UNIVERZA V LJUBLJANI
FAKULTETA ZA ELEKTROTEHNIKO

MAGISTRSKA NALOGA

ROBERT ROSTOHAR

UNIVERZA V LJUBLJANI
FAKULTETA ZA ELEKTROTEHNIKO

MAGISTRSKA NALOGA

Razvojno mikrokrmilniško okolje M68HC11 za pedagoške potrebe

Robert Rostohar

Mentor: prof. dr. Franc Bratkovič

LJUBLJANA, 1999

Zahvala

Iskreno se zahvaljujem vsem, ki so mi pomagali pri izdelavi moje magistrske naloge. Še posebej pa se zahvaljujem:

- mentorju prof. dr. Francu Bratkoviču, ki mi je omogočil izdelavo magistrske naloge
- dr. Tadeju Tumi, ki me je z koristnimi nasveti usmerjal skozi celotno delo
- staršem in prijateljem, ki so me vzpodbujali pri delu in mi stali ob strani

Kazalo

Seznam uporabljenih kratic **1**

1. Povzetek **3**

2. Uvod **7**

3. Mikrokrmilniški sistem **9**
3.1 Zgradba sistema 9
3.1.1 Mikrokrmilnik M68HC11A1 13
3.1.2 Zunanji pomnilnik 15
3.1.3 Naslovni dekoder 16
3.1.4 Paraleni vmesnik PIA6821 17
3.1.5 Priključek RS232 s pretvornikom nivojev 19
3.1.6 Priključek za vhodno / izhodne enote 20
3.1.7 Priključek za razširitev 21
3.1.8 Napajanje 22
3.2 Naslovni prostor 23
3.3 Priključitev na osebni računalnik 24
3.4 Prototipni sistem 25

4. BIOS - Basic Input Output System **27**
4.1 Lastnosti 27
4.2 Komunikacija 28
4.3 Zgradba programa 28
4.4 Ukazi 30
4.4.1 Zapis podatkov (Output) 30
4.4.2 Branje podatkov (Input) 31
4.4.3 Branje kazalca sklada (Stack) 32
4.4.4 Zagon programov (Execute) 34
4.5 Odziv na prekinitve in reset 34
4.6 Omejitve 36

5. Programsko razvojno orodje **37**
5.1 Značilnosti 37
5.2 Grafični vmesnik 38
5.2.1 Datoteke 39
5.2.2 Okno “Editor” 40

5.2.3 Modul “Assembler” 41
5.2.4 Modul “Download” 42
5.2.5 Modul “Debugger” 42
5.2.5.1 Okno “Disassembler” 43
5.2.5.2 Okna “Registers” 47
5.2.5.3 Okno “Stack” 48
5.2.5.4 Okno “Watch” 49
5.2.5.5 Okno “Memory” 50
5.2.6 Serijski terminal 51
5.3 Instalacija 52

6. Zaključek 53

Priloga: Izpis programa BIOS-a 55

Seznam uporabljenih virov 61

Izjava 63

Seznam uporabljenih kratic

AccA	Accumulator A
AccB	Accumulator B
AccD	Double Accumulator
ACK	Acknowledge
AS	Address Strobe
ADC	Analog to Digital Converter
BIN	Binary
BIOS	Basic Input Output System
CCR	Condition Code Register
COP	Computer Operating Properly
CPU	Central Processing Unit
CS	Chip Select
CTS	Clear To Send
DEC	Decimal
DIL	Dual In Line
EEPROM	Electrical Erasable Programmable ROM
EPROM	Erasable Programmable ROM
GND	Ground
HEX	Hexadecimal
IC	Integrated Circuit
IDE	Integrated Development Environment
IDT	Integrated Development Tool
IRQ	Interrupt Request
IX	Index Register X
IY	Index Register Y
LED	Light Emitting Diode
LSB	Less Significant Bit
LVI	Low Voltage Inhibit
MSB	Most Significant Bit
NAK	Not Acknowledged
NMI	Non-maskable Interrupt
OCT	Octal
OE	Output Enable
PC	Program Counter
PCB	Print Circuit Board

PLCC	Plastic Lead Chip Carrier
PIA	Parallel Interface Adapter
RAM	Random Access Memory
ROM	Read Only Memory
RTI	Real Time Interrupt
RTS	Request To Send
RXD	Receive Data
SCI	Serial Communications Interface
SIL	Single In Line
SP	Stack Pointer
SPI	Serial Peripheral Interface
SWI	Software Interrupt
TTL	Transistor Transistor Logic
TXD	Transmit Data
VRL	Voltage Reference Low
VRH	Voltage Reference High
WE	Write Enable

1. Povzetek

Ta magistrska naloga predstavlja integrirano razvojno okolje “HC11 IDE” za mikrokrmilnik Motorola M68HC11. Naloga obsega izdelavo samega večnamenskega mikrokrmilniškega sistema (Development Board) s pripadajočim BIOS-om in izdelavo programske opreme za osebni računalnik, ki predstavlja integrirano razvojno orodje (IDT).

Mikrokrmilniški sistem je zelo univerzalen, zato ga lahko v kombinaciji z razvojnim orodjem uporabljamo za izdelavo najrazličnejših aplikacij. Ko pa je aplikacija oz. program razvit, pa lahko sistem deluje tudi samostojno in služi kot prototipni sistem.

Mikrokrmilniški sistem sestavljajo naslednje enote:

- mikrokrmilnik Motorola M68HC11A1
- 8 kBytes EPROM (pomnilnik za BIOS ali samostojni program)
- 8 kBytes RAM (uporabniški programski pomnilnik)
- 8 kBytes RAM (uporabniški podatkovni pomnilnik)
- paralelni vmesnik PIA6821
- priključek RS232 s pretvornikom nivojev za serijski vmesnik
- priključek za vhodno / izhodne enote (Input / Output Connector)
- priključek za razširitev (Expand Connector)
- priključek za napajanje (Power Supply)
- stabilizator napajanja in blokiranje pri podnapetosti (LVI)
- tipka za reset (Manual Reset)

Sistem je zasnovan na 8-bitnem mikrokrmilniku M68HC11A1 (Single-Chip Microcontroller), ki ima vgrajenih že celo vrsto enot. Na vezju je dodanih še nekaj enot oz. gradnikov, ki še dodatno povečajo zmogljivosti samega mikrokrmilnika in omogočajo priključitev še raznih dodatnih zunanjih enot, za katere se odloči uporabnik. Zaradi tega je celoten sistem zelo odprt in predstavlja učinkovito orodje za razvoj aplikacij.

Za nalaganje programov (Download) v mikrokrmilnik in bralno - pisalni dostop do poljubne pomnilniške lokacije ter za zagon programov in možnost izvajanja programov po korakih skrbi “Basic Input Output System” (BIOS). To je poseben strojni program, ki je zapisan v EPROM-u mikrokrmilniškega sistema in omogoča hitro komunikacijo med sistemom in nadzorno enoto, ki poteka preko serijskega vmesnika RS232. Mikrokrmilnik sprejema ukaze od nadzorne enote in se ustrezno odzove. Nadzorna enota je običajno osebni računalnik.

Glavne značilnosti mikrokrmilniškega sistema:

- mikrokrmilnik M68HC11A1 v razširjenem načinu delovanja (Extended Mode)
  - 8-bitni CPU z izboljšanim M6800/M6801 naborom ukazov (Instruction Set)
  - 256 Bytes RAM
  - 512 Bytes EEPROM
  - 12 vhodno / izhodnih linij z dodatnimi funkcijami
  - 8-kanalni analogno / digitalni pretvornik (ADC)
  - 16-bitni časovnik (Timer) s posebni funkcijami (Input Capture, Output Compare)
  - sinhroni serijski vmesnik (SPI)
  - asinhroni serijski vmesnik (SCI)
  - prekinitveni sistem
- pomnilnik in naslovni prostor
  - 8 kBytes EPROM
  - 16 kBytes RAM
  - 4 ∗ 8 kBytes za zunanje enote (možnost razširitve)
- paralelni vmesnik PIA6821
  - 2 ∗ 8-bitna dvosmerna vrata (Bi-directional Port)
  - 4 dodatne vhodno / izhodne linije s posebnimi funkcijami
  - prenos podatkov s protokolom (Handshake Control)
  - linije z visoko impedančnim stanjem (High Impedance 3-State Lines)
- Basic Input Output System - BIOS
  - nalaganje uporabniških programov
  - zagon programov in možnost izvajanja programov po korakih
  - bralno / pisalni dostop do celotnega naslovnega prostora (64 kBytes)
  - komunikacija preko RS232 asinhronega serijskega vmesnika (SCI)

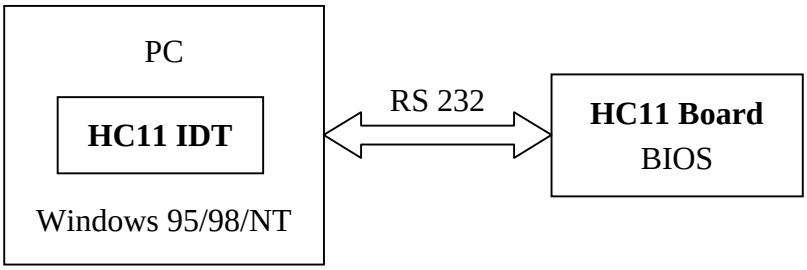
Za načrtovanje aplikacij je dodatno narejen posebni programski paket “HC11 IDT” (integrirano razvojno orodje), ki je grafični vmesnik za komunikacijo z uporabnikom in deluje na osebnem računalniku z Windows 95/98/NT operacijskim sistemom. To je uporabniško prijazen programski paket, ki predstavlja zelo močno razvojno orodje in vsebuje vse, kar je potrebno pri razvoju programov. Razvojno orodje je povezano z mikrokrmilniškim sistemom preko RS232 serijskega vmesnika in uporablja funkcije, ki jih nudi njegov BIOS.

Z razvojnim orodjem lahko pišemo programe, jih urejamo, odpravljamo morebitne napake v sintaksi in jih prevajamo v strojno kodo ter jih nato naložimo v mikrokrmilniški sistem. Naložene programe nato lahko poženemo ali pa jih izvajamo po korakih. Lahko uporabljamo tudi posebne prekinitvene točke tam, kjer želimo, da se program ustavi. Poleg tega lahko pregledujemo in urejamo celotni naslovni prostor, v katerem so poleg pomnilnikov tudi notranji registri mikrokrmilnika, paralelnega vmesnika in morebitnih dodatnih zunanjih enot. Vse to nam omogoča, da lahko preverimo, ali napisani program deluje, kot je bilo načrtovano.

Glavne značilnosti razvojnega orodja "HC11 IDT":

- vgrajen urejevalnik teksta (Editor)
- vgrajen makro-zbirniški prevajalnik (Macro Assembler)
- vgrajen inverzni prevajalnik (Disassembler)
- podpora simbolov (Symbols)
- nalaganje programov (Download)
- zagon programov (Run)
- izvajanje programov po korakih (Single Stepping)
- prekinitvene točke (Breakpoints)
- opazovalni elementi (Watches)
- pregled in urejanje vsebine vseh registrov
- pregled in urejanje celotnega naslovnega prostora (64 kBytes Memory Map)
- podpora osnovnih številskih sestavov (DEC, BIN, OCT in HEX)
- vgrajen serijski terminal
- sprotna pomoč (Online Help)

Blok shema strojne in programske opreme integriranega razvojnega mikrokrmilniškega okolja "HC11 IDE" je prikazana na sliki 1.1.



Slika 1.1: "HC11 IDE" - strojna in programska oprema

Mikrokrmilniški sistem skupaj z BIOS-om in zmogljivim programskim razvojnim orodjem omogoča celotno razvojno okolje za mikrokrmilnik M68HC11. Po svojih lastnostih in zmogljivostih se približuje podobnim profesionalnim razvojnim okoljem, po ceni pa spada med cenene izvedbe.

Razvojno mikrokrmilnško okolje M68HC11 je predvideno predvsem za pedagoško delo, saj ga odlikuje univerzalni in enostavni mikrokrmilniški sistem ter prijazen grafični vmesnik, ki omogoča enostavno delo. Namenjeno je še posebej študentom, da si z majhnimi stroški izgradijo svoj lastni sistem, ki ga lahko uporabljajo v najrazličnejše namene.

2. Uvod

Na vsakem koraku življenja nas vedno bolj spremljajo mikroprocesorji in mikrokrmilniki. Njihovo področje uporabe se vedno bolj razširja in jih danes najdemo že skoraj v vsaki malo zahtevnejši elektronski napravi, kjer nadomeščajo obsežna digitalna logična vezja. V zadnjih nekaj desetletjih smo priča zelo hitremu razvoju mikrokrmilnikov. Prvi so bili relativno enostavni in počasni ter majhnih zmogljivosti. Novejši pa so kompleksni, izredno hitri in zelo zmogljivi, zato se uporabljajo na najrazličnejših področjih. Vseeno pa imajo mnogo stvari še vedno skupnih. Čeprav so novejši mikrokrmilniki hitrejši, imajo močnejšo procesorsko enoto in veliko več vgrajenih enot, pa so načini izvajanja programov in razvoja aplikacij podobni. Če dobro poznamo lastnosti enostavnejših mikrokrmilnikov, lahko to v večini primerov uporabimo tudi pri zahtevnejših mikrokrmilnikih.

