

Programování mikropočítačů Freescale HCS08 pomocí volně dostupných nástrojů

Michal Hrdý

Bakalářská práce
2014



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2013/2014

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Michal HRDÝ**
Osobní číslo: **A10795**
Studijní program: **B3902 Inženýrská informatika**
Studijní obor: **Informační a řídicí technologie**
Forma studia: **prezenční**

Téma práce: **Programování mikro počítačů Freescale HCS08 pomocí volně dostupných nástrojů**

Zásady pro vypracování:

1. Vyhledejte a popište dostupná integrovaná vývojová prostředí, případně samostatné překladače a ladící programy pro mikro počítače Freescale s jádrem HCS08 se zaměřením na jazyky C a C++.
2. Vyberte jedno volně dostupné prostředí nebo navrhnete vlastní prostředí složené ze samostatných nástrojů.
3. Vytvořte podrobný návod k použití zvoleného prostředí.
4. V tomto prostředí vytvořte také ukázkové programy a návod k jejich přeložení, nahrání do mikro počítače a ladění.

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. BARR, Michael a Anthony J MASSA. Programming embedded systems. 2nd ed. Sebastopol: O'Reilly, 2006, xxi, 301 s. ISBN 978-0-596-00983-0.
2. CATSOULIS, John. Designing Embedded Hardware. Sebastopol: O'Reilly Media, 2005. ISBN 978-0-596-00755-3.
3. MANN, Burkhard. C pro mikrokontroléry: ANSI-C, kompilátory C, spojovací programy – linkery, práce s ATMELE AVR a MSC-51, příklady programování v jazyce C, nástroje pro programování, tipy a triky. Praha: BEN, 2003. ISBN 80-730-0077-6.
4. PINKER, Jiří. Mikroprocesory a mikropočítače. Praha: BEN – technická literatura, 2004. ISBN 80-730-0110-1.
5. VÁŇA, Vladimír. Začínáme pracovat s mikrokontroléry HC08 NITRON: příručka pro naprosté začátečníky. 1. vyd. Praha: Grada, 2003, 95 s. Knihovna programátora. ISBN 80-730-0124-1.

Vedoucí bakalářské práce:

Ing. Jan Dolinay, Ph.D.

Ústav automatizace a řídicí techniky

Datum zadání bakalářské práce:

28. února 2014

Termín odevzdání bakalářské práce:

13. června 2014

Ve Zlíně dne 28. února 2014

prof. Ing. Vladimír Vašek, CSc.
děkan



prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně 11.6.2014

.....
podpis diplomanta

ABSTRAKT

Tato bakalářská práce se zabývá zdarma dostupnými vývojovými nástroji pro mikropočítače Freescale řady HC(S)08. Úvod obsahuje nastínění historie mikroprocesorů společně se základními pojmy. Dále se pak věnuje obecnému popisu dané řady mikropočítačů a jejich mikroprocesoru. Na konci teoretické části je pak vysvětlena činnost jednotlivých vývojových nástrojů při procesu tvorby programu. V praktické části je pak uveden popis dostupných vývojových prostředí včetně komerčních. Následuje popis nastavení volně dostupného překladače SDCC v Code Blocks IDE a popis práce v tomto prostředí včetně ukázkového programu.

Klíčová slova: mikropočítač, HC(S)08, vývojové nástroje

ABSTRACT

This bachelor thesis is focused on the free development tools for Freescale microcontroller series HC(S)08. Introduction contains a brief look on history of microprocessors together with the basic concepts. It then deals with the general description of microcontrollers and their microprocessor from the given series. At the end of the theoretical part is explained the activity of individual development tools in the process of making program. The practical part follows with a description of the available development environments including the commercial ones. Thesis then continues with a manual on how to set SDCC compiler in Code Blocks IDE and how to work with it, including the sample program.

Keywords: microcontroller, HC(S)08, development tools

Poděkování, motto

OBSAH

ÚVOD	9
I TEORETICKÁ ČÁST	10
1 MIKROPOČÍTAČE	11
1.1 HISTORIE.....	11
1.2 POJMY	12
2 MIKROPOČÍTAČE FREESCALE ŘADY HCS08	15
2.1 OBECNÝ POPIS ŘADY HCS08	15
2.2 PERIFERNÍ MODULY.....	16
2.3 CPU08	16
2.3.1 Vlastnosti.....	16
2.3.2 Programátorský model (registry CPU).....	17
3 PROGRAMOVÁNÍ MIKROPOČÍTAČE	21
3.1 PROGRAMOVACÍ JAZYK C	21
3.2 KOMPILOVÁNÍ, LINKOVÁNÍ A RELOKOVÁNÍ.....	22
3.2.1 Kompilování.....	23
3.2.2 Linkování	24
3.2.3 Startup kód	25
3.2.4 Relokace	26
3.3 DOWNLOAD A DEBUGOVÁNÍ.....	26
3.3.1 Debug monitory	26
3.3.2 Emulátory	27
3.3.3 Simulátory	28
II PRAKTICKÁ ČÁST	29
4 DOSTUPNÉ VÝVOJOVÉ NÁSTROJE	30
4.1 CODEWARRIOR.....	30
4.2 COSMIC	31
4.3 IMAGECRAFT ICC08.....	32
4.4 SDCC	33
5 VOLBA PROSTŘEDÍ	34
5.1 NASTAVENÍ SDCC V CODE BLOCKS IDE	34
6 PRÁCE S PROSTŘEDÍM	38
6.1 PARAMETRY PRO NASTAVENÍ SDCC.....	38
6.2 ZALOŽENÍ A NASTAVENÍ PROJEKTU	39
6.2.1 Projekty s více zdrojovými soubory.....	43
7 UKÁZKOVÝ PROGRAM	45
7.1 POPIS PROGRAMU	45
7.2 NASTAVENÍ CEST A PŘEKLAD	46
7.3 VÝSTUPNÍ SOUBORY KOMPILÁTORU	47
7.4 LADĚNÍ A NAHRÁNÍ PROGRAMU DO PAMĚTI MIKROPOČÍTAČE.....	48
ZÁVĚR	51

SEZNAM POUŽITÉ LITERATURY.....	52
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	53
SEZNAM OBRÁZKŮ	54
SEZNAM TABULEK.....	CHYBA! ZÁLOŽKA NENÍ DEFINOVÁNA.
SEZNAM PŘÍLOH.....	56

ÚVOD

V současné době jsou mikropočítače nejrozšířenějšími počítači v celosvětovém měřítku. Rozsah jejich nasazení je velmi široký. V nejjednodušších variantách se jich používá ke zvýšení "inteligence" takových zařízení, jako jsou výrobky spotřební elektroniky (radiopřijímač, televizor apod.), výrobky měřicí techniky (osciloskopy, voltmetry aj.), ale i výrobky v dalších odvětvích (řízení benzinových stojanů, automatů na cigarety apod.).

Každý takový mikropočítač vyžaduje vlastní sadu nástrojů specifickou pro daný typ mikropočítače. Ceny takových nástrojů se mohou pohybovat v řádech tisíců až desetitisíců a jsou tak pro nekomerční vývoj programů velkým nákladem.

Cílem bakalářské práce je proto seznámit se s dostupnými vývojovými nástroji pro mikropočítače řady HC(S)08 a z volně dostupných poté buď sestavit vlastní vývojové prostředí, nebo se zaměřit na použití jednotlivých nástrojů. K danému prostředí nebo jednotlivým nástrojům pak je uveden podrobný návod pro nastavení a použití. U všech použitých nástrojů platí, že jsou k dispozici zdarma. Dále obsahuje praktickou ukázkou použití s ukázkovým programem.

V teoretické části je popsána historie mikroprocesorů a to zejména od firem Intel a Motorola. Následuje část obsahující přehled základních pojmů mikroprocesorové techniky.

Další část obsahuje popis základních vlastností mikropočítačů řady HCS08, jejich periférií a CPU s registry. Teoretická část zabývající se programováním obsahuje popis procesu vzniku programu. Zaměřuje se na úkony prováděné vývojovými nástroji jako je překlad programu, linkování a relokační programu.

I. TEORETICKÁ ČÁST

1 MIKROPOČÍTAČE

V současnosti se jako mikropočítač označuje zařízení menších rozměrů než je osobní počítač a na rozdíl od nich plní většinou jednu danou úlohu.

1.1 Historie

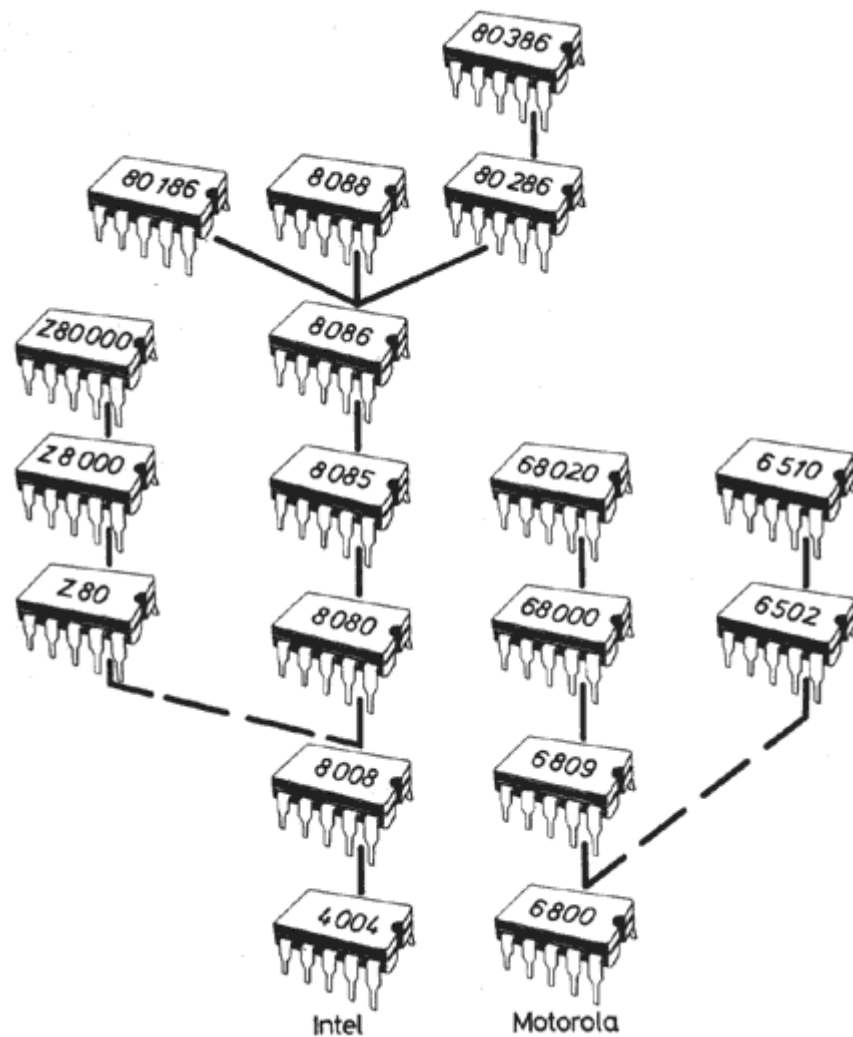
Význam slova mikropočítač se s časem měnil. Zatímco dříve se pod tímto pojmem skrývaly téměř všechny počítače s mikroprocesorem, dnes se takto označují počítače, které jsou oproti stolním počítačům výrazně menší a jsou často specializovány pro určité účely.

Historie mikropočítačů je spjata s vývojem mikroprocesorů. První mikroprocesory vznikly ve firmě Intel na začátku 70. let, kdy se jeden vývojový inženýr rozhodl, že pro řešení zadané úlohy nepoužije množství specializovaných obvodů, ale vytvoří univerzální programovatelný obvod, který byl schopen zpracovávat různé úkoly pouhým přeprogramováním. Tak vznikl první čtyřbitový mikroprocesor Intel 4004. Nicméně tento ani pozdější osmibitový 8008 nezaznamenaly většího rozšíření. Toho dosáhl až v roce 1974 typ Intel 8080A. Ve stejném roce byl uveden na trh konkurenční mikroprocesor firmy Motorola – 6800. Z Intel 8080A vychází také mikroprocesor Z 80 firmy Zilog, který byl ve stejném roce vytvořen pracovníky, kteří opustili firmu Intel. O rok později byl představen vylepšený typ osmibitového I 8080A – a to I 8085. Tyto procesory pak obsadily trh na dlouhou dobu. Byly konstruovány vývojové prostředky a školní pomůcky pro návrh mikropočítačových systémů s těmito mikroprocesory. V té době asi nejznámější postavila dvojice Jobs a Wozniak roku 1976 s procesorem od Motoroly a nazvali ho Apple II.

Dalším vývojovým stupněm byly 16 bitové mikroprocesory. I zde slavila úspěch firma Intel s I 8086 z roku 1978 a později nová varianta I 8088, která byla 16 bitová a s vnějším komunikačním 8 bitovým datovým sběrnici. V roce 1981 uvedla firma IBM na trh osobní počítač IBM PC s čipem I 8086, který ovládl trh. Úspěšný byl také později typ PC AT s novějším I 80286 z roku 1982. Nové typy počítačů s výkonnějšími procesory pak následovali přibližně každé 2 roky (r. 1985 32 bitový I80386, r. 1989 I80486, r. 1993 Intel Pentium, ...). Společně s Intelem se na poli mikroprocesorové techniky prosazuje i Motorola, jejíž mikroprocesory jsou základem počítačů Apple Macintosh. Ta např. jako reakci na I8086 vyrobila 32 bitový M68000.

Pro využití v oblastech jako pokladny v obchodech, domácí elektronice atd. bylo však zbytečné využívat výkonné univerzální mikroprocesory s obsáhlým instrukčním souborem.

Proto začaly vznikat tzv. monolitické mikropočítače. Tyto mikropočítače zahrnují jednotlivé stavební bloky počítače na jednu součástku. Jsou opět nabízeny řadou firem, např. Intel (8048, 8051, 8096 a 32 bitový 80196) a Motorola (6805, 6804, 68HC11, 16 bitový 68HC16 a 32 bitová řada 68300).[1][3]



Obr. 1: Počáteční vývoj mikroprocesorů firem Intel a Motorola[3]

1.2 Pojmy

Aritmeticko-logická jednotka (ALU) - obstarává vykonávání jednotlivých aritmetických a logických operací. Je ovládána řídicími signály z řadiče na základě instrukcí, kterými je určen typ prováděné operace. Zpracování operandů probíhá většinou paralelně nebo sérioparalelně.[1]

Řadič – úkolem řadiče je vyzvednout instrukci z hlavní paměti a uložit ji do dekodéru instrukce. To je prováděno prostřednictvím čítače instrukcí. V dekodéru se pak dekóduje operační znak a na jeho základě se vygenerují řídicí signály.

