Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Mikroprocesory s jádrem ARM

Štěpán Novák

Bakalářská práce
2008

Abstrakt

Práce se zabývá vývojovými nástroji a obsluhou mikrokontrolérů s jádrem ARM. Nejprve je proveden krátký přehled vnitřní architektury a procesoru Atmel AT91SAM7X256. V další části jsou představeny bezplatné integrované vývojové prostředky a vysvětlena jejich konfigurace, jako upravená verze programu Eclispe pro jazyk C/C++, speciální edice GCC kompilátoru a GDB debuggru nazývaná Yagarto, OpenOCD k vytvoření mostu mezi počítačem a mikroprocesorem. Nakonec bylo vytvořeno několik demonstračních úloh pro zkušební desku Olimex SAM7-EX256.

Klíčová slova

ARM, ARM7TDMI, mikroprocesor AT91SAM7X256, ARM-USB-TINY, Eclipse, GNU nástroje, GCC, Yagarto, OpenOCD

Abstract

This paper is about development tools and service microcontrollers of ARM core. At first, is made a brief overview of the internal processor technology and Atmel AT91SAM7X256. In other parts are presented open sourced integrated development enviroment and explained their configuration, as modified version of Eclispe for C/C++ language, special editions of GCC compiler and GDB debugger called Yagarto, OpenOCD to create a bridge between the computer and microprocessor. At last, was programmed several examples for development board Olimex SAM7-EX256.

Keywords

ARM, ARM7TDMI, CPU AT91SAM7X256, ARM-USB-TINY, Eclipse, GNU toll-chain, GCC, Yagarto, OpenOCD

Poděkování

Chci poděkovat všem lidem, kteří přispěli k vypracování této práce, především pak vedoucímu projektu, panu Ing. Martinovi Hájkovi za pomoc při řešení dílčích úkolů, za čas, po který se mi věnoval v rámci odborných konzultací a za poskytnutí studijních materiálů.

Obsah

Seznam obrázků

Úvod

ARM je souhrnné označení pro mikroprocesorová jádra, která jsou v současné době jedničkou na poli nízkoenergetických zařízení. S procesory s jádrem ARM se setkáváme denně snad ve všech složitějších elektronických zařízeních, jako jsou například mobilní telefony, DVD přehrávače, plazmové televizory, GPS navigace, aktivní prvky počítačových sítí či telemetrické snímače v dopravě.

Oblast kolem rodiny ARM je velice široká, proto jsem si zvolil konkrétní cíle a zaměřil se na jádro s označením ARM7TDMI. I když se jedná o nejnižší vývojový stupeň, nabízí obrovské možnosti ve využití. Technologie je kompatibilní s celou následující řadou až po ARM11. Proto jeho základní znalost může posloužit i pro pozdější vývoj mnohem složitějších zařízení.

Pro praktické testy byla zvolena vývojová deska Olimex SAM7-EX256, která je osazena mikroprocesorem Atmel AT91SAM7X256.

Cílem práce je základní seznámení s technologiemi ARM, popis základních vlastností procesoru AT91SAM7X256 a programovacích technik s využitím volně dostupných a bezplatných softwarových řešení, která jsou také známa pod označením open-source.

V první kapitole je velice obecně nastíněna architektura a její vznik, druhá obsahuje rozdělení nejpoužívanějších jader, třetí se zabývá vývojovými nástroji. Ve čtvrté je popsána instalace a nastavení programů. V poslední jsou vysvětleny nejdůležitější části vzorového programu a ovládaní mikroprocesoru.

1 Popis architektury ARM

1.1 Historie

Architektura ARM (Advanced RISC Machine) byla vyvinuta pod vedením Roger Wilson a Steve Furber firmou Acorn RISC Machines (Obr. 1) [7]. Hlavním cílem bylo dosažení co nejrychlejšího zpracovávaní dat za velmi nízkou cenu. V roce 1985 byl představen první procesor s označením ARM1, ale až o rok později se firma dočkala plně funkčního modelu ARM2 s 32bitovou sběrnicí a 32bitovým adresovým prostorem. Tím se stal nejjednodušším a velice funkčním mikroprocesorem světa s pouhými 30tis transistory. Jeho slabší výkon než Intel 80286 byl způsoben tím, že nevlastnil cache paměť. Následovník ARM3 už byl vybaven 4 kB cache.

Po velkém úspěchu se společnost rozdělila a vznikla nová divize Advanced RISC Machines. V roce 1991 byla vydána nová verze procesoru ARM6, který ve svém novém PDA použila firma Aplle. Cílem firmy bylo prodávat samostatná jádra procesoru a vzniklo dodnes velice populární ARM7TDMI. Se stovkami milionů kusů se objevila v každém druhém elektronickém zařízení, které využívalo mikroprocesor. Později vznikla další a mnohem dokonalejší jádra ARM8 – ARM11, ty se dnes používají nejen v nejdokonalejších a nejmodernějších přenosných zařízeních. Po určitých právních nejasnostech tuto technologii má ve svých procesorech i firma Intel, pod označením StrongARM nebo Xscale (více na [9]).

Obr. 1: Oficiální logo ARM

1.2 Obecné vlastnosti jádra ARM

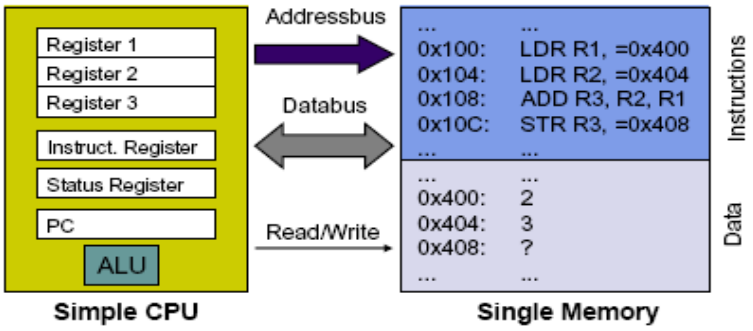
Technologie ARM je světově nejpoužívanějším 32bitovým mikroprocesorem, dokládá to 2,5miliardy prodaných kusů a 75% podíl na celosvětovém trhu. Jeho největší výhody spočívají ve vynikajícím výkonu, nízké spotřebě a jednoduchém programování. ARM je otevřenou technologií a to nám zaručuje vysokou kompatibilitu s jinými systémy. Jádra ARM jsou vybavena technologií Thumb®, která umožňuje používat procesory i v 16bitovém režimu.

Procesory byly původně optimalizovány pro programovací jazyk Java™, ale v dnešní době jsou podporovány v závislosti na vývojovém prostředí všechny nejběžnější programovací nástroje (C/C++, assembler). Pro obrovskou univerzálnost těchto systémů se nachází jejich uplatnění v nejrůznějších moderních elektronických zařízeních.

Jádra ARM obsahují mnoho vylepšení, která usnadňují jejich pozdější použití v konkrétních projektech (více na [8]).

- Jazelle® – zvýšená podpora pro programování v jazyce Java
- TrustZone™ – zabezpečená oblast paměti, která usnadňuje používání operačních systémů (Windows CE, Linux, Apple)
- Intelligent Energy Manager – řídí optimální zátěž procesoru za účelem snížení odebírané energie, současně při zachování požadovaných výkonnostních nároků (mobilní telefony, PDA, GPS navigace)
- OptimoDE™ – nástroj pro rychlou a nepřetržitou práci s údaji v reálném čase (zpracování zvuku a obrazu, telemetrické systémy)

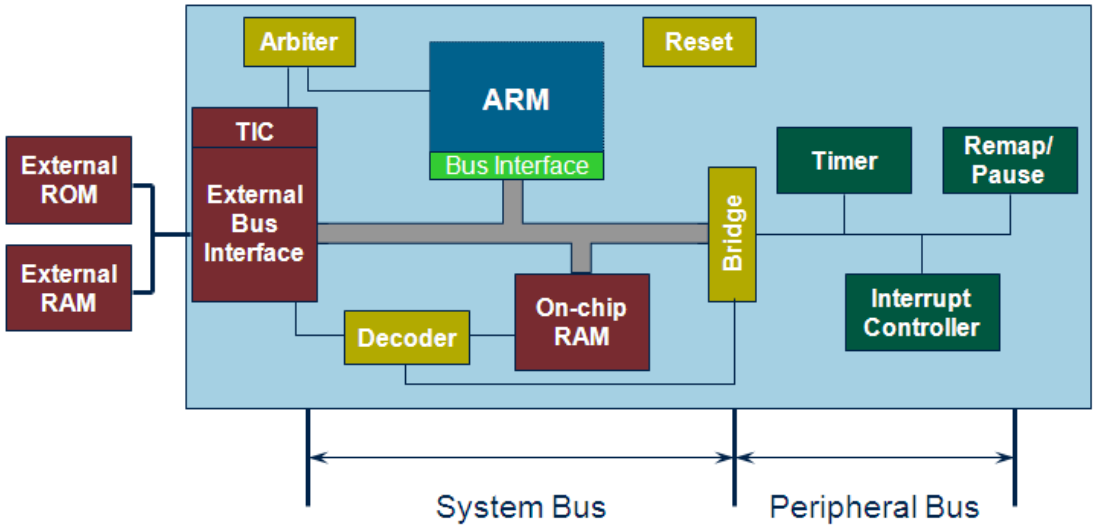
Všechny procesory s jádrem ARM využívají Von Neumannovu technologii, to znamená, že data i instrukce se nacházejí v jednom adresním prostoru. K přenosu dat se využívá sběrnice, která má tři funkční části. První se stará o určení cílové adresy v paměti, druhá o přenos dat a poslední určuje, zda se jedná o čtení nebo zápis dat (Obr. 2) [10].



Obr. 2: Von Neumannova architektura

Na vývoji jader ARM spolupracuje a ve svých procesorech využívá mnoho firem, proto je snahou navrhovat produkty co nejvíce kompatibilní a s pevně stanovenými standardy. Pro příklad uvádím tyto významné firmy: Atmel, Intel, Siemens, Epson, Microsoft, Sony, Motorola, Yamaha, Panasonic, Fujitsu, 3com... [10]

Tyto firmy využívají ve svých procesorech pouze jádro, ostatní komponenty jako obvody pro řízení času, paměti, periférie (Ethernet, USART, přerušení), jsou dodávány výrobci daných procesorů (Obr. 3) [10]. Externí zařízení s jádrem komunikují po standardizované sběrnici. Výrobci tak můžou přizpůsobovat mikroprocesory na míru svým požadavkům, ale i zachovat vzájemnou kompatibilitu, to znamená, že program z jednoho procesoru bude fungovat i v procesoru od konkurenční firmy (více na [7]).



Obr. 3: Blokové schéma procesoru s jádrem ARM

2 Typy jader ARM

2.1 ARM7

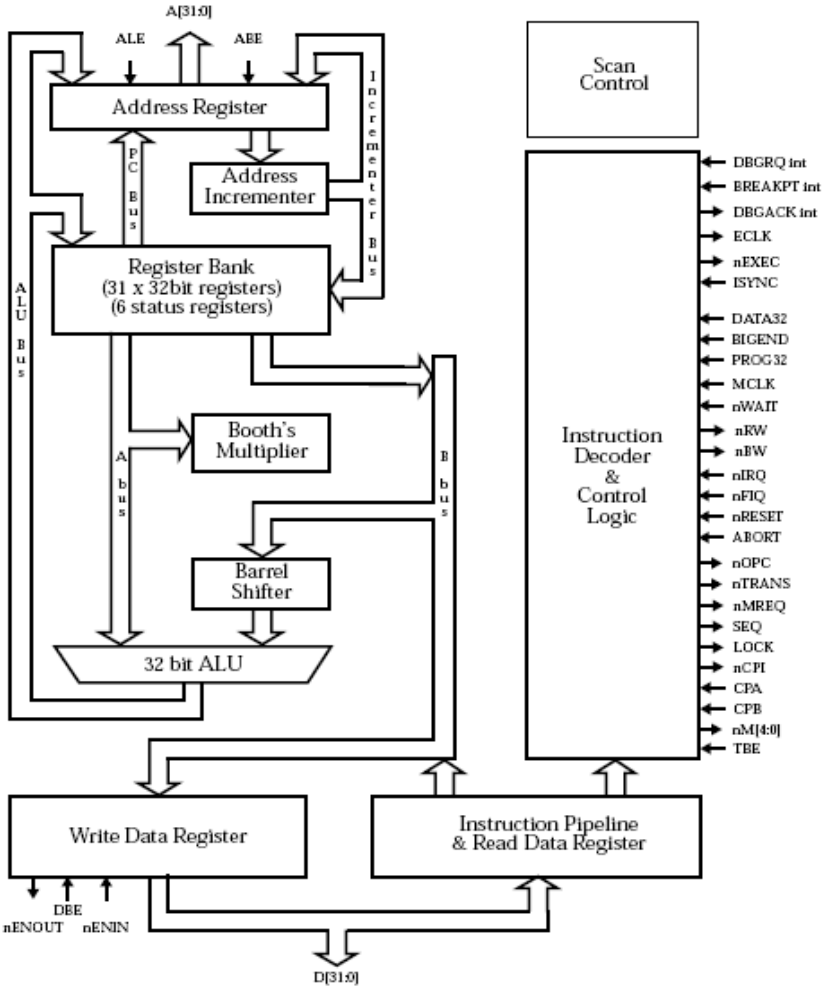
Jedná se nejjednodušší používanou řada 32bitového RISC jádra ARM. Velkou výhodou je malá spotřeba a práce i s nízkým napájecím napětím. Podporuje Thumb® 16-bitovou instrukční sadu. Výkon tohoto procesoru dosahuje až 130 MIPs. Jsou vyráběny 0.25 um, 0.18 um a 0.13 um technologií (kompletní informace na [3]). Tento typ je podporován nejrůznějšími vývojovými nástroji, jak komerčními tak i volně šiřitelnými. Výstupní kód je kompatibilní s jádrem ARM9 i ARM10. Jsou vyvinuty čtyři typy jader, které jsou uzpůsobeny na konkrétní vlastnosti:

- ARM7TDMI – základní integrované jádro
- ARM7TDMI-S – rozšířená verze předchozího jádra
- ARM7EJ-S – jádra optimalizovaná pro vývoj jazykem Java™

- ARM720T – verze s cache pamětí optimalizovaná pro operační systémy jako Windows CE, Linux, Palm OS, Symbian OS

Tento typ jádra je vhodný pro využití v jednodušších aplikacích na zpracování audia (MP3, WMA, AAC), nebo například pro vstupní rozhraní telefonů, pagerů.

