Posledný prístup na stránku: 11.11.2007

[7] *AT91SAM7 32 bit based ARM Based Microcontrollers Overview*
`http://atmel.com/products/AT91/overview.asp`
Posledný prístup na stránku: 11.11.2007

[8] *ARM7TDMI Technical Reference Manual*
`http://atmel.com/dyn/resources/prod_documents/DDI0029G_7TDMI_R3_trm.pdf`
Posledný prístup na stránku: 11.11.2007

[9] *Using the I2C bus*
`http://www.robot-electronics.co.uk/htm/using_the_i2c_bus.htm`
Posledný prístup na stránku: 14.11.2007

[10] *RS232*
http://rs232.hw.cz/
Posledný prístup na stránku: 14.11.2007.

[11] *B., Dado:* **Universal Serial Bus**
Bratislava, FIIT STU, 2005, Materiál ku predmetu Periférne zariadenia

[12] *Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, Philips:* **Universal Serial Bus Specification**
April 27, 2000 Revision 2.0

[13] *DA56-11GWA*
http://www.ges.cz/sheets/d/dx56.pdf
Posledný prístup na stránku: 16.12.2007.

[14] *74HC573N*
http://www.datasheetcatalog.com/datasheets_pdf/7/4/H/C/74HC573N.shtml
Posledný prístup na stránku: 16.12.2007.

[15] *AT91SAM7S Microcontroller Series Schematic Check List*
http://www.datasheetcatalog.com/datasheets_pdf/7/4/H/C/74HC573N.shtml
Posledný prístup na stránku: 9.12.2007.

[16] *Multi-ICE Version 2.2 User Guide*
http://infocenter.arm.com/help/topic/com.arm.doc.dui0048f/DUI0048.pdf
Posledný prístup na stránku: 1.12.2006.

[17] *MAX3222, MAX3232, MAX3237, MAX3241*
http://datasheets.maxim-ic.com/en/ds/MAX3222-MAX3241.pdf
Posledný prístup na stránku: 16.12.2006.

[18] *BC1602A*
http://www.fush.lv/Bolymin/Data/BC1602A.pdf
Posledný prístup na stránku: 16.12.2006.

[19] *Znakové LCD displeje*
http://www.cmail.cz/doveda/lcd/
Posledný prístup na stránku: 16.12.2006.

[20] *DOG SERIES 3.3V*
http://www.sos.sk/a_info/resource/d/ea/dog.pdf
Posledný prístup na stránku: 2.5.2008.

[21] *SIPMOC Small-Signal Transistor (P channel MOSFET)*
`http://www.datasheetcatalog.org/datasheet/siemens/Q62702-S633.pdf`
Posledný prístup na stránku: 12.5.2008.

[22] *HEXFET Power MOSFET*
`http://www.irf.com/product-info/datasheets/data/irf9520pbf.pdf`
Posledný prístup na stránku: 12.5.2008.

Dodatok A

Používateľská príručka pre firmware a obslužný program pre PC

Firmware je tá časť projektu, ktorá sa nachádza vo forme inštrukcií vo vytvorenom mikropočítači. Jeho úlohou je test mikropočítača a komunikácia s hostiteľským PC. Komunikácia sa odohráva po sériovej linke. Na komunikáciu s hostiteľským počítačom je určený USART1. Parametre sériového spojenia sú:

- rýchlosť spojenia 9600 baudov
- žiadna parita
- žiadny flow-control
- 1 stop bit

Firmware poskytuje funkcie na prácu s displejom a to zapnutie a inicializovanie displeja, výpis znakov na displej, nastavovanie kurzora na požadovanú pozíciu.

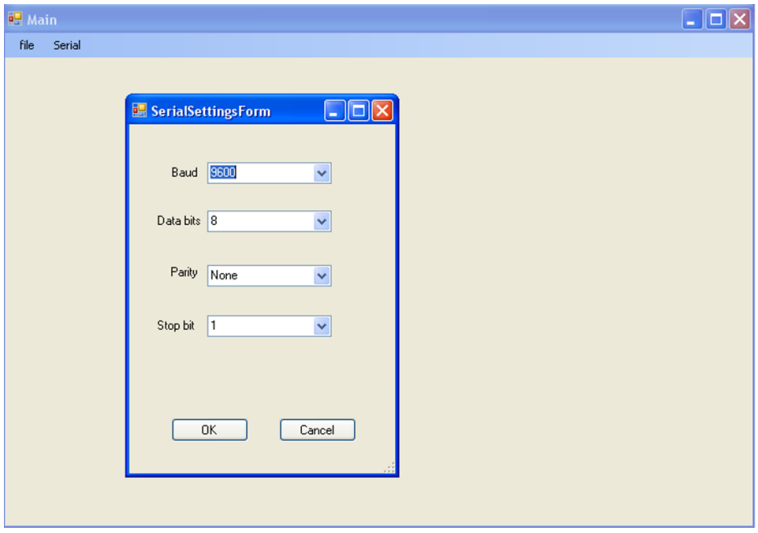
Firmware rozlišuje dva komunikačné módy: neštruktúrovaný mód, v ktorom iba echuje prijaté znaky po sériovej linke, a štruktúrovaný mód, v ktorom sa medzi mikropočítačom a hostiteľským PC posielajú špeciálne formátované správy s konkrétnym významom. Do štruktúrovaného módu a von zo štruktúrovaného módu sa prechádza pomocou programu v PC, ktorý má na prechod medzi módmi implementované dve tlačidlá. Firmware okrem toho, že inicializuje mikropočítač a otestuje ho, tak na podporu vyvíjania programov implementuje nasledovné funkcie:

- nahrávanie obsahu pamäte
- výpis obsahu pamäte
- nahrávanie obsahu registrov

- výpis obsahu registrov
- spustenie programu
- možnosť pokračovať v prerušenom programe po breakpointe

Firmware ďalej implementuje komunikačný protokol s hostiteľským PC. Jednotlivé implementované funkcie firmwaru sa ovládajú cez používateľský program v PC.

A.1 Nastavenie parametrov sériového portu



Obr. A.1: Nastavenie sériového portu

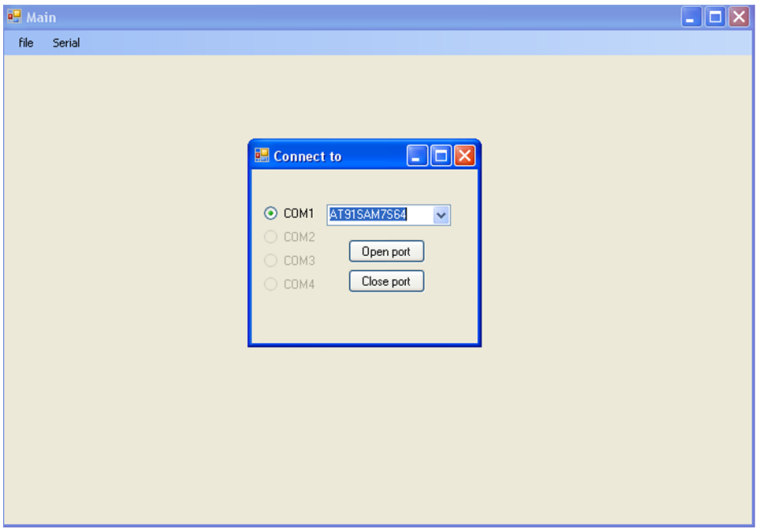
Na obrázku vidno formulár s výberom nastavenia sériového portu. Umožňuje nastaviť rýchlosť komunikácie, počet dátových bitov, paritu a počet stop bitov.