Dalším z úkolů řadiče je zpracování adresní části instrukce. Ta určuje operandy pro danou operaci. Adresy operandů se pak vypočítávají dle různých způsobů adresování. [1]

Paměť - Každý mikropočítač pro svou funkci nutně vyžaduje dva typy paměti – paměť programu a paměť dat.

V programové paměti je uložen vlastní program v podobě instrukcí, které řídí chování mikropočítače, a dále rovněž různé konstanty a neměnné tabulky, se kterými program pracuje. Pro realizaci programové paměti se dnes u mikropočítačů používají téměř výhradně paměti typu FLASH.

Jako datová paměť se využívá výhradně statická paměť RAM, u níž informace zaniká s vypnutím napájecího napětí. Mikropočítače bývají navíc často vybaveny ještě malou pamětí EEPROM (Electrically Erasable PROM), která slouží k zálohování důležitých dat.[7]

Sběrnice – z hardwarového pohledu je sběrnice soubor souběžných vodičů, na které se paralelně připojují jednotlivé bloky základní jednotky. V počítači se obvykle nachází tyto typy:

- datová sběrnice
- adresová sběrnice
- řídicí sběrnice
- specializované systémové sběrnice

Vnitřní sběrnice pak propojuje jednotlivé bloky procesoru.[1]

Instrukční soubor – obsahuje všechny instrukce, které je dané zařízení schopno zpracovat. Tyto instrukce jsou zapisovány pomocí symbolických názvů a ty jsou potom překládány pomocí softwarového překladače do strojového kódu, který dokáže mikroprocesor zpracovat. Existuje několik typů instrukcí:

- aritmetické
- logické
- posuvy a rotace

- instrukce přesunů
- instrukce skoků

Procesory se pak dle rozsahu instrukčního souboru dělí na CISC (complex instruction set computer) a RISC (reduced instruction set computer).[1]

Mikroprocesor – integrovaný logický obvod s určitým stupněm integrace. Vykonává aritmetické a logické operace dané programem. Řídí také ostatní části mikropočítače (zpracovává data v paměti, řídí tok dat ze vstupních obvodů do mikropočítače a jejich zpracování a též řídí tok dat z počítače přes výstupní obvody).[7]

Monolitický mikropočítač – tento typ mikropočítače zahrnuje mikroprocesor, potřebnou paměť pro data a program, vstupní a výstupní obvody na jediném čipu. Celý mikropočítač je tak k dispozici v jediném integrovaném obvodu.[7]

Registr – paměťová buňka v procesoru nebo periferním zařízení o kapacitě jeden byte, slabika nebo slovo.[5]

Zásobník – typ paměti, užívaný k uložení obsahu pracovních registrů a pro automatické ukládání návratových adres při volání podprogramů a při obsluze přerušení. Zásobník je charakteristický výběrem dat, který je prováděn v opačném pořadí, než v jakém jsou do něj ukládána. Výběr dat tedy není libovolný ale sekvenční a to typu LIFO (poslední dovnitř, první ven).[5]

Kompilátor – překladač z vyššího programovacího jazyka do strojově orientovaného jazyka. Oproti kompilovaným jazykům existují ještě interpretované. Ty nejsou kompilovány, ale vyžadují interpret, který má za úkol provedení kódu na daném systému.[5]

Linker – Program sloužící k vytvoření zaveditelného modulu (programu), umožňující slučovat soubory.[5]

Debugger – nástroj pro odhalování chyb v programu. Většinou umožňuje náhled do paměti s daty případně jejich editaci. Umožňuje také krokování programu a užití breakpointů k zastavení programu v určité fázi jeho vykonávání.

2 MIKROPOČÍTAČE FREESCALE ŘADY HCS08

V následujících podkapitolách bude oveden obecný popis řady HCS08 a jejich periferií s popisem procesoru.

2.1 Obecný popis řady HCS08

HCS08 je nástupcem řady HC08, který přináší několik vylepšení a zároveň je zpětně kompatibilní s kódem napsaným pro HC08. Každé MCU (microcontroller unit) této řady se skládá z jádra HCS08 a několika paměťových a periferních modulů.

Jádro pak obsahuje:

- HCS08 CPU
- Background debug controller (BDC)
- Podporu až pro 32 zdrojů přerušení/resetu
- Dekódování adres na úrovni čipu

HCS08 CPU vykonává všechny instrukce instrukčního souboru. Maximální taktovací frekvence CPU je 40 MHz (hodinový signál je generován krystalovým nebo interním generátorem). CPU tedy pracuje na frekvenci 40 MHz, zatímco sběrnice dosahuje maxima 20 MHz (polovina frekvence CPU).

Background debug controller je zabudován do CPU a umožňuje přístup do obvodů generujících adresy a do registrů procesoru. BDC obsahuje jeden hardwarový breakpoint. Další sofistikovanější breakpointy jsou většinou obsaženy v oddělených debug modulech. BDC umožňuje přístup do vnitřního registru a míst paměti pomocí jediného pinu na MCU.

Ačkoli rozložení paměti jednotlivých odnoží HCS08 se liší, některé základní aspekty jsou ovládány dekodovací logikou jádra HCS08, která se v jednotlivých typech nemění. Registry pro vstupní/výstupní (I/O) porty a stavové registry pro periferní zařízení se začínají od adresy \$0000 a pokračují až do 32, 64, 96, nebo 128 bytů. Prostor od konce těchto registrů až po \$107F je rezervován pro statickou RAM. Prostor začínající na \$1800 je pak rezervován pro tzv. high-page registry. To jsou stavové a ovládací registry, ke kterým není tak často přistupováno jako k direct page registrům. Například registry pro nastavení systému, do kterých se zapisuje pouze jednou po resetu, mohou být umístěny

právě ve „vyšší“ části (high-page) tohoto prostoru, aby uvolnily místo v přímém adresovém prostoru pro častěji využívané registry a RAM. Zbývající místo od \$1C00 až po \$FFFF je rezervováno pro FLASH paměť nebo ROM. Posledních 64 míst (\$FFC0 - \$FFFF) je označováno jako prostor vektorů (pro až 32 vektorů přerušení a resetu).[2]

2.2 Periferní moduly

Kombinace periferních modulů obsažených na specifických odnožích HC08 se mohou velmi lišit. Vždy však budou obsahovat paměť pro program a data, generator časového signálu a ladící modul. MCU pak může obsahovat moduly:

- 4K – 60K bytů FLASH nebo ROM paměti
- 128 – 4K bytů statické RAM
- Asynchronní sériový I/O (SCI)
- Synchronní sériový I/O (SPI a IIC)
- Časovací/PWM moduly (TPM)
- Přerušení od klávesnice (KBI)
- Analogově číslicový převodník (ADC)
- Moduly pro generování časových signálů
 - Úplný interní generátor (ICG) schopný operovat bez dodatečných externích komponent
 - Tradiční Piercův oscilátor
- Debugovací modul [2]

2.3 CPU08

CPU08 je centrální procesorová jednotka řady M68HC08. Je plně kompatibilní řadou M68HC05 na úrovni strojového kódu.

2.3.1 Vlastnosti

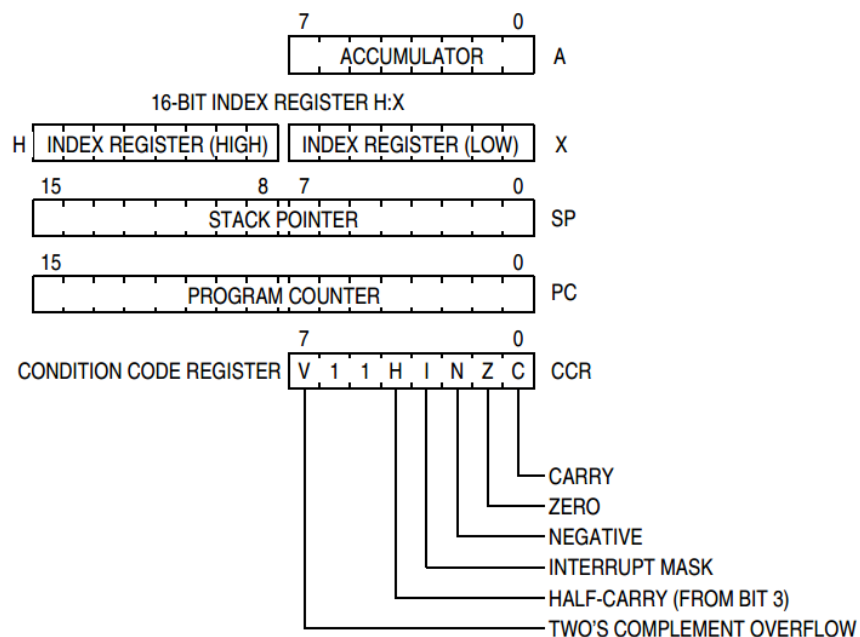
CPU obsahuje 16 bitový ukazatel zásobníku s instrukcemi pro manipulaci se zásobníkem, 16 bitový indexový registr (H:X) s manipulačními instrukcemi pro vyšší a nižší byte, sběrnici s frekvencí 8MHz, 64Kbytu programového/datového paměťového prostoru, 16 adre-

sovacích módů, 78 nových operačních kódů, přemísťování dat v paměti bez použití akumulátoru, rychlé instrukce pro násobení 8x8 bitů a dělení 16/8 bitů, rozšiřitelnost pro adresování nad 64K bytů, flexibilní definice sběrnice umožňující rozšíření o periferie pro zvýšení výkonu jako je kontroler pro přímý přístup k paměti (DMA). [4]

2.3.2 Programátorský model (registry CPU)

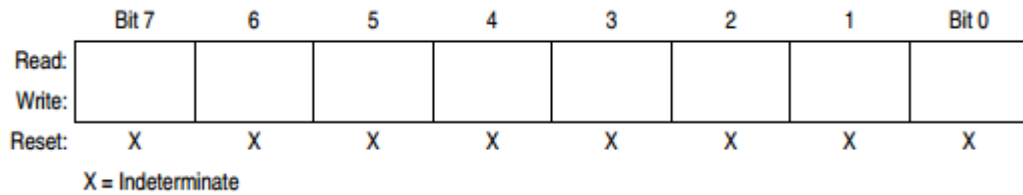
Programátorský model CPU08 se skládá z:

- 8 bitového akumulátoru
- 16 bitového indexového registru
- 16 bitového ukazatele zásobníku
- 16 bitového programového čítače
- 8 bitového registru příznaků



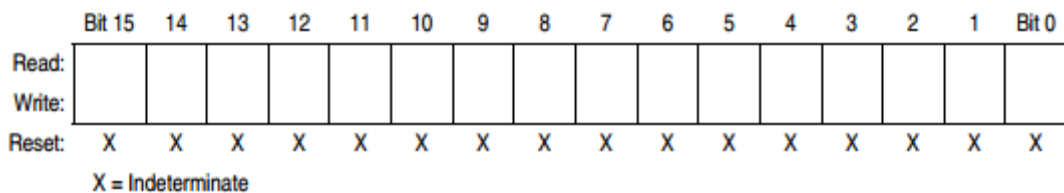
Obr. 2: Registry CPU[4]

Akumulátor (A) je víceúčelový 8 bitový registr. CPU jej využívá k ukládání operandů a výsledků aritmetických i nearitmetických operací.



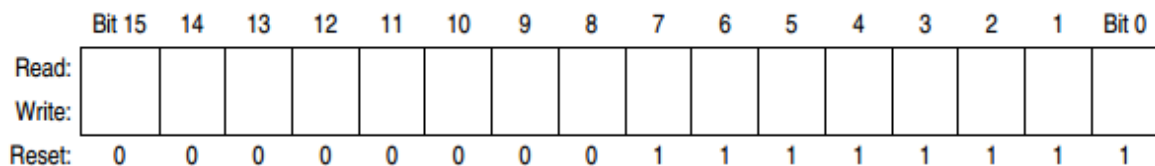
Obr. 3: Akumulátor[4]

Indexový registr (H:X) umožňuje uživateli indexovat nebo adresovat 64K bytovou paměť. Vyšší byte tohoto zásobníku je označován jako H, nižší je pak označován X. V indexovacích adresovacích módech CPU využívá obsah H:X k určení efektivní adresy operandu. H:X může také sloužit jako dočasné úložiště dat.



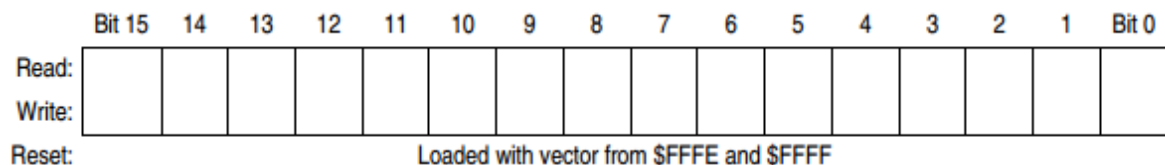
Obr. 4: Indexový registr[4]

Ukazatel zásobníku (SP) je 16-bitový registr, který obsahuje adresu následujícího prvku na zásobníku. Při resetu je ukazatel nastaven na hodnotu \$00FF kvůli zajištění kompatibility s řadou mikropočítačů M6805. (resetovací instrukce RSP nastavuje nejméně významný byte na hodnotu \$FF, nejméně významný byte nijak neovlivňuje) Adresa v ukazateli zásobníku se snižuje s tím, jak jsou data nahrávána (push) na zásobník a naopak se zvyšuje, když jsou nahrávána ze zásobníku (pull). SP tak ukazuje vždy na další volný byte na zásobníku.



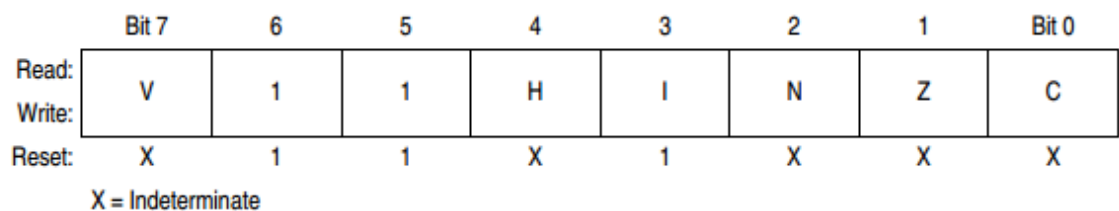
Obr. 5: Ukazatel zásobníku[4]

Programový čítač (PC) 16-bitový registr obsahující adresu následující instrukce nebo operandu k načtení. Adresa v PC se automaticky inkrementuje na následující sekvenční místo v paměti pokaždé, když je načtena instrukce nebo operand. Operace skoku, větvení a přerušení nahrají do PC jinou adresu než tu následující sekvenční oblasti. Během resetu je do PC nahrán obsah resetovacího vektoru z adresy \$FFFE a \$FFFF. Toto reprezentuje adresu první instrukce, která se má vykonat po resetu.