Na (Obr. 4) [3] je vnitřní blokové schéma jádra, které se skládá v pravé části z dekodéru instrukcí a vstupů externích funkcí (IRQ, FIQ, paměť, JTAG-IEC), ty přes vstupně výstupní registry komunikují po sběrnicích s ostatními části jádra (ALU, banka funkčních registrů). ALU jednotka je řízena registrem adres, který je inkrementován počítadlem.



Obr. 4: Blokové schéma jádra ARM7TDMI

2.1.1 Specifikace

- 32bitový RISC procesor
- napájení 1,8 V

- příkon nižší než 0,4 mW
- pracovní frekvence 60 – 110 MHz
- architektura typu Von Neumann
- 3stavové zpracování instrukcí

2.1.2 Pracovní režimy

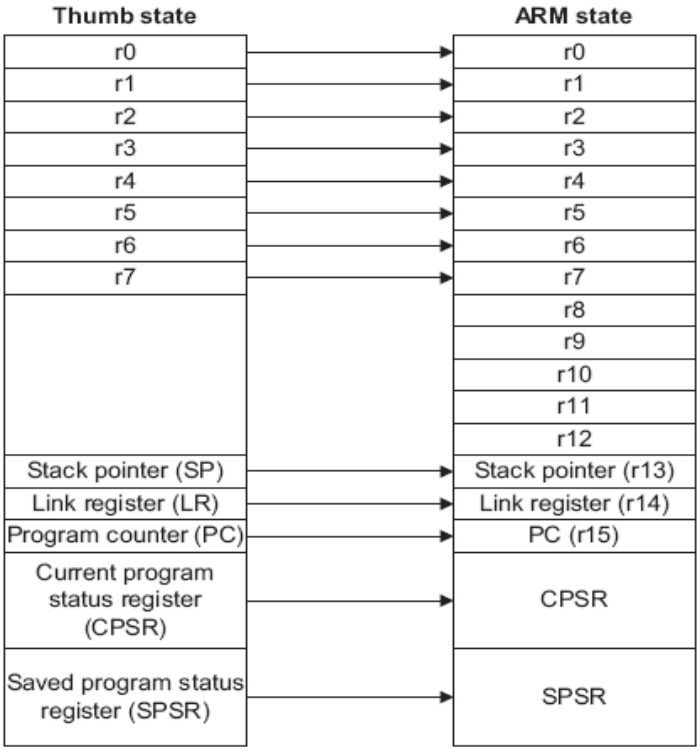
Procesorové jádro má 37 celočíselných 32bit registrů, podporuje 32bit, 16bit a 8bitové datové typy. Je možné využívat 7 uživatelských režimů [10]:

- USR – normální programový režim
- FIQ – režim datového přenosu (rychlá obsluha IRQ, přenos dat prostřednictvím DMA)
- IRQ – režim pro základní přerušovací služby
- SVC – chráněný režim pro operační systém
- ABT – režim pro případ, kdy se prováděná instrukce nepodaří správně vykonat
- UND – režim pro případ vykonání nedefinované instrukce
- SYS – privilegovaný režim pro uživatelskou aplikaci

2.1.3 Popis registrů

Každý z výše uvedených pracovních režimů má svoji sadu registrů, používá se vždy aktivní. Pro Thumb mód je redukován volný počet registrů (Obr. 5) [3].

- R0 – R12 univerzální 32bitové registry
- R13 (SP) ukazatel do zásobníku adres
- R14 (LR) registr pro uložení obsahu registru R15 při vykonání instrukce typu BL
- R15 (PC) programový čítač
- R16 (CPSR) stavový registr



Obr. 5: Rozložení registrů v ARM7TDMI

2.1.4 Instrukční sada

Jádro podporuje dva režimy instrukcí. Základní 32bitový a 16bitový Thumb, které dovolují programátorovi uzpůsobit program na dané zaměření. Thumb mód ušetří až 40% paměťového prostoru, ale nevýhodou je znatelné zpomalení mikroprocesoru. Lze využívat i 8bitových datových typů. Jádro podporuje klasickou sadu jazyka assembler [3].

- 15 typů instrukcí pro větvení programu (B, BL, BLX, BX, ...)
- 16 typů instrukcí pro aritmetické a logické operace (AND, ADD, SUB, MUL)
- 10 typů instrukcí pro přesun dat (MOV, LDR, STR, SWP, ...)
- SWI instrukce pro softwarové přerušení

2.2 ARM9

Tato řada nabízí až dvojnásobně větší výkon než ARM7, též se jedná o 32bitovou architekturu typu RISC. I zde je možno využívat 16bitového Thumb® módu. Vyrábí se ve dvou provedeních ARM920T a ARM922T. Obě tyto verze jsou určené především pro použití s operačním systémem (Symbian OS, Palm OS, Linux, Windows CE), liší

se pouze vestavěnou cache pamětí. V první verzi je dostupných 2 x 16 kb a ve druhé pouze 2 x 8 kb v duálním režimu.

Procesorů s jádrem ARM9 se využívá v oblasti nových mobilních zařízeních, které zvládají i náročné aplikace jako videotelefonování. Dále se s nimi můžeme setkat v set-top-boxech, počítačových routerech, WiFi zařízeních, v herních konzolích. Tímto procesorem je možné zpracovávat video ve formátech MPEG2, dokonce i MPEG4. A proto si našly uplatnění v digitálních kamerách a fotoaparátech. Ale nejen to, jejich výkon dostačuje i na telemetrické systémy, které se čím dál častěji objevují v řízení dopravy (více na [7]).

2.3 ARM11

Nejvýkonnější jádro založené na technologii ARM. V současné době se využívá v nejmodernějších přístrojích, smartphonech, digitálních televizích, set-top-boxech, DVD přehrávačích. Uplatnění naleznou i v systémech pro rozpoznávání řeči a obrazu. S touto technologií je dosahováno dobrých výkonů v 3Dgrafice (podpora OpenGL), a proto byly použity i v počítačových grafických kartách firmou nVidia.

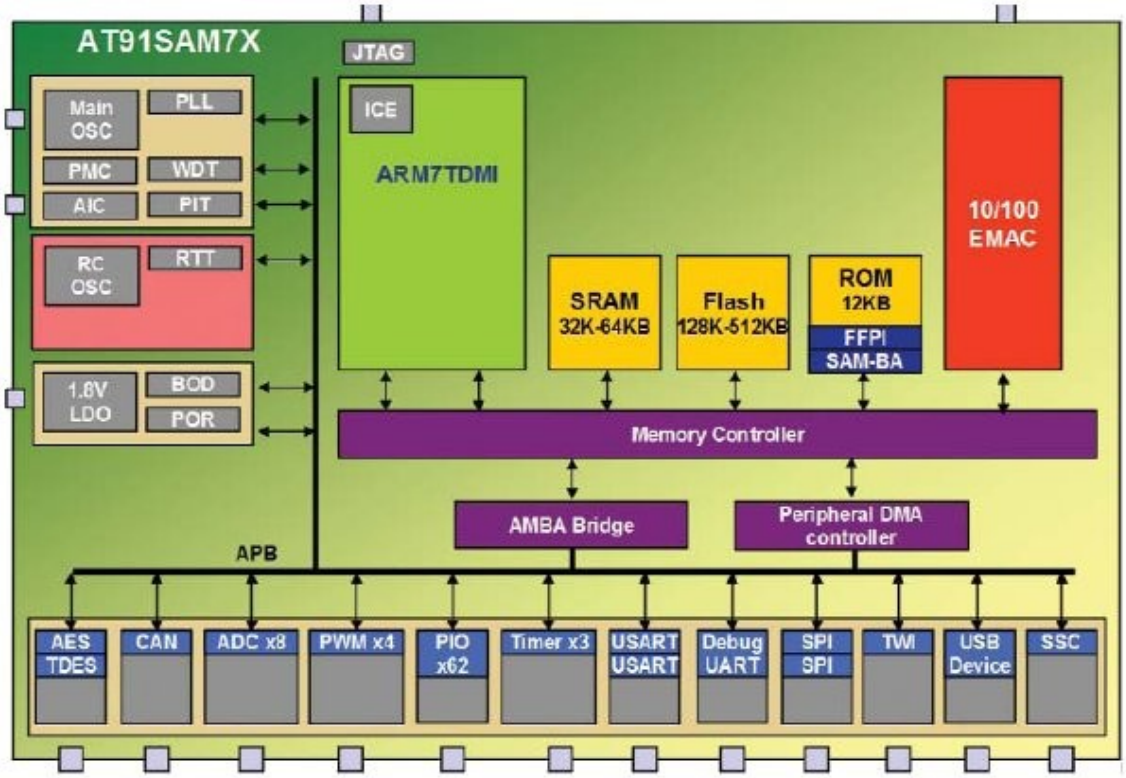
Jádro pracuje s napětím 1,2 V a jeho spotřeba včetně cache paměti je pouhých 0,6 mW. Podporuje Thumb-2 mód, který řeší výkonnostní problémy předchůdce. Nárůst výkonu je až o 35%. Je zajištěna podpora až 4 x 64 Kkb paměti cache a možnost využívat 64bitové datové typy. Samozřejmě nechybí podpora pro programování v Jave a mód on-chip zabezpečení oblastí, které využívají operační systémy (více na [7]).

3 Procesor AT91SAM7X256

Procesor je založen na jádře ARM7TDMI a je členem řady vysoce integrovaných RISC procesorů s flash pamětí a podporou 32bitových datových typů. Obsahuje 256 kB flash paměti a 64 kB RAM paměti. Je možné použít i redukovanou instrukční 16bitovou sadu Thumb-2.

Usnadněnou a rychlou komunikaci s perifériemi zajišťuje podpora DMA. Programování tohoto procesoru je možno provádět i prostřednictvím JTAG-IEC rozhraní nebo pomocí sériového kanálu. Procesor disponuje chráněnou pamětí, která zamezuje nechtěné přepsání důležitých částí programu a nesprávnou uživatelskou manipulaci.

Procesor má mnoho vestavěných periferií a to snižuje počet externích komponent, což zjednodušuje návrh zařízení a šetří místo. Například 802.3 Ethernet MAC, CAN rozhraní, USART, SPI, AD převodník, USB 2.0 (Obr. 6) [11].