Mikrokrmilniki Motorola M68HC11 spadajo med zmogljivejše 8-bitne mikrokrmilnike. Imajo izpopolnjeno in močno procesorsko enoto in kar nekaj vgrajenih dodatnih enot. Primerni so za realizacijo raznih aplikacij, kar se kaže tudi v praksi, saj jih najdemo kot krmilne enote v različnih elektronskih napravah.

Cilj te naloge je izdelati integrirano razvojno okolje za omenjeni mikrokrmilnik M68HC11, ki se bo uporabljal predvsem v pedagoške namene. Okolje je sestavljeno iz mikrokrmilniškega sistema in integriranega razvojnega orodja. Pri tem mora biti mikrokrmilniški sistem univerzalen, a še vedno enostaven in cenen, da ga lahko študenti sami zgradijo s pomočjo enostavnih napotkov in brez velikih stroškov. Integrirano razvojno orodje pa je programski paket za osebni računalnik, ki predstavlja prijazen in enostaven grafični vmesnik. Vsebuje vse potrebne komponente za pisanje, prevajanje, nalaganje, izvajanje in preverjanje programov.

Izdelano integrirano razvojno okolje predstavlja strojno in programsko opremo, ki omogoča profesionalno načrtovanje aplikacij za omenjeni mikrokrmilnik. V primerjavi z obstoječimi razvojnimi sistemi je njegova prednost predvsem v nizki ceni in univerzalnosti samega mikrokrmilniškega sistema ter v zelo zmogljivem programskem paketu za razvoj programov, ki pa je še vedno enostaven za uporabo. Ta programski paket je bil narejen z zgledovanjem po profesionalnih razvojnih orodjih in tako zajema vsa potrebna orodja, ki omogočajo in olajšajo razvoj aplikacij. Poleg načrtovanja programov nam v povezavi z mikrokrmilniškim sistemom omogoča tudi preverjanje izvajanja programov v realnem času (Real Time). Vse to kaže na zmogljivo integrirano razvojno okolje, ki je zaradi nizke cene dostopno širokemu krogu uporabnikov. Razvojno okolje je zato zelo primerno za študente, saj ga lahko uporabljajo vsi, od začetnikov naprej pa vse do izkušenih na tem področju.

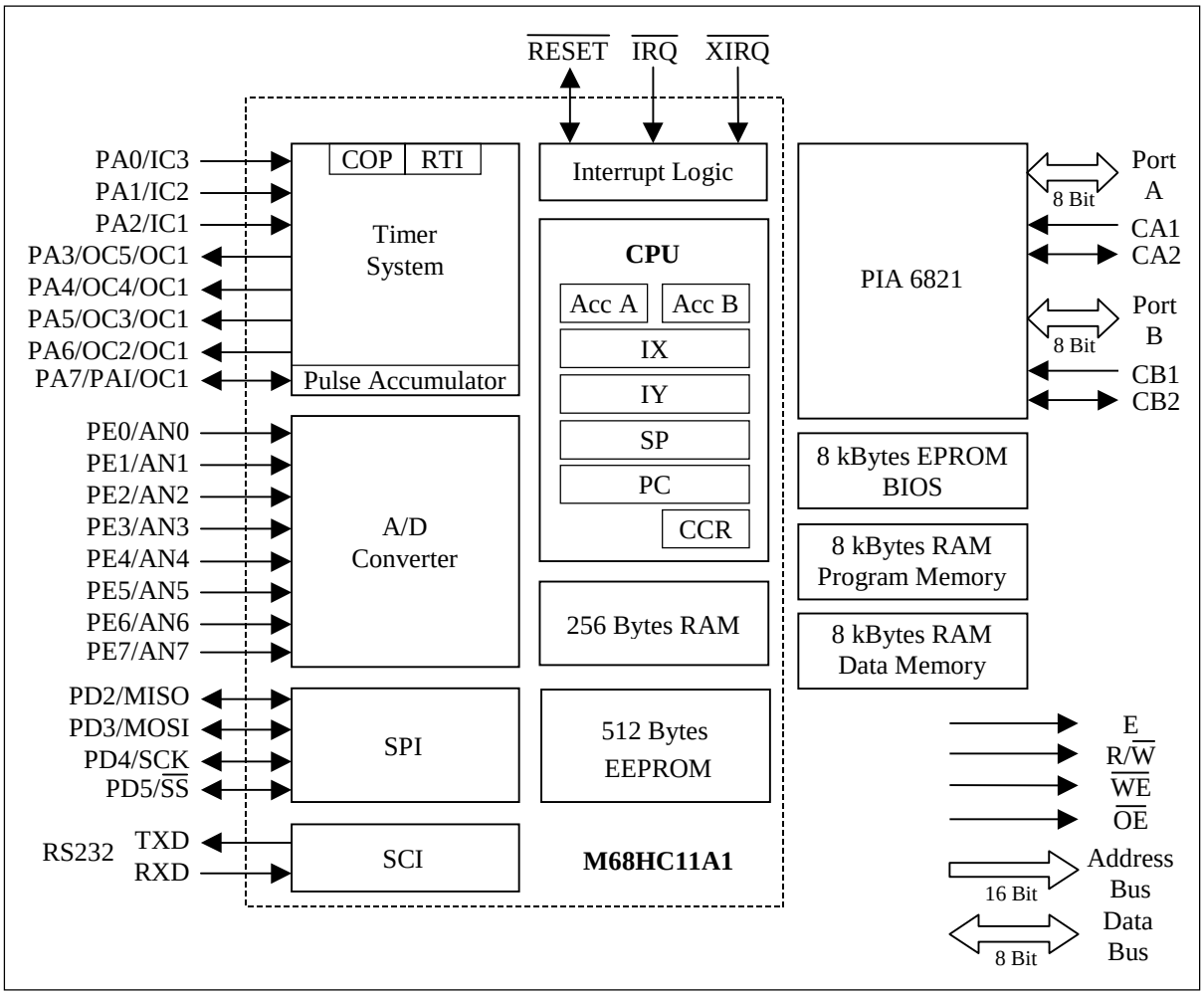
Običajni ceneni razvojni sistemi (Low Cost Development Board & Tools) sicer nudijo kvalitetno izdelane mikrokrmilniške sisteme na majhnih in optimiziranih tiskanih vezjih, vendar pa imajo manj zmogljiva programska razvojna orodja, ki so običajno tudi manj prijazna za uporabo. Te pomanjkljivosti seveda ne veljajo za profesionalne razvojne sisteme, ki pa so zaradi zelo visoke cene največkrat nesprejemljivi za povprečnega uporabnika.

3. Mikrokrmilniški sistem

V tem poglavju je opisana zgradba in delovanje mikrokrmilniškega sistema. Natančno so opredeljene lastnosti in funkcije posameznih gradnikov ter njihova vloga v sistemu. Poleg tega so podana tudi navodila za priključitev zunanjih enot in za priključitev nadzorne enote oz. osebnega računalnika.

3.1 Zgradba sistema

Mikrokrmilniški sistem je zasnovan na mikrokrmilniku M68HC11A1 z dodatnimi pomnilniki (EPROM in RAM) ter paralelnim vmesnikom PIA6821. Blok shemo prikazuje slika 3.1.



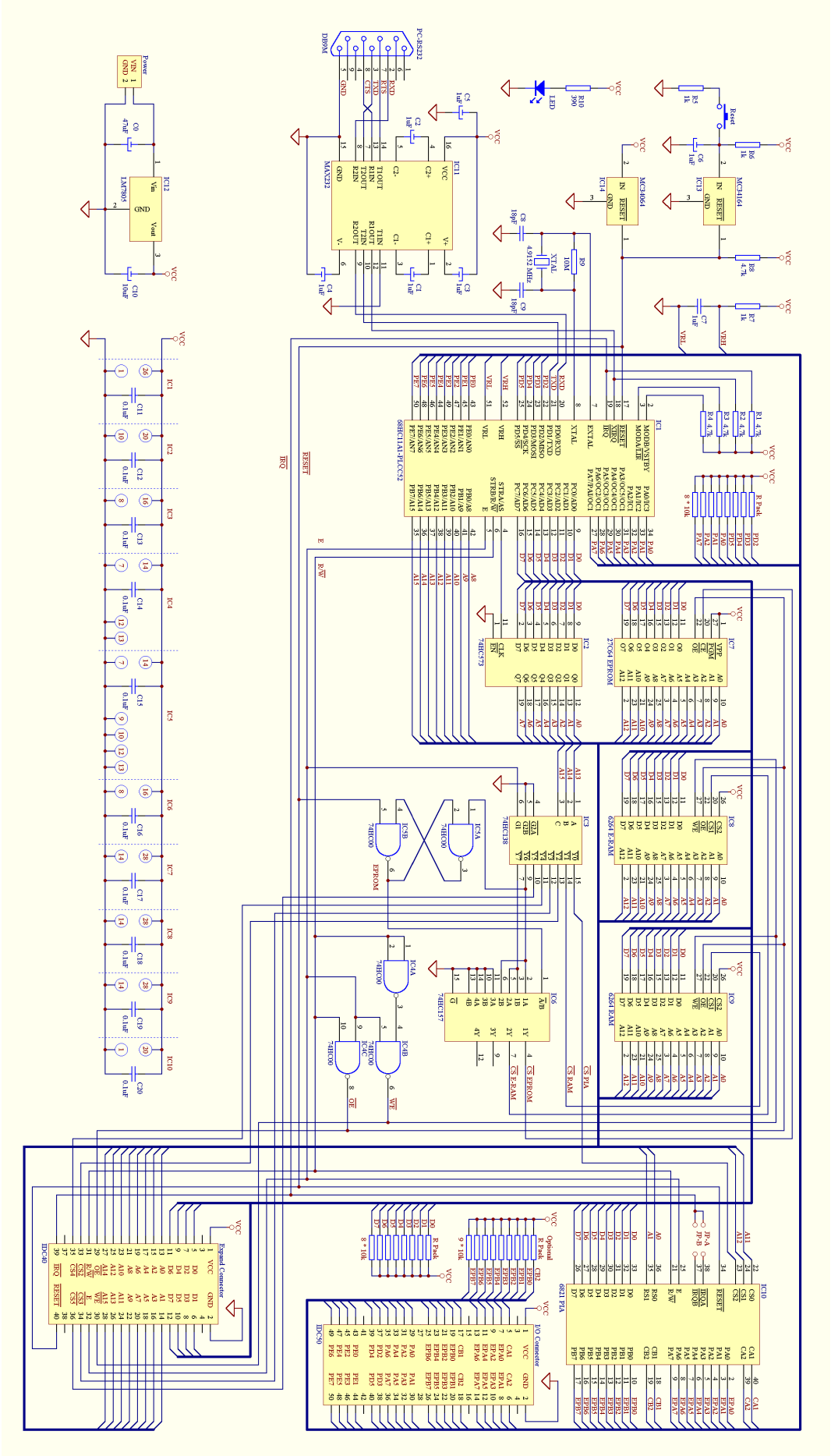
Slika 3.1: Blok shema mikrokrmilniškega sistema

Celotni sistem predstavlja razširjeni mikrokrmilnik M68HC11. Z dodanim statičnim RAM pomnilnikom mu razširimo pomnilnik za programsko kodo in podatke. V dodanem EPROM pomnilniku pa je zapisan BIOS. S paralelnim vmesnikom PIA6821 pa še dodatno povečamo število vhodno / izhodnih linij celotnega sistema. Podrobno shemo celotnega razvojnega mikrokrmilniškega sistema prikazuje slika 3.2.