Obr. 6: Programový čítač[4]

Stavový registr příznaků (CCR) 8-bitový registr obsahující masku přerušení a 5 flagů indikující výsledný stav právě provedené instrukce. Bity 5 a 6 jsou permanentně nastaveny na logickou 1.



Obr. 7: Registr příznaků[4]

V – příznak přetečení. Je nastaven, pokud dojde k přetečení při výpočtu s dvojkovým doplňkem.

H – poloviční přenos. Je nastaven, pokud dojde k přenosu mezi 3. a 4. bitem akumulátoru během ADD nebo ADC operace. Tento příznak je potřebný u BCD aritmetických operací.

I – maska přerušení. Pokud je nastavena maska přerušení, jsou všechna přerušení zakázána. Při přerušení se automaticky nastaví na 1 potom, co se obsah registrů uloží na zásobník, předtím než se načte vektor přerušení. Pokud se objeví přerušení a I je nastaven na 1 jsou obsloužena dle priority až potom co je bit I vynulován.

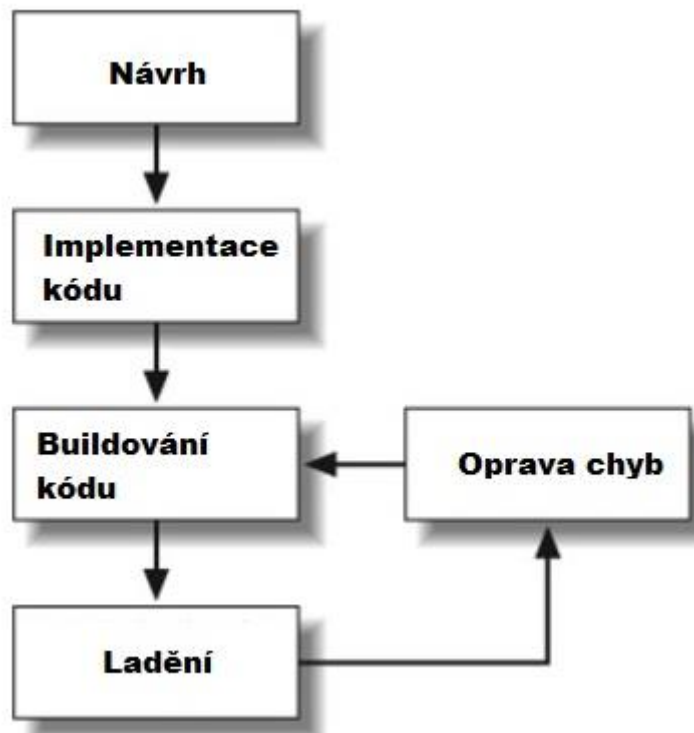
N – znaménkový příznak. CPU nastaví tento příznak, pokud výsledek aritmetické operace, logické operace nebo manipulace s daty vyprodukuje negativní výsledek.

Z – nulový příznak. Je nastaven, pokud výsledek aritmetické operace, logické operace nebo manipulace s daty vyprodukuje výsledek \$00.

C – příznak přenosu. Je nastaven, pokud sčítací operace vyprodukuje přenos, nebo když odečítací operace vyžaduje přenos. Některé logické operace a instrukce pro manipulaci s daty také nulují nebo nastavují tento příznak (bit test, posuny a rotace). [4]

3 PROGRAMOVÁNÍ MIKROPOČÍTAČE

Jak je vidět z obrázku 8, vývojový cyklus softwaru začíná designem a implementací kódu. Potom obvykle následuje cyklus buildování (sestavení výsledného programu), stažení programu do paměti cílového zařízení, vyhledávání chyb a jejich následná oprava.



Obr. 8: Vývoj programu[6]

Jelikož se jedná o zjednodušený diagram, nejsou zde zobrazeny některé části, jako je profilování a optimalizace. Profilování slouží ke zjištění určitých parametrů jako např. ve které části stráví procesor nejvíce času. Optimalizace je proces, který se snaží zajistit co nejefektivnější chod programu pomocí různých technik jako je implementace časově kritického kódu v jazyku s. adres. Velmi často tyto techniky závisí na kompilátoru, procesoru nebo operačním systému.

3.1 Programovací jazyk C

Programovací jazyk C je jednou z mála konstant na poli mikropočítačové techniky. Je používán programátory embedded systémů více než jakýkoliv jiný jazyk. Nebylo tomu tak vždy, nicméně C je nejbliže tomu co by se dalo označit jako standard u těchto zařízení.

Protože úspěšný vývoj softwaru často závisí na výběru vhodného jazyka pro daný projekt, je s podivem že jeden jazyk se ukázal vhodný jak pro 8-bitové tak pro 64-bitové procesory v systémech s byty až mnoha megabyty paměti a pro vývojové týmy od jednoho po desítky vývojářů. Přesto C tuto širokou oblast pokrývá.

Možná proto, že C má mnoho výhod. Je malý a celkově jednoduchý na naučení, kompilátory jsou dostupné pro prakticky každý dnes vyráběný procesor a existuje velká základna zkušených C programátorů. Navíc C má výhodu nezávislosti na procesoru což umožňuje vývojářům se zaměřit na vývoj samotné aplikace a jejích algoritmů než na detaily dané procesorové architektury. Tyto výhody však nabízejí i jiné programovací jazyky takže je otázkou proč se prosadilo právě C.

Nejspíše nejsilnější stránkou tohoto jazyka je oproti jiným, jako jsou Pascal nebo FORTRAN je to, že se jedná o velmi „nízkourovňový“ vyšší programovací jazyk. To znamená, že dává uživateli výjimečnou úroveň přímé kontroly nad hardwarem bez obětování výhod vyšších programovacích jazyků.

Je málo populárních vysokoúrovňových jazyků, které se s C mohou rovnat v kompaktnosti a efektivnosti kódu pro téměř jakýkoliv procesor. A i mezi těmito snad jen C umožňuje programátorům takto snadnou interakci s hardwarovou vrstvou.[6]

3.2 Kompilování, linkování a relokování

Pokud nástroje pro tyto úkony běží na stejném systému, jako poběží program, který produkuje, mohou si o daném systému udělat samy obrázek. To však neplatí u mikropočítačových systémů, kde tyto nástroje běží na vývojovém počítači který je odlišný od cílové hardwarové platformy. Existuje mnoho věcí, které vývojové nástroje mohou provádět automaticky, pokud mají dobře definovanou cílovou platformu. Tato automatizace je možná protože vývojové nástroje mohou využít vlastností daného hardwaru a operačního systému, na kterém program poběží. Pokud například budeme uvažovat, že vaše programy budou spouštěny pouze na IBM kompatibilních PC s operačním systémem Windows, váš kompilátor může automatizovat a z vašeho pohledu skrýt určité kroky při buildování programu. Vývojové nástroje pro mikropočítače však nemohou samy činit závěry o cílové platformě. Místo toho jim uživatel musí sám poskytnout dodatečné informace.

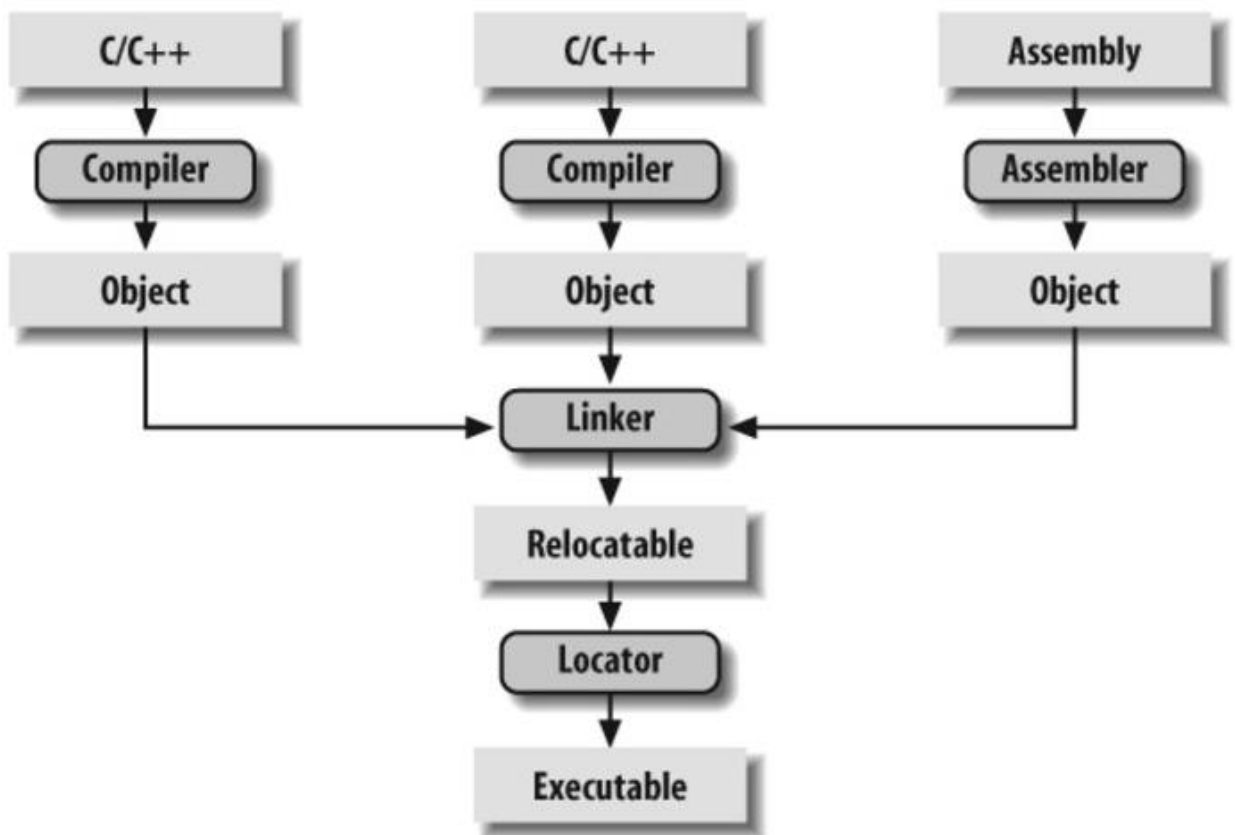
Proces konverze zdrojového kódu na spustitelný binární obraz zahrnuje 3 kroky:

1. Každý zdrojový soubor je kompilován do objektového souboru.

2. Všechny výsledné objektové soubory z bodu 1 jsou pak linkovány do jediného objektového souboru, kterému se říká přemístitelný program (relocatable program).

3. Adresy fyzické paměti jsou přiřazeny relativním offsetům v přemístitelném programu v procesu zvaném relokace (přemístění).

Výsledek posledního kroku je pak soubor obsahující spustitelný binární obraz, který se dá spustit na cílovém systému. [6]



Obr. 9: Příprava binárního obrazu programu[6]

3.2.1 Kompilování

Úkolem kompilátoru je tedy překlad programů napsaných v jazyku čitelném pro člověka do ekvivalentního zápisu operačních kódů pro určitý procesor. Z tohoto pohledu je i assembler druh kompilátoru (kompilátor jazyka symbolických adres), který ale provádí mnohem jednodušší překlad typu jedna ku jedné, tedy z jednoho řádku daného jazyka na jeden ekvivalentní operační kód.

Překladač běžící na jednom zařízení a produkující kód pro jiné zařízení se nazývá křížový překladač (cross compiler). Mikropočítačová technika se využitím těchto překladačů vyznačuje.

Nehledě na vstupní jazyk (C, C++, JSA...), výstupem křížového překladače je objektový soubor. To je speciálně formátovaný binární soubor, který obsahuje sadu instrukcí a data vzniklé jako výsledek překladu. Tento soubor sice obsahuje části spustitelného kódu, ale sám o sobě být spuštěn nemůže.

Obsah objektového souboru si můžeme představit jako velkou flexibilní datovou strukturu. Struktura tohoto souboru je pak definována standardy jako jsou COOF nebo ELF. Většina těchto souborů začíná hlavičkou popisující následující sekce. Každá z těchto sekcí obsahuje jeden nebo více bloků kódu nebo dat pocházejících ze zdrojového kódu, které kompilátor seřadil do návazných sekcí.

Soubor také většinou obsahuje tabulku symbolů, ve které jsou zaznamenány jména alokace všech proměnných a funkcí referencovaných ve zdrojovém souboru. Části této tabulky mohou být neúplné, protože ne všechny proměnné a funkce jsou definovány v jednom zdrojovém souboru. Je pak na linkeru, aby vyřešil tyto reference. [6]

3.2.2 Linkování

Všechny výstupní soubory kompilace se následně musí zkombinovat. Objektové soubory samy o sobě jsou nekompletní a je potřeba aby linker tyto nedostatky doplnil.

Výstupem linkeru je nový objektový soubor, který obsahuje veškerý kód a data ze vstupních souborů a je ve stejném objektovém formátu. Provádí to tak že spojí všechny sekce vstupních souborů. V průběhu tohoto spojování také vyhledává symboly, u kterých chybí definice. Např. pokud jeden objektový soubor obsahuje nevyřešenou referenci na proměnnou a tato proměnná je deklarována v některém z ostatních souborů, linker je propojí. Nevyřešená reference bude nahrazena referencí na skutečnou proměnnou. Pokud se proměnná nachází např. na offsetu 14 datové sekce, její záznam v tabulce symbolů bude nyní obsahovat tuto adresu. Pokud je stejný symbol deklarován ve více než jednom objektovém souboru, linker nemůže pokračovat. Proces se pak ukončí většinou vypsáním chyby pro uživatele.

Na druhou stranu pokud zůstane některá reference nevyřešena pod spojení všech souborů, linker se ji pokusí vyřešit sám. Může se totiž jedna o referenci na funkce jako *memcpy*,

strlen nebo *malloc* které jsou součástí standardní knihovny jazyka C, takže linker otevře každou tuto knihovnu která a projde jejich tabulky symbolů. Pokud objeví funkci nebo proměnnou stejného jména, přidá je do daného kódu nebo datové sekce výstupního objektového souboru.

Po pospojování celého kódu a datových sekcí a vyřešení všech symbolových referencí, linker vyprodukuje soubor obsahující speciální přemístitelnou kopii programu. Jinými slovy kompletní program, který však neobsahuje žádné adresy paměti, ale pouze offsety. Pokud by se nejednalo o mikropočítačový systém byl by proces hotov.