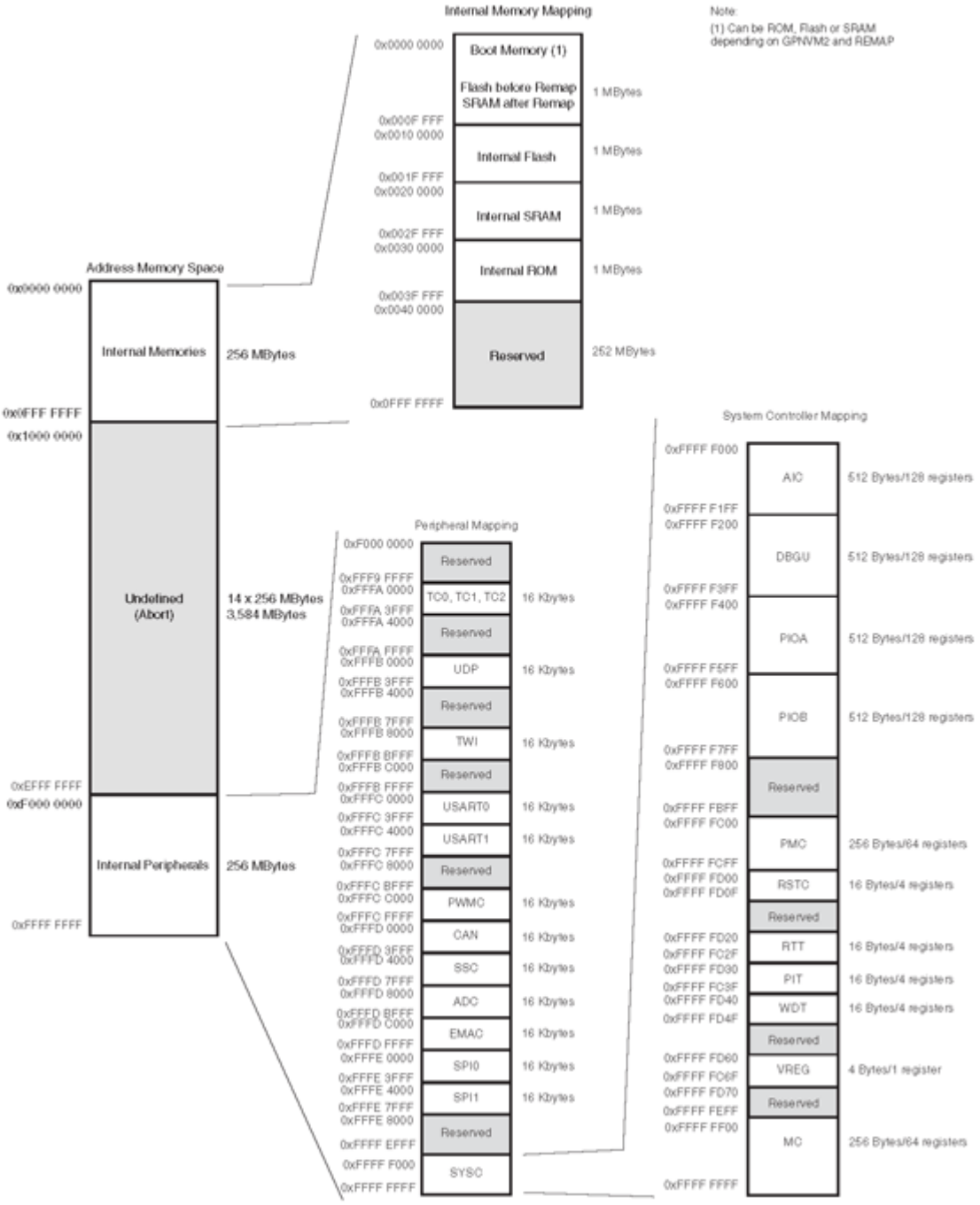


Obr. 6: AT91SAM7X256 - blokové schéma

Na obrázku (Obr. 6) je vidět základní blokové schéma procesoru. Jádro ARM7TDMI komunikuje pouze s paměťovým řadičem, což je velice důležité, protože všechny funkce procesoru se řídí pouze zápisem na správnou adresu řadiče. Každý blok v paměti, funkční registr i okolní periferie, mají svoji unikátní adresu (Obr. 7) [2].

Paměťový řadič komunikuje přímo s RAM, ROM i flash pamětí. Kvůli vysokým rychlostem dosahovaných na Ethernet sítích není možné EMAC 10/100 připojit přes sběrnici, proto je spojen přímo. Dále je ještě přes dva různé mosty, klasický a s podporou DMA pro rychlou komunikaci, připojena sběrnice procesoru.

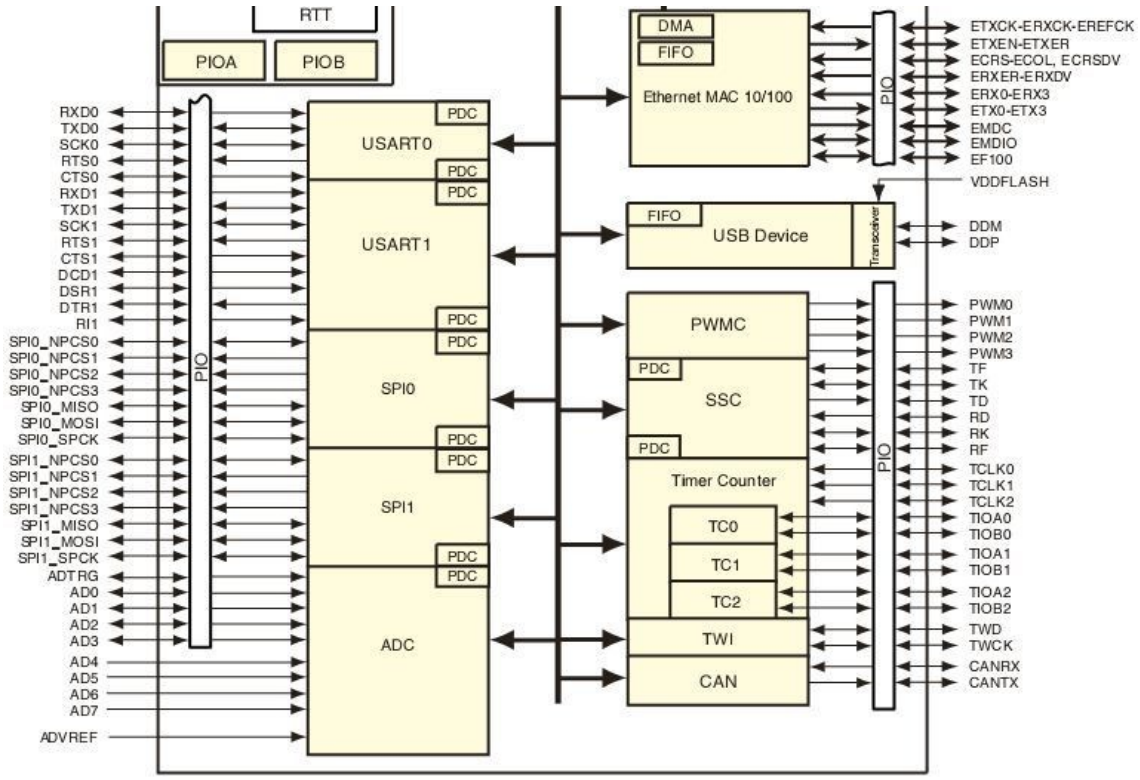
Na sběrnici jsou připojeny všechny periferie. Hlavní oscilátor procesoru, RC oscilátor, softwarově řízený napěťový zdroj, časovače, vstupně výstupní porty, USB, CAN... (Obr. 8) [11].



Obr. 7: Rozdělení adresového prostoru AT91SAM7X256

Procesor je řízen krystalem 3 – 20 MHz, ale tato frekvence se nerovná taktovací frekvenci. V obvodu řízení času jsou děličky a jedna násobička kmitočtu. Pomocí zvolených parametrů (v závislosti na krystalu) můžeme určit taktovací frekvenci přesně podle našich potřeb. Procesor má také vestavěný napětím řízený oscilátor, který je možný nastavit zápisem do daného registru, jeho frekvence může být až 55 MHz. Tento vnitřní oscilátor je používán procesorem po resetu, než se softwarově nastaví hlavní. Ale také pro programování či debugger.

V procesoru jsou tři nezávislé časovače RA, RB, RC. Jejich nejčastějším využitím je obsluha tří časových přerušení pod označením IRQ. FIQ je označení pro nejrychlejší obsluhu přerušení, jaké jde u tohoto typu procesoru docílit. Procesor má ještě další nezávislý časovač, který dokáže vyvolávat přerušení. Uplatnění najde například v udržování reálného času, na tuto funkci je také uzpůsoben. Procesor obsahuje několik vstupů pro vyvolání externího přerušení. U tohoto zařízení je možné řídit spotřebu energie, v závislosti na vykonávané funkci. Průměrná spotřeba se pohybuje mezi 3 – 3,6mW. WatchDog hlídá správný běh instrukcí v procesoru (více na [2],[11]).



Obr. 8: Rozdělení periférií a portů AT91SAM7X256

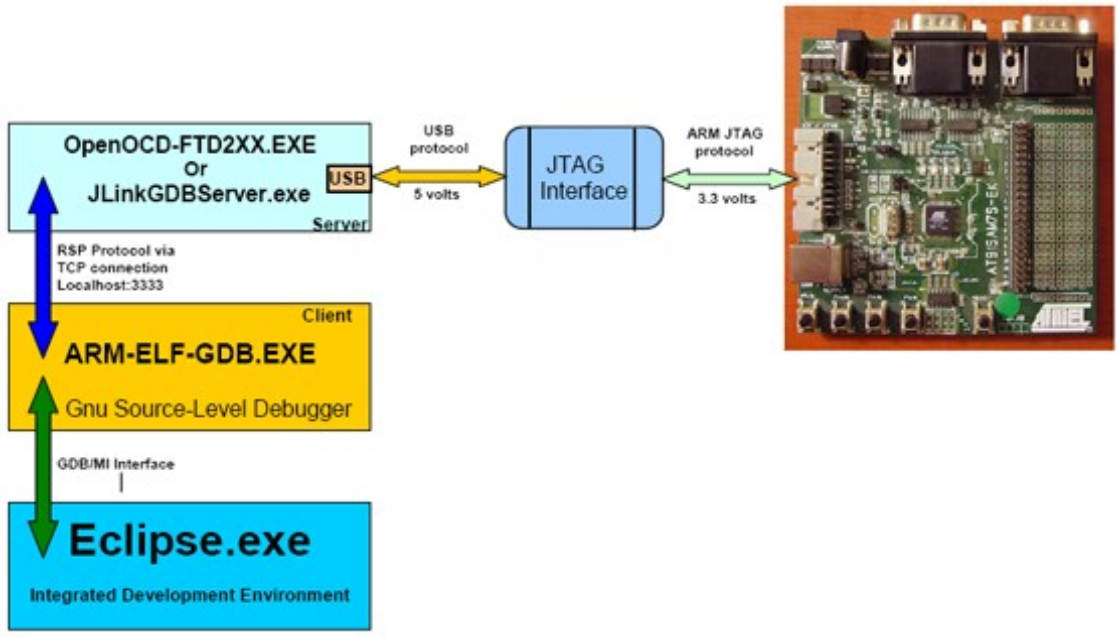
4 Dostupné vývojové prostředky

Většina dnešních vývojových prostředí pro programování mikroprocesorů, jakými jsou komerční balíčky IAR® EWARM, ARM RealView® nebo CrossWorks, jsou velice propracované a integrují v sobě všechny nástroje pro práci s procesorem bez větších obtíží a složitějšího nastavení. Na svých stránkách prezentují vzorové příklady s návody a také možnost stažení knihoven přesně na míru pro konkrétní procesor nebo typ. Avšak tato řešení bývají obvykle i finančně náročná. Existuje i možnost použití volně šiřitelných programů (open-source), které se dnes stávají více konkurenceschopné. Velkým

problémem je, že zatím neexistuje žádný souhrnný balíček, který by integroval všechny komponenty a nabídl komplexní vývojové prostředí. Abychom dosáhli stejných vlastností, jako u komerčních produktů, musíme navzájem propojit několik různých programů. Každý má na starosti určitou činnost, jako například kompilování projektů, komunikaci s procesorem nebo on-chip ladění. Všechny tyto komponenty může zastřešovat a ovládat jediný program, například Eclipse [6].

Fyzické programování mikroprocesoru AT91SAM7X256 se provádí různými cestami, přes sériový port, USB, nebo JTAG-IEC rozhraní. Nejvhodnější je použití JTAGu, protože je schopen posloužit i při on-chip ladění, ostatní přenosové cesty tuto možnost nepodporují.

Obrázek (Obr. 9) [4] nastiňuje komunikaci mezi IDE (integrované vývojové prostředí) a mikroprocesorem. Eclipse ovládá GDB komponentu Yagarta, která řídí on-chip ladění procesoru, ta mu naopak zpět vrací údaje. GDB komunikuje po vnitřním síťovém rozhraní počítače (localhost) s programem OpenOCD, který obsluhuje USB s JTAGem a ten už je přímo spojen s vývojovou deskou, která je osazena procesorem AT91SAM7X256 [4].



Obr. 9: Komunikace vývojových prostředků s mikroprocesorem

4.1 JTAG Olimex ARM-USB-TINY

Obr. 10: Olimex JTAG ARM-USB-TINY

Levný, ale velice vydařený JTAG od firmy Olimex (Obr. 10) [12]. Slouží jako most mezi počítačem, kde je připojen přes USB 2.0 rozhraní a procesorem s 2 x 10pinovým konektorem. Tento JTAG je podporován v nejrůznějších programech jako IAR (od verze 5.00) CrossWorks (od verze 1.7) a v neposlední řade open-source programem OpenOCD. Tento JTAG umí obsluhovat všechny ARM procesory v závislosti na softwaru, který využíváme. Lze ho použít jak na on-chip ladění aplikací, tak i k samotnému zápisu do flash paměti procesoru, pokud to konkrétní typ mikroprocesoru dovoluje. Pracuje s napětím 2 – 5 V [12].