A.2 Výber sériového portu a mikropočítača

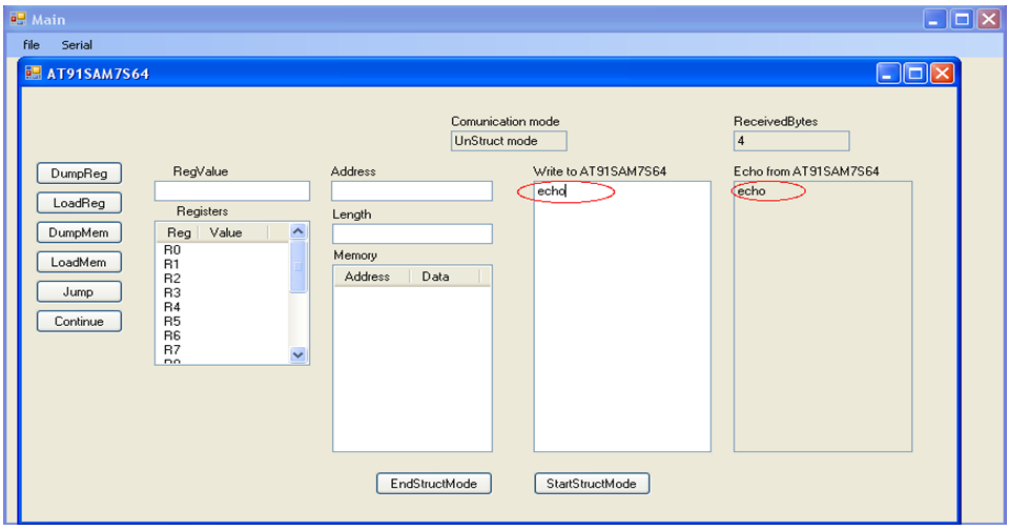
V tomto formulári sa vyberá port na komunikáciu medzi hostiteľským počítačom a mikropočítačom.

A.3 Echo

V neštruktúrovanom móde mikropočítač echuje prijaté znaky späť po sériovej linke. Vidno to na obrázku.

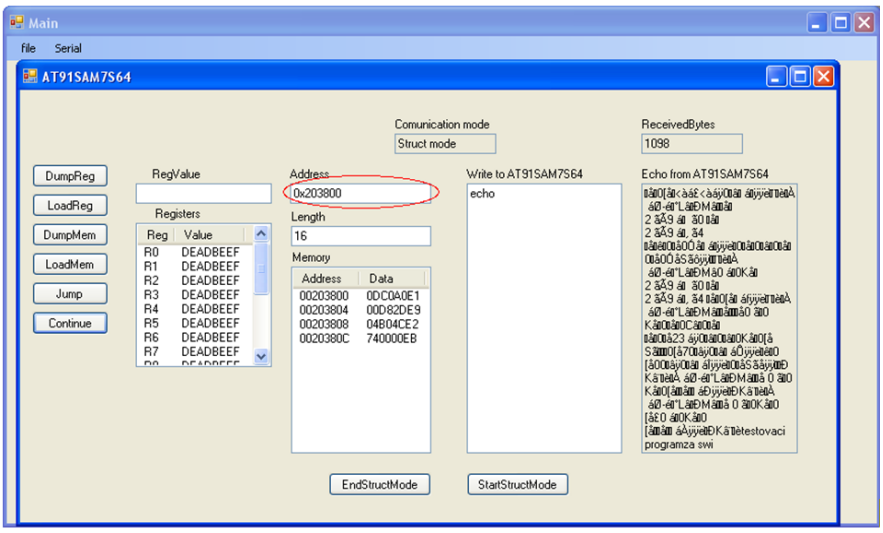


Obr. A.2: Výber sériového portu



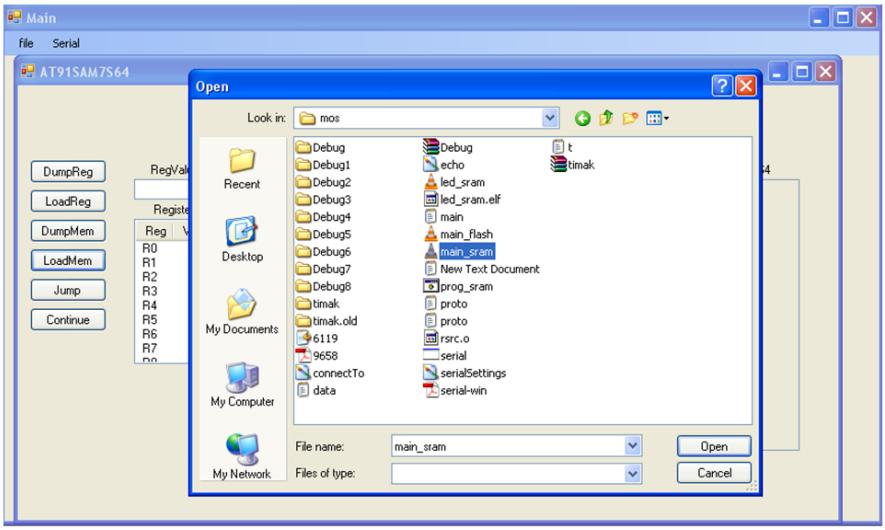
Obr. A.3: Neštruktúrovaný mód komunikácie

A.4 Nahrávanie súboru do mikropočítača



Obr. A.4: Nahrávanie súboru do mikropočítača

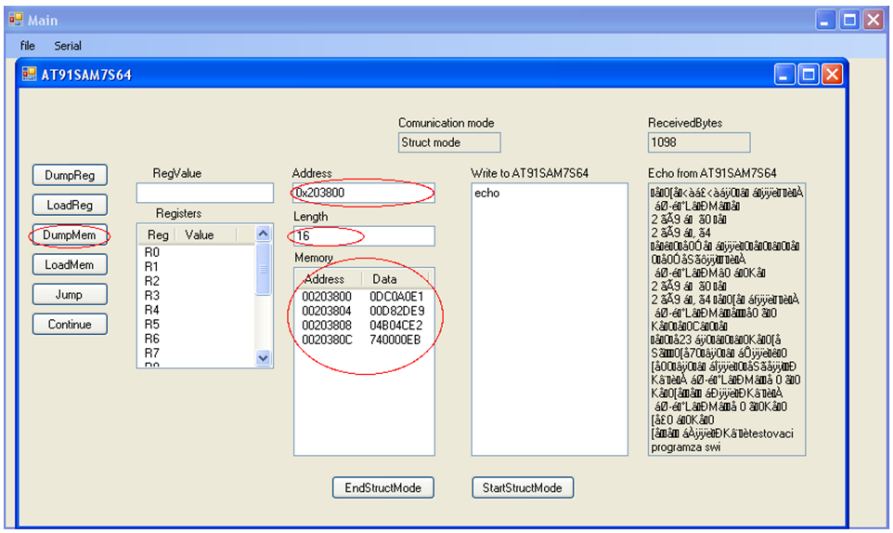
Pred samotným nahrávaním súboru je potrebné zvoliť adresu ,na ktorú sa bude nahrávať. Potom stačí kliknúť na tlačítko "LoadMem". V oblasti "echo from" AT91SAM7S64 je "odechovaný" vykonateľný program. Toto sa môže stať vtedy keď sa najskôr neprepneme do štruktúrovaného módu.



Obr. A.5: Formulár na výber programu

Ak klikneme na tlačítko "LoadMem", (a sme v štruktúrovanom móde) tak sa objaví formulár na výber programu na nahratie do mikropočítača. Stačí odkliknúť tlačítko "Open" a program sa nahrá na zvolenú adresu ak je adresa správna.

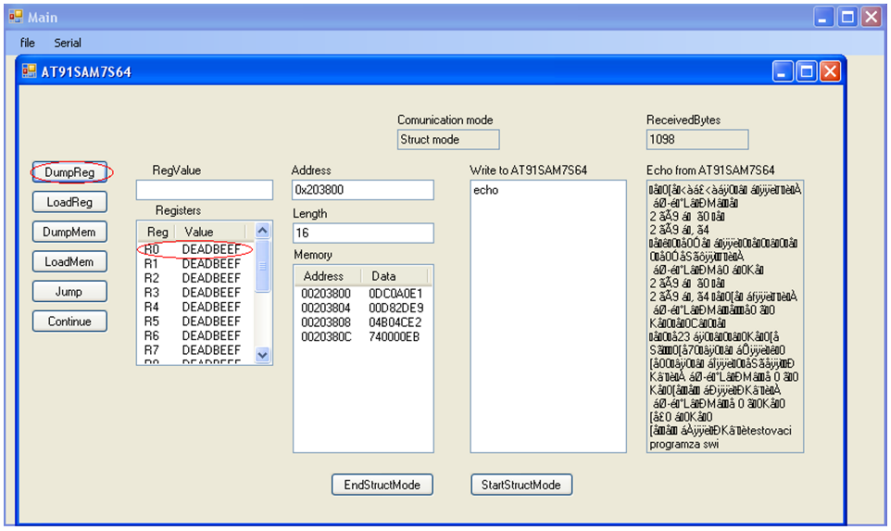
A.5 Vypísanie obsahu pamäte



Obr. A.6: Vypísanie obsahu pamäte

Na výpis obsahu pamäte treba zadať adresu, dĺžku vypisovanej oblasti a kliknúť na tlačítko "DumpMem". V poli "Memory" sa zobrazí obsah pamäti.