Adresy symbolů jsou při linkovacím procesu relativní. I když váš mikropočítačový systém užívá operační systém, budete potřebovat absolutně umístěný (lokovaný) binární obraz. Pokud používáte operační systém, jeho kód a data jsou nejspíše umístěny v přemístitelném souboru také. [6]

3.2.3 Startup kód

Jednou z věcí, kterou tradičně automaticky provádí vývojové nástroje je vložení startovacího kódu. To je malý blok kódu v jazyce symbolických adres, který připraví, jakým způsobem se bude zpracovávat kód psaný ve vyšším programovacím jazyce. Každý vyšší programovací jazyk má vlastní nároky na prostředí, ve kterém běží. Například C používá zásobník. Prostor pro tento zásobník musí být alokovan ještě předtím, než může být spuštěn software napsaný v C.

Většina křížových kompilátorů obsahuje soubor s názvem *startup.asm*, *crt0.s* (zkratka pro C runtime) nebo podobný. Lokace a obsah tohoto souboru je většinou popsána v dokumentaci kompilátoru.

Startup kód se obvykle skládá z těchto akcí:

1. Zákaz všech přerušení
2. Zkopírování dat z ROM do RAM
3. Vynulování neinicializované oblasti dat
4. Alokování prostoru pro zásobník a jeho inicializace
5. Inicializace ukazatele zásobníku na procesoru
6. Zavolání *main*

Typicky tento kód také obsahuje několik instrukcí, které se volají po této proceduře. Tyto instrukce budou vykonány pouze v případě, že program vyššího jazyka bude ukončen. S ohledem na daný systém mohou být tyto instrukce využity k pozastavení procesoru, resetu systému nebo předání ovládní debugovacím nástroji.

3.2.4 Relokace

Nástroj, který provádí konverzi z přemístitelného programu na spustitelný binární obraz, se nazývá lokátor (umíst'ovač). Provádí asi nejjednodušší úkon z celého řetězce. Ve skutečnosti většinu práce v tomto kroku vykoná uživatel tím, že nástroji poskytne informaci o paměti cílového zařízení. Lokátor použije tuto informaci k přiřazení adres fyzické paměti každé kódové a datové sekci přemístitelného programu. Potom vyprodukuje výstupní binární obraz paměti, který může být nahrán do cílového zařízení. Někdy může být lokátor jako samostatný nástroj, většinou je však součástí linkeru.[6]

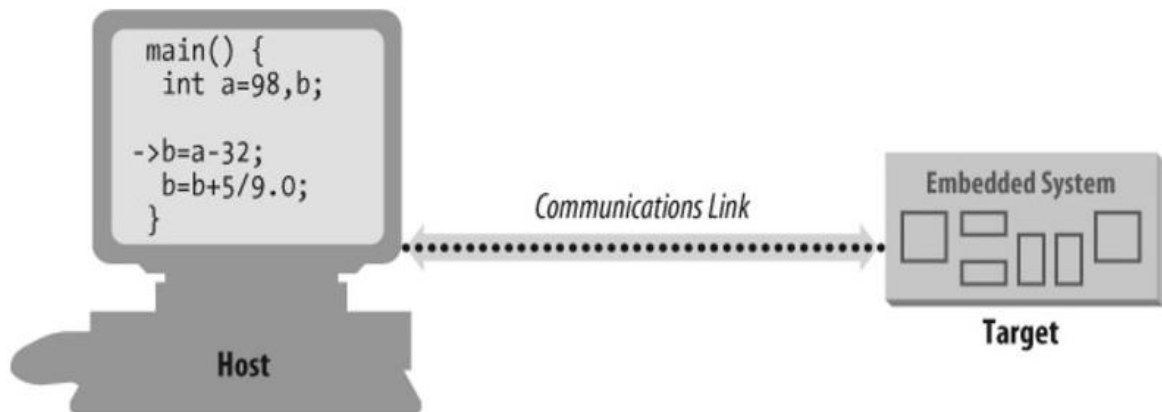
3.3 Download a debugování

Jakmile máme hotový binární obraz uložen v souboru, můžeme tedy program nahrát do cílového zařízení a spustit ho. Se správnými nástroji je pak možno nastavovat breakpointy v programu nebo sledovat jeho provedení.

3.3.1 Debug monitory

Debug monitor nebo také ROM monitor je malý program, který je uložen v nevolatilní paměti cílového hardwaru který zajišťuje několik operací potřebných během vývoje. Jedním z jeho úkolů je zajištění základní inicializace hardwaru. Debug monitor umožňuje stažení a spuštění programu v RAM pro zjištění chyb v tomto programu. Monitor pak obsahuje funkce, jako jsou příkazy pro stažení a spuštění programu, čtení a zápis do paměti a registrů procesoru, porovnávání nebo zobrazování bloků paměti a nastavování inicializační konfigurace hardwaru.

Vzdálený debugger - používá se ke stažení, provedení a debugování softwaru přes sériový port nebo síťové připojení mezi hostem a cílem. Grafické rozhraní na hostujícím zařízení pak vypadá jako jakýkoliv běžný debugger.



Obr. 10: Vzdálené debugování[6]

Vzdálené debuggery jsou jedním z nejpoužívanějších nástrojů pro nahrávání do cílového zařízení a testování programu a to hlavně díky své ceně. Nicméně mají své nevýhody, jako jsou nemožnost debugování startup kódu, nutnost spouštění z RAM a potřeba komunikačního kanálu mezi zařízeními.

3.3.2 Emulátory

Emulátory poskytují mnohem více funkcionality než vzdálený debugger. Oproti těm navíc umožňuje debugování startup kódu a běh programu z ROM, nastavování breakpointů blížících z ROM a spouštění testů vyžadujících více RAM než systém obsahuje.

Emulátor samotný je také embedded systém se svou vlastní kopií cílového procesoru, RAM, ROM a softwaru. Jsou proto celkem drahé. Stejně jako dálkový debugger využívá grafické rozhraní na hostujícím počítači.

Emulátory mají navíc kromě softwarových také hardwarové breakpointy. Ty umožňují zastavení programu nejen na základě instrukcí, ale mohou reagovat také na přerušení, čtení a zápis do paměti.

Dalším podobným typem debugovacích nástrojů jsou **background debug** módy. Typicky nejsou tak nákladné a nabízejí víceméně stejnou funkcionalitu. Jsou založeny na debugovacím interface a testovacím obvodu který je součástí většiny moderních procesorů.

3.3.3 Simulátory

Simulátor je program kompletně běžící na hostujícím počítači. Ten simuluje (tzn. napodobuje) funkcionalitu a instrukční sadu cílového procesoru. Uživatelské rozhraní je víceméně stejné jako u vzdáleného debuggeru. Simulátory mají sice mnoho nevýhod, jsou však nepostradatelné na začátku vývoje kdy vývojáři nemají ještě dostupný cílový hardware pro experimentování. Největší nevýhodou simulátoru je že simuluje pouze procesor. Cílové systémy většinou obsahují ještě další periferie.

II. PRAKTICKÁ ČÁST

4 DOSTUPNÉ VÝVOJOVÉ NÁSTROJE

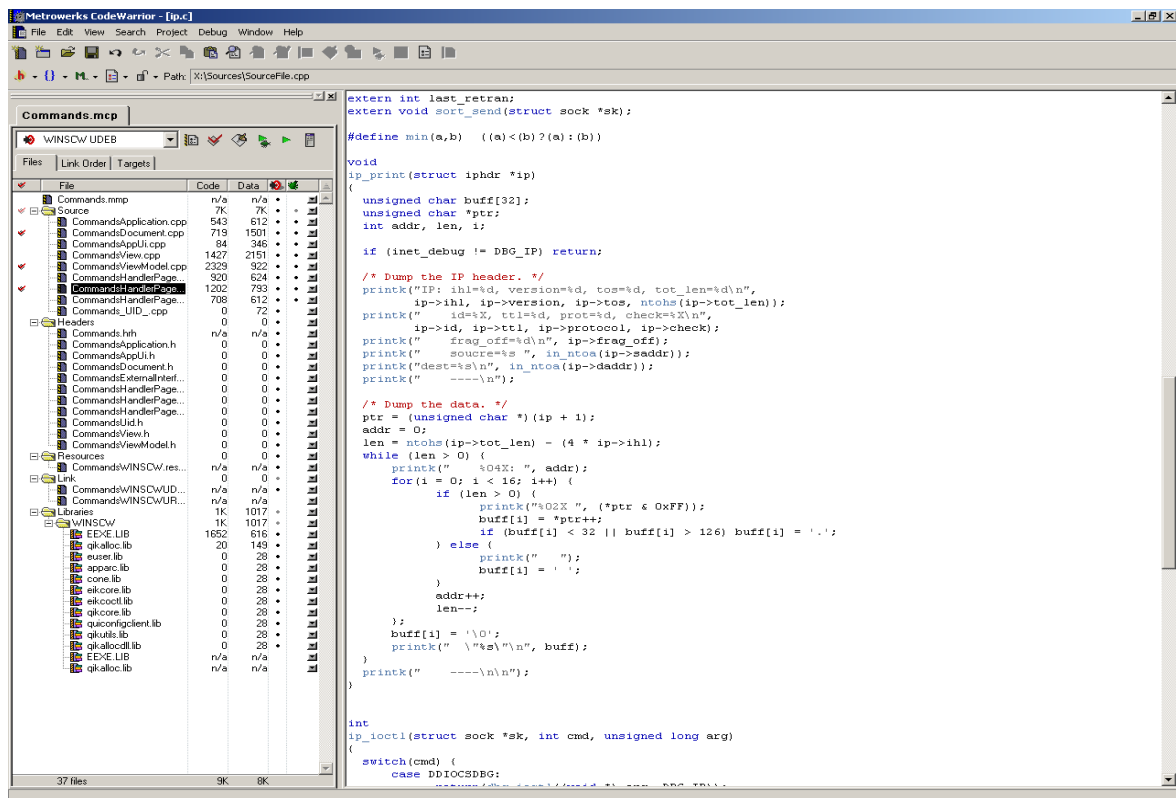
V této části bude následovat popis vývojových nástrojů pro mikropočítače Freescale s jádrem HC08/HCS08. Jsou zde zahrnuty jak nástroje poskytované zdarma, vyvíjené převážně komunitou, tak profesionální placená prostředí.

4.1 Codewarrior

Codewarrior je komerční sada vývojových nástrojů zahrnující v sobě assembler, C/C++ kompilátor s případným omezením podle verze balíku, integrovaný FLASH programátor a ladící program se simulátorem daného mikropočítače. Existuje distribuce jak s vlastním grafickým rozhraním tak verze založená na Eclipse IDE. Dále se dělí na 3 placené verze (Professional, Standard, Basic) a verzi zdarma (Special Edition), která je ale omezena velikostí kódu. Omezení velikosti kódu obsahuje i verze Basic avšak umožňuje zpracování programu 2 krát většího než Special (u řady S08 je to 64kB pro Special a 128kB pro Basic). Verze Standard je pak neomezená, co se týče velikosti kódu. Všechny jmenované verze nabíí tzv. kernel-aware debug. Jde o funkci ladícího nástroje, která umožňuje zobrazovat struktury operačního systému v paměti (např. informace o procesech, schránkách apod.). Verze Professional oproti ostatním nabízí podporu více operačních systémů pro tuto funkci.

Cena tohoto prostředí se pohybuje od 395 až po 4995 dolarů (tj. zhruba 8000 až 100 000 Kč dle kurzu v době psaní této práce). Podrobné informace s tabulkou porovnávací funkce jednotlivých verzí jsou k vidění na stránce

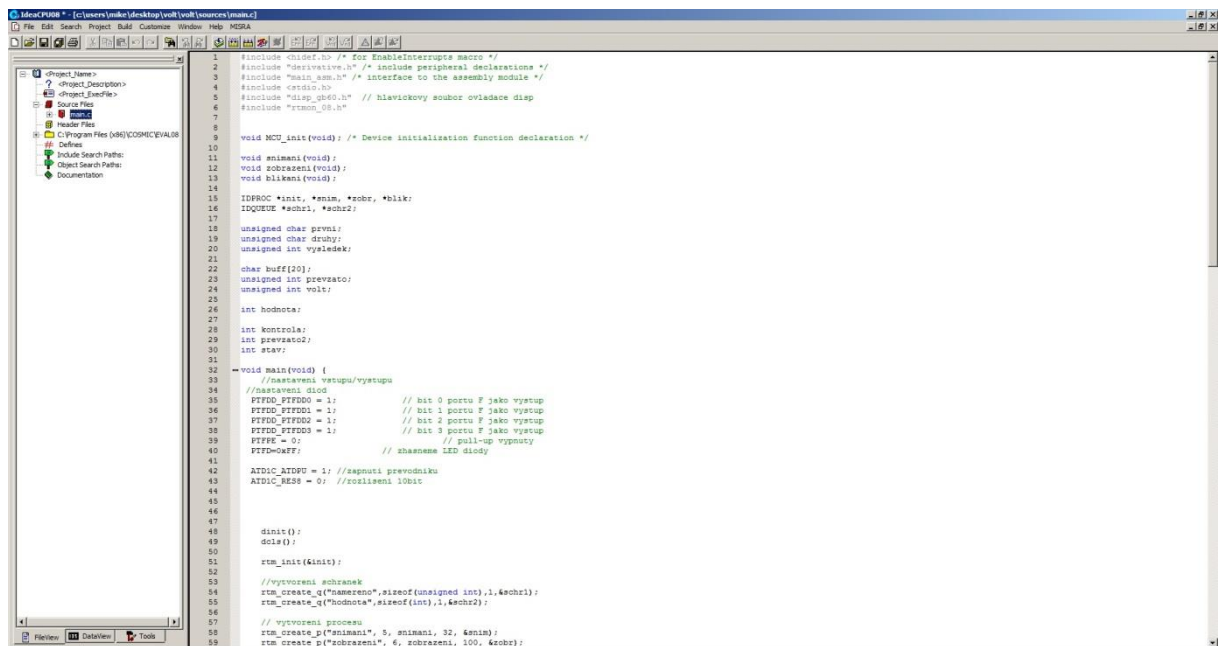
http://www.freescale.com/webapp/sps/site/overview.jsp?code=CW_SUITES&tid=CWH.