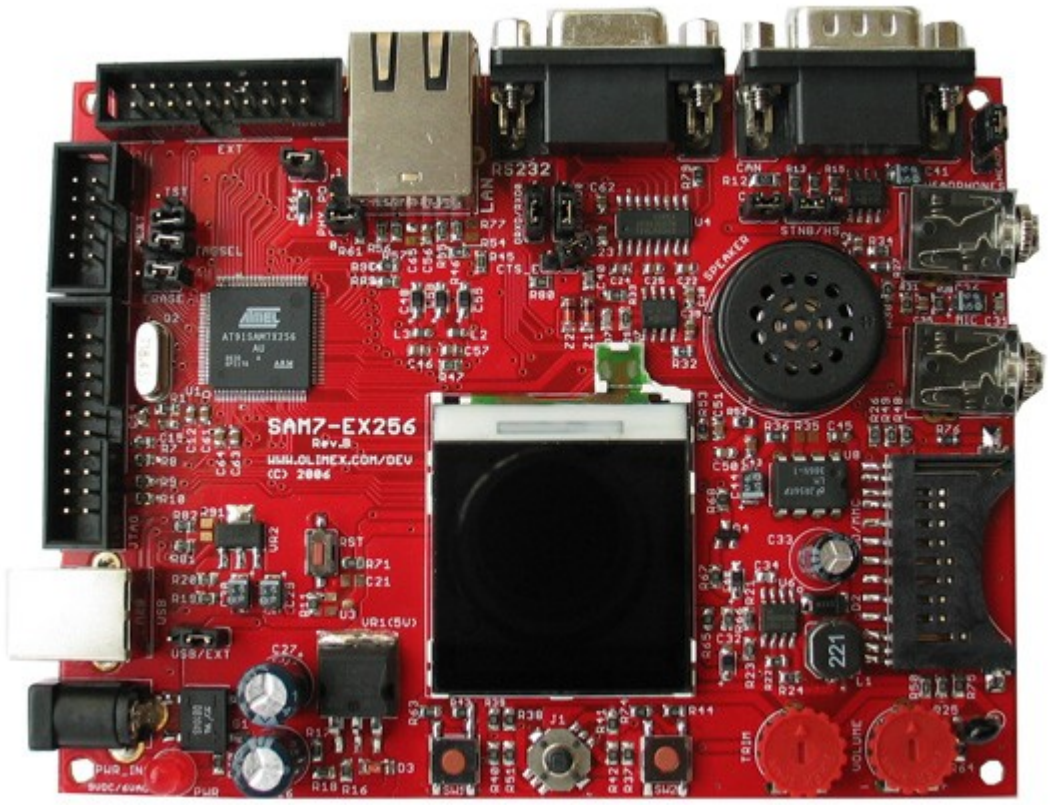
4.2 Vývojová deska Olimex SAM7-EX256

Jedná se o vývojovou desku pro procesor AT91SAM7X256 od firmy Atmel, která nabízí téměř kompletní sadu možností pro práci s procesorem na vestavěných zařízeních (Obr. 11) [13].

Základní specifikace vývojové desky:

- MCU: AT91SAM7S256 16/32bit ARM7TDMI™ s 256 KB Program Flash, 64K Bytes RAM, USB 2.0, RTT, 10bit ADC 384 ksps, 2 x UARTs, TWI (I2C), SPI, 3 x 16bit časovač, 4 x PWM, SSC, WDT, PDC (DMA) pro všechny periferie až do 60 MHz
- standardní JTAG konektor s ARM 2 x 10 piny zapojenými pro programování nebo debugging s ARM-JTAG

- NOKIA 6610 128 x 128 TFT 12 bit COLOR LCD displej s podsvícením
- Ethernet 10/100 PHY s KS8721BL
- USB konektor
- Dva kanály RS232
- SD-MMC konektor
- Joystick se čtyřmi směry, dvě samostatná tlačítka
- Audio vstup a audio výstup pro mikrofon a sluchátka, na desce je integrován reproduktor a potenciometr pro regulaci hlasitosti
- Potenciometr zapojen do ADC
- Termistor zapojen do ADC
- Integrovaný regulátor napětí 3.3 V zatižitelný do 800 mA
- Jeden napájecí vstup, požadované vstupní napětí 6 V AC nebo DC, zařízení může být také napájeno pomocí USB
- Napájecí kontrolka LED
- Filtrační kondenzátor pro napájení
- Resetovací obvod
- Resetovací tlačítko
- 18.432 MHz krystal osazený v patici
- Rozšiřující konektory pro všechny vývody procesoru
- Rozměry: 128 x 98mm (5 x 3.8")

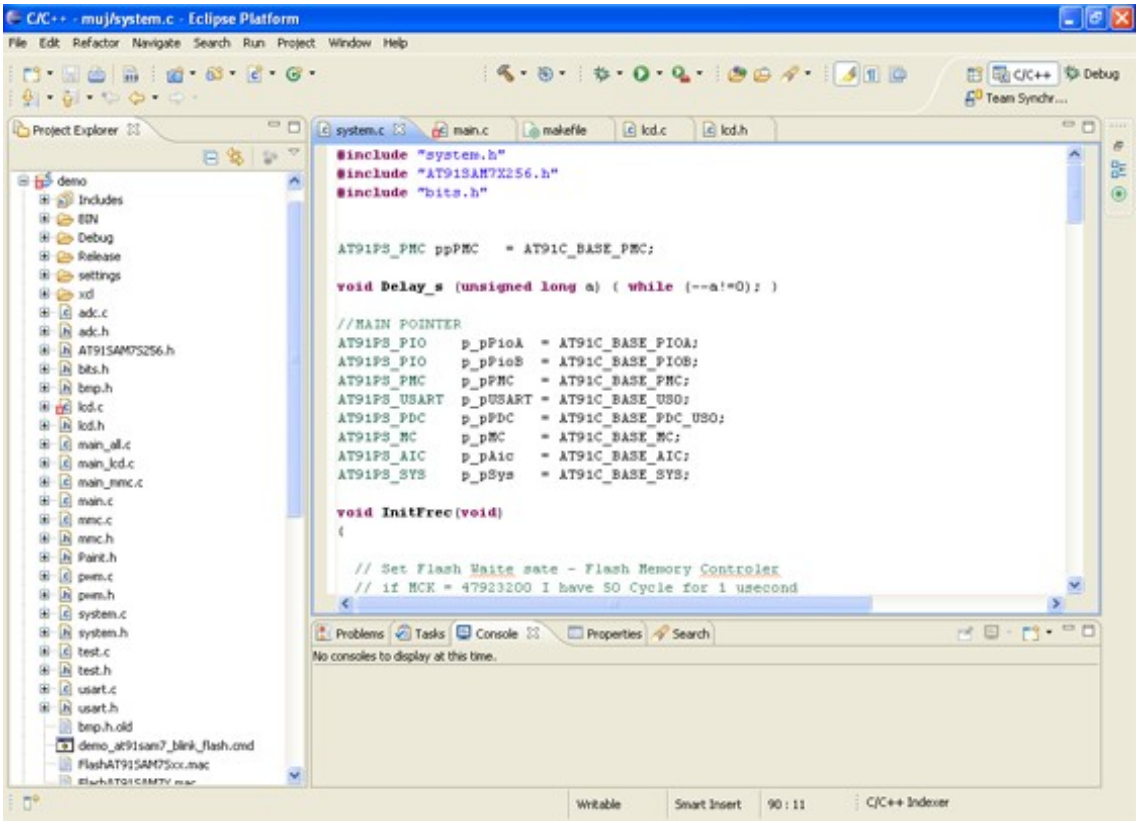


Obr. 11: Vývojová deska Olimex SAM7-EX256

4.3 Eclipse™

Eclipse je open-source integrovaným vývojovým prostředím (IDE), původně zaměřeným na programování v Java. Díky jeho univerzálnosti a přidáním několika vylepšení se výborně hodí na vývoj v jazyce C/C++ a integruje v sobě systémy řízení jako CVS a SVN. Eclipse slouží především pro psaní programů a umožňuje spravovat a využívat ostatní nezbytné externí doplňky (překladač, komunikace s mikroprocesorem, debugger).

Systém je naprogramován v Java, a proto je ho možné použít na všech operačních systémech kde je nainstalován Java Runtime Environment, jako například Linux nebo Windows [6].



Obr. 12: Eclipse

Rozložení programu Eclipse je velice intuitivní (Obr. 12) [6]. V levé části je panel souborů našeho projektu, uprostřed je okno pro zobrazení zdrojových kódů. Ve spodní části se nachází konzole, kde zobrazují všechny výstupy od spolupracujících programů. V pravé části si můžeme otevřít výpis proměnných a používaných funkcí. V horní části jsou ovládací prvky programu.

4.4 Yagarto

Je to implementace GNU ARM tollchainu pro jazyk C/C++ [14]. Mezi jeho nástroje například patří:

- GCC/G++ – C/C++ kompilátor
- AS – assembler kompilátor
- LD – linker
- GDB – debugger

V porovnání s ostatními GNU ARM toolchainy má Yagarto tu výhodu, že je nativně sestaven pro Windows. Většina ostatních jako například Cygwin a MinGW je ur-

čena spíše pro Linux a ve Windows jsou s nimi problémy. Ale ta největší výhoda je v tom, že Yagarto bylo vyvinuto pro Eclipse a je s ním plně kompatibilní [14].

4.5 Make

Většina všech GNU projektů psaných v jazyce C/C++ používá pro jeho sestavení Make, který je více znám spíše z Linuxu. Jelikož Yagarto tuto komponentu neobsahuje, existuje nativní verze pro Windows s názvem GnuWin32 [15]. Přináší větší efektivitu při překladu, při změnách v projektu není nutné kompilovat vše znovu, pouze změněné soubory. Projekty by také měly být více přenositelné mezi různými systémy.

4.6 OpenOCD

Jedná se o externí doplněk k IDE, který zprostředkovává komunikaci mezi počítačem a mikroprocesorem. Je plně kompatibilní s Eclipse a využívá se ho jak k zápisu do flash paměti procesoru, tak i k on-chip ladění programů přes JTAG rozhraní. OpenOCD podporuje všechny nejznámější JTAGy a plně spolupracuje s procesory ARM od firmy Atmel řady AT91SAM7 [16].

5 Konfigurace vývojového prostředí

5.1 Instalace

Před samotným nastavováním a používáním jednotlivých komponent je potřeba nainstalovat následující programy. Všechna pozdější nastavení a konfigurační soubory se budou vztahovat přesně na tyto verze programů. Po zkušenostech bylo zjištěno, že s přechodem na novější verze se velice často mění jejich vlastnosti. Jako operační systém je použit Windows XP SP2.

Instalované programy (dostupné na přiloženém CD-ROM disku):

- SUN Java Runtime Environment v6u7
- Eclipse Genymade 3.4.0 IDE for C/C++ developers
- OpenOCD r204
- Yagarto 2.18 (gcc 4.2.2)
- GnuWin32 – make 3.81

- Olimex OpenOCD ARM-USB-TINY Drivers
- Zylin Eclipse plugin

Po instalaci je vhodné zkontrolovat systémové proměnné v operačním systému Windows a popřípadě je doplnit o chybějící. To později usnadní práci s jednotlivými programy tím, že není nutné zadávat celou cestu ke spouštěcím souborům. Toto nastavení se zobrazí v *Start/Ovládací panely/Systém/Upřesnit/Proměnné prostředí/Systémové proměnné*, proměnná *Path* [4].

```
C:\ARM\yagarto\bin;C:\ARM\openocd\bin;C:\ARM\GnuWin32\bin
```

Mimo jiných je potřeba zkontrolovat tyto tři cesty. ARM je cílová složka, kam byly programy nainstalovány. Cesty k různým programům se oddělují středníkem.

Kontrola nainstalovaných programů se provede spuštěním příkazové řádky *Start/Spustit* a příkazem *cmd*, kde se zadají následující tučně vyznačené příkazy.

Java:

```
C:\>java -version
java version "1.6.0_07"
Java(TM) SE Runtime Environment (build 1.6.0_07-b06)
Java HotSpot(TM) Client VM (build 10.0-b23, mixed mode, sharing)
```

GnuWin32:

```
C:\>make
make: *** No targets specified and no makefile found.  Stop.
```

Yagarto:

```
C:\>arm-elf-gcc
arm-elf-gcc: no input files
```

OpenOCD:

```
C:\>openocd-ftd2xx
Info:    openocd.c:93 main(): Open On-Chip Debugger (2007-09-05
09:00 CEST)
Error:   configuration.c:122 parse_config_file(): couldn't open
config file
```

Pokud výstupy vypadají následovně, je vše plně funkční. Eclipse vyžaduje pro svoji práci minimální verzi programu Java 1.6.0 a vyšší.

Instalace JTAG rozhraní se provádí klasickým přidáním nového hardwaru do systému Windows. Po zapojení JTAGu do USB počítače, je nalezen nový hardware a je vyžadována cesta k driverům toho zařízení. Tyto drivery jsou dodávány současně s JTAGem firmou Olimex. Po úspěšné instalaci by se měla ve *Správci zařízení* objevit položka *Olimex OpenOCD JTAG interface*.

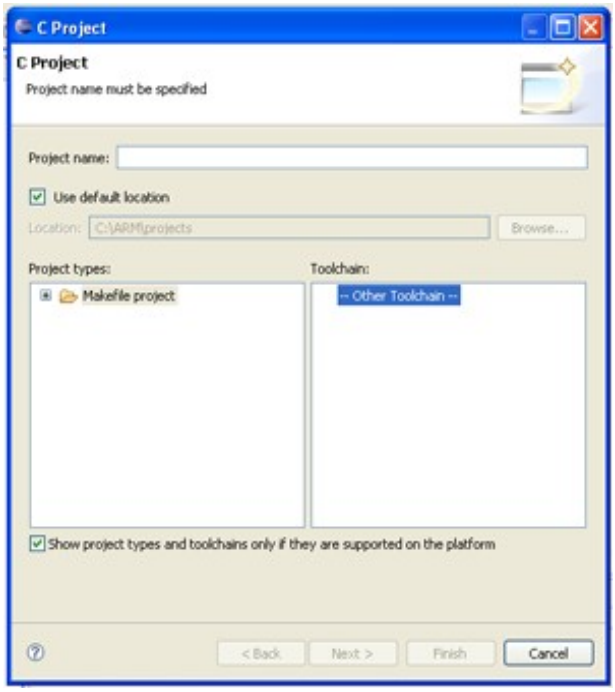
Pro správnou funkci všech komponent je dále potřeba nainstalovat *Zylin plugin* pro Eclipse, který řeší některé problémy s operačním systémem Windows, především je důležitý pro on-chip ladění. Instalace se provede v Eclipse pomocí *Help/Software update*, kde se přidá nový zdroj pro aktualizaci:

```
http://www.zylin.com/zylincdt
```

Poté se provede instalace pluginu, je vyžadován restart Eclipse. Tento plugin jde implementovat pouze od verze programu Eclipse 3.4 a novější [4].