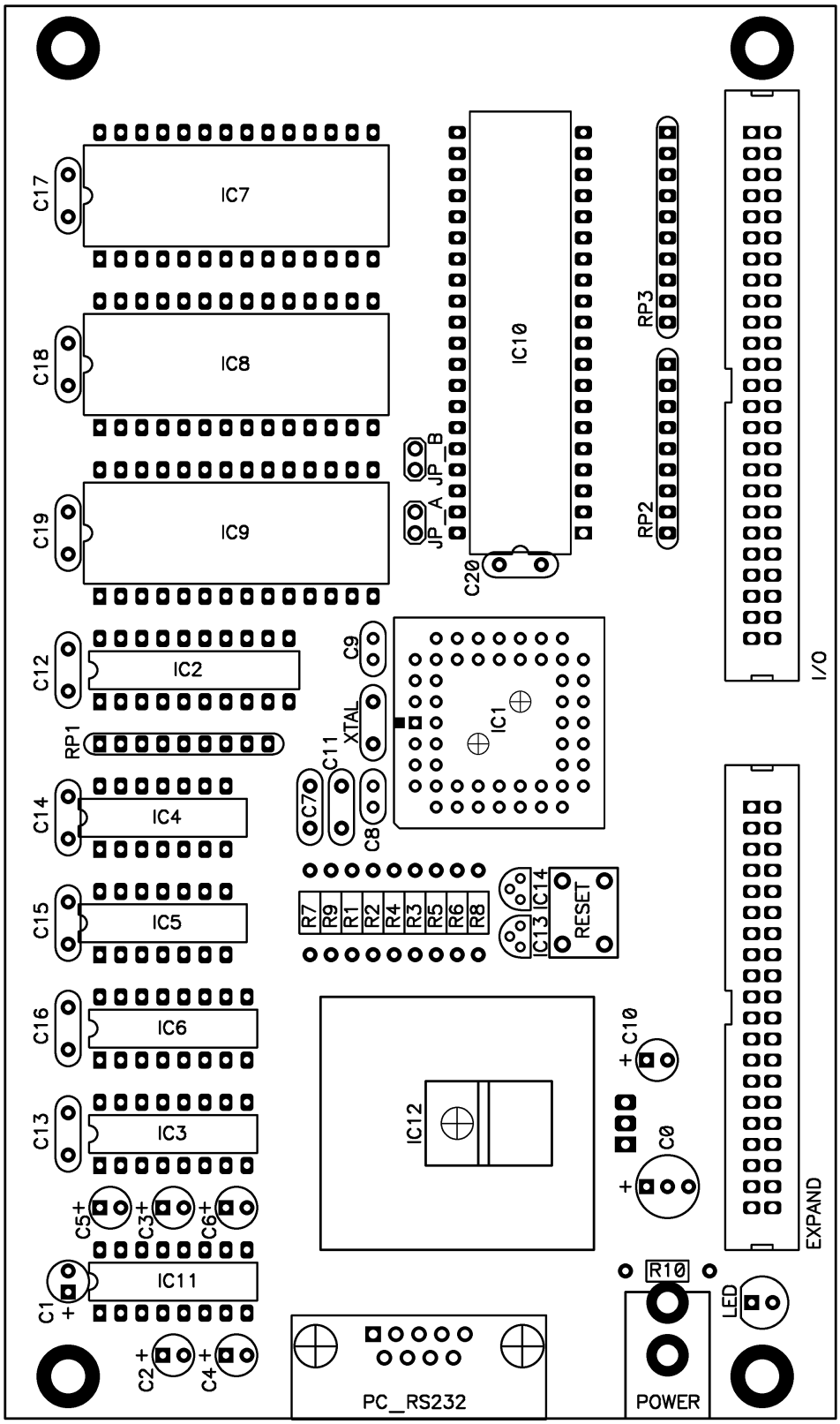
Mikrokrmilniški sistem sestavljajo naslednji gradniki:

- IC1 - mikrokrmilnik Motorola M68HC11A1 - PLCC52
- IC7 - 8 kBytes EPROM 27C64 - DIL28 (pomnilnik za BIOS)
- IC8 - 8 kBytes statični RAM 6264 - DIL28 (programski pomnilnik)
- IC9 - 8 kBytes statični RAM 6264 - DIL28 (podatkovni pomnilnik)
- IC2 - osmiški transparentni zapah 74HC573 - DIL20 (naslovni demultipleksor)
- IC3 - dekoder 3/8 74HC138 - DIL16 (naslovni dekoder)
- IC6 - 4 ∗ dvokanalni multipleksor 74HC157 - DIL16 (logika za naslovni dekoder)
- IC5 - 4 ∗ NAND logična vrata 74HC00 - DIL14 (logika za naslovni dekoder)
- IC4 - 4 ∗ NAND logična vrata 74HC00 - DIL14 (logika za Read / Write signale)
- IC10 - paralelni vmesnik PIA6821 - DIL40
- IC11 - pretvornik nivojev TTL ⇔ RS232 MAX232 - DIL16
- IC12 - 5V napetostni stabilizator LM7805
- IC13 - LVI podnapetostno senzorsko vezje MC34164P-5 TO-226AA
- IC14 - LVI podnapetostno senzorsko vezje MC34064P-5 TO-226AA
- kvarčni kristal XTAL 4.9152 MHz
- Power Supply (priključek za napajanje)
- PC - RS232 Connector DB9M (priključek za serijski vmesnik)
- I/O Connector IDC50 (priključek za vhodno / izhodne enote)
- Expand Connector IDC40 (priključek za razširitev)
- Jumper JP-A in JP-B (izbira izvora IRQ linije)
- LED (indikator prisotnosti napajanja)
- Reset (tipka za reset)

Poleg naštetih gradnikov so v vezju še nekateri pasivni elementi (upori in kondenzatorji). To so predvsem Pull-up upori in blokirni kondenzatorji. Pull-up upori so vezani med linije, ki lahko delujejo kot vhodi, in med napajanje +5V (Vcc). S tem je na linijah, ki nimajo definiranega napetostnega potenciala, vsiljeno logično stanje “1”, kar prepreči morebitno nevarno nihanje napetostnega nivoja. Blokirni kondenzatorji pa so vezani med napajalno linijo Vcc in GND posameznih digitalnih integriranih vezjih in zmanjšujejo motnje, ki nastajajo pri preklapljanju teh vezij iz enega logičnega stanja v drugo. Ostali pasivni elementi pa so potrebni za delovanje nekaterih gradnikov (npr. pretvornik nivojev TTL ⇔ RS232).


IC1
68HC11A1-PLCC52
IC2
74HC573
IC3
74HC138
IC4
74HC00
IC5
74HC157
IC6
IC7
27C64 EPROM
IC8
6264 E-RAM
IC9
6264 RAM
IC10
6821 PIA
IC11
MAX232
IC12
LM7805
IC13
MC34164
IC14
MC34064
PC-RS232
DB9M
Power
Reset
LED
XTAL
4.9152 MHz
R Pull
E Pull
Expand Connector
I/O Connector
IDC40
IDC50
JP-A
JP-B
VCC
GND
IRQ
RESET
E
R/W
CS PIA
CS RAM
CS E-RAM
CS EPROM
EPROM
WR
OE

Vezje mikrokrmilniškega sistema je izdelano na dvostranski tiskani ploščici debeline 1,5 mm in velikosti 17 cm ∗ 10 cm. Ploščica je prevlečena tudi z zaščitnim lakom. Prostorski razpored elementov na tiskani ploščici v originalni velikosti prikazuje slika 3.3.



Slika 3.3: Tiskana ploščica - razpored elementov

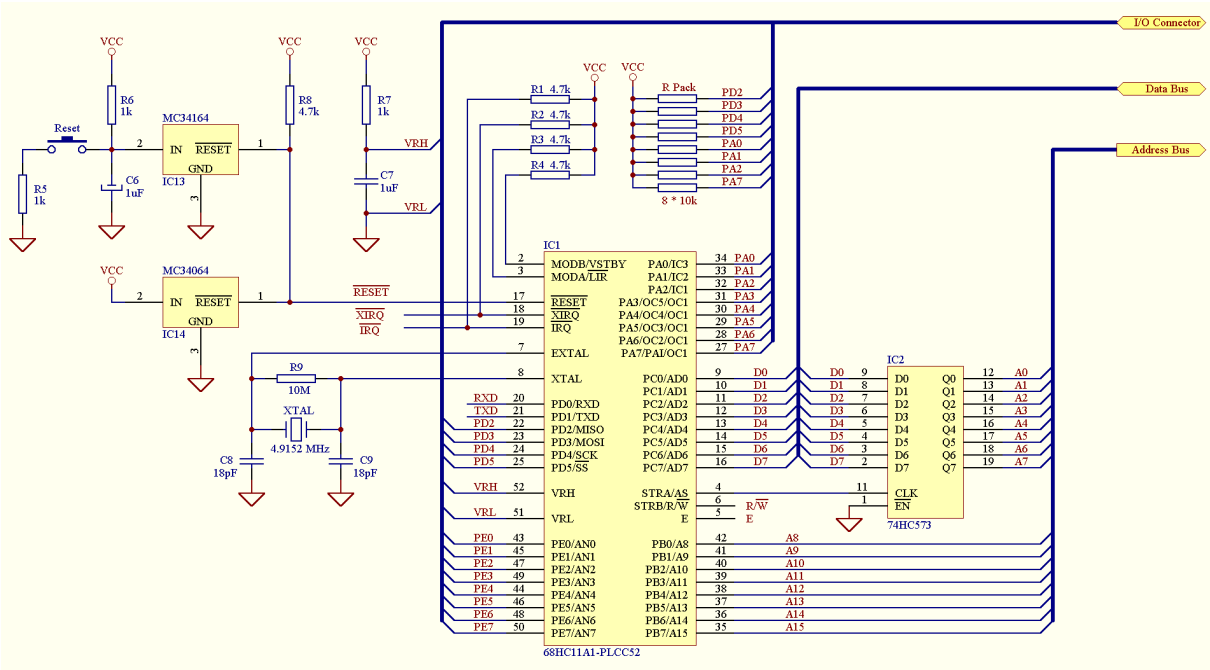
3.1.1 Mikrokrmilnik M68HC11A1

Mikrokrmilnik Motorola M68HC11A1 [2] je zmogljiv in izpopolnjen 8-bitni mikrokrmilnik iz družine mikrokrmilnikov HC11 [1]. Poleg zelo močne procesorske enote ima vgrajenih tudi kar nekaj dodatnih enot, med katerimi so predvsem vhodno - izhodni vmesniki.

Glavne značilnosti mikrokrmilnika M68HC11A1:

- 8-bitni CPU z izboljšanim M6800/M6801 naborom ukazov
  - dva 8-bitna akumulatorja AccA in AccB ali 16 bitni akumulator AccD
  - dva 16-bitna indeksna registra IX in IY
  - 16-bitni kazalec sklada SP
  - 16-bitni programski števec PC
  - 8-bitni pogojno programski register CCR
  - naslavljanje: takojšnje, direktno, razširjeno, indeksno, vsebujoče in relativno
- 256 Bytes RAM
- 512 Bytes EEPROM
- 64 kBytes naslovni prostor (Memory Map)
- 8 vhodno / izhodnih linij PA0 .. PA7 z dodatnimi funkcijami, vezanimi na časovnik
- 4 vhodno / izhodne linije PD2 .. PD5 z dodatnimi funkcijami, vezanimi na SPI
- 8 analognih vhodnih linij PE0 .. PE7 z 8-bitnim A/D pretvornikom (ADC)
- 16-bitni časovnik s posebnimi funkcijami:
  - 3 ∗ 16-bitni registri za zajemanje vhodov (Input Capture)
  - 5 ∗ 16-bitni registri za postavljanje izhodov (Output Compare)
  - 8-bitni števec impulzov (Pulse Accumulator)
  - prekinitev realnega časa (Real Time Interrupt)
- sinhroni serijski vmesnik (SPI)
- asinhronski serijski vmesnik (SCI)
- prekinitveni sistem (Reset, NMI, IRQ, …)
- COP Watchdog
- ohišje PLCC52

Mikrokrmilnik M68HC11A1 deluje v razširjenem načinu delovanja (Extended Mode). Z zunanjim svetom je povezan preko vhodno / izhodnih vmesnikov ter preko 16-bitnega naslovnega vodila (Address Bus) in 8-bitnega podatkovnega vodila (Data Bus). Naslovno in podatkovno vodilo omogočata priključitev zunanjega programskega in podatkovnega pomnilnika ter morebitnih dodatnih zunanjih enot v skupni 64 kBytes naslovni prostor. Vključitev mikrokrmilnika v mikrokrmilniški sistem prikazuje slika 3.4.



Slika 3.4: Vključitev mikrokrmilnika v sistem

Razširjeni način delovanja mikrokrmilnika je določen z logično “1” na sponkah MODA in MODB, ki se vzorči ob resetu. Ker je spodnjih 8 bitov naslovnega vodila (A0 .. A7) časovno multipleksirano s podatkovnim vodilom (D0 .. D7), je dodan osmiški transparentni zapah 74HC573 (IC2), ki demultipleksira naslovno vodilo. Za avtomatsko krmiljenje zapaha skrbi mikrokrmilnik preko linije AS. Smer prenosa podatkov določa mikrokrmilnik preko linije za branje / pisanje (R/W). Trenutek branja / pisanja oz. sinhronizacijo pa mikrokrmilnik določa preko linije E (Clock Output), ki je urin takt s frekvenco, enako 1/4 frekvence oscilatorja. Perioda urinega takta predstavlja tudi en strojni cikel (Machine Cycle) mikrokrmilnika. Za oscilator je uporabljen zunanji kvarčni kristal (XTAL) frekvence 4,9152 MHz.

Mikrokrmilnik ima izpopolnjeno logiko za resetiranje oz. postavljanje v definirano začetno stanje. Za to služi dvosmerna linija RESET, ki jo lahko uporabljamo kot vhod za zunanji reset ali pa kot izhod za prepoznavanje reseta. Mikrokrmilnik se samodejno resetira ob priključitvi na napajanje (Power-On Reset). Z dvema dodatnima senzorjema za prepoznavanje prenizke napetosti v obliki integriranih vezij IC13 in IC14 pa dosežemo, da je mikrokrmilnik v stanju reseta tudi ob padcu napajalne napetosti in ob pritisku na Reset tipko (Manual Reset).