Obr. 11: CodeWarrior IDE

4.2 Cosmic

Prostředí firmy Cosmic software obsahuje kromě kompilátoru také grafické prostředí s editorem kódu a správcem souborů a projektů. Dále prostředí obsahuje debugger ve formě samostatné aplikace. Tento debugger existuje v několika verzích lišících se dle cílového zařízení případně verze se simulátorem. Tato sada nástrojů je dostupná ve dvou verzích. Kromě placené verze je tedy ještě dostupná verze zkušební, která obsahuje omezení velikosti programu 4kB. Cenu placené verze výrobce na svých stránkách neuvádí. Podrobnější informace na webu <http://www.cosmic-software.com/hc08.php>.



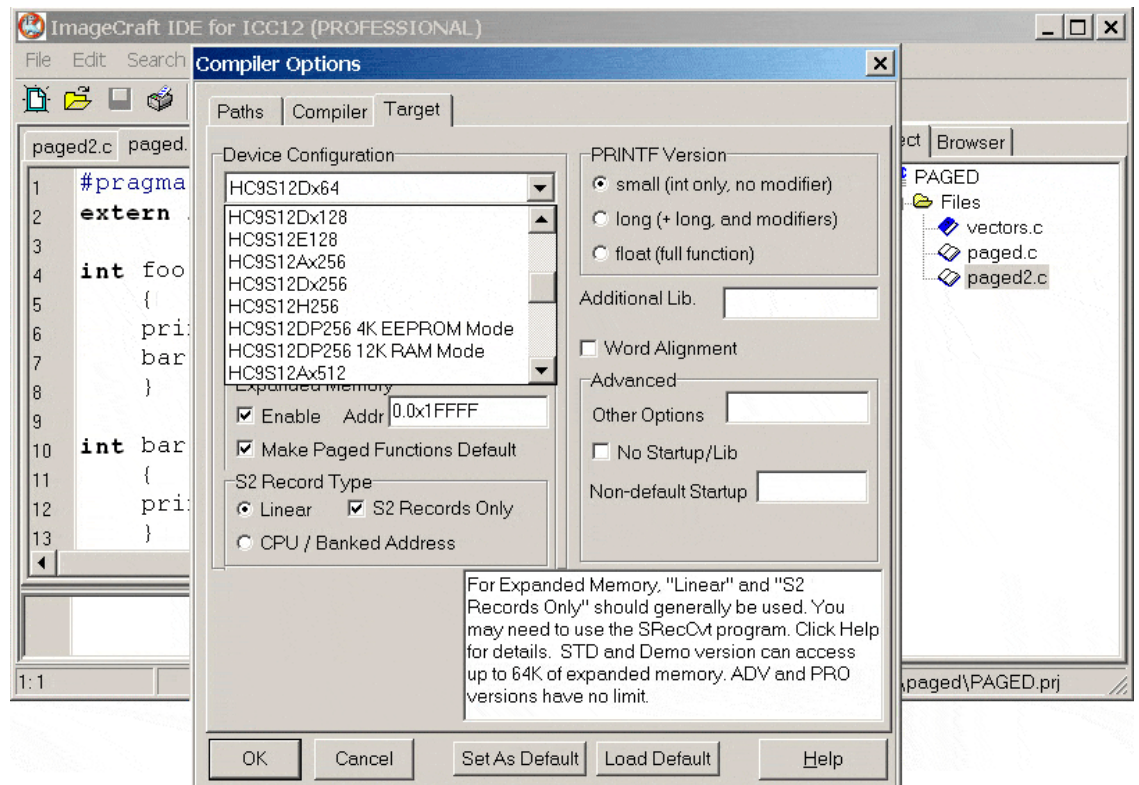
```
1 #include <stdint.h> /* For EnableInterrupts Macro */
2 #include "derivative.h" /* include peripheral declarations */
3 #include "main_asm.h" /* interface to the assembly module */
4 #include <stdio.h>
5 #include "disp_gpio.h" // hlavickovy soubor ovladace disp
6 #include "rtmon_08.h"
7
8
9
10 void MCU_init(void); /* Device initialization function declaration */
11 void snimani(void);
12 void zobrazeni(void);
13 void blikani(void);
14
15 IDPROC *init, *snim, *zobr, *blik;
16 IDQUEUE *sch1, *sch2;
17
18 unsigned char prvni;
19 unsigned char druhe;
20 unsigned int vysledek;
21
22 char buff[20];
23 unsigned int prevzat0;
24 unsigned int volt;
25
26 int hodnota;
27
28 int kontrola;
29 int prevzat02;
30 int stav;
31
32 --void main(void) {
33     //nastaveni vstupu/vystupu
34     //nastaveni diod
35     PFD0_PFD00 = 1; // bit 0 portu F jako vystup
36     PFD0_PFD01 = 1; // bit 1 portu F jako vystup
37     PFD0_PFD02 = 1; // bit 2 portu F jako vystup
38     PFD0_PFD03 = 1; // bit 3 portu F jako vystup
39     PTFE = 0; // pull-up vypnuty
40     PFD0=0xFF; // zhasneme LED diody;
41
42     ATD1C_ATD0P = 1; //zapnutí převodníku
43     ATD1C_RES0 = 0; //rozlišení 10bit
44
45
46
47
48     dinit();
49     dclr();
50
51     rtm_init(&init);
52
53     //vytvoreni schrank
54     rtm_create_q("nastaveni", sizeof(unsigned int), 1, &sch1);
55     rtm_create_q("hodnota", sizeof(int), 1, &sch2);
56
57     // vytvoreni procesu
58     rtm_create_p("snimani", 5, snimani, 32, &init);
59     rtm_create_p("zobrazeni", 6, zobrazeni, 100, &sch2);
60 }
```

Obr. 12: Cosmic IDE

4.3 Imagecraft ICC08

Jedná se opět o komerční sadu nástrojů distribuovaných v několika verzích podle cílového hardware. Verze ICC08 podporuje všechny HC08 a HCS08 mikropočítače. Jako všechny předchozí produkty obsahuje grafické rozhraní s editorem kódu a správcem projektů, ANSI C kompilátor, assembler s linkerem. Dostupný je ve verzi Standard a Professional. Verze Professional navíc obsahuje nástroj Code Compressor™ který podle výrobce dokáže zmenšit velikost výsledného programu o 5-10%. Vzhledem k tomu že výrobce ukončil vývoj těchto nástrojů, není k nim poskytována technická podpora a programy nefungují pod Windows Vista a novější. Cena je 249 dolarů za Standard a 369 za Professional verzi. Více informací je k nalezení na stránce

https://www.imagecraft.com/devtools_FreescaleOther.html.



Obr. 13: Imagecraft IDE

4.4 SDCC

SDCC (small device C compiler) je open source sada pro překlad programů pro osmibitové mikroprocesory dostupná zdarma. Současná verze podporuje procesory založené na Intel MCS51, Dallas varianty DS80C390, Freescale vycházející z řady HC08, Zilog řady Z80 a STMicroelectronics STM8. Celý zdrojový kód je distribuován pod licencí GNU GPL.

V balíku se nachází sada konzolových aplikací pro překlad a vývoj. Kromě základních nástrojů jako je kompilátor, assembler a linker v různých verzích dle výrobce daného hardwaru jsou k dispozici ještě simulátor, ladící program (debugger) a jiné. Domovská stránka projektu je <http://sdcc.sourceforge.net>.

5 VOLBA PROSTŘEDÍ

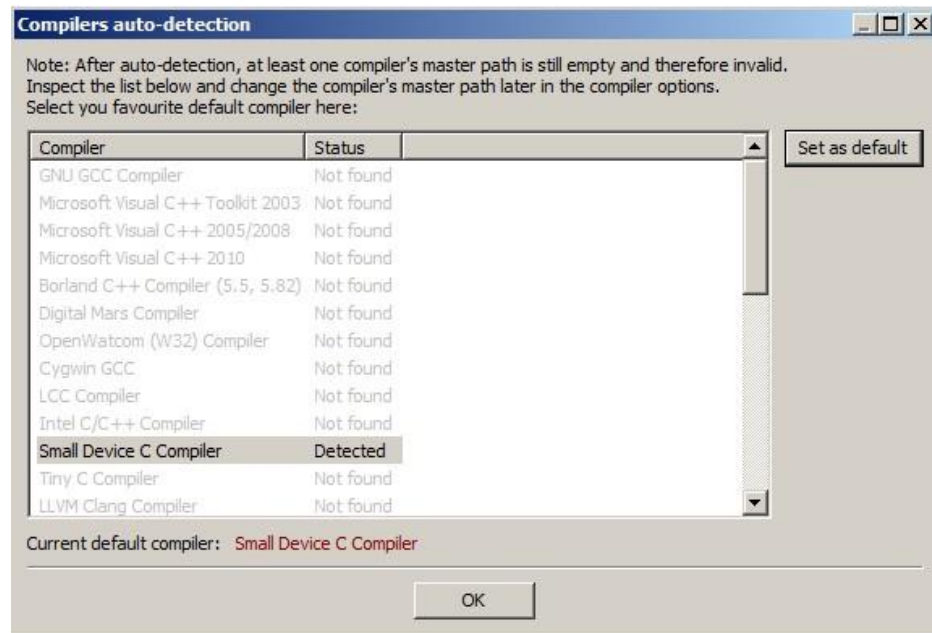
Základním požadavkem pro volbu vývojových nástrojů je dle zadání jejich dostupnost zdarma. V následujících částech práce proto budeme pracovat s SDCC. Jako editor kódu a správce projektů bude použito Code Blocks IDE volně dostupné z <http://www.codeblocks.org/>. Jedná se o modulární IDE zaměřené na vývoj aplikací v C/C++. Při výběru IDE připadal v úvahu ještě Eclipse pro který dokonce existuje plugin pro integraci SDCC. Ten však nefunguje správně s novějšími verzemi tohoto IDE. Zvolil jsem proto Code blocks a to z několika důvodů. Kromě faktu že jsem znal Code Blocks z dřívější doby hrála význam také jednoduchost s jakou se v něm dá nastavit externí překladač a svoji roli může pro někoho hrát fakt že Eclipse zabírá více systémové paměti.

5.1 Nastavení SDCC v Code Blocks IDE

Nejprve je tedy třeba stáhnout nejnovější verzi nástrojů. IDE Code Blocks je možno stáhnout z adresy uvedené výše. SDCC je pak dostupné z <http://sdcc.sourceforge.net/>.

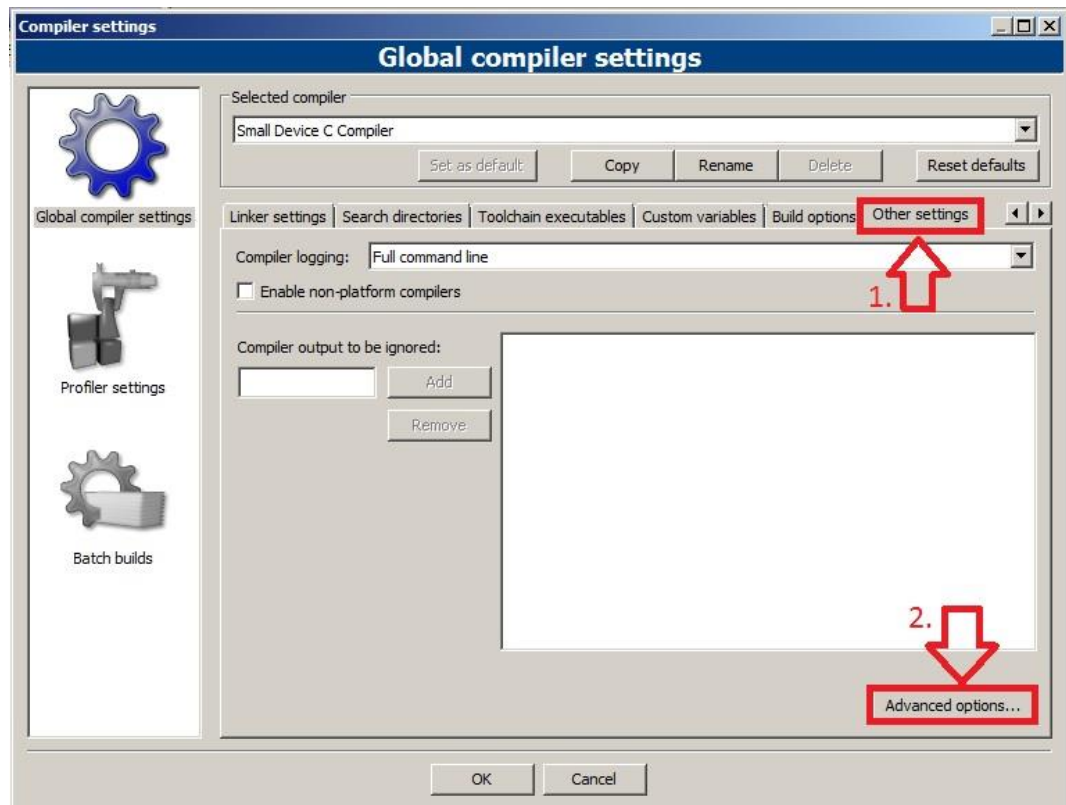
Nejdříve je nutno nainstalovat balík s kompilátorem SDCC. K tomuto kroku je jen potřeba dodat, že na konci instalace se musí potvrdit přidání složky s `sdcc.exe` do systémových cest. Kompilátor je pak možno volat z příkazového řádku pouze zadáním `sdcc`.

Dalším krokem je instalace Code Blocks. Instalace opět nevyžaduje většího popisu. Po instalaci a prvním spuštění je pak potřeba nastavit výchozí kompilátor pro IDE. V okně s dotazem které se objeví po prvním spuštění tedy najdeme Small Device C Compiler a nastavíme jej jako výchozí tlačítkem Set as default.



Obr. 14: Nastavení výchozího kompilátoru

Dále je potřeba nastavit jakým způsobem předává IDE parametry kompilátoru a upravit koncovku výstupních objektových souborů pro linker. Z hlavní lišty Code Blocks tedy vybereme nabídku Settings a zvolíme Compiler. Otevře se dialogové okno s několika záložkami. Zde zvolíme záložku Other settings.