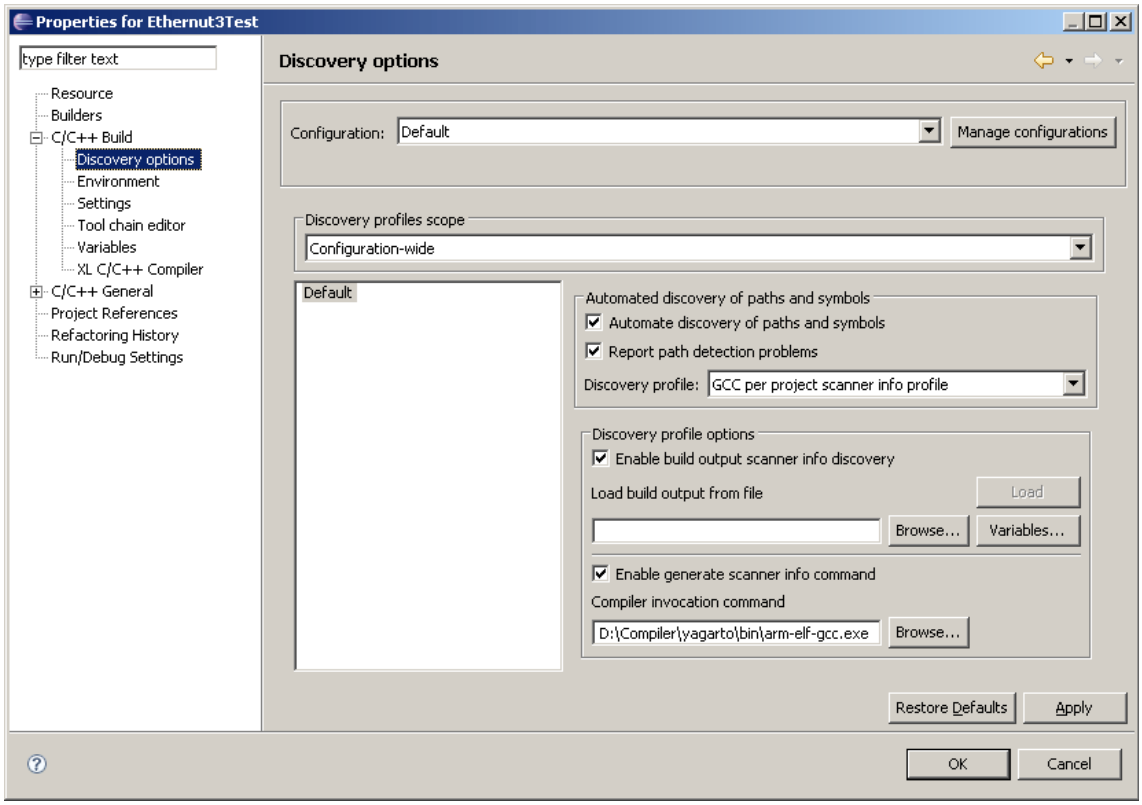
5.2 Vytvoření nového projektu

Po spuštění programu Eclipse je nutné nejprve nastavit *perspektivu C/C++*, aby bylo možné vytvářet projekty pro tento programovací jazyk. Změna se provede ve *Window/Open Perspective/Other*, kde se zvolí *C/C++*. Nový projekt je možno založit v záložce *File/New/C Project*. Zde se vybere položka *Other toolchain* (Obr. 13) [6] a zvolí se název a pracovní oblast. V různých verzích Eclipse se také mohou nacházet předpřipravená nastavení pro technologii Cygwin GCC, MinGW GCC. Pokud chceme používat Yagarto, musí se toto nastavení vytvořit ručně [14].



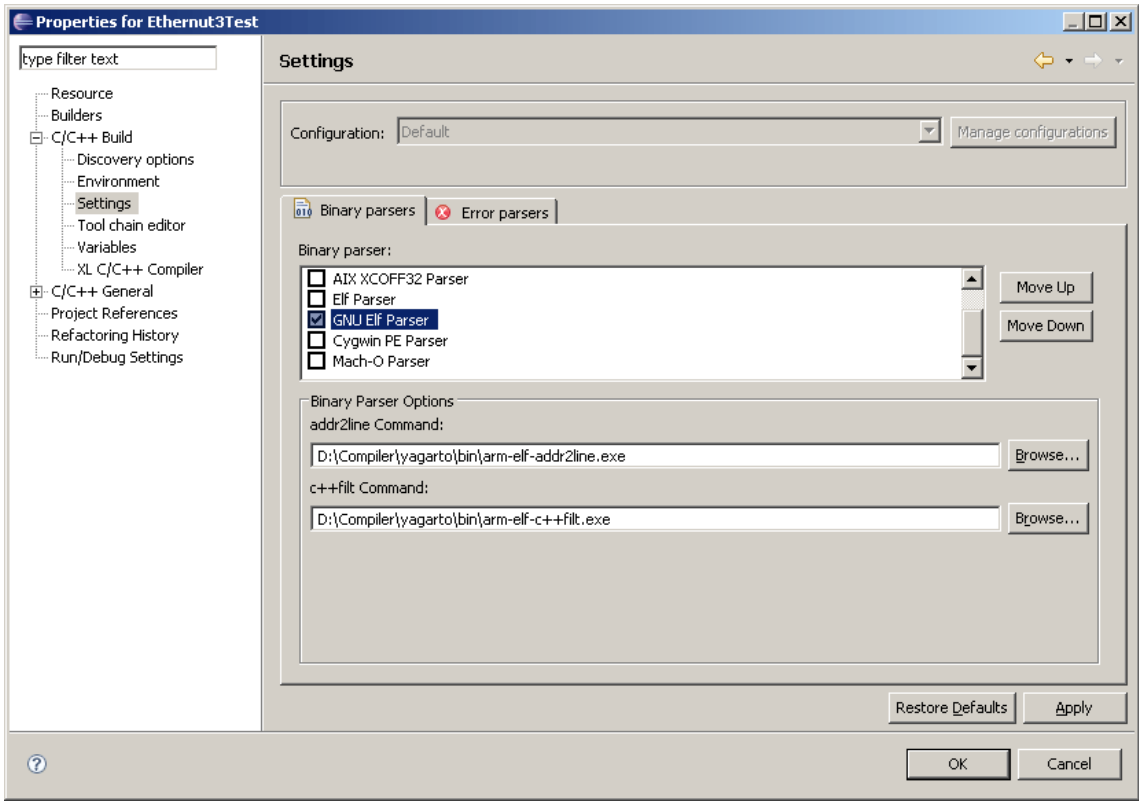
Obr. 13: Vytvoření nového projektu

V záložce *Project* je vhodné vypnout položku *Build automatically*, aby se projekt bez svolení nepřekládal, protože na starších počítačích trvá překlad i několik sekund. Nyní je potřeba v projektu nastavit, aby se pro kompilaci použilo Yagarto GCC, to se provádí v záložce *Project/Properities*, kde se zvolí položka *C/C++ Build/Discovery Options*. Do položky *Compiler invocation comand* se zadá cesta k souboru *arm-elf-gcc.exe*, který se nachází ve složce s nainstalovaným Yagartem. Položku *Load build from output file* je vhodné nechat prázdnou (Obr. 14) [6].



Obr. 14: Nastavení kompilátoru v Eclipse - Discovery options

V záložce *C/C++ Build/Setting/Binary parsers* se musí zvolit pouze *GNU Elf Parser*, po rozevření této položky je také potřeba nastavit cestu do složky Yagarta pro soubory *arm-elf-addr2line.exe* a *arm-elf-c++filt.exe* (Obr. 15) [6].



Obr. 15: Nastavení kompilátoru v Eclipse – Settings

Nyní je projekt nastaven a lze začít s programováním. V levém sloupci (Project explorer) se můžou vytvářet nové soubory a složky a obsahuje mnoho funkcí, které usnadňují práci s projektem, mimo jiné i nástroj Import. Aby bylo možné vysvětlit práci s Eclipse a jeho komponenty, je v dalších částech textu předpokládáno úspěšné naimportování vytvořeného projektu z přiloženého CD-ROM disku. Samotný projekt a jeho jednotlivé funkce jsou vysvětleny v poslední kapitole.

Hotový projekt je možné zkompilovat příkazem *Project/Build All*. Z dříve nastavených vlastností bude ke kompilaci použit GCC toolchain Yagarto. K sestavení projektu slouží soubor makefile, kde jsou definovány všechny vstupní a výstupní vazby. Podle jeho konfigurace by měly vzniknout objektové soubory, ale také velice důležitý binární soubor (main.bin), který se zapisuje do flash paměti procesoru a systémový soubor (main.out) využívaný pro on-chip ladění aplikací [14].

5.3 Zápis do flash paměti procesoru

Most mezi počítačem a procesorem tvoří JTAG Olimex ARM-USB-TINY, o řízení přenosu dat se stará program OpenOCD, který je možné implementovat přímo do IDE.

V Eclipse je připraven nástroj, který ovládá externí zařízení. Jeho konfiguraci je možné provést v záložce *Run/External Tools/External Tools configuration*, kde se v levé části nachází tlačítko *New launch configuration* pro vytvoření nové konfigurace. Pro pozdější orientaci je doporučeno zvolit název například USB TINY, ale především nastavit propojení s OpenOCD programem (Obr. 16) [6].

Location:

```
C:\ARM\openocd\bin\openocd-ftd2xx.exe
```

V tomto poli je žádána přímá cesta k programu, který zajistí spojení s procesorem.

Working directory:

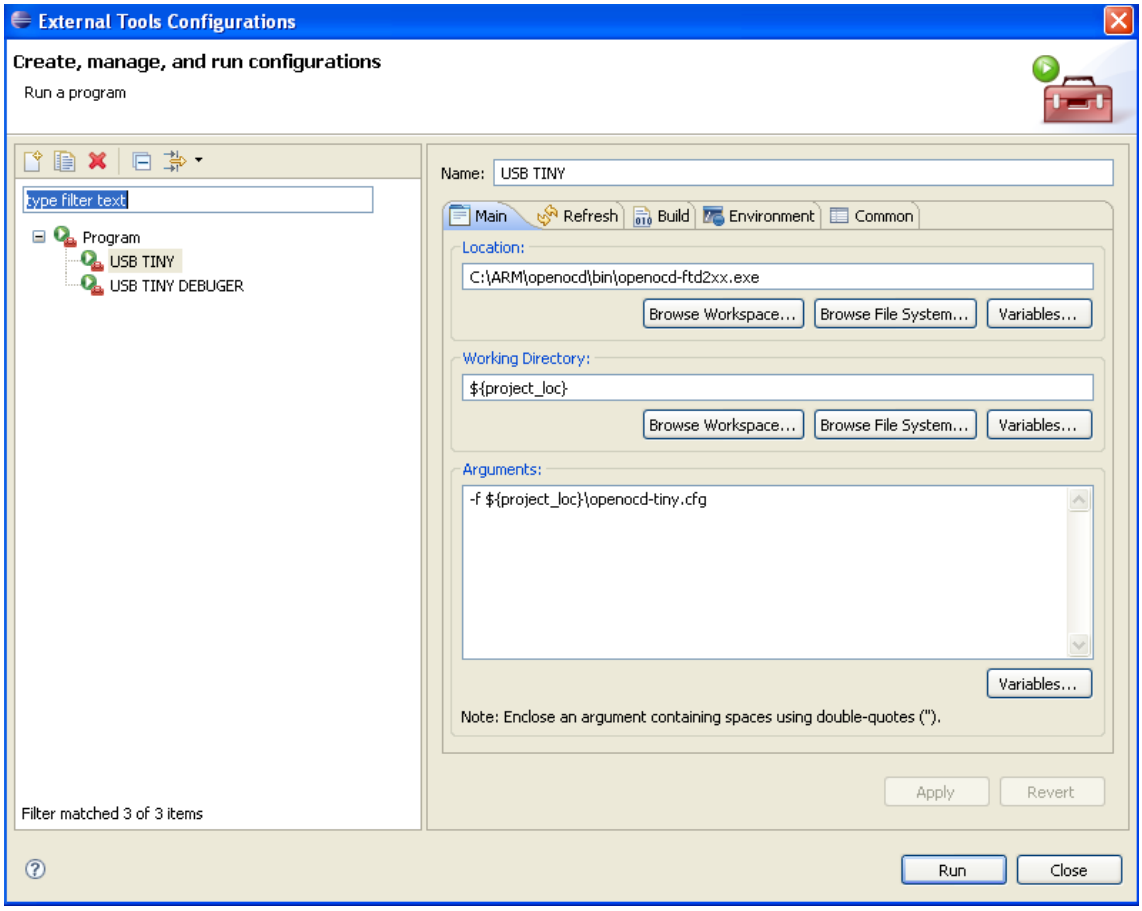
```
${project_loc}
```

Pracovní adresář, ve kterém má program OpenOCD hledat například binární soubor pro přenos do procesoru. Je tu možné zadat absolutní cestu a nebo využít jednu z automatických proměnných v Eclipse. Pokud je zvolena, je potřeba při použití této externí komponenty dodržet otevření projektu. Například v editačním okně musí být zobrazen soubor main.c a kurzor umístěn uvnitř, jinak tato procedura neproběhne a Eclipse vypíše chybovou hlášku.