Vhodno / izhodne linije PA0 .. PA7 in PD2 .. PD5 z dodanimi Pull-up upori ter analogne vhodne linije PE0 .. PE7 so vezane na priključek za vhodno / izhodne enote. Linija IRQ služi kot vhod za maskirano prekinitev in linija XIRQ kot vhod za nemaskirano prekinitev (NMI).

Linija VRH je vezana na napajanje +5V (Vcc) in predstavlja zgornjo referenčno napetost A/D pretvornika. Spodnja referenčna napetost pretvornika VRL pa je 0V (GND).

3.1.2 Zunanji pomnilnik

Mikrokrmilnik M68HC11A1 je izvedba, ki nima vgrajenega notranjega ROM pomnilnika. Vsebuje le bralno / pisalni pomnilnik manjše kapacitete (256 Bytes RAM) in električno brisljivi in programirljivi pomnilnik tudi manjše kapacitete (512 Bytes EEPROM). Zato je nujno, da sistemu dodamo še zunanji pomnilnik za programsko kodo in podatke.

Dodani zunanji pomnilnik je sestavljen iz treh enot:

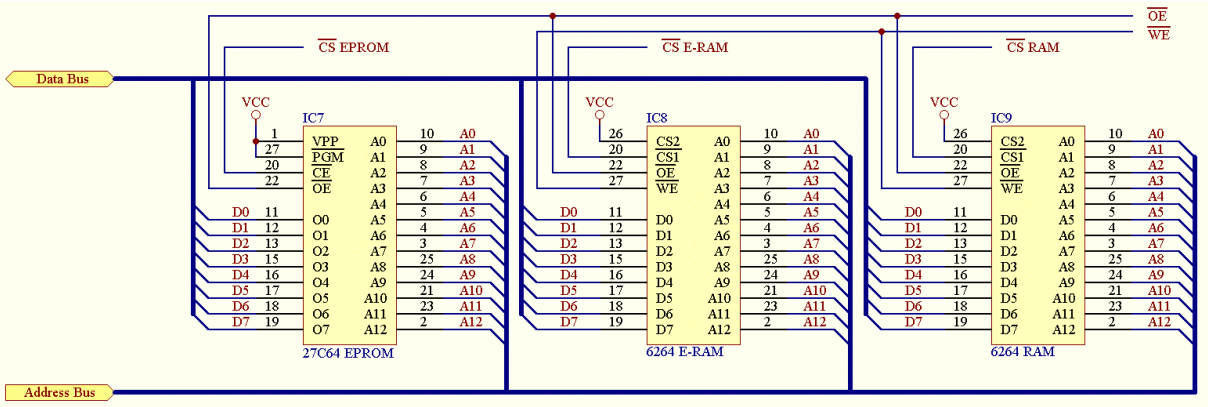
- uporabniški programski pomnilnik (IC8): 8 kBytes RAM (naslov: $E000 .. $FFFF)
- uporabniški podatkovni pomnilnik (IC9): 8 kBytes RAM (naslov: $2000 .. $3FFF)
- pomnilnik za BIOS (IC7): 8 kBytes EPROM

Uporabniški programski pomnilnik je statični bralno / pisalni pomnilnik, namenjen za začasno shranjevanje programske kode uporabniškega programa v času razvoja aplikacije. Naslovni prostor programskega pomnilnika je izbran tako, da zajema tudi Reset in prekinitvene vektorje mikrokrmilnika, ki se nahajajo na najvišjih naslovih ($FFD6 .. $FFFF).

Uporabniški podatkovni pomnilnik je tudi statični bralno / pisalni pomnilnik, ki je namenjen predvsem shranjevanju začasnih podatkov uporabniškega programa. Na ta način povečamo notranji bralno / pisalni pomnilnik in omogočimo tudi načrtovanje programov, ki uporabljajo večjo količino podatkov. Kadar želimo še povečati kapaciteto programskega pomnilnika, lahko uporabimo podatkovni pomnilnik tudi za shranjevanje programske kode, vendar pa tak program ne more delovati kot samostojna aplikacija, ampak le v razvojnem okolju.

Pomnilnik za BIOS je EPROM, v katerem je trajno shranjen strojni program BIOS-a. Kadar pa razvojno ploščo pretvorimo v prototipni sistem, je ta pomnilnik namenjen za trajni zapis programske kode samostojnega programa.

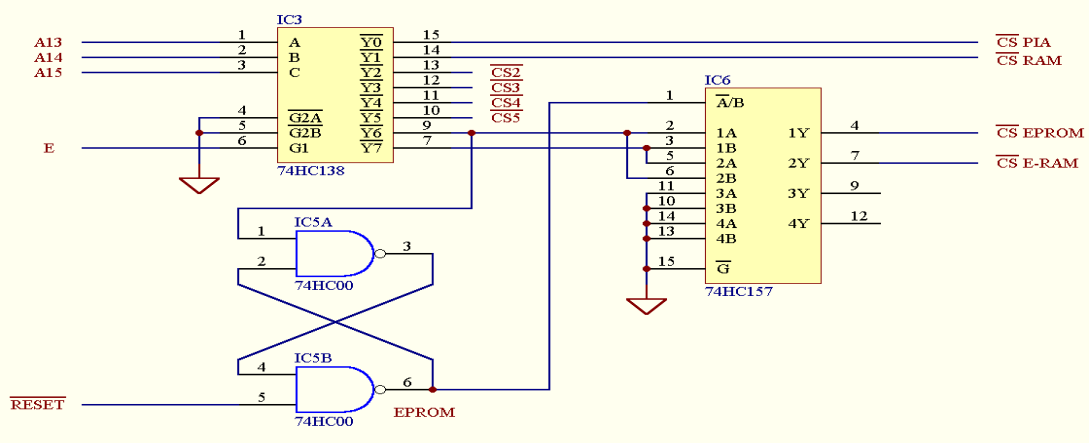
Vključitev dodanega zunanjega pomnilnika v mikrokrmilniški sistem prikazuje slika 3.5.



Slika 3.5: Vključitev zunanjega pomnilnika v sistem

3.1.3 Naslovni dekoder

Naslovni dekoder je logično vezje, ki celotni 64 kBytes naslovni prostor mikrokrmilnika razdeli na osem enakih zaporednih blokov velikosti 8 kBytes. Vsakemu bloku pripada po ena linija za izbiranje (CS). Na sliki 3.6 je prikazana zgradba naslovnega dekoderja.



Slika 3.6: Naslovni dekoder

Naslovno dekodiranje se izvede s pomočjo dekoderja 3/8 74HC138 (IC3), ki glede na trenutni naslov (preko naslovnih linij A13 .. A15) izbere pripadajoči blok in postavi ustrezno linijo za izbiranje v aktivno stanje. Slika 3.7 prikazuje razdelitev naslovnega prostora na bloke.

Naslov (Hex)	Velikost	Blok	Aktivni CS
$0000 .. $1FFF	8192 Bytes	Paralelni vmesnik	CS PIA
$2000 .. $3FFF	8192 Bytes	Podatkovni pomnilnik	CS RAM
$4000 .. $5FFF	8192 Bytes	Zunanja enota z CS2	CS2
$6000 .. $7FFF	8192 Bytes	Zunanja enota z CS3	CS3
$8000 .. $9FFF	8192 Bytes	Zunanja enota z CS4	CS4
$A000 .. $BFFF	8192 Bytes	Zunanja enota z CS5	CS5
$C000 .. $DFFF	8192 Bytes	BIOS program / Programski pomnilnik (ob Resetu)	CS EPROM / CS E-RAM
$E000 .. $FFFF	8192 Bytes	Programski pomnilnik / BIOS program (ob Resetu)	CS E-RAM / CS EPROM

Slika 3.7: Razdelitev naslovnega prostora na bloke

RS flip-flop, ki je sestavljen iz logičnih NAND vrat (IC5), in dvokanalni multipleksor (IC6) nam ob Resetu postavita na naslov: $E000 .. $FFFF blok z BIOS programom, ki definira tudi začetni Reset vektor mikrokrmilnika. Ob prvem bralno / pisalnem dostopu, ki ga izvede BIOS do bloka na naslovu: $C000 .. $DFFF, pa se avtomatsko zamenjata naslova blokov za BIOS in programski pomnilnik.

3.1.4 Paraleni vmesnik PIA6821

Paralelni vmesnik PIA6821 [3] omogoča priključitev najrazličnejših dodatnih zunanjih enot na mikrokrmilniški sistem. S tem povečamo število vhodno / izhodnih linij mikrokrmilnika. Lahko uporabljamo tudi nadzor nad tokom podatkov in na ta način še dodatno povečamo zmogljivosti celotnega mikrokrmilniškega sistema.

Glavne značilnosti paralelnega vmesnika PIA6821:

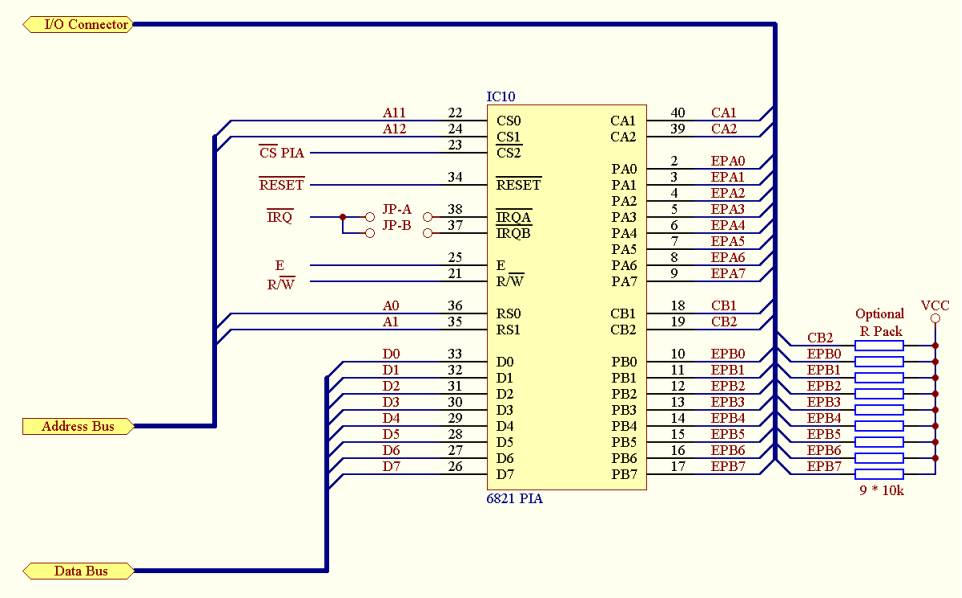
- 8-bitni Port A (EPA0 .. EPA7):
  - dvosmerna vrata (vsaka linija se da posebej programirati kot vhod ali izhod)
  - možnost uporabe nadzora nad tokom podatkov (CA1 in CA2)
- vhodna linija CA1:
  - za zahtevo po oskrbi priključene enote na vratih A ali po prekinitvi IRQA
- vhodno / izhodna linija CA2:
  - vhod: zahteva po oskrbi priključene enote na vratih A ali po prekinitvi IRQA
  - izhod: krmiljenje priključene enote na vratih A
- 8-bitni Port B (EPB0 .. EPB7):
  - dvosmerna vrata (vsaka linija se da posebej programirati kot vhod ali izhod)
  - možnost uporabe nadzora nad tokom podatkov (CB1 in CB2)
  - tristanjski izhodni gonilniki (3-State Output Buffer)
- vhodna linija CB1:
  - za zahtevo po oskrbi priključene enote na vratih B ali po prekinitvi IRQB
- vhodno / izhodna linija CB2:
  - vhod: zahteva po oskrbi priključene enote na vratih B ali po prekinitvi IRQB
  - izhod: krmiljenje priključene enote na vratih B
- izhodni liniji za zahtevo po prekinitvi IRQA in IRQB
  (izhod z odprtim ponorom - Open Drain Output)

Mikrokrmilnik je s paralelnim vmesnikom povezan preko šestih registrov, ki omogočajo nastavljanje delovanja vmesnika in branje ali postavljanje vhodnih oz. izhodnih linij. Registri zavzemajo 4 pomnilniške lokacije, ki so zrcaljenje na naslovih $1800 .. $1FFF. Na sliki 3.8 so prikazani omenjeni registri paralelnega vmesnika.

Simbol	Naslov (Hex)	Ime registra
DPRA	$1800	Data Direction / Peripherial Register A
CRA	$1801	Control Register A
DPRB	$1802	Data Direction / Peripherial Register B
CRB	$1803	Control Register B

Slika 3.8: Registri paralelnega vmesnika PIA6821

Vključitev paralelnega vmesnika PIA6821 v mikrokrmilniški sistem prikazuje slika 3.9.