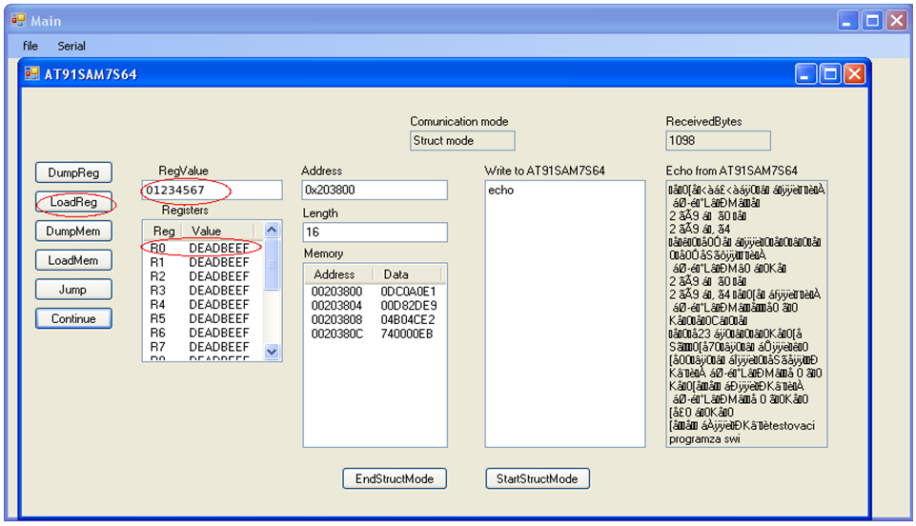
A.6 Vypísanie obsahu registrov



Obr. A.7: Výpis obsahu registrov

Na vypísanie obsahu registra treba zakliknúť register, ktorý chceme vypísať a stlačiť tlačítko "DumpReg".

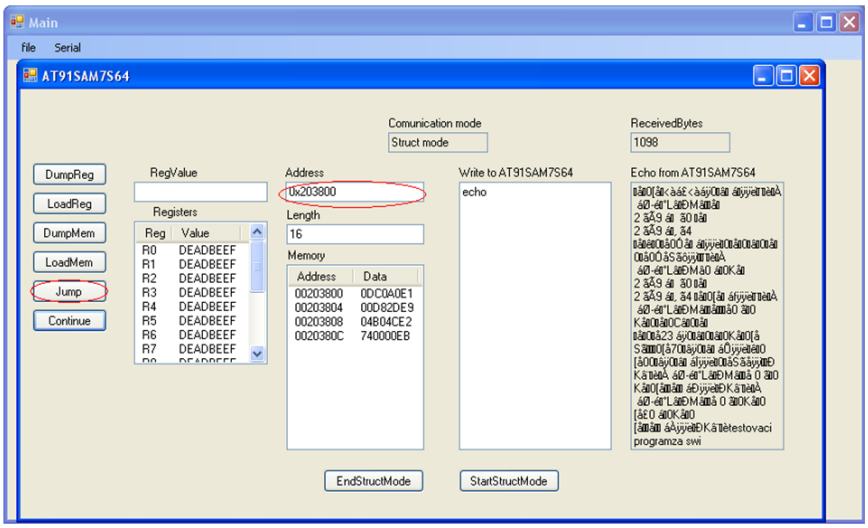
A.7 Nahrávanie obsahu registrov



Obr. A.8: Nahrávanie obsahu registrov

Na nahratie registra je potrebné označiť daný register, vyplniť pole “RegValue” a kliknúť na tlačítko “LoadReg”.

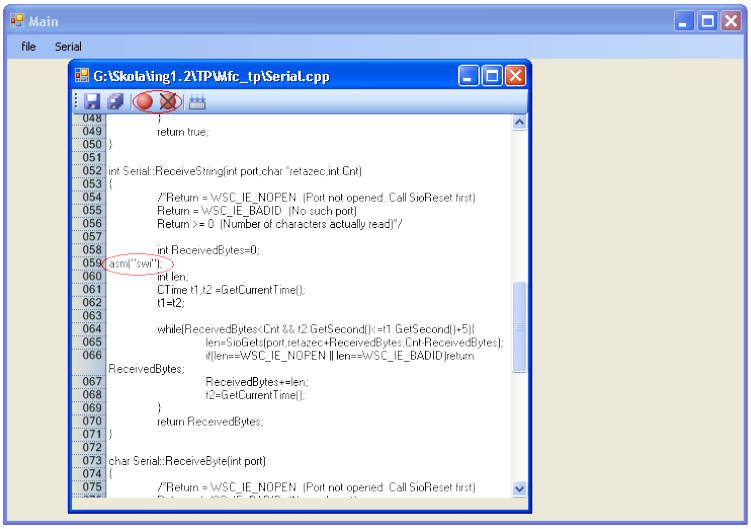
A.8 JUMP- začiatok vykonávania používateľského programu



Obr. A.9: Spustenie programu

Na začatie vykonávania programu, treba vložiť začiatočnú adresu programu a kliknúť na tlačítko “Jump” .

A.9 Vloženie breakpointu



Obr. A.10: Pridanie a odobranie breakpointov

Na vloženie breakpointu treba kliknúť na prvé vyznačené tlačítko na obrázku A.10 . Breakpoint sa vloží pred riadok s kurzorom.

A.10 Vymazanie breakpointu

Na vymazanie breakpointu treba kliknúť na druhé vyznačené tlačítko na obrázku A.10 . Breakpoint sa vymaže ak sa nachádza v riadku s kurzorom.

A.11 Continue - pokračovanie v prerušenom programe

Na predošlých obrázkoch vidno tlačítko "Continue". Slúži na pokračovanie v behu programu, ktorý bol prerušený breakpointom.

A.12 Dodatok pre programovanie mikropočítača AVR

Viaceré komponenty ATmega32 sú realizované manuálne, teda využitím pinov vstupno/ výstupných portov procesora na účel požadovaný týmto projektom. Konkrétne zapojenia sú zdokumentované nižšie.

A.13 Dátová zbernica

Keďže procesor ATmega32 nepodporuje vyvedenie externej adresnej zbernice, bolo potrebné zrealizovať implementáciu zbernice manuálne. Na tento účel bol využitý vstupno-výstupný port A procesora. Zbernica sa využíva na posielanie dát radiču USB rozhrania, do záchytných registrov pre 7-segmentové displeje a LCD displej.

A.14 Dekóder

Pripojenie zariadení k zbernici je realizované prostredníctvom dekódera 74LS137N. Jeho vstupy A, B, C sú vyvedené na tri inak nevyužité piny procesora 22, 23 24, rovnako ako aktivačný vstup (pin 28). Výstupy y0 až y5 aktivujú záchytné registre pre 3 dvojciferné displeje DA56-11GNA. Výstup y6 aktivuje záchytný register LCD displeja. Pin y7 je nevyužitý.

A.15 USB

Riadiace signály RD#, WR, TXF# sú vyvedené na voľné piny procesora 25-27. Riadiaci signál RXF# je vyvedený na pin externého prerušenia INT2.