Arguments:

```
-f ${project_loc}\openocd-tiny.cfg
```

Argument *-f* slouží k připojení konfiguračního souboru, ve kterém jsou příkazy pro OpenOCD řídící celé spojení s procesorem. Proměnná ${project_loc} je vysvětlena výše a směřuje do adresáře, kde je umístěn projekt [14].



Obr. 16: Eclipse – konfigurace externích zařízení

Nyní je konfigurace v programu Eclipse kompletní, pokud se použije tlačítko *Run*, proběhne připojení k procesoru a provedou se všechny úkony, které jsou předefinovány v konfiguračním souboru. Externí komponentu je možné spouštět i automaticky po překladu projektu, ale je potřeba v souboru makefile nastavit správné údaje pro spuštění. Parametry jsou předpřipraveny na přiloženém CD-ROM disku.

5.3.1 Konfigurační soubor pro OpenOCD

openocd-tiny.cfg:

```
#daemon configuration
telnet_port 4444
gdb_port 3333
```

Nastavení komunikačního kanálu.

```
#interface
interface ft2232
ft2232_device_desc "Olimex OpenOCD JTAG TINY A"
```

```
ft2232_layout "olimex-jtag"
ft2232_vid_pid 0x15BA 0x0003
jtag_speed 2
jtag_nsrst_delay 200
jtag_ntrst_delay 200
```

Je důležité zvolit správný název používaného JTAGu v celém jeho znění, které je uvedeno v dokumentaci OpenOCD. Dále je možné nastavení rychlosti přenosu. Lze ji zvětšit na rychlost tři, ale pro spolehlivější přenos je doporučeno použít tato nastavení. Rychlost se zásadní měrou podílí na době programování procesoru. Při výskytech chyb při přenosu je možné zvýšit dobu zpoždění.

```
#use combined on interfaces or targets that can't set TRST/SRST
separately
reset_config srst_only srst_pulls_trst
#jtag scan chain
#format L IRC IRCM IDCODE (Length, IR Capture, IR Capture Mask,
IDCODE)
jtag_device 4 0x1 0xf 0xe
#target configuration
daemon_startup reset
#target
#target arm7tdmi
target arm7tdmi little run_and_init 0 arm7tdmi
run_and_halt_time 0 30
```

V popisované části se provádí další nastavení JTAGu, je zvolen typ jádra, se kterým se pracuje.

```
# flash-options AT91
target_script 0 reset sam7flash.script
working_area 0 0x00200000 0x4000 nobackup
flash bank at91sam7 0 0 0 0 0
```

Tyto příkazy provádí nastavení flash paměti procesoru a importují script, který provede reset procesoru, ve kterém je možné provést zápis do paměti. Musí být nastaven rozsah paměti podle zvoleného typu CPU. Parametry se vztahují k AT91SAM7X256.

sam7flash.script:

```
halt
```

```
sleep 10

mww 0xfffffd44 0x00008000   # disable watchdog
mww 0xfffffd08 0xa5000001   # enable user reset
mww 0xfffffc20 0x00000601   # CKGR_MOR : enable the main oscillator
sleep 10
mww 0xfffffc2c 0x00481c0e   # CKGR_PLLR: 96.1097 MHz
sleep 10
mww 0xfffffc30 0x00000007   # PMC_MCKR : MCK = PLL / 2 ~= 48 MHz
sleep 10
mww 0xffffff60 0x003c0100   # MC_FMR: flash mode (FWS=1,FMCN=60)
```

Po probuzení procesoru následuje zpoždění, poté se v postupných krocích provede reset s nastavením oscilátoru na 48 MHz, aby bylo možné provést zápis do paměti.

```
arm7_9 dcc_downloads enable
sleep 10
poll

#flash probe 0       #find blash
#flash info 0       #find information about uProcesor
flash protect 0 0 15 off
flash erase 0 0 15
flash write 0 main.bin 0x0

reset run
sleep 10
shutdown
```

Pokud bude vyžadováno on-chip ladění, tak se po resetu neprovádí už žádné další kroky, o zbytek se postará GDB komponenta programu Yagarto a další konfigurační soubor. Před zápisem do flash paměti je nejprve potřeba vypnout její ochranu. Poté se provede vymazání a následně zápis binárního souboru. *Flash probe* vypíše oblast se zablokovanými bity, *flash info* zobrazí informace o CPU. Vhodné například pro odzkoušení spojení s procesorem nebo zjištění volné kapacity. V novějších verzích programu OpenOCD se změnila syntaxe některých příkazů a je nutné použít například write_image a erase_sector, starší už nejsou podporovány (více na [16]).

5.4 On-chip debugger

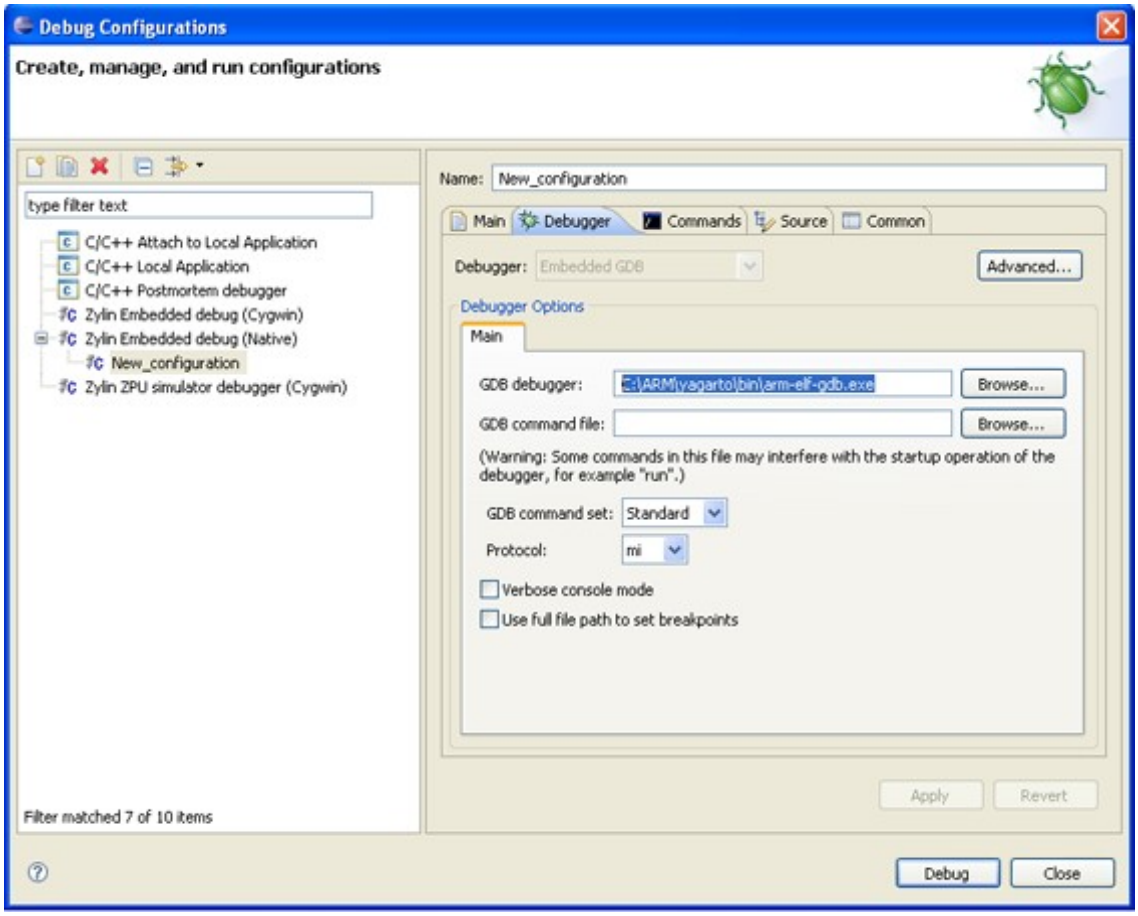
Procesory s jádrem ARM7 podporují on-chip ladění programů. V následujících řádcích je popsána základní obsluha v programu Eclipse s GDB ovladači Yagarto. Pro spojení s procesorem se opět použije program OpenOCD a JTAG Olimex ARM-USB-TINY. Před začátkem práce je důležité mít nainstalovaný výše zmíněný Zyling plugin pro Eclipse.

V Eclipse je potřeba vytvořit v nástroji *External Tools* novou položku, která bude zajišťovat spojení s procesorem. Provádí se to stejně jako v předchozím případě pro zápis do flash paměti (Obr. 16) [6]. Jediným rozdílem je jiný konfigurační script, který je nazván *openocd-tiny-gdb.cfg*. Resetovací script bude naprosto stejný, jako pro zápis do paměti, ale s tím rozdílem, že je vyžadováno pouze vytvoření spojení. Proto je importován odlišný soubor.

```
target_script 0 reset sam7flash-gdb.script
```

Místo původního souboru je importován *sam7flash-gdb.script*, který obsahuje pouze první část pro reset z původního souboru a*m7flash.script*. Důležité je, že neobsahuje příkaz *shutdown*, který ukončí spojení. To má za následek, že po vykonání instrukcí, zůstane spojení aktivní a pomocí debuggru jsou ve spolupráci s a*rm-elf-gdb.exe* vykonávány další instrukce související s ovládáním procesoru.

Tlačítko pro přepnutí do debug perspektivy se nachází v pravém horním rohu Eclipse (Obr. 18) [6]. V záložce *Run/Debug configuration* je možné zvolit *Zyling Embedded debug (Native)* a vytvořit pomocí tlačítka *New lounch configuration* novou konfiguraci. V záložce Main je vybrán projekt, ke kterému se má vztahovat nastavení a také vstupní soubor (Obr. 17) [6]. V tomto případě se jedná o *main.out*. Yagarto podporuje i jiné vstupy, například *.elf*. V další záložce *Debugger* je potřeba nastavit do položky GDB Debugger cestu k souboru, který se bude starat o chod. Tento nástroj je součástí Yagarto a cesta bude vypadat například takto: *C:\ARM\yagarto\bin\arm-elf-gdb.exe*. Položka *GDB command file* se nevyplňuje.



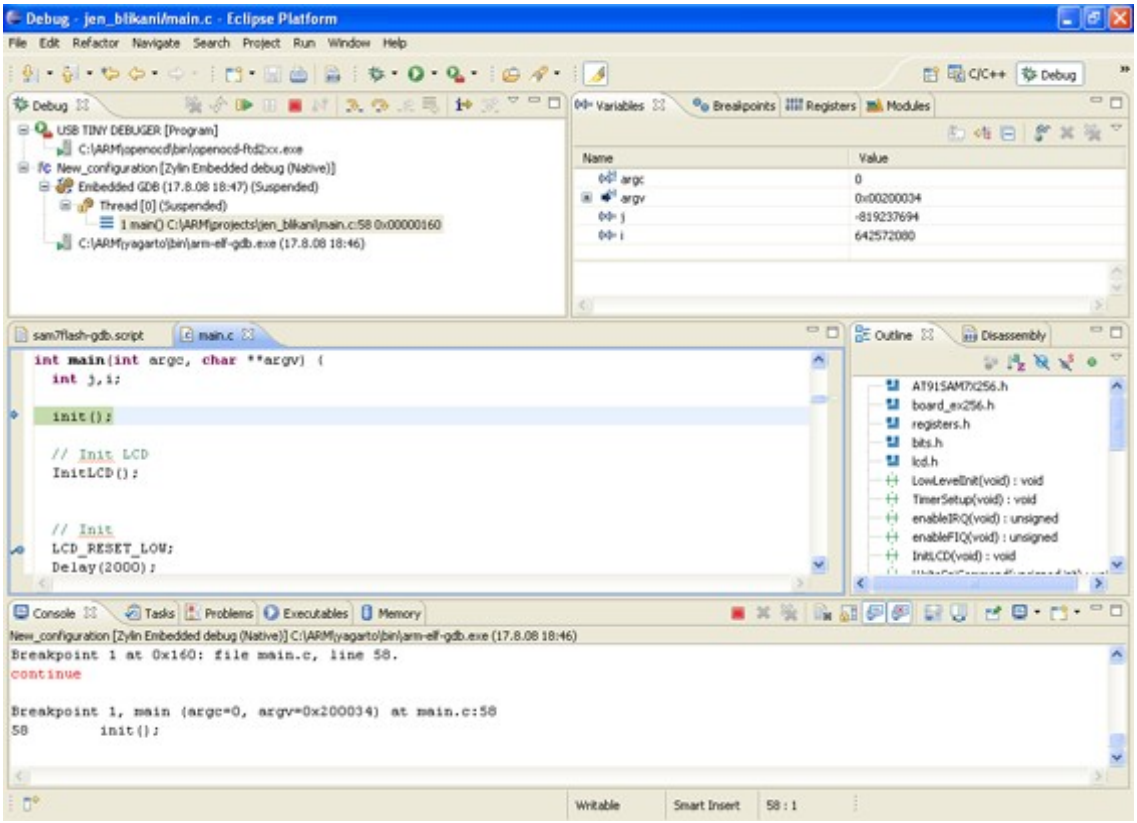
Obr. 17: Eclipse – Eclipse -Debug Configuration

V záložce *Commands* je potřeba do položky *Initialize commnands* vložit spouštěcí kód pro on-chip ladění.

```
target remote localhost:3333
monitor reset
monitor sleep 500
monitor poll
monitor soft_reset_halt
monitor arm7_9 force_hw_bkpts enable
# WDT_MR, disable watchdog
monitor mww 0xFFFFFD44 0x00008000
# RSTC_MR, enable user reset
monitor mww 0xfffffd08 0xa5000001
# CKGR_MOR
monitor mww 0xFFFFFC20 0x00000601
```

```
monitor sleep 10
# CKGR_PLLR
monitor mww 0xFFFFFC2C 0x00481c0e
monitor sleep 10
# PMC_MCKR
monitor mww 0xFFFFFC30 0x00000007
monitor sleep 10
# PMC_IER
monitor mww 0xFFFFFF60 0x00480100
monitor sleep 100
# needed for gdb 6.8 and higher
set mem inaccessible-by-default off
load
break main
continue
```

Tento script provede základní nastavení propojeni GDB s OpenOCD a nastaví nejnutnější parametry v procesoru, vypne Watchdog, provede uživatelský reset a přenastaví oscilátor CPU. Vše je zakončeno příkazem *continue*, což znamená, že příkazy budou dále pokračovat, o ně už se stará GDB nástroj Yagarta podle požadavků na debuging. Po provedení všech scriptů, se nastaví ukazatel na první řádek programu (Obr. 18).