Slika 3.9: Vključitev paralelnega vmesnika v sistem

Paralelni vmesnik je vključen v mikrokrmilniški sistem kot pomnilnik s 4 lokacijami. Za izbiro čipa poskrbi naslovni dekoder (CS PIA) v kombinaciji z delom naslova (A11 in A12). Za izbiro registra pa poskrbi spodnji del naslova (A0 in A1). Podatkovno vodilo (D0 .. D7) je priključeno direktno na mikrokrmilnik. Smer prenosa podatkov določa mikrokrmilnik preko linije za branje / pisanje (R/W). Za sinhronizacijo pri branju in pisanju paralelnega vmesnika poskrbi mikrokrmilnik preko linije E. Inicializacija paralelnega vmesnika se izvede sočasno z resetom mikrokrmilnika preko linije RESET.

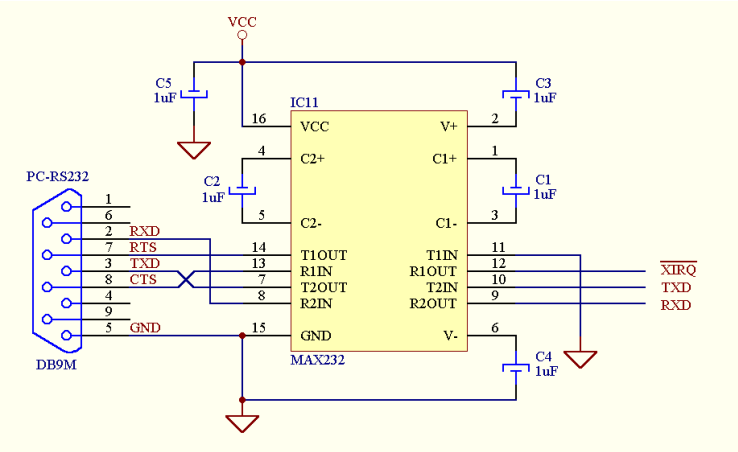
Vhodno / izhodne linije vrat A (EPA0 .. EPA7) in vrat B (EPB0 .. EPB7) ter kontrolne linije vrat A (CA1, CA2) in vrat B (CB1, CB2) so vezane na priključek za vhodno / izhodne enote. Vse linije, ki so lahko v visoko impedančnem stanju (EPB0 .. EPB7 in CB2), imajo dodane Pull-up upore v obliki uporovne verige. S tem vsilimo logično stanje “1” , ko so definirane kot vhodne linije, in na ta način preprečimo morebitno nevarno nihanje napetostnega nivoja. Kadar pa aplikacija želi izkoristiti možnost, da so linije lahko v visoko impedančnem stanju in sama poskrbi za ustrezno zaključitev linij, se lahko odstranijo Pull-up upori oz. uporovna veriga, ki je nameščena na podnožju SIL 10.

Paralelni vmesnik lahko preko linij IRQA in IRQB zahteva od mikrokrmilnika prekinitev izvajanja programa in sproži izvajanje prekinitvenega podprograma. To dosežemo s povezavo IRQA (veznik JP-A) ali IRQB (veznik JP-B) ali obeh na linijo IRQ mikrokrmilnika in s programsko omogočitvijo maskirane prekinitve. Skupaj lahko povežemo oba možna izvora prekinitev IRQA in IRQB, saj sta to izhoda z odprtim ponorom. Na ta način bosta lahko oba izvora povzročila prekinitev. Izvor prekinitve pa je v takem primeru vedno potrebno ugotoviti programsko s pomočjo branja kontrolnih registrov paralelnega vmesnika.

3.1.5 Priključek RS232 s pretvornikom nivojev

Mikrokrmilnik ima vgrajen univerzalni asinhronski serijski vmesnik (SCI). S pomočjo tega vmesnika komunicira mikrokrmilniški sistem z nadzorno enoto oz. osebnim računalnikom v času razvoja aplikacije. Omenjeni serijski vmesnik pa lahko uporablja tudi uporabniški program kot običajni asinhroni serijski port.

Komunikacija poteka dvosmerno (Full Duplex) preko dveh linij. Podatki se pošiljajo preko oddajne linije TXD (Transmit Data) in prejemajo preko sprejemne linije RXD (Receive Data). Medtem ko mikrokrmilnik uporablja TTL napetostne nivoje, pa običajne asinhrone serijske enote uporabljajo RS232 nivoje. Zato je dodano posebno integrirano vezje MAX232 (IC11), ki pretvarja nivoje TTL ⇔ RS232. Liniji TXD in RXD, ki uporabljata RS232 standard, sta vezani na priključek RS232, ki se običajno uporablja za priključitev osebnega računalnika. Slika 3.10 prikazuje RS232 priključek in pretvornik nivojev.



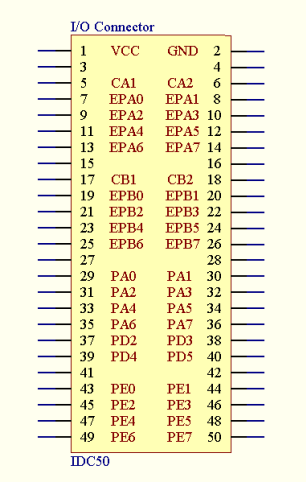
Slika 3.10: Priključek RS232 s pretvornikom nivojev

Na priključku RS232 imamo še dve dodatni liniji RTS in CTS, ki tudi uporabljata RS232 standard. Preko RTS linije lahko prepoznamo prisotnost mikrokrmilniškega sistema. Ta linija je v aktivnem stanju, kadar je sistem priključen in ima ustrezno napajanje. Kadar pa sistem ni priključen ali pa nima napajanja, pa je linija v neaktivnem stanju. Linija CTS pa se uporablja kot izvor za nemaskirano prekinitev (NMI). Priključena je preko pretvornika nivojev na linijo XIRQ mikrokrmilnika. Obe liniji RTS in CTS uporablja programsko razvojno orodje, ki deluje na osebnem računalniku in je z mikrokrmilniškim sistemom povezano preko priključka RS232, ki uporablja DB9 moški konektor.

Elektrolitski kondenzatorji C1 .. C4 so dodatni zunanji gradniki integriranega vezja MAX232, ki so potrebni za pridobivanje simetričnih RS232 napetostnih nivojev iz enojnega pozitivnega napajanja +5V.

3.1.6 Priključek za vhodno / izhodne enote

Za povezavo mikrokrmilniškega sistema z različnimi zunanjimi enotami imamo na razpolago skupno 40 vhodno / izhodnih linij (32 digitalnih in 8 analognih), ki so vezane na priključek za vhodno / izhodne enote (I/O Connector). Priključek za zunanje enote je prikazan na sliki 3.11.



Slika 3.11: Priključek za vhodno / izhodne enote

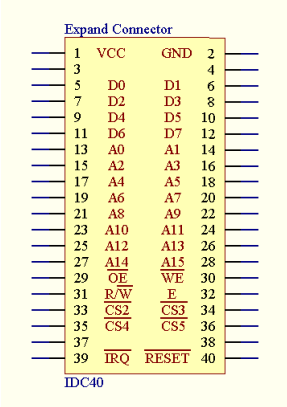
Priključek za vhodno / izhodne enote vsebuje naslednje linije:

- Vcc in GND: +5V napajanje za zunanje enote
- CA1 (vhod) in CA2 (vhod / izhod): kontrolne linije za PIA Port A
- EPA0 .. EPA7: 8-bitni dvosmerni PIA Port A
- CB1 (vhod) in CB2 (vhod / izhod): kontrolne linije za PIA Port B
- EPB0 .. EPB7: 8-bitni dvosmerni PIA Port B
- PA0 .. PA7: 8-bitni dvosmerni Port A z dodatnimi funkcijami, vezanimi na časovnik
  - PA0 / IC3 (vhod): Digital Input / Input Capture 3
  - PA1 / IC2 (vhod): Digital Input / Input Capture 2
  - PA2 / IC1 (vhod): Digital Input / Input Capture 1
  - PA3 / OC5 / OC1 (izhod): Digital Output / Output Compare 5 / Output Compare 1
  - PA4 / OC4 / OC1 (izhod): Digital Output / Output Compare 4 / Output Compare 1
  - PA5 / OC3 / OC1 (izhod): Digital Output / Output Compare 3 / Output Compare 1
  - PA6 / OC2 / OC1 (izhod): Digital Output / Output Compare 2 / Output Compare 1
  - PA7 / PAI / OC1 (vhod / izhod): Digital I/O / PA Input / Output Compare 1
- PD2 .. PD5: 4-bitni dvosmerni Port D z dodatnimi funkcijami, vezanimi na SPI
  - PD2 / MISO (vhod / izhod): Digital Input Output / Master In Slave Out
  - PD3 / MOSI (vhod / izhod): Digital Input Output / Master Out Slave In
  - PD4 / SCK (vhod / izhod): Digital Input Output / Serial Clock
  - PD5 / SS (vhod / izhod): Digital Input Output / Slave Select
- PE0 .. PE7 (AN0 .. AN7): 8-bitni analogni vhodni Port E

3.1.7 Priključek za razširitev

Mikrokrmilniški sistem je načrtovan zelo odprto in zato omogoča tudi nadgradnjo s strani uporabnika. Sistemu lahko dodajamo razne dodatne enote, ki jih vključimo v naslovni prostor mikrokrmilnika. To so lahko najrazličnejši pomnilniki (RAM, ROM, EPROM, EEPROM, …) in različni drugi periferni vmesniki, ki so primerni za priključitev na mikrokrmilnik oz. mikroprocesor. Na ta način lahko sistem nadgradimo, da ustreza še tako zahtevni aplikaciji.

Za nadgradnjo sistema se uporablja posebni priključek za razširitev (Expand Connector), na katerega priključujemo dodatne enote. Ta priključek vsebuje vse signale mikrokrmilniškega sistema, ki so potrebni za nadgradnjo sistema. Priključek za razširitev prikazuje slika 3.12.



Slika 3.12: Priključek za razširitev

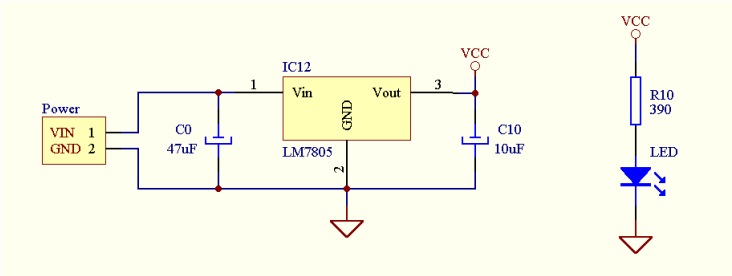
Priključek za razširitev vsebuje naslednje linije:

- Vcc in GND: +5V napajanje za zunanje enote
- D0 .. D7 (Data Bus): 8-bitno podatkovno vodilo
- A0 .. A15 (Address Bus): 16-bitno naslovno vodilo
- OE (Output Enable): izhodna kontrolna linija za omogočanje izhoda
- WE (Write Enable): izhodna kontrolna linija za omogočanje pisanja
- R/W (Read / Write): izhodna kontrolna linija za branje / pisanje
- E (Clock Output): izhod urinega takta (1/4 frekvence oscilatorja)
- IRQ (Interrupt Request): vhod za maskirano prekinitev
- RESET: vhodno / izhodna reset linija mikrokrmilnika

Mikrokrmilnik obravnava dodane zunanje enote kot pomnilniške lokacije z bralno / pisalnim dostopom v svojem 64 kBytes naslovnem prostoru. Z njimi je povezan preko naslovnega in podatkovnega vodila ter kontrolnih linij (OE, WE ali R/W). S pomočjo IRQ vhoda lahko zunanja enota zahteva tudi maskirane prekinitve. Preko vhodno / izhodne reset linije pa lahko mikrokrmilnik resetira zunanjo enoto ali pa zunanja enota izvrši reset mikrokrmilnika.

3.1.8 Napajanje

Vsi aktivni gradniki v vezju mikrokrmilniškega sistema so napajani iz enotnega napetostnega vira +5V (Vcc). Ta napetostni vir je na razpolago tudi zunanjim enotam na priključku za vhodno / izhodne enote in na priključku za razširitev. Za konstantno enosmerno pozitivno napetost 5V skrbi napetostni stabilizator LM7805 (IC12), ki je pritrjen na hladilno telo. Zaradi tega lahko stabilizator bolj obremenimo (napajamo lahko tudi dodatne zunanje enote). Na sliki 3.13 je prikazan priključek za napajanje in stabilizator napetosti.



Slika 3.13: Priključek za napajanje in stabilizator napetosti

Priporočljiva vhodna napetost stabilizatorja je glajena enosmerna napetost okoli 9V, ki jo mora uporabnik pripeljati na priključek za napajanje (Power). Napetost je lahko sicer tudi nižja ali pa višja. Pri tem moramo biti pazljivi, ker prenizka vhodna napetost povzroči tudi prenizko napajalno napetost Vcc. Previsoka vhodna napetost pa lahko povzroči prekomerno segrevanje napetostnega stabilizatorja s pripadajočim hladilnim telesom.

Največja skupna poraba samega vezja mikrokrmilniškega sistema je določena za najslabši možni primer in znaša pod 300mA pri 5V (teoretični tehnični podatki). V praksi se izkaže, da je povprečna poraba bistveno manjša.