Obr. 15: Pokročilé nastavení

Otevře se další nabídka. V poli Command nejdříve vybereme Compile single file to object file a pole Command line macro upravíme na tvar `$compiler $options $includes -o $object -c $file`. Tímto říkáme v jakém tvaru se budou předávat parametry a soubory kompilátoru. Pokud si tento zápis rozdělíme na jednotlivé části, získáme:

\$compiler – volání kompilátoru (sdcc.exe)

\$options – parametry pro nastavení kompilátoru

\$includes – cesty k souborům nebo jednotlivé soubory vkládané do kódu pomocí `#include`

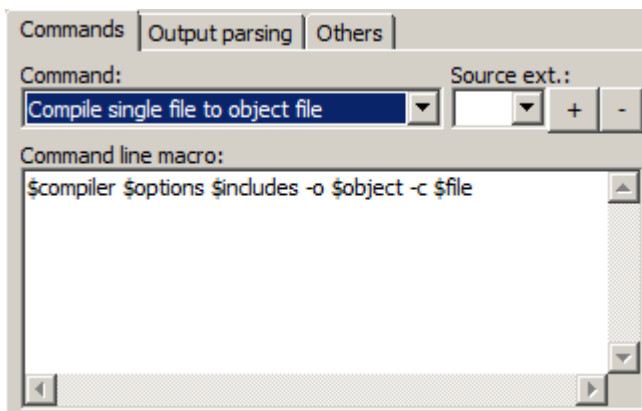
-o \$object – výstupní objektový soubor překladače

-c – parametr, který kompilátoru říká, že chceme pouze kompilovat (neprovádí se linkování)

\$file – vstupní soubor se zdrojovým kódem v jazyce C (např. main.c)

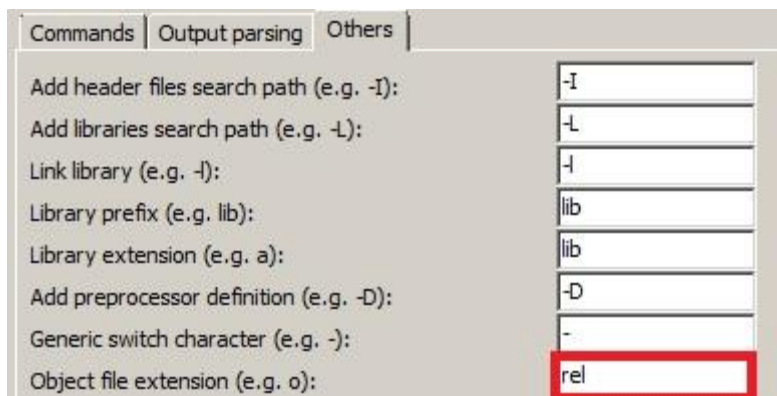
Parametr `-c` je použit pro rozdělení buildovacího procesu na kompilování a linkování které se provádí zvlášť. Je to z důvodu vícesouborových projektů kde je třeba nejprve všechny soubory zkompilovat zvlášť a až pak předat výsledné objektové soubory k linkování. Exis-

tuje zde tedy ještě možnost nastavení, kde lze upravit jak parametry předávat pro linker. Tu však není třeba nijak upravovat.



Obr. 16: Nastavení předávaných parametrů

Dále ve stejném okně vybereme ještě záložku Others. Zde jen upravíme koncovku výstupního objektového souboru kompilátoru na *rel*.



Obr. 17: Nastavení koncovky *rel*

Nyní je kompilátor připraven k použití. Výhodou tohoto sestavení je hlavně jednoduchost a přehlednost díky správci projektů. Kompilování je zjednodušeno na stisk tlačítka a není potřeba zadávat parametry ručně.

6 PRÁCE S PROSTŘEDÍM

V následující části probereme několik parametrů pro nastavení kompilátoru SDCC, dále si popíšeme jak nastavit založit a nastavit projekt ve zvoleném prostředí.

6.1 Parametry pro nastavení SDCC

Parametry jsou kompilátoru předávány ve tvaru *sdcc [parametry] soubor*. Kde *sdcc* je samotný kompilátor a soubor značí vstupní soubor s kódem. Níže následuje výpis některých podstatných parametrů.

--std-c99 - slouží pro kompilování v normě ISO C99 (existují ještě --std-c89 a --std-c11 pro normy ISO C90 a ISO C11).

-mhc08 - generování kódu pro Freescale/Motorola HC08 (68HC08) řadu procesorů.

-I<cesta> - dodatečné lokace kde bude preprocesor hledat hlavičkové soubory *.h.

-L --lib-path <absolutní cesta> - říká linkeru kde hledat dodatečné knihovny. Cesta musí být zadána absolutně.

--opt-code-speed – kompilátor bude optimalizovat kód pro rychlost, výsledný kód bude ale nejspíše větší.

--opt-code-size – kompilátor optimalizuje kód na co nejmenší velikost výsledku, kód však může být pomalejší.

-c --compile-only – zkompiluje a asembluje kód, ale nezavolá linker.

--out-fmt-s19 – výstup linkeru (výsledný objektový kód) bude ve formátu Motorola S19 místo výchozího Intel IHX.

-o <cesta/soubor> - výstupní cesta kde bude nebo soubor kde bude uložen výsledek operace

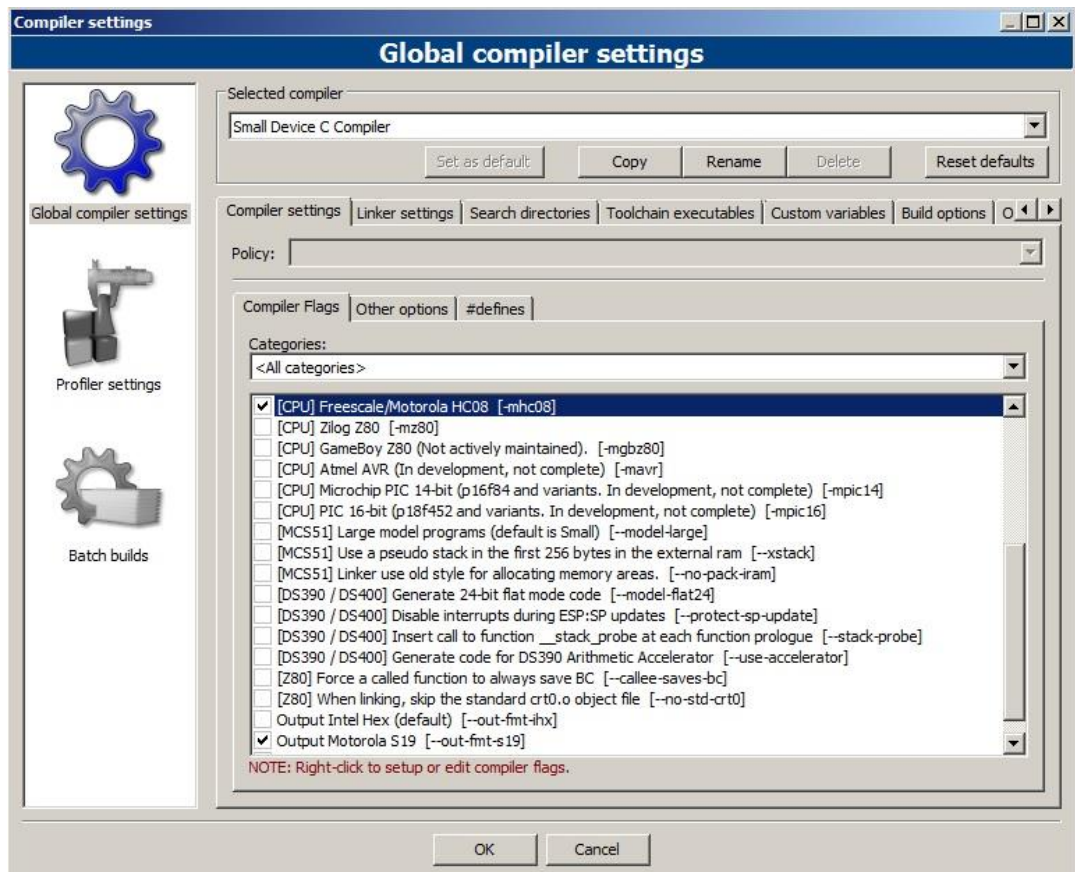
--xram-loc <hodnota> - počáteční lokace externí RAM (muže být hexadecimání nebo decimální číslo, např. --xram-loc 0x8000 nebo --xram-loc 32768). Výchozí hodnota je 0.

--code-loc <hodnota> - počáteční lokace kódového segmentu paměti. Výchozí hodnota 0.

--data-loc <hodnota> - počáteční lokace datového segmentu interní RAM.

--stack-loc <hodnota> - umístění zásobníku v paměti

Všechny tyto parametry se Code Blocks nastavují v nastavení kompilátoru. Opět tedy vybereme Settings z hlavní lišty a Compiler. Okno se otevře na záložce Compiler kde je možno zvolit většinu důležitého nastavení pomocí zaškrťávacích polí. Zde je pro nás důležité zvolit typ procesoru čili zatrhnout možnost Freescale/Motorola HC08. Dále si můžeme změnit formát výstupního formátu volbou Output Motorola S19. V podzáložce Other options pak můžeme přidat `-std-c99` případně `-std-sdcc99` pro specifikaci normy jazyka C.



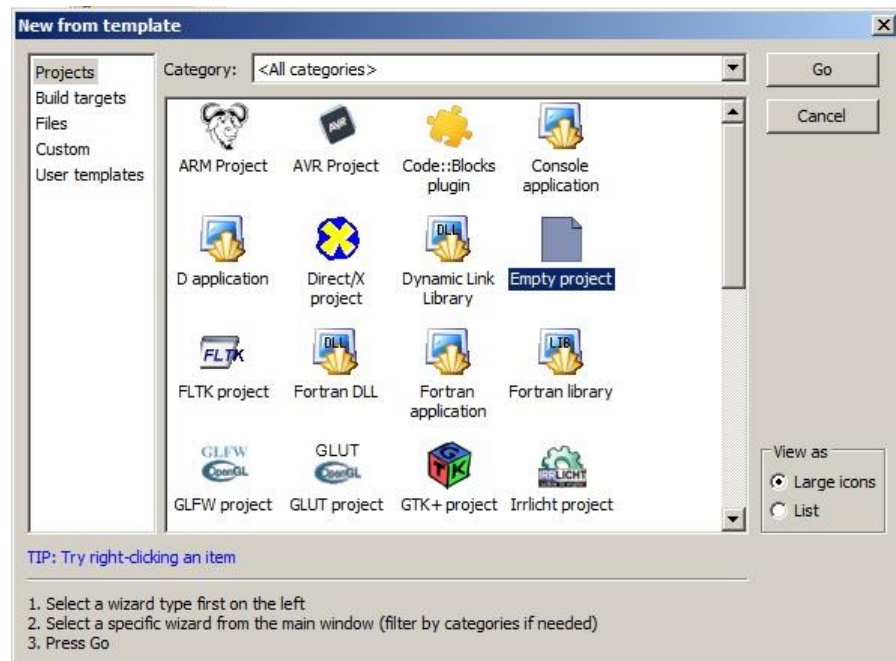
Obr. 18: Nastavení parametrů

V záložce linker pak můžeme přidat knihovny pro linkování pomocí tlačítka Add. Okno Other options zde slouží např. pro dodatečné informace o paměti zařízení (parametry jako `-code-loc` `-data-loc`).

6.2 Založení a nastavení projektu

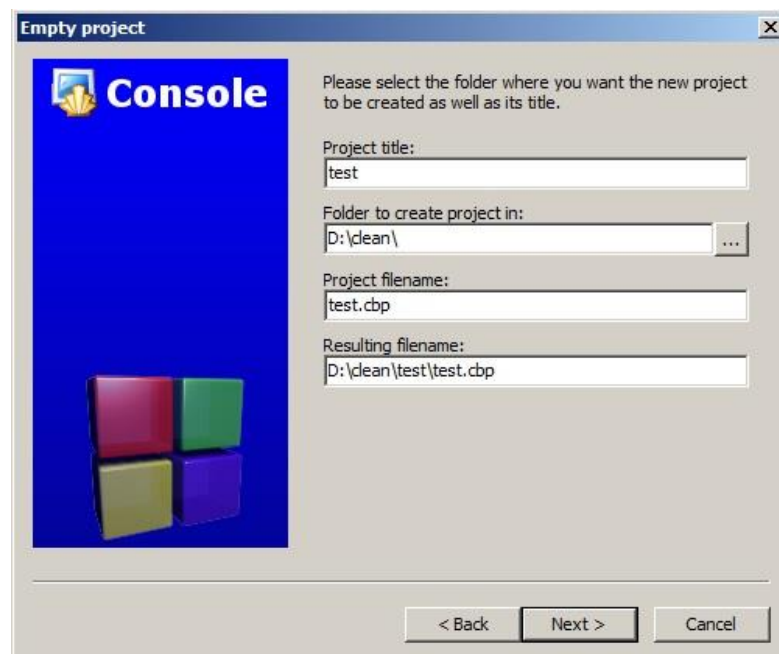
V této části následuje popis založení a nastavení projektu, poté bude ještě vysvětlena práce s projekty obsahujícími více zdrojových souborů.

Pro založení nového projektu vybereme File z hlavní lišty Code Blocks, dále New a Project. Z následné nabídky zvolíme Empty project a potvrdíme tlačítkem Go.



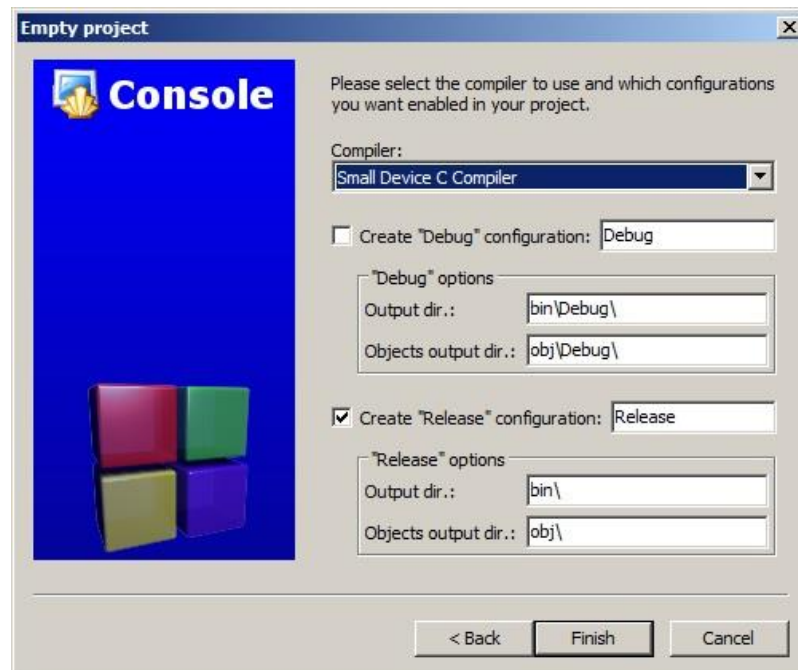
Obr. 19: Výběr typu projektu

V dalším dialogovém okně pak zadáme název projektu a vybereme cílový adresář pro uložení. Dále se přesuneme tlačítkem Next.