Obr. 18: Eclipse – Debug perspektiva

Debug perspektiva se skládá z několika částí (Obr. 18), v záložce Debug jsou zobrazeny všechny připojené periferie. V první odrážce je možné vidět aktivní spojení s OpenOCD a ve druhé fungující GDB komponentu, která ovládá on-chip ladění. U jednotlivých periferií je indikace stavu, zelená šipka znamená bezproblémový chod, pokud se zobrazí červený obdélník, vyskytla se chyba.

V pravé horní části jsou záložky, kde se zobrazují aktuální stavy proměnných, výpis registrů a také výpis aktivních breakpointů. V pravé části je možné zobrazit strukturu projektu, jako jsou vložené soubory *.h* nebo výpis používaných funkcí, ale také ASM výpis právě prováděného příkazu. V prostřední části se nachází zdrojový kód, zelená linka značí aktuální pozici v programu.

Například pomocí *Run/Step into*, se dá program krokovat a sledovat všechno dění v okolních blocích. Breakpoint se vloží přímo do programu pomocí *Run/Toggle Breakpoint*. Po skončení on-chip ladění je vhodné všechny procesy řádně ukončit a uzavřít spojení s procesorem (více na [14]).

6 Program pro vývojovou desku Olimex SAM7-EX256

V této části práce je popsán vzorový program pro mikroprocesor AT91SAM7X256, který je osazen na vývojové desce Olimex SAM7-EX256. Celý zkompilovaný projekt se nachází na přiloženém CD-ROM disku.

Program je pro větší přehlednost rozdělen do několika souborů, každý z nich má na starosti jinou část obsluhy procesoru. Při kompilaci jsou spojeny do výsledného výstupu podle pravidel, která jsou uvedena v souboru makefile. Možnosti tohoto procesoru jsou velice rozsáhlé, proto zde není možné vše předvést na příkladech. Všechny informace jsou podrobně popsány v několika dokumentacích od firmy Atmel. Program na přiloženém CD-ROMu demonstruje ty nejvíce používané funkce, bez kterých se často neobejde žádný projekt. Je to ovládání vstupů a výstupů na portech procesoru, časové přerušení a samozřejmě prvotní nastavení, které je potřeba provést pro správnou činnost CPU.

Časové přerušení je nastaveno na 50 ms, kde se inkrementuje proměnná, po dosažení 1 s se změní úroveň výstupního portu, což má za následek blikání displeje. Mezitím jsou čteny vstupní porty, na které jsou připojena tlačítka, jimiž můžeme vypnout nebo zapnout zápis do výstupních registrů [2].

6.1 Struktura programu

Jak již bylo popsáno, program se skládá z mnoha souborů, které jsou nejčastěji napsány v jazyce C nebo v assembleru. Základem všeho jsou knihovny pro ovládaní všech vlastností procesoru. Soubor se jmenuje *AT91SAM7X256.h*. Je volně dostupný na webových stránkách firmy Atmel a uzpůsobený pro GNU nástroje, jako je například GCC. Jedná se o velice rozsáhlý soubor, který nám definuje a pojmenovává všechny adresy na sběrnici procesoru. Například pro ovládání portu je potřeba na určité adrese nastavit hodnotu, ta se pak zapíše do výstupního registru.

```
#define AT91C_BASE_PIOA      ((AT91PS_PIO)     0xFFFFF400) // (PIOA)
Base Address

typedef struct _AT91S_PIO {
     AT91_REG    PIO_SODR; // Set Output Data Register
     AT91_REG    PIO_CODR; // Clear Output Data Register
```

```
    ...
} AT91S_PIO, *AT91PS_PIO;
```

Proměnná AT91C_BASE_PIOA, které je přidělena adresa pro komunikaci po sběrnici. Je přetypovaná na AT91PS_PIO, což je pointr na strukturu, kde jsou nadefinovány všechny funkce proveditelné s výstupním registrem. Například PIO_SODR, která slouží k nastavení vysoké úrovně portu. Vše přináší jednoduché ovládaní všech vlastností a nastavení procesoru.

V souboru *register.h* jsou funkce z předchozího souboru nadefinovány.

```
AT91PS_PIO    pPIOA  = AT91C_BASE_PIOA;
...
```

Je vytvořena proměnná pPIO, které je přidělena hodnota z AT91C_BASE_PIOA, ta ovládá registr pro port A.

V souboru *bits.h* jsou definovány jednotlivé bity portu, aby se mohly ovládat samostatně.

Pokud je vše nastaveno, můžeme nyní jednoduše ovládat výstupní registr.

```
pPIOA->PIO_OER    = BIT20;    // Configure PA20 as output
pPIOA->PIO_SODR   = BIT20;    // Set PA20 to HIGH
```

Pro ukázku jsou uvedeny dva příklady, první nastaví port A BIT20 (PA20) jako výstupní. Druhý příkaz přepne PA20 na vysokou úroveň. Tímto způsobem se ovládají všechny periferie a nastavení.

Program také obsahuje soubor *board_ex256.h*, který určuje velikost paměti RAM a flash paměti v procesoru a taktovací frekvenci vnitřního oscilátoru. Soubor je volně dostupný na webových stránkách firmy Atmel.

Soubor *crt.S* je napsán v assembleru, tedy nejnižším programovacím jazykem pro mikroprocesory. Je to spouštěcí kód, který všechny neinicializované proměnné nastaví na nulu a tím umožňuje main() v jazyce C používat jako hlavní funkci pro běh programu. Jsou zde definovány vektory přerušení, což obdobně slouží ve spojení s GCC k využití funkce pro obsluhu přerušení. V praxi to znamená, že po přetečení registru časového přerušení program pokračuje na funkci Timer0IrqHandler().

Isrsupport.c slouží k ovládání přerušení. Například k jeho vypnutí či zapnutí. V *lowlevelinit.c* se nastavuje taktovací frekvence procesoru a základní vstupní obslužné vektory. *Timersetup.c* slouží k detailnímu nastavení časového přerušení. V *timerisr.c* je

funkce pro vykonání přerušení a nakonec prioritní blok celého projektu *main.c*, ve kterém jsou všechny hlavní vykonávané příkazy.

Projekt ještě obsahuje další soubory jako *libc.a, libgcc.a, libm.a***,** což jsou knihovny s funkcemi podporované v GCC. Je možné je stáhnout ze složky s nainstalovaným programem Yagarto. Soubory s koncovkou *.cfg* a *.script* jsou vysvětleny v předchozích kapitolách zaměřených na OpenOCD. *Demo_at91sam7_blink_flash.cmd* definuje pozice jednotlivých pamětí v procesoru [4].

6.2 Nastavení hlavního oscilátoru procesoru

Před začátkem používání procesoru je potřeba nastavit jeho taktovací frekvenci. Krystal, který je připojen, má frekvenci 18,432 MHz. I přesto s tímto krystalem dokáže procesor běžet na frekvencích kolem 50 MHz. Má vestavěnou děličku i násobičku kmitočtu fázovým závěsem, pomocí které je možné kmitočet softwarově ovládat zápisem do stanovených registrů. Po restartu procesor běží z vestavěného napětím řízeného oscilátoru, pro zapnutí hlavního je nutné provést postupně několik kroků.

```
AT91C_BASE_MC->MC_FMR = ((AT91C_MC_FMCN)&(50 <<16)) |
AT91C_MC_FWS_1FWS; // Set internal oscilator
AT91C_BASE_WDTC->WDTC_WDMR= AT91C_WDTC_WDDIS; // Watchdog Disable
```

První příkaz nastaví vnitřní oscilátor na kmitočet 30 MHz, aby byly proveditelné následující kroky. Modifikace bude trvat 50 cyklů zhruba 1 us. Před zapnutím hlavního oscilátoru je potřeba vypnout WatchDog.

```
pPMC->PMC_MOR = (( AT91C_CKGR_OSCOUNT & (0x06 <<8) | AT91C_CKGR_MOSCEN
)); // 1 Enabling the Main Oscillator
while(!(pPMC->PMC_SR & AT91C_PMC_MOSCS)); // Wait the startup time
```

Příkaz spustí hlavní oscilátor procesoru, proces trvá 56 strojových cyklů (8 x 6). Jelikož z předchozího kroku je nastavena frekvence na 30 MHz, jeden strojový cyklus proběhne za 30,51 us a celé zapnutí oscilátoru se projeví po 1,46 ms. Následující cyklus čeká na dokončení nastavení, které provede zápis do PMC_SR registru.

```
pPMC->PMC_PLLR = ((AT91C_CKGR_DIV & 14) |
                      (AT91C_CKGR_PLLCOUNT & (10<<8)) |
                     (AT91C_CKGR_MUL & (72<<16)));
```

V tomto kroku se zapne vstup z vnějšího krystalu a je možné nastavit taktovací frekvenci hlavního oscilátoru. K výpočtu nám slouží jeden dělící registr, který můžeme

zvolit od 0 – 255. Jeden násobící registr, ke kterému je připočtena hodnota 1, použitelný rozsah je 0 – 2047. To umožňuje velice variabilní a přesné nastavení taktovací frekvence. Do dělícího registru je zadáno číslo *DIV = 5* a do násobícího *MUL = 25+1*. Frekvence krystalu je 18,432 MHz, kterou vydělíme pěti a vynásobíme dvaceti šesti. Výsledkem je frekvence 95,8464 MHz. Opět následuje čekací cyklus na provedení změny.

```
pPMC->PMC_MCKR = AT91C_PMC_CSS_PLL_CLK | AT91C_PMC_PRES_CLK_2;
```

Jelikož s touto frekvencí nemůže procesor pracovat, je na výstupu vestavěna ještě jedna dělička frekvence dvěma. Tu inicializujeme tímto příkazem. Výsledný taktovací kmitočet procesoru bude 47,923 MHz [5].

6.3 Nastavení časového přerušení

Mikroprocesor AT91SAM7X256 má několik časových přerušení, zde je popsáno nejčastěji používané IRQ. V postupných krocích a zápisem do několika registrů je čas přerušení nastaven na 50 ms. Před začátkem používaní je potřeba správně nastavit vektor tohoto přerušení, což se děje v souboru *crt.S*.

Před použitím je nejprve potřeba zapnout hodiny pro časovač 0 a poté zapnout AIC výstupní registr také pro časovač 0. Toto nastavení je provedeno ve funkci init() na začátku celé inicializace.