Z napetostnim virom Vcc lahko napajamo tudi dodane zunanje enote. Pri veliki obremenitvi napajalnega vira se bo zaradi tega začel segrevati napetostni stabilizator s hladilnim telesom. Če je segrevanje premočno, je potrebno obstoječe hladilno telo zamenjati z večjim ali pa zmanjšati obremenitev. Kljub temu pa naj skupna obremenitev dodanih zunanjih enot ne presega 500mA pri 5V.

Za indikacijo prisotnosti napajanja je uporabljena svetleča dioda (LED), ki je nameščena neposredno zraven priključka za napajanje.

Vsa digitalna integrirana vezja imajo v neposredni bližini nameščene blokirne kondenzatorje, ki so vezani med Vcc in GND. Ti kondenzatorji zmanjšujejo napetostne konice, ki nastajajo pri preklapljanju digitalnih vezij, kar dodatno stabilizira napajalno napetost.

3.2 Naslovni prostor

Naslovni prostor mikrokrmilnika sestavlja 65536 8-bitnih lokacij (64 kBytes Memory Map). V njem so vključeni notranji pomnilnik in registri mikrokrmilnika ter zunanji pomnilnik in registri paralelnega vmesnika. Preostali del naslovnega prostora pa je večinoma rezerviran za dodatne zunanje enote in je razdeljen na štiri dele, izmed katerih ima vsak svojo izbirno linijo. Tudi tiste dele neuporabljenega naslovnega prostora, ki niso rezervirani, lahko uporabimo za vključitev dodatnih zunanjih enot, vendar pa mora uporabnik sam poskrbeti za generiranje ustreznih izbirnih linij. Podrobna razdelitev naslovnega prostora je prikazana na sliki 3.14.

Naslov (Hex)	Velikost	Blok
$0000 .. $00FF	256 Bytes	Notranji RAM mikrokrmilnika (Zero Page)
$0100 .. $0FFF	3840 Bytes	Neuporabljeno
$1000 .. $103F	64 Bytes	Notranji registri mikrokrmilnika
$1040 .. $17FF	1984 Bytes	Neuporabljeno
$1800 .. $1803	4 Bytes	Registri paralelnega vmesnika
$1804 .. $1FFF	2044 Bytes	Zrcaljeni registri paralelnega vmesnika
$2000 .. $3FFF	8192 Bytes	Zunanji RAM (podatkovni pomnilnik)
$4000 .. $5FFF	8192 Bytes	Rezervirano za zunanjo enoto z izbirno linijo CS2
$6000 .. $7FFF	8192 Bytes	Rezervirano za zunanjo enoto z izbirno linijo CS3
$8000 .. $9FFF	8192 Bytes	Rezervirano za zunanjo enoto z izbirno linijo CS4
$A000 .. $BFFF razen $B600 .. $B7FF	7680 Bytes	Rezervirano za zunanjo enoto z izbirno linijo CS5
$B600 .. $B7FF	512 Bytes	Notranji EEPROM mikrokrmilnika
$C000 .. $DFFF	8192 Bytes	Zunanji EPROM (BIOS)
		Za samostojno aplikacijo oz. prototipni sistem: Zunanji RAM (podatkovni pomnilnik)
$E000 .. $FFFF	8192 Bytes	Zunanji RAM (programski pomnilnik)
		Ob Resetu: Zunanji EPROM (BIOS)
		Za samostojno aplikacijo oz. prototipni sistem: Zunanji EPROM (samostojni program)

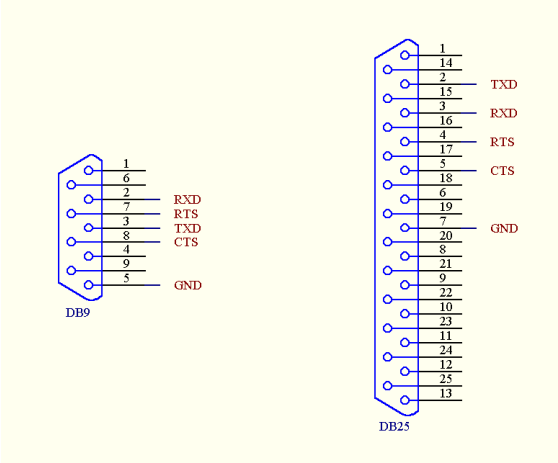
Slika 3.14: Razdelitev naslovnega prostora

Mikrokrmilnik ima bralno / pisalni dostop do poljubne lokacije iz naslovnega prostora. Pri tem lahko uporablja različne načine naslavljanja (direktno, razširjeno, indeksno in relativno). Direktno naslavljanje je rezervirano za prvih 256 Bytes naslovnega prostora (Zero Page), na katerih se nahaja notranji RAM mikrokrmilnika.

3.3 Priključitev na osebni računalnik

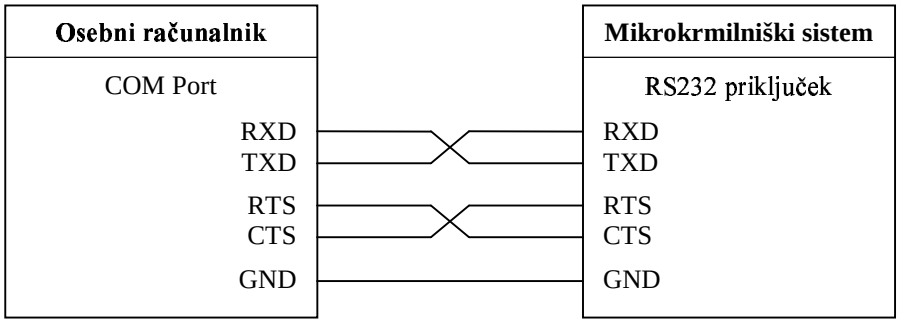
V času razvoja aplikacije je mikrokrmilniški sistem priključen na osebni računalnik. Povezana sta preko asinhronega serijskega vmesnika RS232. Na osebnem računalniku lahko uporabimo katerikoli prosti standardni serijski komunikacijski port (COM Port). Na mikrokrmilniškem sistemu pa uporabimo vgrajeni serijski vmesnik (SCI), ki je dostopen preko RS232 priključka.

Povezava je izvedena preko štirih signalnih linij (RXD, TXD, RTS, CTS) in mase (GND). Priključek RS232 na mikrokrmilniški strani uporablja DB9 moški konektor. Priključek za serijski COM Port na osebnem računalniku pa ima lahko DB9 ali DB25 moški konektor. Slika 3.15 prikazuje izgled DB9 in DB25 konektorja z označenimi linijami, ki so potrebne za povezavo mikrokrmilniškega sistema in osebnega računalnika.



Slika 3.15: DB9 in DB25 konektor

Za povezavo je uporabljena tako imenovana “Null Modem” povezava. Pri tem je potrebno križno povezati linijo TXD ene strani z linijo RXD druge strani. Prav tako je potrebno križno povezati tudi linijo RTS z linijo CTS. Liniji GND pa morata biti povezani skupaj. Vezalna shema priključitve mikrokrmilniškega sistema na osebni računalnik je prikazana na sliki 3.16.

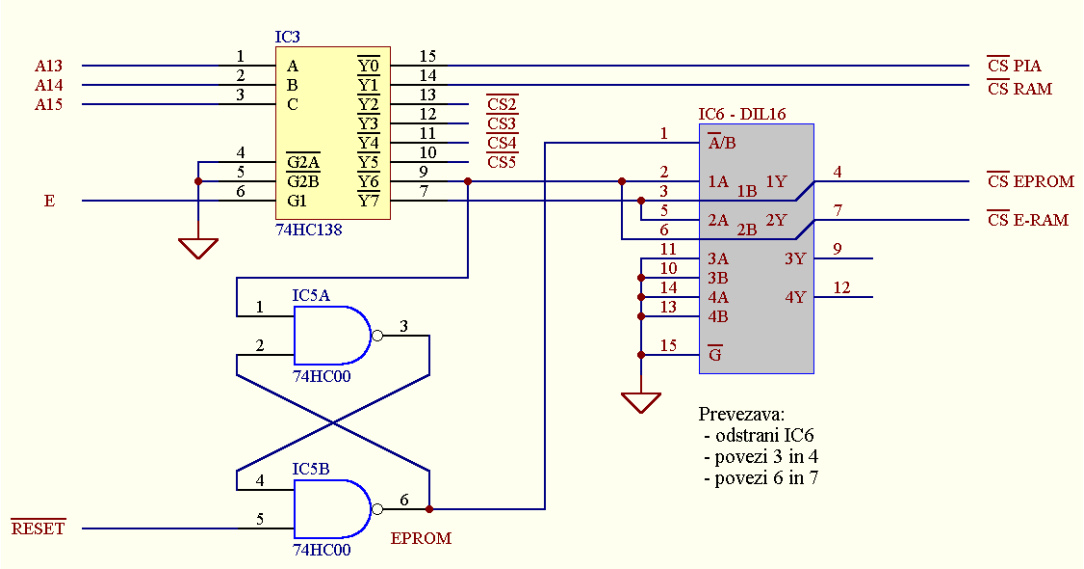


Slika 3.16: Priključitev mikrokrmilniškega sistema na osebni računalnik

3.4 Prototipni sistem

Mikrokrmilniški sistem se predvsem uporablja kot razvojna plošča v povezavi s programskim razvojnim orodjem za načrtovanje aplikacij. Ko pa je aplikacija narejena in deluje, kot je bilo načrtovano, pa lahko z minimalnimi posegi mikrokrmilniški sistem spremenimo v prototipni sistem, ki deluje popolnoma samostojno.

Če želimo mikrokrmilniški sistem spremeniti v prototipni sistem, moramo naprej zamenjati BIOS, ki je shranjen v EPROM-u (IC7) s samostojnim uporabniškim programom. To lahko naredimo preprosto tako, da preprogramiramo EPROM. Teoretično bi to bilo sicer že dovolj, če mikrokrmilniški program nikoli ne bi poskusil nasloviti poljubne lokacije iz bloka na naslovu $C000 .. $DFFF. To bi imelo namreč za posledico avtomatsko prestavitev EPROM-a iz naslova $E000 .. $FFFF na naslov $C000 .. $DFFF, kar bi za program pomenilo nepopravljivo napako (omenjeni avtomatski preklop sicer uporablja BIOS). Zato je potrebno spremeniti naslovni dekoder tako, da izklopimo avtomatski preklop naslovov. To lahko naredimo na več načinov. Ena izmed možnosti je prikazana na sliki 3.17.



Slika 3.17: Spremenjeni naslovni dekoder za prototipni sistem

Iz vezja mikrokrmilniškega sistema odstranimo integrirano vezje 74HC157 (IC6) in skupaj povežemo priključni sponki 3 in 4 ter 6 in 7. Na ta način dosežemo, da bo EPROM trajno prisoten na naslovu $E000 .. $FFFF (aktivna izbirna linija CS EPROM). Poleg tega pa se RAM (IC8), ki je bil uporabljen za programski pomnilnik ob času razvoja aplikacije, pojavi na naslovu $C000 .. $DFFF (aktivna izbirna linija CS E-RAM). Lahko ga uporabimo kot dodatni podatkovni ali programski pomnilnik.

4. BIOS - Basic Input Output System

BIOS je optimiziran strojni program, napisan v makro-zbirnem jeziku, ki skrbi za programsko povezavo med mikrokrmilniškim sistemom in nadzorno enoto, ki je običajno osebni računalnik. Komunikacija poteka preko asinhronega serijskega vmesnika RS232. Pri načrtovanju in preverjanju aplikacij nam BIOS omogoča uporabo potrebnih ukazov s strani nadzorne enote in generiranje ustreznih odzivov s strani mikrokrmilnika. Program BIOS-a je zapisan v EPROM-u mikrokrmilniškega sistema.

4.1 Lastnosti

Nadrejena nadzorna enota (Master) s pomočjo različnih ukazov sproži zahtevano operacijo, ki jo podrejeni mikrokrmilniški sistem (Slave) izvrši in jo z ustreznim odzivom tudi potrdi. Poleg tega nam BIOS tudi omogoča, da mikrokrmilniški sistem generira ustrezni odziv ob programski prekinitvi ali resetu mikrokrmilnika, kar lahko prepozna nadzorna enota.