Obr. 20: Nastavení názvu projektu

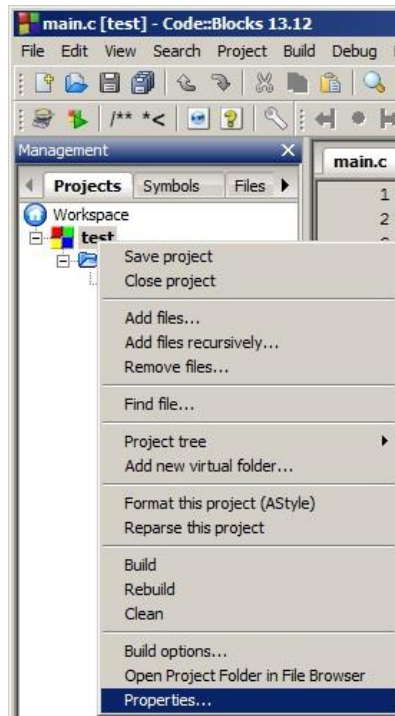
Na poslední stránce se ujistíme, že je v položce Compiler vybrán Small Device C Compiler a z dvou možných voleb necháme zatrženo jen Create „Release“ configuration.



Obr. 21: Výběr kompilátoru

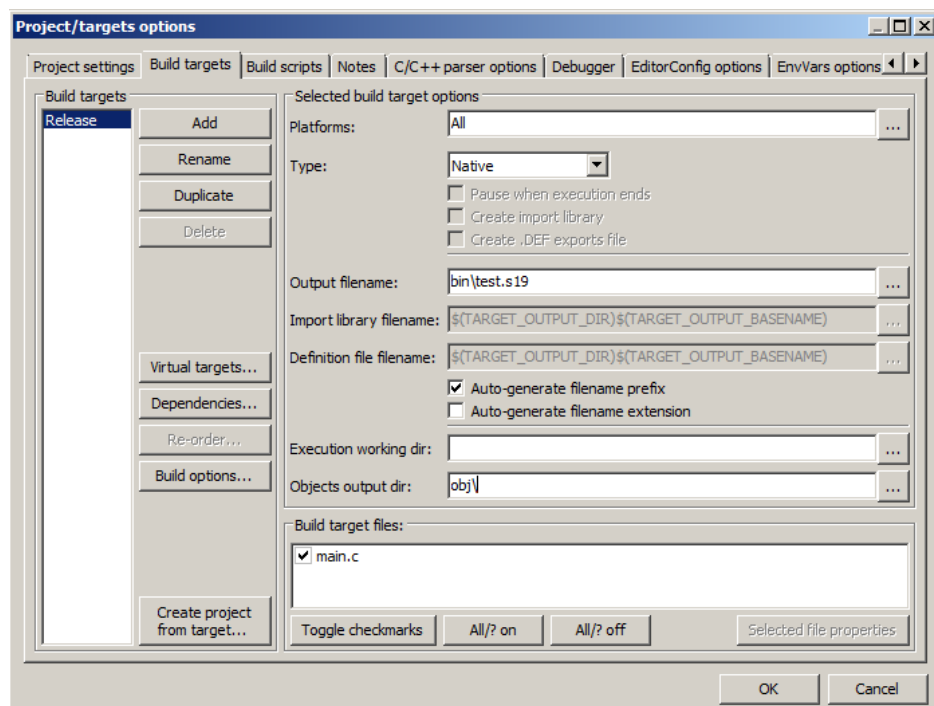
Tvorbu projektu pak dokončíme kliknutím na Finish.

Nyní se přesuneme do nastavení projektu. V projekt manažeru klikneme pravým tlačítkem na název projektu a zvolíme Properties.



Obr. 22: Vlastnosti projektu

Zde se přesuneme do záložky Build targets a položky Type změníme na Native. Tímto je projekt připraven k použití. Nový soubor pak přidáme opět pomocí nabídky File z hlavní lišty, kde vybereme New a zde Empty file.



Obr. 23: Nastavení typu projektu

6.2.1 Projekty s více zdrojovými soubory

Jelikož soubory vícesouborových projektů musíme nejdříve zkompileovat a až pak výsledné soubory předat k linkování, musí být tato činnost rozdělena do dvou kroků. V tom prvním se jednotlivé zdrojové soubory jeden po druhém předávají kompilátoru pro překlad (SDCC neumí kompilovat více souborů najednou). V dalším kroku je nutné všechny výsledné soubory předat linkeru a to v takovém pořadí aby hlavní soubor byl na prvním místě. Pokud máme tedy např. soubory:

foo1.c (obsahující funkce)

foo2.c (obsahuje další funkce)

foomain.c (obsahuje funkci main)

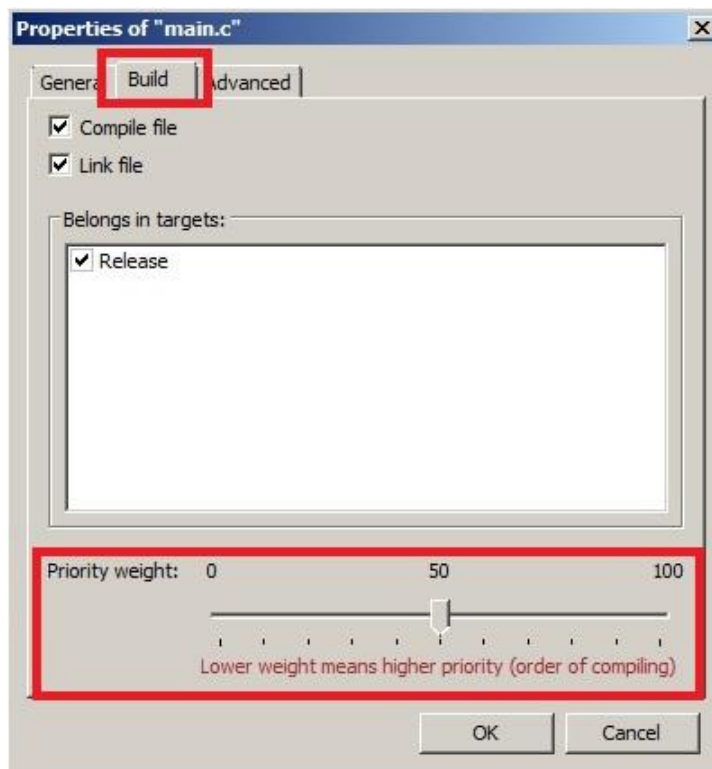
Nejdříve zkompilejeme zvlášť pomocí příkazů:

```
sdcc -c foo1.c, sdcc -c foo2.c a sdcc -c foomain.c
```

Výsledné soubory s příponou .rel pak předáme k linkování příkazem:

```
sdcc foomain.rel foo1.rel foo2.rel
```

S tím, že soubor s funkcí *main* musí být na prvním místě v předávaných souborech. V prostředí Code Blocks toho docílíme nastavením priority pro linkování jednotlivých souborů. Kliknutím pravým tlačítkem myši na soubor v projekt manažeru a výběrem Properties se dostaneme do vlastností souboru. Nastavení priority provedeme pomocí posuvníku ve spodní části záložky Build.



Obr. 24: Nastavení priority souboru

Platí, že menší číslo znamená větší prioritu. Soubor s hlavní funkcí by tedy měl mít vždy nejmenší číslo.

7 UKÁZKOVÝ PROGRAM

Jako ukázkový poslouží program pro blikání LED diody na portu PTC modelu Motorola MC68HC908GP32. K tvorbě byl využit soubor *mc68hc908gp32.h* obsahující deklarace periférií, který je součástí instalace SDCC.

7.1 Popis programu

Program začíná vložením *mc68hc908gp32.h* pomocí

```
#include <mc68hc908gp32.h>
```

Dále následuje deklarace funkce *cekej* a hlavní funkce *main*.

```
void cekej(void);
```

```
void main(void) {
```

```
    DDRC |= 0x04;
```

```
    while(1) {
```

```
        PTC ^= 0x04; // prepínání PTC2 pomocí XOR
```

```
        cekej();    // čekání mezi blikáními
```

```
    }
```

```
}
```

Kde *DDRC |= 0x04* nastaví port PTC2 jako výstup. Následuje nekonečná smyčka *while* která pomocí *PTC ^= 0x04*; přepíná PTC2 za využití funkce XOR. Před opakováním celé smyčky se zavolá funkce *cekej* která slouží jako prodleva při přepínání diody. Na konci programu se pak nachází definice funkce *cekej*.

```
void cekej() {
```

```
    unsigned int i;
```

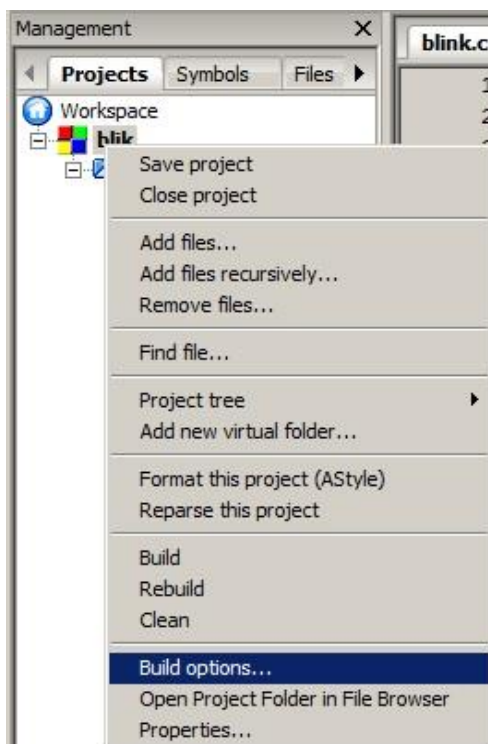
```
    for (i = 0; i < 0xFFFF; i++) PTC;
```

```
}
```

Ta jen inkrementuje proměnnou *i* do šestnáctkové hodnoty FFFF.

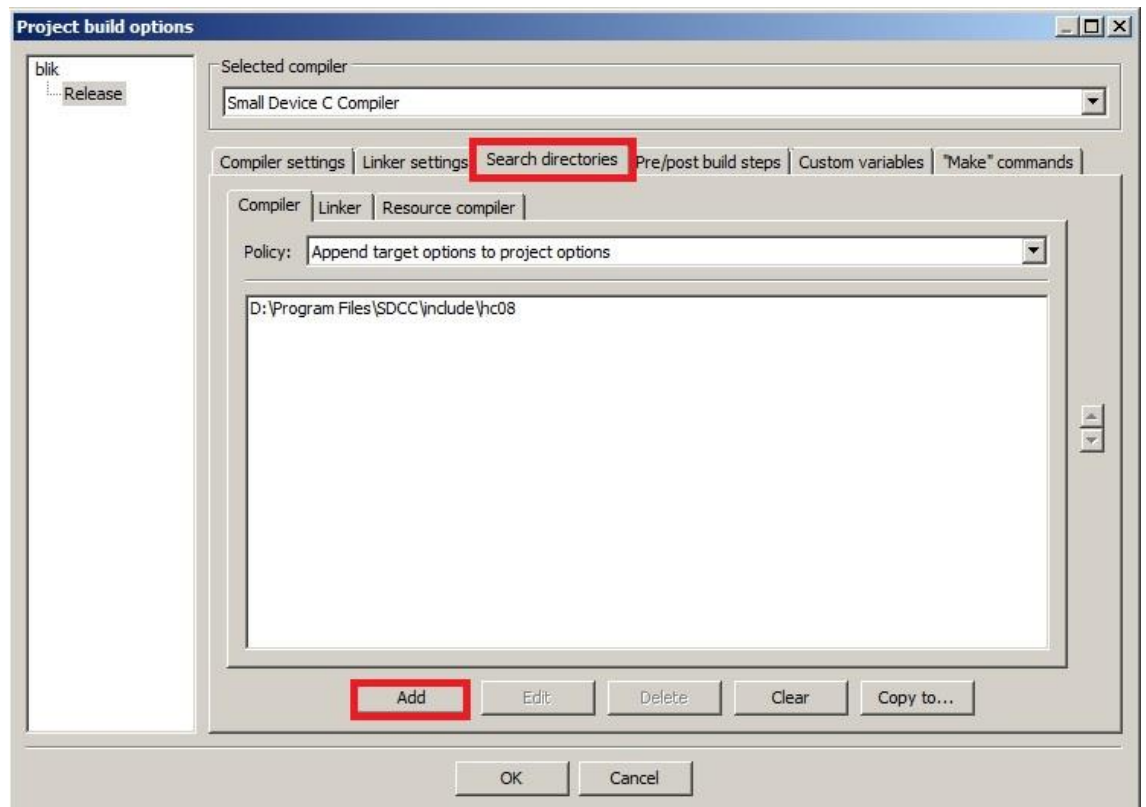
7.2 Nastavení cest a překlad

Vytvoření a nastavení projektu už bylo popsáno výše. Dále v projektu vytvoříme nový soubor, do nějž program píšeme. Před překladem je třeba ještě nastavit cesty, kde bude překladač hledat hlavičkové soubory .h. K nastavení těchto cest se dostaneme např. pravým kliknutím na projekt a volbou Build options.



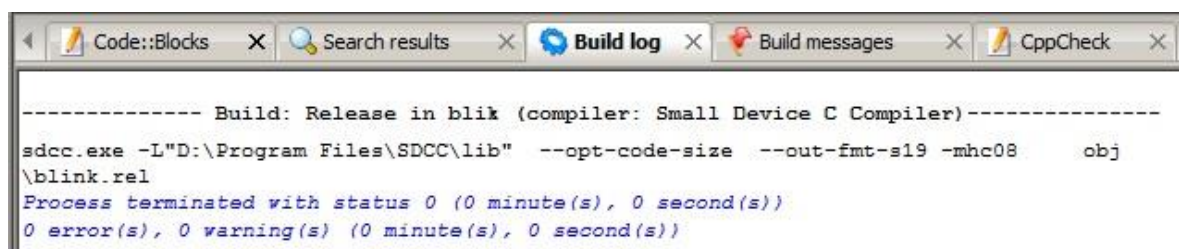
Obr. 25: Build options

V následném okně se přesuneme do záložky Search directories kde pomocí tlačítka Add a jeho dialogového okna najdeme složku s hlavičkovými soubory a přidáme je do seznamu. V mém případě je to například cesta `D:\Program Files\SDCC\include\hc08`.



Obr. 26: Nastavení cest

Nyní můžeme program přeložit např. pomocí klávesové zkratky CTRL-F9 nebo kliknutím pravým tlačítkem na název projektu a volbou možnosti Build. Pokud vše proběhne v pořádku měl by překladač vrátit hodnotu 0 jako na obrázku 27.