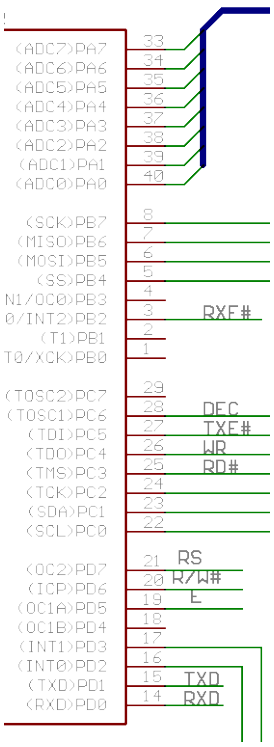
A.16 LCD

Riadiace vstupy displeja E, R/W# a RS sú vyvedené priamo na piny procesora 19-21.

A.17 Prehľad vybraných pinov ATmega32 a ich alternatívnych funkcií

Zariadenie	Číslo pinu	Funkcia
Zbernica	40	D0
	39	D1
	38	D2
	37	D3
	36	D4
	35	D5
	34	D6
	33	D7
Dekóder	22	A
	23	B
	24	C
	28	OE DEC
USB	25	RD#
	26	WR
	27	TXE#
	3	RFX#
LCD	19	E
	20	R/W#
	21	RS