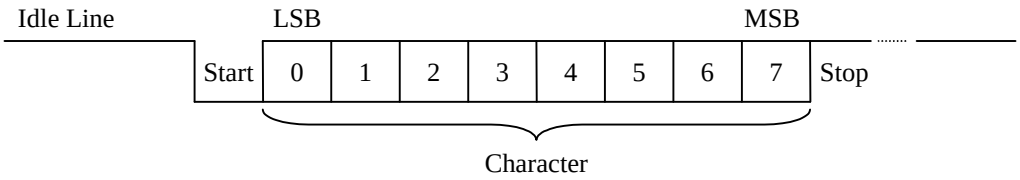
Basic Input Output System (BIOS) omogoča:

- nalaganje programov iz osebnega računalnika v mikrokrmilniški sistem (Download)
- zagon programov (Run)
- koračno izvajanje programov (Single Stepping)
- zaustavljanje programov s pomočjo nemaskirane prekinitve (NMI)
- pregled in urejanje vsebine registrov centralno procesne enote (CPU Registers)
- bralno / pisalni dostop do vseh lokacij iz celotnega naslovnega prostora (64 kBytes)
- prepoznavanje prekinitev in reseta mikrokrmilnika

Zaradi preprečitve morebitnih napak, ki bi se lahko pojavile pri komunikaciji med nadzorno enoto in mikrokrmilniškim sistemom, imamo dodane posebne mehanizme. Zato imajo ukazi, pri katerih se prenašajo podatki, dodano 8-bitno vsoto za preverjanje (Check Sum). Poleg tega nadzorna enota tudi zahteva za vsak ukaz, ki ga pošlje, ustrezno potrditev razumljivosti ukaza s strani mikrokrmilniškega sistema. Če ukaz ni bil potrjen, ga mora nadzorna enota ponoviti. Kadar pa mikrokrmilniški sistem generira odziv zaradi prekinitve ali reseta, se ta odziv dvakrat ponovi, kar zelo zmanjša verjetnost, da bi prišlo do nesporazuma pri prepoznavanju tega odziva. Z vsemi temi metodami močno zmanjšamo možnost, da bi se zaradi motenj v komunikaciji pojavila napaka, ki je ne bi prepoznali in je ne bi mogli popraviti.

4.2 Komunikacija

Za komunikacijo med nadzorno enoto in mikrokrmilniškim sistemom je uporabljen standard RS232. Ta omogoča dvosmerni asinhroni serijski prenos podatkov po dveh linijah, kjer se podatki prenašajo kot biti, združeni v znake. BIOS uporablja za prenos znake dolžine 8 bitov brez paritete. Format podatkov je prikazan na sliki 4.1.



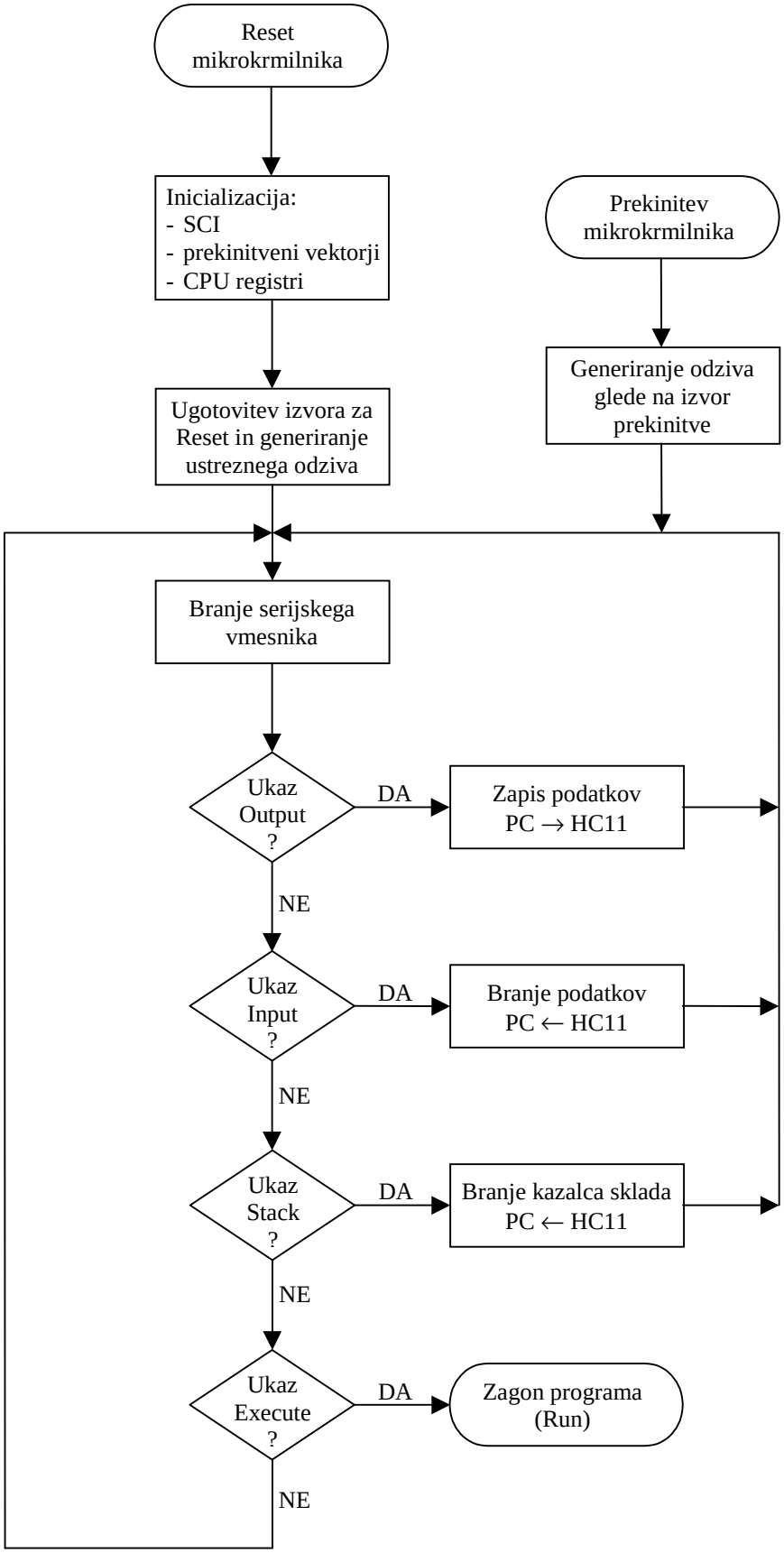
Slika 4.1: Format podatkov

Okvir enega znaka sestavlja: začetni (Start) bit, 8-bitni znak (Character) in končni (Stop) bit. Vrstni red prenosa bitov pri znaku poteka od bita z najmanjšo utežjo (LSB) do bita z največjo utežjo (MSB). Neaktivno stanje linije (Idle) predstavlja logična “1”.

Hitrost prenosa je 38400 bitov na sekundo (Baud Rate), kar omogoča hitro komunikacijo, ki je predvsem potrebna pri nalaganju programov v mikrokrmilniški sistem.

4.3 Zgradba programa

Diagram poteka (Flow Chart) programa BIOS-a je prikazan na sliki 4.2. Program ob resetu mikrokrmilnika najprej inicializira serijski vmesnik SCI in nastavi ustrezne vektorje za prekinitve ter inicializira CPU registre. Nato ugotovi izvor reseta in generira pripadajoči odziv. Za tem se začne glavna zanka programa, v kateri se čaka na ukaz iz nadzorne enote, ki ga pošlje preko serijskega vmesnika. Ukaz se nato izvrši in program se vrne v glavno zanko. Edina izjema je ukaz za zagon uporabniškega programa (Run), pri katerem program BIOS-a prepusti izvajanje uporabniškemu programu. V program BIOS-a se lahko vrnemo le preko reseta mikrokrmilnika ali pa prekinitve (maskirane ali nemaskirane). V primeru prekinitve bo BIOS najprej ugotovil izvor prekinitve in generiral pripadajoči odziv. Po tem pa BIOS nadaljuje izvajanje programa v glavni zanki, kjer čaka na nov ukaz. Prekinitev je možna le, ko ima nadzor uporabniški program (če je omogočena). V primeru, da je zahteva za prekinitev prišla, ko ima nadzor BIOS, se prekinitev izvede šele, ko dobi nadzor uporabniški program (če je zahteva po prekinitvi še vedno prisotna).



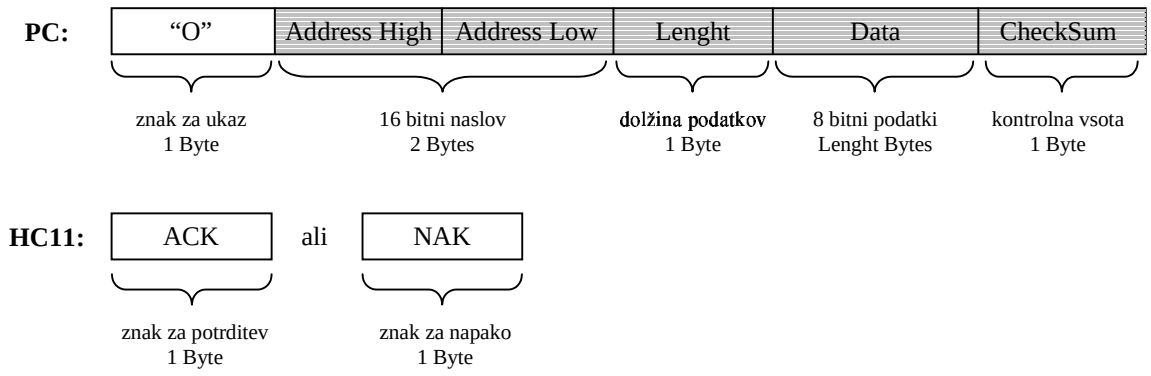
Slika 4.2: Diagram poteka

4.4 Ukazi

Ukazi BIOS-a omogočajo bralno / pisalni dostop do vseh pomnilniških lokacij iz celotnega naslovnega prostora mikrokrmilnika in do registrov centralno procesne enote. Poleg tega nam BIOS omogoča tudi zagon programov oz. nadaljevanje izvajanja po prekinitvi. Vse to se izvaja preko štirih ukazov: zapis podatkov (Output), branje podatkov (Input), branje kazalca sklada (Stack) in zagon programa (Execute).

4.4.1 Zapis podatkov (Output)

Ta ukaz je namenjen za zapis podatkov v zaporedne pomnilniške lokacije mikrokrmilniškega sistema iz celotnega naslovnega prostora. Format ukaza prikazuje slika 4.3.



Slika 4.3: Ukaz "Output"

Ukaz za zapis podatkov, ki ga pošlje osebni računalnik (PC), sestavljajo:

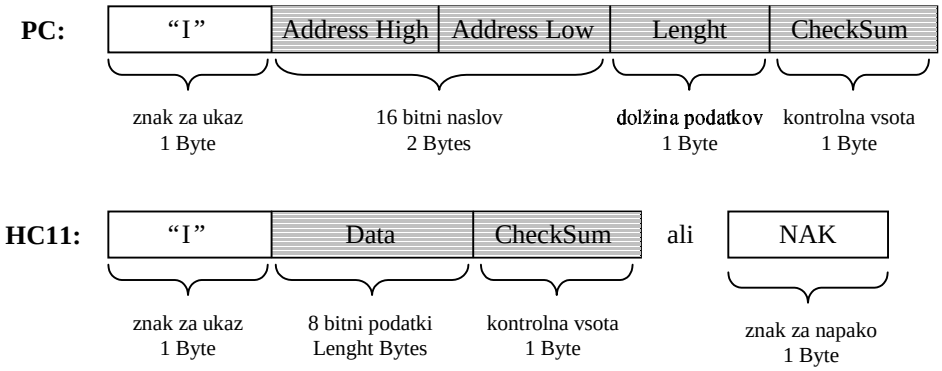
- znak za prepoznavanje ukaza Output ("O")
- 16-bitni naslov (razdeljen na zgornji in spodnji del: Address High in Address Low), ki določa začetni naslov, kamor bodo zapisani podatki
- dolžina podatkov, ki bo zapisana (Lenght)
- blok 8-bitnih podatkov, ki bodo zapisani (Data)
- kontrolna vsota (CheckSum), ki zagotovi, da je 8-bitna vsota vseh polj, razen znaka za ukaz (zasenčena polja na sliki), enaka nič

Mikrokrmilniški sistem (HC11) potrdi ukaz z znakom ACK, če ga je pravilno razumel in je bila pravilna kontrolna vsota. V nasprotnem primeru pa odgovori z znakom NAK.

S pomočjo ukaza za zapis podatkov lahko nalagamo programsko kodo ali zapisujemo podatke v pomnilnike mikrokrmilniškega sistema. Ravno tako lahko zapisujemo vrednosti v registre mikrokrmilnika in paralelnega vmesnika ter morebitnih dodanih zunanjih enot. Posredno pa lahko zapisujemo vrednosti tudi v registre centralno procesne enote.

4.4.2 Branje podatkov (Input)

Ta ukaz omogoča branje podatkov iz zaporednih pomnilniških lokacij mikrokrmilniškega sistema s celotnega naslovnega prostora. Format ukaza prikazuje slika 4.4.