Obr. 27: Úspěšný překlad

7.3 Výstupní soubory kompilátoru

Na závěr uvedeme popis výstupních souborů SDCC.

soubor.asm – assemblerovský zdrojový soubor vytvořený kompilátorem

soubor.lst – assemblerovský seznam

soubor.rst – assemblerovský seznam doplněný o informace z linkeru

soubor.sym – seznam symbolů vytvořený assemblerem

soubor.rel – objektový soubor vytvořený assemblerem, vstup linkeru

soubor.map – mapa paměti pro nahrávací modul vytvořená linkerem

soubor.mem – soubor se záznamem o využití paměti

soubor.ihx nebo soubor.s19 - nahrávací modul ve formátu Intel hex format nebo Motorola S19 (formát Motorola S19 lze nastavit pomocí --out-fmts19)

soubor.adb – přechodný soubor obsahující ladící informace potřebné k vytvoření .cdb souboru (pomocí --debug)

soubor.cdb – volitelný soubor (pomocí --debug) obsahující ladící informace

soubor.omf – volitelný soubor formátu AOMF nebo AOMF51 obsahující ladící informace (opět vytvářen pomocí --debug). Význam této zkratky je absolute object module format a je používán některými ladícími nástroji.

soubor.dump* - odkládací soubor pro ladění samotného kompilátoru (generován pomocí --dumpall)

7.4 Ladění a nahrání programu do paměti mikro počítače

Vzhledem k tomu, že Code Blocks je univerzální vývojové prostředí, neobsahuje ladící nástroje zaměřené mikro počítače HC(S)08 ani nástroj pro nahrání výsledného programu do mikro počítače. K simulaci je možno využít konzolový simulátor uCsim (shc08.exe) nacházející se v podadresáři bin adresáře SDCC.

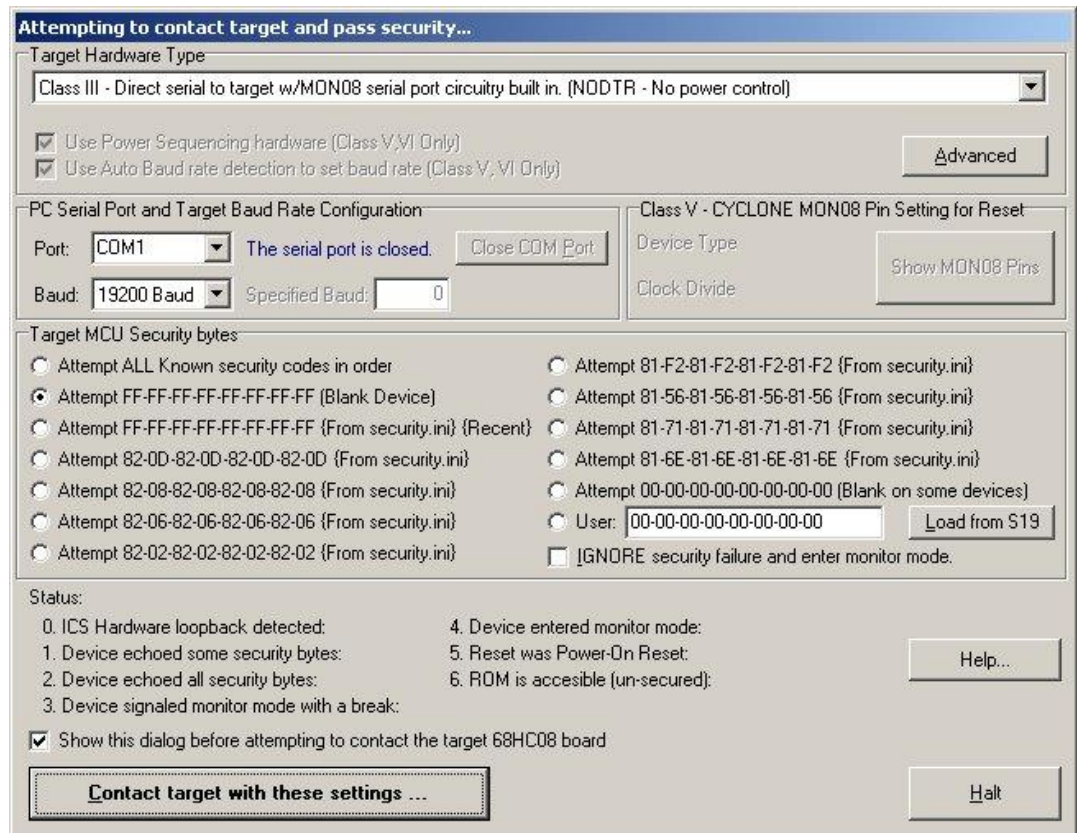
K nahrání programu do paměti je možno použít např. programátor od firmy P&E Micro-computer dostupný z

http://www.pemicro.com/products/product_viewDetails.cfm?product_id=27.

Ten je dostupný zdarma v konzolové verzi a po bezplatné registraci i ve verzi s GUI. Použití programu pro sériové rozhraní je následující.

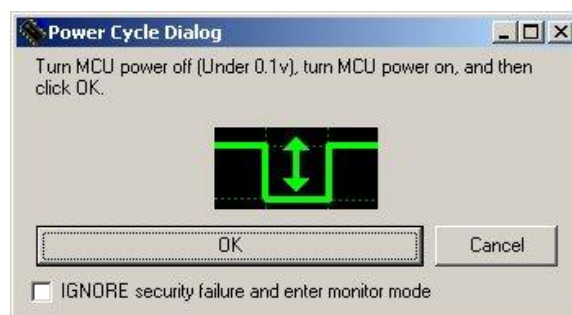
Nejdříve nastavíme parametry připojení tak, že zvolíme nastavení Class III jako typ cílového hardware. Poté zvolíme sériový port v položce Port a rychlost komunikace v položce

Baud. Dále následuje nastavení bezpečnostních bytů. Pokud tyto byty neznáme můžeme paměť zařízení jen smazat. Tyto byty se také dají načíst externě z .s19 souboru.



Obr. 28: Dialog nastavení připojení

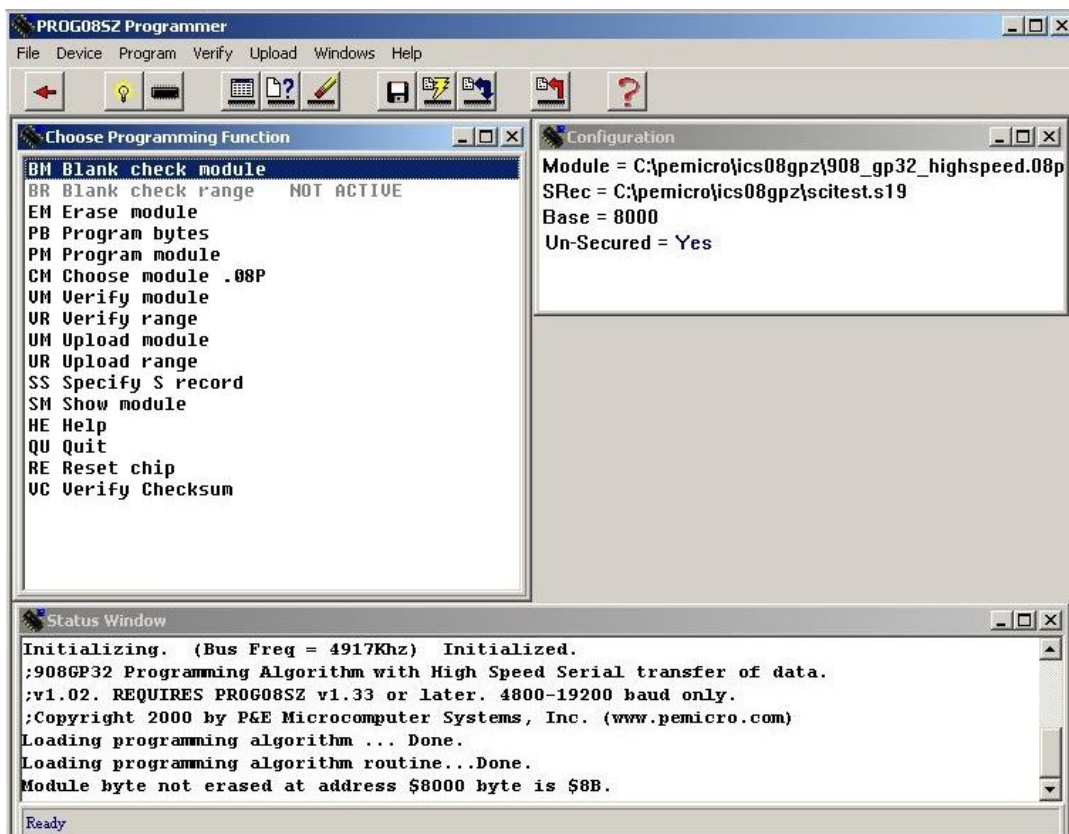
Po úspěšném připojení budete dotázáni na vypnutí a zapnutí napájení mikropočítače, které potvrdíte tlačítkem OK.



Obr. 29: Dialog napájení

Poté se vás program dotáže na programovací algoritmus. Ty jsou ke stažení na adrese http://www.pemicro.com/support/flash_list_menu.cfm.

Po výběru programovacího algoritmu se zpřístupní funkce programátoru dostupné pomocí tlačítek v horní liště. Pokud paměť obsahuje předešlý program, vymažeme ji tlačítkem *Erase module*. Následně vybereme náš nový program pomocí *Specify S record* kde vybereme vstupní .s19 soubor který se po překladu programu nachází ve složce projektu a následně jej nahrajeme do FLASH paměti mikropočítače pomocí tlačítka *Program module*.



Obr. 30: Okno programu

ZÁVĚR

Cílem práce bylo vyhledat jednotlivé vývojové nástroje pro mikropočítače HCS08 nebo celá IDE ať už zdarma nebo komerční a s těmi se seznámit. Poté vybrat jednotlivé nástroje nebo vytvořit vlastní IDE ze zdarma dostupných nástrojů pro editaci, překlad, ladění a nahrání programu do mikropočítače. K tomuto pak vytvořit podrobný návod s ukázkovým programem.

V práci je uveden popis několika komerčních IDE (jmenovitě ImageCraft ICC08, Code-warrior pro HCS08 a Cosmic C6808) a volně dostupného SDCC kompilátoru se sadou nástrojů a knihoven.

Jako výchozí prostředí bylo použito Code Blocks IDE s překladačem SDCC. Code Blocks byl upřednostněn hlavně díky své modulárnosti a tedy možnosti celkem jednoduše nastavit externí překladač.

Práce obsahuje návod pro nastavení kompilátoru ve jmenovaném IDE, nastavování SDCC pomocí parametrů, popis předávání a nastavení parametrů v Code Blocks.

Dále je uveden návod použití daného sestavení obsahující popis založení a nastavení projektu a popis problematiky vícesouborových projektů. Práce je pak zakončena ukázkovým programem s ukázkou nastavení cest pro ke hlavičkovým souborům a popisem práce se softwarem nahrání výsledného programu do paměti mikropočítače.

SEZNAM POUŽITÉ LITERATURY

- [1] ANTOŠOVÁ, Marcela a Vratislav DAVÍDEK. Číslicová technika. Dotisk prvního vydání. České Budějovice: Kopp, 2005, 286 s. ISBN 80-7232-207-9.
- [2] FREESCALE SEMICONDUCTOR. HCS08 Family: Reference Manual. 2007. Dostupné z: http://www.freescale.com/files/microcontrollers/doc/ref_manual/HCS08RMV1.pdf
- [3] Guide to processor history. In: Syntheticlones silicone chips [online]. 2003 [cit. 2014-04-16]. Dostupné z: <http://www.chips.5u.com/idxhst.html>
- [4] FREESCALE SEMICONDUCTOR. CPU08 Central Processor Unit: Reference Manual. 2006. Dostupné z: http://www.freescale.com/files/microcontrollers/doc/ref_manual/CPU08RM.pdf
- [5] GALEČKOVÁ, Iveta a Jan GALEČEK. Počítač: základní vědomosti. Olomouc: Alda, 1994, 93 s. ISBN 80-856-0037-4.
- [6] BARR, Michael a Anthony J MASSA. Programming embedded systems: with C and GNU development tools. 2nd ed. Sebastopol: O'Reilly, 2006, xxi, 301 s. ISBN 978-0-596-00983-0.
- [7] Mikroprocesor, mikropočítač, mikrokontrolér, DSP a DSC. In: Mikrokontroléry PIC: Web o číslicové technice a mikrokontrolérech PIC [online]. 2012 [cit. 2014-05-30]. Dostupné z: <http://mikrokontrolery-pic.cz/zaciname/mikroprocesor-mikropocitac-mikrokontroler/>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

RAM	Paměť s přímým přístupem nebo paměť s libovolným výběrem.
ROM	Paměť pouze pro čtení.
FLASH	Elektricky programovatelná paměť s libovolným přístupem.
KB	Jednotka množství dat.
LED	Dioda emitující světlo.
CPU	Centrální procesorová jednotka.
GUI	Grafické rozhraní programu.
IDE	Vývojové prostředí.

SEZNAM OBRÁZKŮ

Obr. 1: Počáteční vývoj mikroprocesorů firem Intel a Motorola[3]

Obr. 2: Registry CPU[4]

Obr. 3: Akumulátor[4]

Obr. 4: Indexový registr[4]

Obr. 5: Ukazatel zásobníku[4]

Obr. 6: Programový čítač[4]

Obr. 7: Registr příznaků[4]

Obr. 8: Vývoj programu[6]

Obr. 9: Příprava binárního obrazu programu[6]

Obr. 10: Vzdálené debugování[6]

Obr. 11: CodeWarrior IDE

Obr. 12: Cosmic IDE

Obr. 13: Imagecraft IDE

Obr. 14: Nastavení výchozího kompilátoru

Obr. 15: Pokročilé nastavení

Obr. 16: Nastavení předávaných parametrů

Obr. 17: Nastavení koncovky rel

Obr. 18: Nastavení parametrů

Obr. 19: Výběr typu projektu

Obr. 20: Nastavení názvu projektu

Obr. 21: Výběr kompilátoru

Obr. 22: Vlastnosti projektu

Obr. 23: Nastavení typu projektu

Obr. 24: Nastavení priority souboru

Obr. 25: Build options

Obr. 26: Nastavení cest

Obr. 27: Úspěšný překlad

Obr. 28: Dialog nastavení připojení

Obr. 29: Dialog napájení

Obr. 30: Okno programu

SEZNAM PŘÍLOH

P I CD nosič

PŘÍLOHA P I: CD NOSIČ

Obsah disku:

- Ukázkový projekt a BP v elektronické podobě