Tabuľka A.1: Prehľad vybraných pinov ATmega32 a ich alternatívnych funkcií



Obr. A.11: Prehľad vybraných pinov

Dodatok B

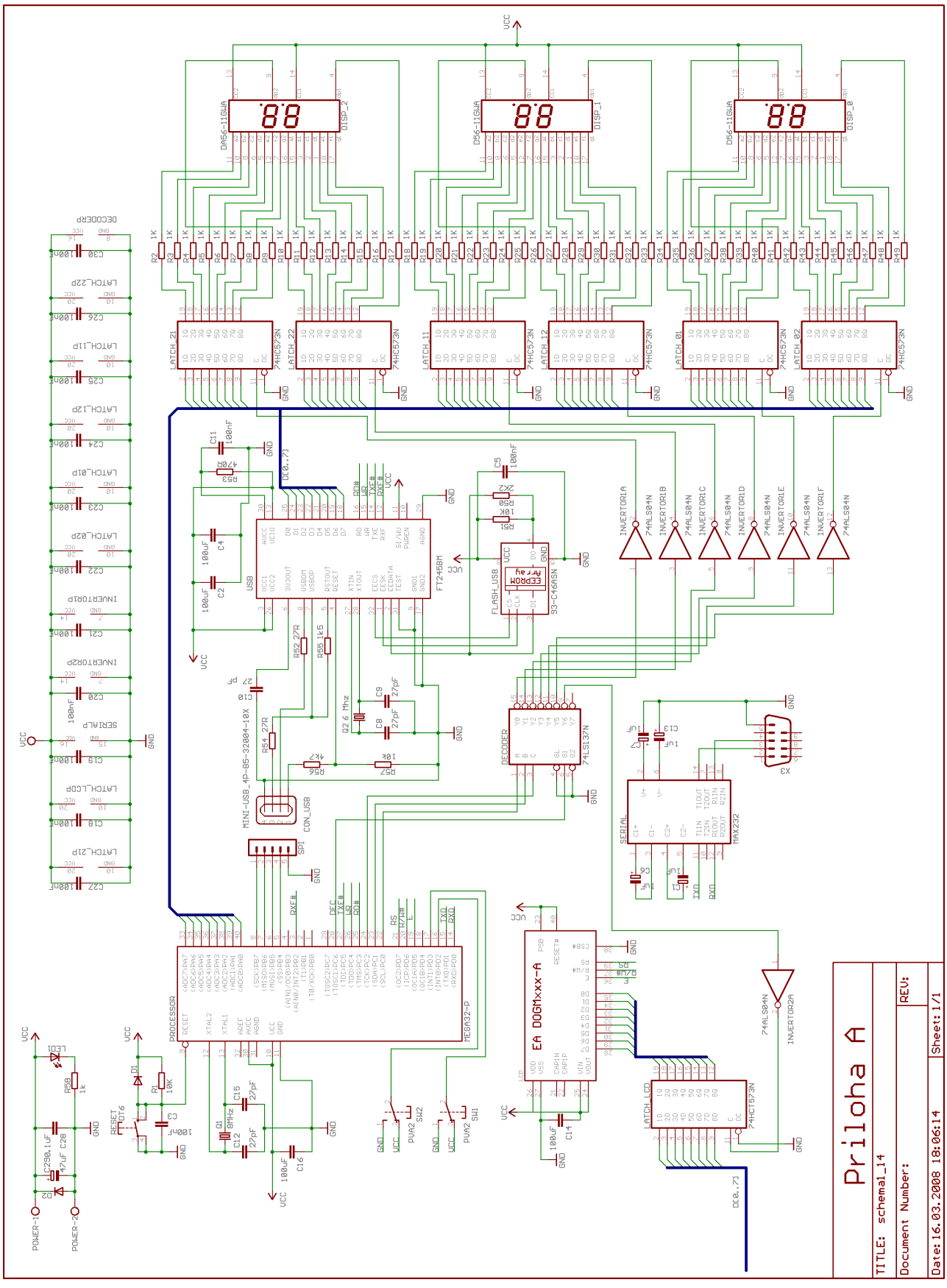
Schémy a dosky plošných spojov

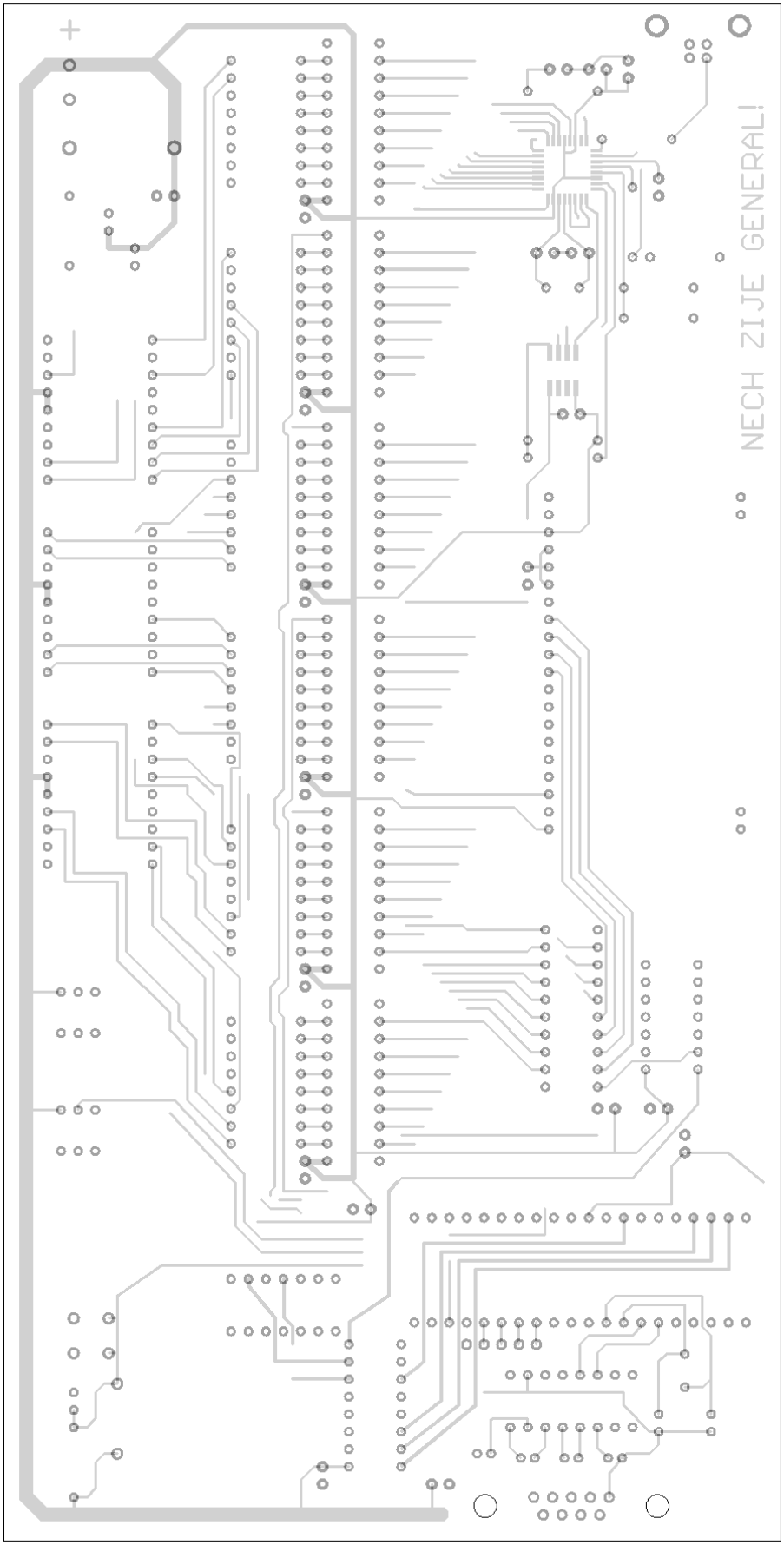
B.1 AVR

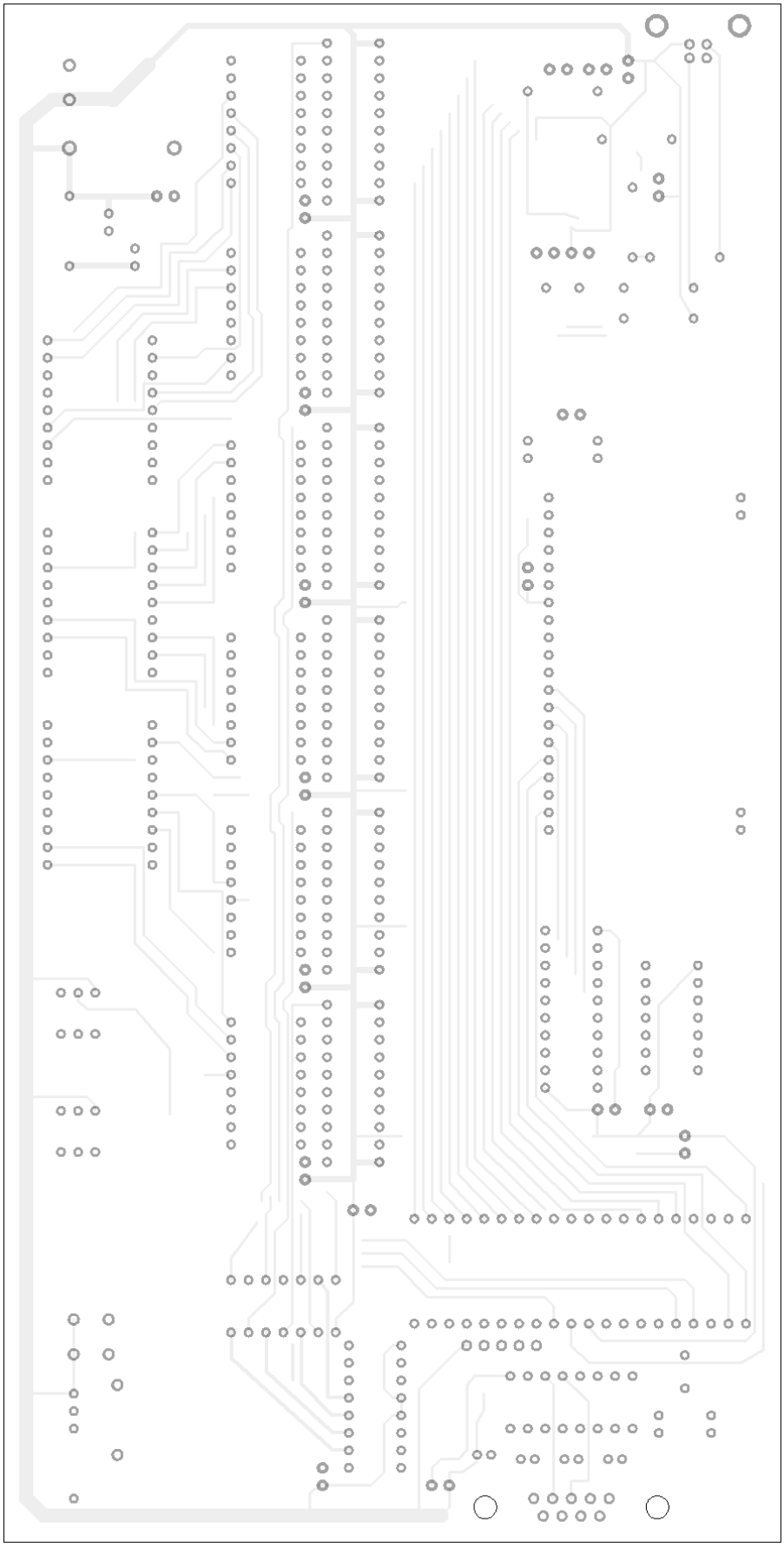
- Schéma
- Vrchná časť dosky plošných spojov
- Spodná časť dosky plošných spojov
- Rozloženie súčiastok