Je vytvořena funkce TimerSetup(), která se nachází v souboru *timersetup.c*, kde je proveden zápis do registrů, určující detailní konfiguraci přerušení.

```
pTCB->TCB_BCR = 0;      // SYNC trigger not used
```

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	SYNC

SYNC = 0 – vypnuto ←
1 – zapnuta synchronizace přerušení

V registru TCR_BCR je možné nastavit pouze jeden bit a to synchronizaci všech tří přerušení. Jelikož je použito pouze jedno časové přerušení, je nastavena do registru nula.

```
pTCB->TCB_BMR = 0x15;   // external clocks not used
```

7	6	5 4	3 2	1 0
-	-	TC2XC2S	TC1XC1S	TC0XC0S

TC0XC0S = 00 – TLCK0 (PA4)
01 – vypnuto ←
10 – TIOA1 (PA15)
11 – TIOA2 (PA26)

Přerušení je možné řídit třemi externími hodinami. V tomto projektu chceme použít vlastní časování procesoru. Pro blok časovače nula se nastaví binární hodnota na 01b. Tím je zvolen jako zdroj hodin vnitřní oscilátor. Do bloku pro ostatní časovače je zapsána stejná hodnota.

```
pTC->TC_CCR = 0x5;      // enable the clock and start it
```

7	6	5	4	3	2	1	0
–	–	–	–	–	SWTRG	CLKDIS	CLKEN

CLKEN = 0 – bez efektu
1 – zapnutí hodin časovače ←

CLKDIS = 0 – bez efektu ←
1 – vypnutí hodin časovače

SWTRG = 0 – bez efektu
1 – spuštění hodin časovače ←

V tomto kroku jsou zapnuty a spuštěny hodiny časovače.

```
pTC->TC_CMR = 0x4004;   //TC Channel Mode Register
```

23	22	21	20	19	18	17	16
–	–	–	–	LDRB		LDRA	
15	14	13	12	11	10	9	8
WAVE	CPCTRG	–	–	–	ABETRG	ETRGEDG	
7	6	5	4	3	2	1	0
LDBDIS	LDBSTOP	BURST		CLKI	TCCLKS		

Nastavení hodin:

TCCLKS = 000 – TIMER_CLOCK1 (MCK/2 = 24027420 hz)
001 – TIMER_CLOCK2 (MCK/8 = 6006855 hz)
010 – TIMER_CLOCK3 (MCK/32 = 1501713 hz)
011 – TIMER_CLOCK4 (MCK/128 = 375428 hz)
100 – TIMER_CLOCK5 (MCK/1024 = 46928 hz) ←
101 – XC0
101 – XC1
101 – XC2

Byla zvolena kombinace 100b, což podle předchozí deklarace MCK zajišťuje kmitočet 46,928 kHz.

Výběr typu inkrementace zásobníku časového přerušení:

CLKI = 0 – inkrementace zásobníku na nástupnou hranu ←
1 – inkrementace zásobníku na sestupnou hranu

Zvolení externího vstupu pro časování:

BURST = 00 – vypnuto ←
01 – XC0
10 – XC1
11 – XC2

Nastavení chování po přetečení zásobníku časového přerušení:

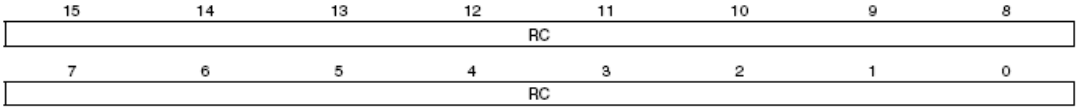
LDBSTOP = 0 – po přetečení nejsou hodiny zastaveny ←
1 – po přetečení jsou hodiny zastaveny

RC compare mód:

CPCTRG = 0 – bez efektu
1 – Spuštění RC Compare módu ←

Zde jsou popsány nejdůležitější vlastnosti tohoto registru, podrobnější popis všech vlastností je uveden v dokumentaci k mikroprocesoru AT91SAM7X256 [2].

```
pTC->TC_RC = 2346;      //TC Register C
```



Registr slouží k nastavení času pro vyvolání přerušení. Jelikož je v předchozích položkách nastavena frekvence na 46,928 kHz, perioda je 21.309239686 us. Pokud je požadováno přerušení 50 ms, je zřejmé, že zásobník musí byt nastaven na hodnotu 2346.

```
pTC->TC_IER = 0x10;     // enable RC compare interrupt
pTC->TC_IDR = 0xEF;     // disable all except RC compare interrupt
```

Toto jsou poslední dva registry, které je nutné nastavit pro správný běh časového přerušení. První zapne vnitřní RC oscilátor pro časový zásobník a druhý vypne externí řízení zásobníku.

Soubor nastaví základní vlastnosti přerušení, aby se s nimi mohlo pracovat. Je potřeba v hlavní části programu zapnout IRQ přerušení, což se provede voláním funkce enableIRQ(), která je opět dostupná na webových stránkách společnosti Atmel [4].

6.4 Ovládání vstupu a výstupu na portech procesoru

Jak už bylo výše naznačeno, k ovládání výstupu je potřeba správně nadefinovat pointry na struktury, které jsou uloženy v základní knihovně k procesoru AT91SAM7X256.

```
pSYS->PIOA_ASR = 0xffffffff;
pSYS->PIOB_ASR = 0xffffffff;
pSYS->PIOA_ODR = 0xffffffff;
pSYS->PIOB_ODR = 0xffffffff;
```

Příkazy umožní práci s okolními periferiemi.

```
pPIOB->PIO_OER    = BIT20;
pPIOB->PIO_SODR   = BIT20;
```

Tato definice nám nastaví PB20 jako výstupní port, ve druhém řádku se provádí zápis do stanoveného registru, který přepne PB20 na vysokou úroveň.

Pro čtení vstupu, v konkrétním případě tlačítek, které jsou umístěny na vývojové desce, je potřeba udělat několik následujících nastavení.

```
pPMC->PMC_PCER = 1 << AT91C_ID_PIOB;
```

Přiřazení zapne časování pro sběrnici POIB.

```
pPIOB->PIO_ODR    = 0xffffffff;
```

Celý port B je nastaven jako vstupní, je povolen zápis do registrů.

```
 pSYS->PIOB_PPUDR  = 0xffffffff;
```

Od portu B byly odstraněny pull-up rezistory.

```
pPIOB->PIO_ODR |= BIT24;
pPIOB->PIO_PER |= BIT24;
```

V prvním řádku je BIT24 nastaven jako vstupní a ve druhém je povolen zápis.

```
if(!((pPIOB->PIO_PDSR) & BIT24))
```

Nyní je vše nastaveno a pomocí tohoto jednoduchého příkazu je možné zjistit stavy jednotlivých portů. Pokud je na BIT24 nízká úroveň, podmínka je vyhodnocena jako kladná a program pokračuje na dalším řádku [5].

6.5 Makefile

V souboru jsou nadefinované jednotlivé vazby pro sestavení projektu.

```
main.o: main.c
```

```
	@ echo "..compiling"
	$(CC) $(CFLAGS) main.c
```

Na prvním řádku je pravidlo main.o, které je závislé na souboru main.c. Na druhém řádku je pouze výpis řetězce, slouží k indikaci právě prováděné činnosti. Na posledním řádku je pravidlo pro kompilaci. $(CC) je proměnná, která obsahuje řetězec *arm-elf-gcc*. V druhé proměnné je *-I./ -c -fno-common -O0 -g*. Po provedení toho příkazu vznikne za pomoci a*rm-elf-gcc.exe* soubor main.o, kompilace se bude řídit pravidly v proměnné $(CFFLAGS). Takto se zkompilují všechny soubory psané v jazyce C. Hlavičkové soubory se připojí automaticky.

```
crt.o: crt.S
	@ echo "..assembling"
	$(AS) $(AFLAGS) crt.S > crt.lst
```

Obdobně se zkompiluje i soubor napsaný v assembleru, jediným rozdílem je použití jiného souboru Yagarta a jiných pravidel, která se definují opět v proměnných.

```
main.out: $(OBJECTS) demo_at91sam7_blink_flash.cmd
	@ echo "..linking"
	$(LD) $(LFLAGS) -o main.out $(OBJECTS) libc.a libm.a libgcc.a
```

Tímto blokem příkazů se slinkují všechny soubory s koncovkou *.o* i v závislosti na mapě paměti procesoru. Do výsledného souboru jsou přidány i GCC knihovny. Vznikne main.out, ze kterého se poté může vygenerovat například binární soubor, mimo jiné se používá i jako řídící pro on-chip ladění.

```
all:  main.out
	@ echo "...copying"
	$(CP) $(CPFLAGS) main.out main.bin
```

Poslední pravidlo all, vytvoří ze slinkovaného main.out pomoci *arm-elf-objcopy.exe* soubor main.bin, který se používá pro zápis do flash paměti procesoru. Toto pravidlo má nejvyšší hodnotu a je voláno programem Eclipse při zahájení překladu.

```
Clean: -del $(OBJECTS) crt.lst main.lst main.out main.bin main.hex
main.map main.dmp
```

Opět pravidlo nejvyšší úrovně a slouží pro vymazání všech souborů vzniklých při překladu projektu. Je zde nepatrná změna proti Linuxové verzi, ze které soubor makefile původně pochází. Ve Windows je nutné pro mazání použít *-del* oproti původnímu *-rm* [4].

Závěr

Práce se zabývá vývojem projektů pro platformy s jádrem ARM a přináší obecné informace o jejich problematice. Hlavní důraz je kladen na programování mikroprocesoru AT91SAM7X256 ve volně šiřitelných aplikacích.

Výsledkem je souhrnný přehled nejdůležitějších informací, které jsou potřeba pro úspěšnou práci s těmito technologiemi. Užitečný bude především těm, kdo začínají s vývojem na technologii ARM a chtějí svého cíle dosáhnout s nejnižšími finančními náklady, protože propracované komerční balíčky jsou cenově velice nákladné. Uplatnění najde jak na akademické půdě, tak i u lidí, kteří mají již zkušenosti s jinými technologiemi, jako jsou například jednočipové procesory řady 8051, nebo procesory založenými na technologii AVR.

Tento materiál je unikátní v tom, že pro vybranou problematiku neexistuje komplexní přehled informací, který je ještě doplněn o názorný příklad. V úvodní části práce je obecné seznámení s ARM, jeho vznikem, popisem jednotlivých vlastností a rozdělením do vývojových řad. Větší pozornost je věnována jádru ARM7, které využívá i procesor AT91SAM7X256. V další části je popsána vývojová deska a JTAG, který zprostředkovává komunikaci mezi počítačem a mikroprocesorem. Následující kapitoly se věnují praktickému použití softwarových nástrojů, jejich nastavení a vzájemnému propojení. Jsou zde popsány základní ovládací prvky, jako například vytvoření nového projektu či propojení s GNU nástroji pro kompilaci. Předposlední část představuje způsoby komunikace počítače s mikroprocesorem. Jedná se o zápis do flash paměti, ovládání a nastavení pro on-chip ladění programů. V poslední kapitole je proveden rozbor nejdůležitějších částí programu.

Většina původních cílů byla splněna, práce obsahuje téměř všechny požadavky, které byly před samotnou implementací vzneseny. Zadání bylo obsahově velice široké, a proto nebylo možné obsáhnout detailně všechny vlastnosti programů či mikroprocesoru. Dokumentace pro každý jednotlivý software a hardware většinou mnohonásobně přesahuje rozsah této bakalářské práce. Je zde především naznačeno, jak začít pracovat s procesory obsahujícími ARM jádra za pomoci open-source programů.

Zdroje

[1] SOLSS, SYMES, *WRIGHT, ARM systém Developers Guide*. 2004 Elsevier Inc. 703 s. ISBN 1-55860-874-5.

[2] *Datasheets – Atmel AT91SAM7X256* [online]. 2007 [revision G] 671 s. Dostupný z WWW: <http://atmel.com/dyn/resources/prod_documents/doc6120.pdf>

[3] *ARM7TDMI Technical Reference Manual* [online]. 2007 284 s. Dostupný z WWW: <http://atmel.com/dyn/resources/prod_documents/DDI0029G_7TDMI_R3_trm.pdf>

[4] James P. LYNCH, *Using Open Source Tools for AT91SAM7 Cross Development* [online]. 2007 [revision C] 198 s. Dostupný z WWW: <*http://www.at91.com/Tool/Controleurs/cVisualisation.php?idVisualisation=212*>

[5] Demo programing code for AT91SAMTX [online]. Dostupný z WWW: <http://olimex.com/dev/soft/arm/SAM7/SAM7_EX256.zip>

[6] Eclipse wiki [online]. Dostupný z WWW: <http://wiki.eclipse.org/Main_Page>

[7] ARM documentation [online]. Dostupný z WWW: <http://infocenter.arm.com/help/index.jsp>

[8] ARM Product Backgrounder [online]. Vydáno 1. 1. 2005. Dostupný z WWW: <http://www.arm.com/miscPDFs/3823.pdf>

[9] ARM CORPORATE BACKGROUNDER [online]. Vydáno 1. 8. 2005. Dostupný z WWW: <http://www.arm.com/miscPDFs/3822.pdf>

[10] Intel, The ARM Architecture [online]. Dostupný z WWW: <ftp://download.intel.com/education/highered/Embedded/

02_ARM_Architecture.ppt>

[11] Nikolaj KOROLJOV, 32bitové procesory Atmel ARM7 [online]. Vydáno 6. 11. 2006. Dostupný na WWW: <http://www.kit-e.ru/articles/micro/2006_06_42.php>

[12] ARM-USB-TINY [online]. 1999-2008. Dostupný z WWW: <http://olimex.com/dev/arm-usb-tiny.html>

[13] SAM7-EX256 [online]. 1999–2008. Dostupný z WWW: <http://olimex.com/dev/sam7-ex256.html>

[14] Yagarto [online]. Dostupný z WWW: <http://yagarto.de/>

[15] GnuWin32 [online]. Dostupný z WWW: <http://gnuwin32.sourceforge.net/>

[16] OpenOCD [online]. Dostupný z WWW: <http://openocd.berlios.de/web/>

Přílohy

Obsah přiloženého CD-ROM disku

Na CD-ROM disku jsou tři složky. První obsahuje zkompilovaný vzorový projekt se zdrojovými soubory. Ve druhé složce se nachází všechny instalované vývojové programy. Poslední obsahuje vypracovanou bakalářskou práci v elektronické podobě, ve formátu PDF.