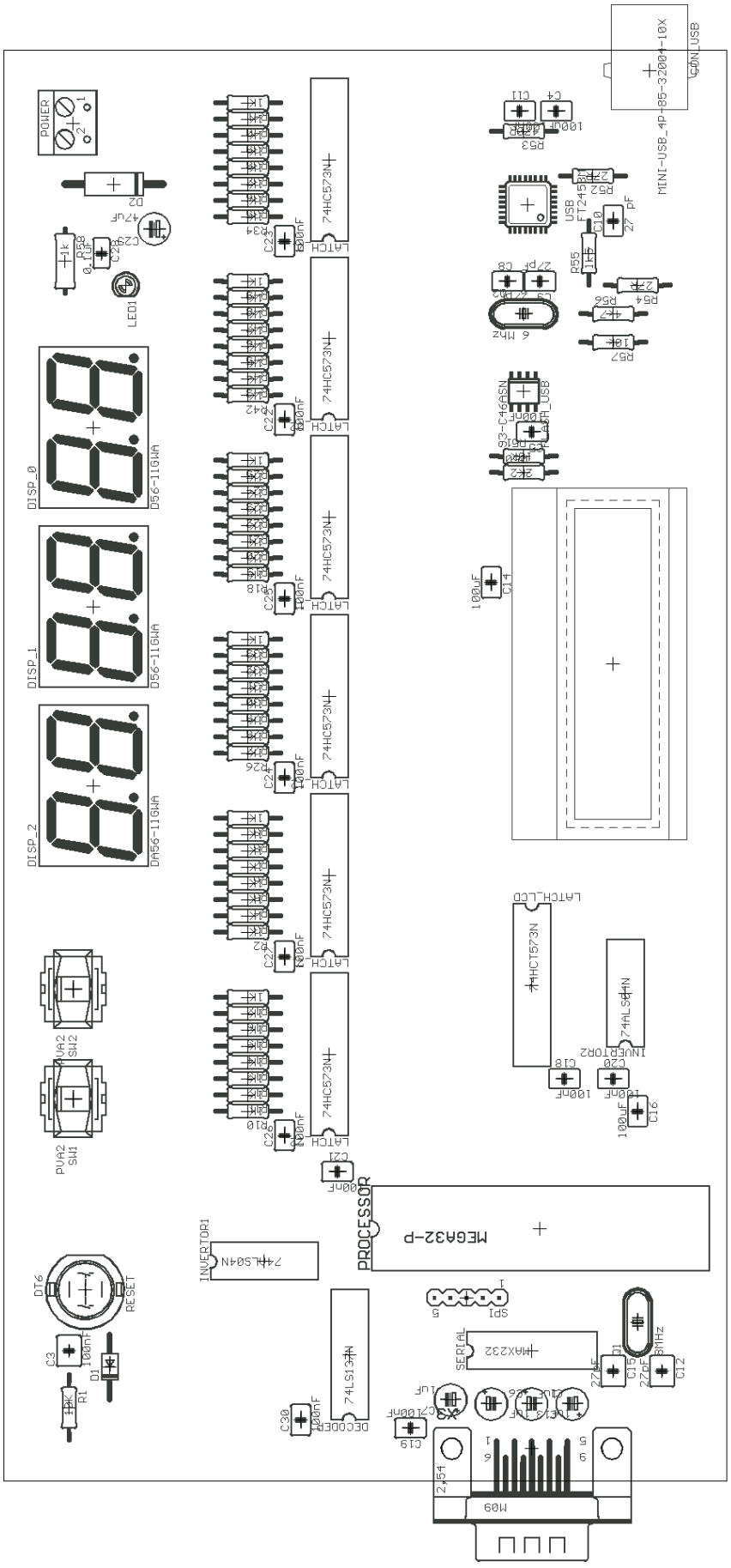
B.2 ARM

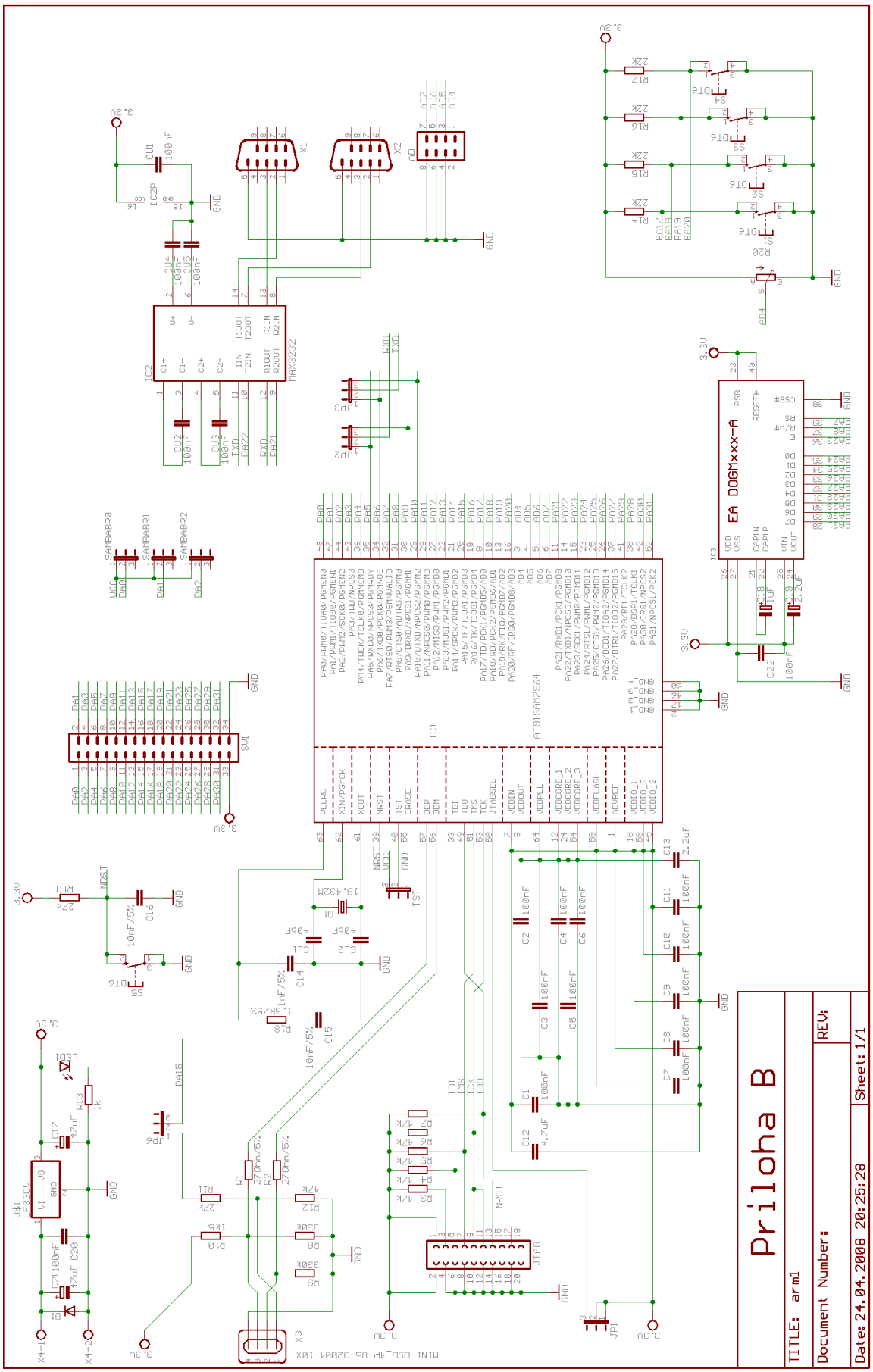
- Schéma
- Vrchná časť dosky plošných spojov
- Spodná časť dosky plošných spojov
- Rozloženie súčiastok

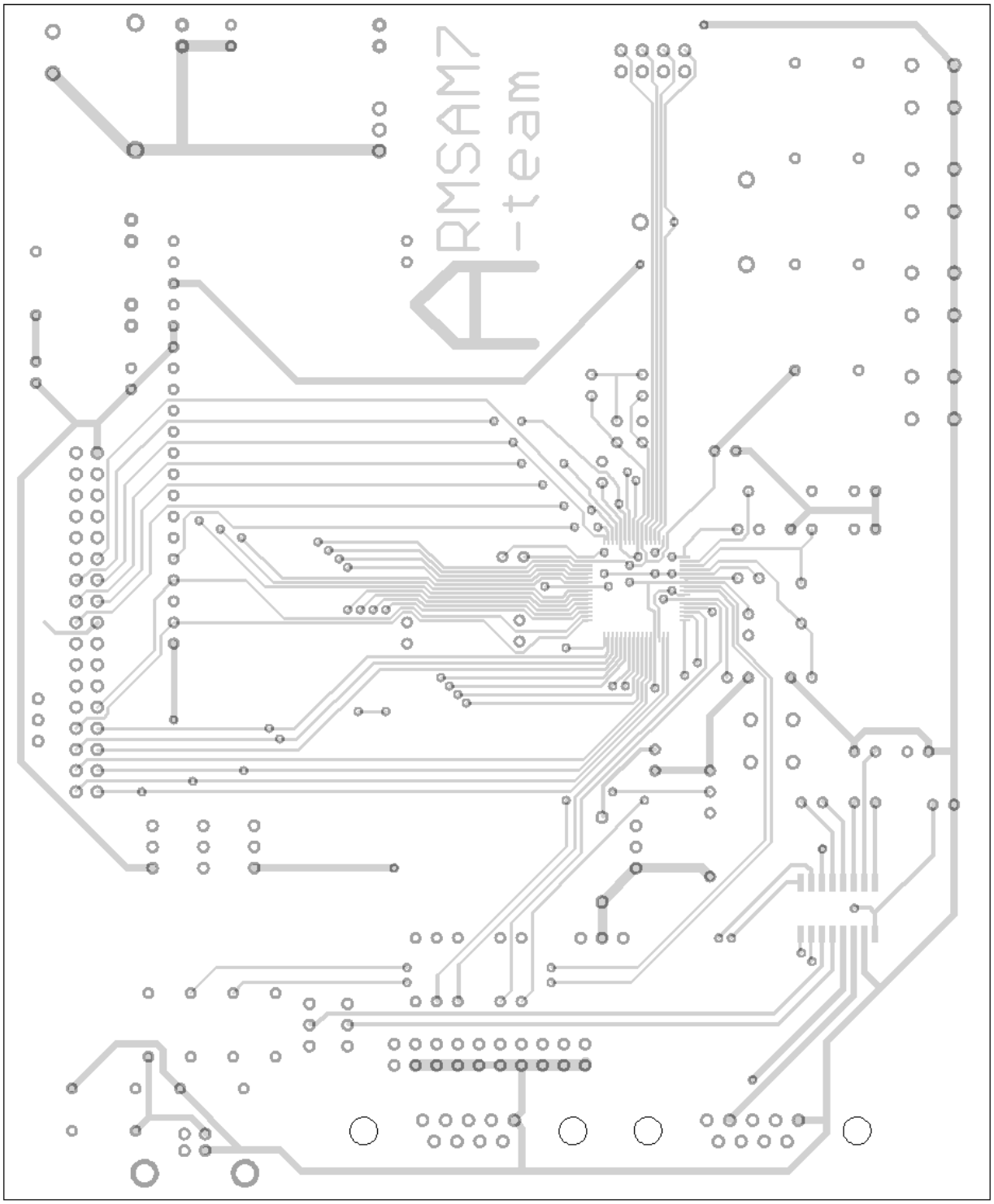

Priloha A
TITLE: schema1_14
Document Number:
REV:
Date: 16.03.2008 18:06:14
Sheet: 1/1

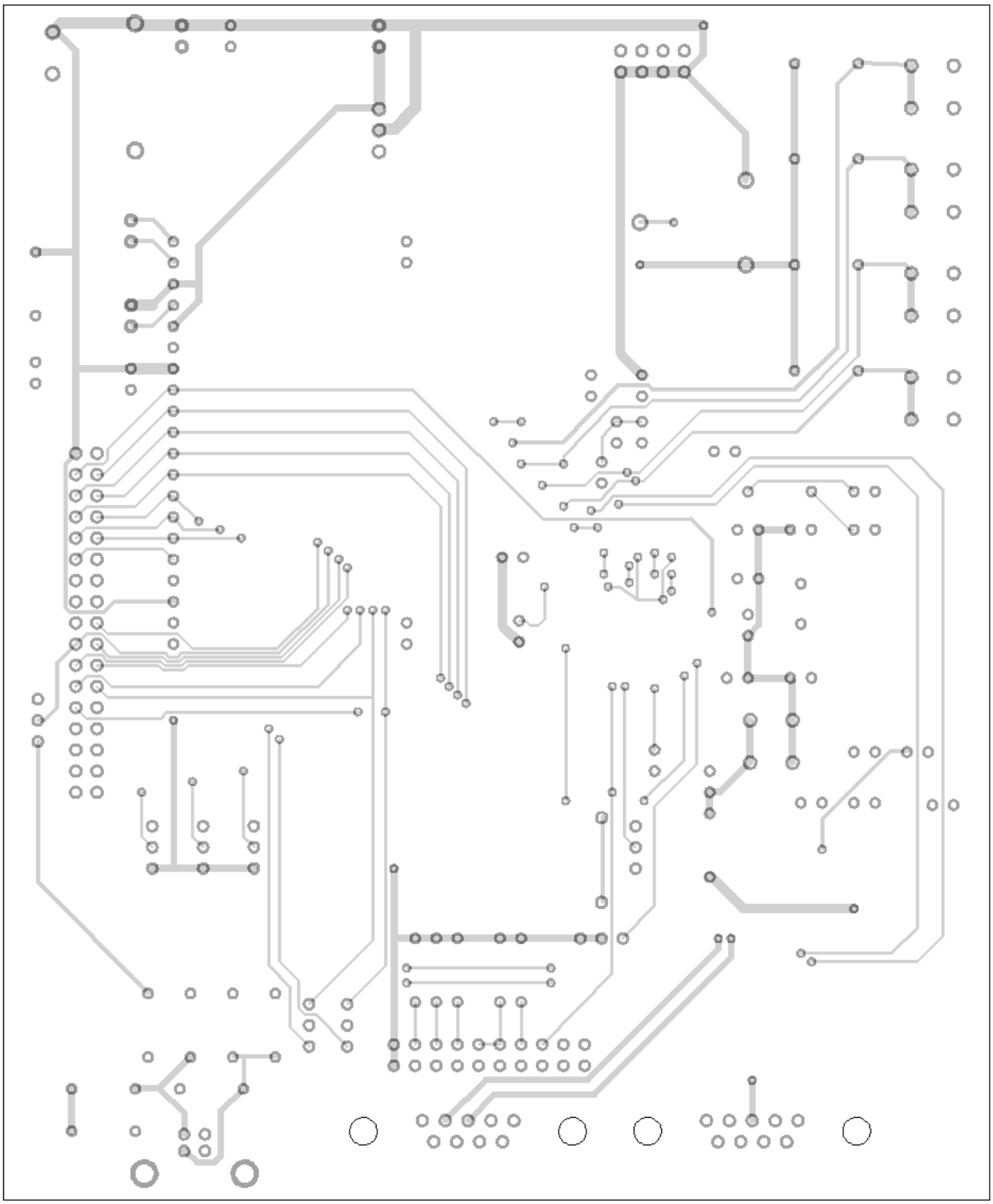

NECH ZIJE GENERÁL!

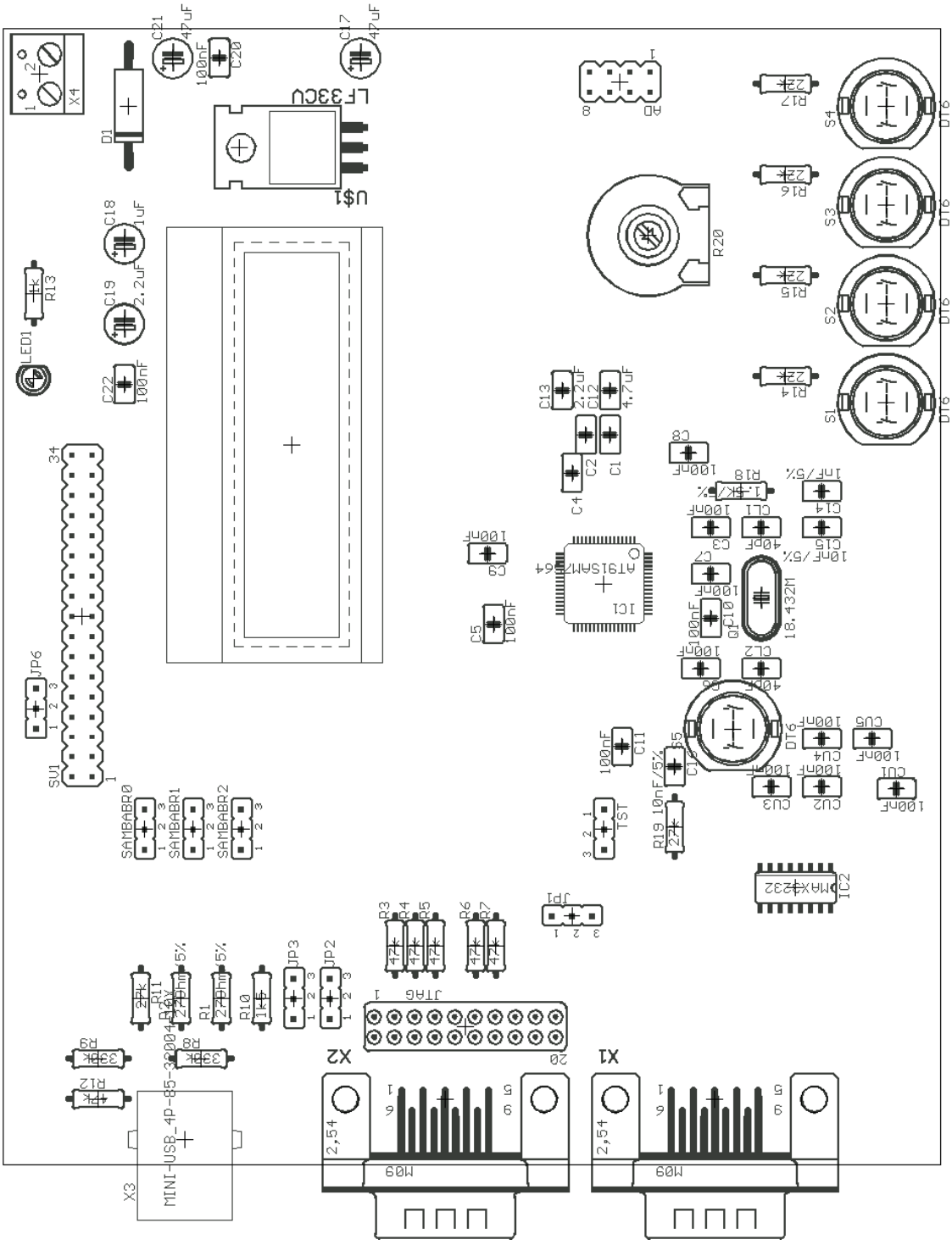



POWER
DISP_0
DISP_1
DISP_2
D56-116WA
DA56-116WA
LED1
D2
47uF
R58
1k
0.1uF
C23
PVA2
SW2
SW1
DT6
RESET
C3
100nF
R1
1K
D1
INVERTOR1
74HC573N
LATCH
C22
C24
C25
C26
C27
C28
100nF
PROCESSOR
ME6A32-P
74LS04N
74LS138N
DECODER
C30
C21
SPI
SERIAL
MAX232
C15
C12
22pF
3MHz
C19
100nF
2,54
M09
USB
FT245BM
R55
R56
R57
47R
10R
6 Mhz
C8
27pF
C9
27 pF
R53
C11
C4
1000uF
MINI-USB_4P-85-32004-10X
CON_USB
93-C46SN
EEPROM_USB
C14
100uF
LATCH_LCD
74HCT573N
74LS04N
INVERTOR2
C18
C20
100nF
C16
100uF


Příloha B
TITLE: arm1
Document Number:
REV:
Date: 24.04.2008 20:25:28
Sheet: 1/1
AT91SAM7S64
EA DOGMxxx-A
MAX3232
MINI-USB_4P-85-32004-10X
JTAG


ARMSAM7
A-team




X4
C21
47uF
C20
100nF
C17
47uF
D1
LF33CV
U$1
C18
1uF
C19
2,2uF
C22
100nF
R13
LED1
R20
R17
R16
R15
R14
S4
S3
S2
S1
DT6
C13
2,2UF
C12
4,7uF
C2
C1
C4
C8
100nF
R18
1,8K/5%
C14
1NF/5%
C15
10nF/5%
C3
CL1
100nF
40pF
C7
100nF
C10
100nF
18,432M
CL2
40pF
C6
100nF
C11
100nF
C5
100nF
C9
100nF
ATMEGA64
IC1
C16
100nF
R19 10nF/5%
27R
TST
MAX232
IC2
CU1
CU2
CU3
CU4
CU5
100nF
JP1
R3
R4
R5
R6
R7
47K
JTAG
X2
X1
2,54
M09
SV1
34
JP6
SAMBABR0
SAMBABR1
SAMBABR2
R11
2K
R14
27Ohm/5%
R1
27Ohm/5%
R10
1K6
JP3
JP2
R9
33K
R8
33K
R12
47K
X3
MINI-USB_4P-85-32004

Dodatok C

Zoznam použitých súčiastok

C.1 Mikropočítač s AVR

Súčiastka	Hodnota	Puzdro	Objednaný počet	Číslo v katalógu	Názov v katalógu
C1,6,7,13	1uF	E2-5	10	S073402	E105 10uF/63V TKR
C2-5,11,14,16,18-28,30	100nF	C2.5-3	50	62687	CDC 0,1uF Y5V RM2,54
C8-10,12,15	27pF	C2.5-3	10	53800	CDC 27pF NP0 RM2,54
C29	47uF	E2,5-5	1	47042	ELRA 100uF/25V KOME
CON_USB	MINI-USB_4P-85-32004-10X	85-32004-10X	1		
D1	DO41-10	diode	1	52103	P600B-B002
D2	C1702-15	diode	5	S002045	ZD1,3W5,1V (BZV85C)

DECODER	74LS137N	DIL16	1	S010758	74HC137
DISP_0 - 2	D56-11GWA	HDSP-R	3	K001616	DA56-11EWA
FLASH_USB	93-C46ASN	SO-08	1	S030796	M93C46-WMN6P
INVERTOR1	74ALS04N	DIL14	2	S025614	74LS04
INVERTOR2	74ALS04N	DIL14	↑		
LATCH_01	74HC573N	DIL20	7	S041802	74HCT573
LATCH_02	74HC573N	DIL20	↑		
LATCH_11	74HC573N	DIL20	↑		
LATCH_12	74HC573N	DIL20	↑		
LATCH_21	74HC573N	DIL20	↑		
LATCH_22	74HC573N	DIL20	↑		
LATCH_LCD	74HCT573N	DIL20	↑		
LCD	EADOGMXXX-A	EADOGM162	1	52086	EADOGM162L-A
LED1	LED3MM	led	1	K001940	L-934HD
PROCESSOR	MEGA32-P	DIL40	1	45176	ATMEGA32L-8PU
Q1	8MHz	HC49/S	1	S029009	Q 8,0 MHz U-4 (Q-8-SS4-30-30/50)
Q2	6	Mhz	1	S029005	Q 6,0 MHz U-4 (Q-6-SS4-30-30/50)
R1,51,57	10K	0207/10	10	S032974	RM 1% 10 K
R2-49,58	1K	0207/10	50	S032860	RM 1% 1 K
R50	2K2	0207/10	10	S032896	RM 1% 2,2 K
R52,54	27R	0207/10	2	S032679	RM 1% 27 R
R53	470R	0207/10	10	S032822	RM 1% 470 R
R55	1K5	0207/10	10	S032878	RM 1% 1,5 K
R56	4k7	0207/10	10	S032936	RM 1% 4,7 K
RESET	DT6	DT6	1	E000397	DT 6 SW
SERIAL	MAX232	DIL16	1	48464	MAX232CPE+

SW1	PVA2	PVA2F	2	E000880	P10S
SW2	PVA2	PVA2F	↑		
USB	FT245BM	QFP-32	1	42416	FT245BLT&R
X3	M09H	con-subd	1	S025347	DMR 09 M#80422

C.2 Mikropočítač s ARM

Súčiastka	Hodnota	Puzdro	Objednaný počet	Číslo v katalógu	Názov v katalógu
C12	4.7uF	C025-025X050	1	S080002	CT 4,7uF 16V
C13	2.2uF	C025-025X050	1	S079992	CT 2,2uF 16V
C14	1nF/5%	C025-025X050	1		
C15,16	10nF/5%	C025-025X050	1		
C17	47uF	E2,5-5	1	47042	ELRA 100uF/25V KOME
C18	1uF	E2,5-5	1	S073402	E105 10uF/63V TKR
C19	2.2uF	E2,5-5	1		
C20	0.1uF	C025-025X050	1	62687	CDC 0,1uF Y5V RM2,54
C21	47uF	E2,5-5	1	47042	ELRA 100uF/25V KOME
CL1-2	40pF	C025-025X050	2		
C1-C11,CU1-5,C22	100nF	C025-025X050	17	62687	CDC 0,1uF Y5V RM2,54
IC1	AT91SAM7S64	LQFP64	1	51918	AT91SAM7S64-AU
IC2	MAX3232	SO16	1	62453	MAX3232EID

IC3	EADOGMXXX-A	EADOGM162	1	52088	EADOGM162W-A
JTAG	FE10-2	con-lsta	1	E006510	LPH20S
LED1	LED3MM	led	2	K001940	L-934HD
Q1	18.432M	HC49U-V	2	S043181	Q 18,432 MHz U-4 (Q-18.432-SS4-30-30/50)
Q2	BSS92	TO92	1	E005777	IRF9520PBF
R1,2	27Ohm/5%	0207/7	10	S032679	RM 1% 27 R
R3-7,12	47k	0207/7	10	S033050	RM 1% 47 K
R8-10	330k	0207/7	10	S033145	RM 1% 330 K
R11,19	27k	0207/7	10	S033021	RM 1% 27 K
R13	1k	0207/7	1	S032860	RM 1% 1 K
R14-17	22k	0207/7	10	S033010	RM 1% 22 K
R18	1.5K/5%	0207/7	1	S032878	RM 1% 1,5 K
R20	TRIM_EU-LI15	pot	1	S010719	PT 15 NV 10 K
S1	DT6	DT6	5	E000397	DT 6 SW
S2	DT6	DT6	↑		
S3	DT6	DT6	↑		
S4	DT6	DT6	↑		
S5	DT6	DT6	↑		
SV1	MA17-2	con-lstb	4	S023904	WWS 20 G
U$1	LF33CV	TO-220	1	S046104	LF33CV
X1	M09H	con-subd	2	S025347	DMR 09 M#80422
X2	M09H	con-subd	↑		
X3	MINI-USB_4P-85-32004-10X	85-32004-10X	1		