Slika 4.4: Ukaz "Input"

Ukaz za branje podatkov, ki ga pošlje osebni računalnik (PC), sestavljajo:

- znak za prepoznavanje ukaza Input ("I")
- 16-bitni naslov (razdeljen na zgornji in spodnji del: Address High in Address Low), ki določa začetni naslov, od koder bodo prebrani podatki
- dolžina podatkov, ki bo prebrana (Lenght)
- kontrolna vsota (CheckSum), ki zagotovi, da je 8-bitna vsota vseh polj, razen znaka za ukaz (zasenčena polja na sliki), enaka nič

Za pravilni ukaz mikrokrmilniški sistem (HC11) vrne prebrane podatke v naslednji obliki:

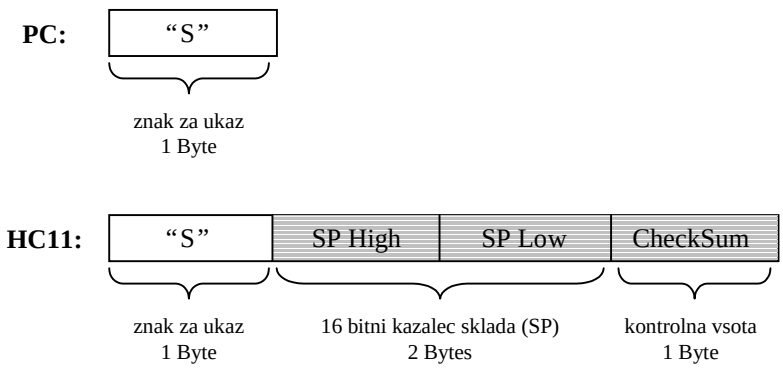
- znak za prepoznavanje ukaza Input ("I")
- blok 8-bitnih prebranih podatkov (Data)
- kontrolna vsota (CheckSum), ki zagotovi, da je 8 bitna vsota vseh polj, razen znaka za ukaz (zasenčena polja na sliki), enaka nič

V primeru nerazumljivosti ukaza ali nepravilne kontrolne vsote pa mikrokrmilniški sistem odgovori z znakom NAK.

Ukaz za branje podatkov uporabljamo za branje podatkov iz pomnilnikov mikrokrmilniškega sistema, registrov mikrokrmilnika in paralelnega vmesnika ter morebitnih dodanih zunanjih enot. Posredno pa ta ukaz omogoča tudi branje registrov centralno procesne enote.

4.4.3 Branje kazalca sklada (Stack)

Ta ukaz omogoča branje vrednosti kazalca sklada (Stack Pointer), ki je eden izmed registrov centralno procesne enote. Format ukaza je prikazan na sliki 4.5.



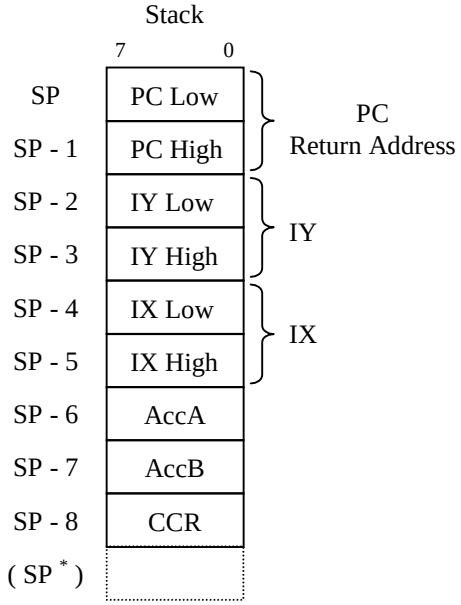
Slika 4.5: Ukaz "Stack"

Ukaz za branje kazalca sklada sproži osebni računalnik, ko pošlje znak "S". Mikrokrmilniški sistem vrne prebrano vrednost kazalca sklada v naslednji obliki:

- znak za prepoznavanje ukaza Stack ("S")
- prebrana 16-bitna vrednost kazalca sklada razdeljena na zgornjih in spodnjih 8 bitov (SP High in SP Low)
- kontrolna vsota (CheckSum), ki zagotovi, da je 8-bitna vsota vseh polj, razen znaka za ukaz (zasenčena polja na sliki), enaka nič

Registri centralno procesne enote niso dostopni v naslovnem prostoru mikrokrmilniškega sistema, ampak le preko posebnih ukazov mikrokrmilnika. Ob prekinitvi pa se registri shranijo na sklad, kar izkorišča program BIOS-a za posredni dostop do omenjenih registrov. Ta ukaz nam omogoča, da preberemo vrednost kazalca sklada, ki hkrati kaže tudi na blok preostalih registrov centralno procesne enote, ki so preslikani v pomnilnik.

Na sliki 4.6 je prikazan sklad, ki vsebuje registre centralno procesne enote in položaj kazalca sklada po prekinitvi mikrokrmilniškega programa.



Slika 4.6: Sklad po prekinitvi mikrokrmilniškega programa

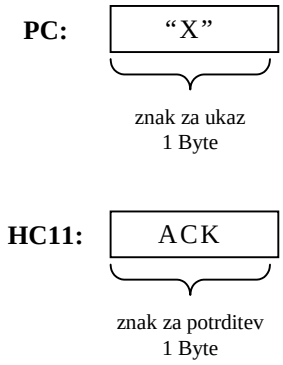
Ob prekinitvi uporabniškega programa se shranijo na sklad registri centralno procesne enote v naslednjem vrstnem redu: programski števec PC (povratni naslov prekinitvenega programa), indeksni register IY in IX, akumulator AccA in AccB, pogojno kodni register CCR. Po prekinitvi prevzame program BIOS-a nadzor nad izvajanjem programske kode. S pomočjo preslikave registrov v sklad je posredno omogočen bralno / pisalni dostop do vseh registrov centralno procesne enote (razen do SP). Kajti prvotno preslikane in morebiti spremenjene vrednosti v skladu se nazaj preslikajo v registre ob koncu prekinitve, kar predstavlja nadaljevanje izvajanja uporabniškega programa.

V času, ko je uporabniški program prekinjen in ima nadzor program BIOS-a, je omogočen bralno / pisalni dostop do registrov kar preko pomnilnika, kjer je sklad. Vrednosti registrov lahko v sklad zapisujemo preko ukaza “Output” in beremo preko ukaza “Input”. Najprej pa moramo seveda prebrati trenutno vrednost kazalca sklada z ukazom “Stack”, da ugotovimo naslove ustreznih pomnilniških lokacij za posamezne registre.

Dejanska vrednost kazalca sklada se ob prekinitvi sicer zmanjša za 9 lokacij, ker se na sklad shranijo registri centralno procesne enote. Na sliki 4.6 je ta kazalec označen s (SP^*). Ukaz “Stack” pa vrne vrednost kazalca sklada, povečano za 9 (na sliki 4.6 označen s SP). Na ta način dosežemo, da uporabnik pri koračnem izvajanju programov (uporabljajo se programske prekinitve) ne opazi vplivov prekinitev na vrednost kazalca sklada. Tako prebrani kazalec sklada vedno kaže na prvo prosto lokacijo za uporabniški program, ki pa ob prekinitvi hkrati kaže tudi na začetek preslikanega bloka registrov centralno procesne enote.

4.4.4 Zagon programov (Execute)

S pomočjo tega ukaza lahko poženemo izvajanje uporabniškega programa oz. nadaljujemo izvajanje prekinjenega programa. Format ukaza je prikazan na sliki 4.7.



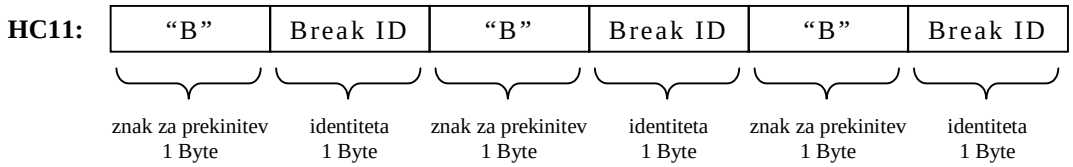
Slika 4.7: Ukaz "Execute"

Ukaz za zagon programa sproži osebni računalnik z znakom "X". Mikrokrmilniški sistem potrdi sprejeti ukaz z znakom ACK in začne izvajanje uporabniškega programa.

Preko tega ukaza program BIOS-a prepusti nadzor izvajanja uporabniškemu programu. To izvede mikrokrmilniški sistem z ukazom RTI (Return from Interrupt). Ob tem se iz sklada naložijo vrednosti registrov v obratnem vrstnem redu, kot so bili shranjeni ob prekinitvi (CCR, AccB, AccA, IX, IY in PC - glej sliko 4.6). Nazadnje se naloži programski števec, ki definira naslov, od koder se bo začelo oz. nadaljevalo izvajanje uporabniškega programa.

4.5 Odziv na prekinitve in reset

Če pride pri izvajanju uporabniškega programa do prekinitve ali reseta mikrokrmilniškega sistema, bo BIOS ugotovil izvor prekinitve in generiral pripadajoči odziv. Format odziva na prekinitev oz. reset prikazuje slika 4.8.



Slika 4.8: Odziv na prekinitev in reset

Odziv sestavljata znak za prekinitev (“B”) in znak za identiteto prekinitve (Break ID), ki se dvakrat ponovita (večja zanesljivost prepoznavanja in ločitev od uporabniških odzivov). Vsi možni izvori prekinitev in njihovi znaki za identiteto so prikazani na sliki 4.9.

ID (Dec)	Izvor prekinitve
0	**Power-on or Manual Reset**
1	**Clock Monitor Fail Reset**
2	**COP Failure Reset**
3	**Illegal Opcode Trap**
4	**Software Interrupt (SWI)**
5	**XIRQ (Pseudo Non-Maskable Interrupt)**
6	IRQ (Maskable Interrupt)
7	Real Time Interrupt (RTI)
8	Timer Input Capture 1
9	Timer Input Capture 2
10	Timer Input Capture 3
11	Timer Output Compare 1
12	Timer Output Compare 2
13	Timer Output Compare 3
14	Timer Output Compare 4
15	Timer Output Compare 5
16	Timer Overflow
17	Pulse Accumulator Overflow
18	Pulse Accumulator Input Edge
19	Serial Peripherial Interface (SPI) Interrupt
20	Serial Communication Interface (SCI) Interrupt

Slika 4.9: Prekinitve in znaki za identiteto

BIOS ima definirane vektorje za vse tri možne izvore reseta mikrokrmilnika. Poleg tega pa ob inicializaciji nastavi še ustrezne vektorje za naslednje prekinitve: programska prekinitev (SWI), nemaskirana prekinitev (XIRQ) in nepravilna koda (Illegal Opcode). Skupno je torej definiranih šest vektorjev za prekinitve (krepko natisnjene na sliki 4.9). Te prekinitve so namreč potrebne za delovanje programskega razvojnega orodja.

Če želi uporabnik izkoristiti tudi odzive, ki jih nudi BIOS za ostale prekinitve, mora ustrezno definirati vektorje za prekinitvene podprograme (preslikati mora pripravljene vektorje iz EPROM-a v programski pomnilnik).

4.6 Omejitve

BIOS je izdelan kot preprost, vendar zelo učinkovit program, ki preko serijskega vmesnika omogoča osebnemu računalniku popoln nadzor nad mikrokrmilniškim sistemom. Kljub temu pa nam postavlja določene omejitve.

Eno izmed takih omejitev nam predstavljajo zaščiteni registri mikrokrmilnika, ki jih lahko spreminjamo le prvih 64 strojnih ciklov po resetu. Ker ima takoj po resetu mikrokrmilnika nadzor program BIOS-a, ki čaka na ukaze uporabnika, ni možno spreminjati vsebine teh zaščitenih registrov. To sicer ne predstavlja večjih težav, saj so zaščiteni registri uporabljeni za nekatere začetne nastavitve mikrokrmilnika, ki jih uporabniku običajno ni treba spreminjati.

Druga taka omejitev predstavlja delno omejena uporaba serijskega vmesnika. Ta je namreč namenjen za komunikacijo med osebnim računalnikom in mikrokrmilniškim sistemom, kadar uporabljamo programsko razvojno orodje. Še vedno pa ga lahko uporablja tudi uporabniški program, vendar ne sme spreminjati nastavitev serijskega vmesnika, kot so hitrost prenosa podatkov, format podatkov …

Zavedati se je potrebno tudi, da so določene funkcije programskega razvojnega orodja (zagon programov, koračno izvajanje in ustavljanje programov) zasnovane na prekinitvah, ki seveda uporabljajo sklad za shranjevanje vsebine registrov. Zato bi lahko nepremišljeno spreminjanje vsebine pomnilnika sklada, kjer so trenutno shranjeni registri centralno procesne enote, povzročilo nezaželene učinke.

5. Programsko razvojno orodje

"HC11 IDT" je programsko razvojno orodje, ki predstavlja integrirana orodja, potrebna pri razvoju aplikacij. Vsebuje prijazen grafični vmesnik za komunikacijo z uporabnikom, ki deluje na osebnem računalniku z Windows 95/98/NT operacijskim sistemom. Povezan je z mikrokrmilniškim sistemom preko serijskega vmesnika in opisanega BIOS-a.

5.1 Značilnosti

Programsko razvojno orodje omogoča pisanje in urejanje programov, odpravljanje napak v sintaksi in prevajanje programov v strojno kodo ter nalaganje programov v mikrokrmilniški sistem. Programe lahko poženemo ali pa izvajamo po korakih (uporabljamo lahko tudi prekinitvene točke). Poleg tega nam omogoča pregledovanje in urejanje vsebine celotnega naslovnega prostora, v katerem so vključeni pomnilniki, notranji registri mikrokrmilnika in paralelnega vmesnika ter morebitne dodane zunanje enote.

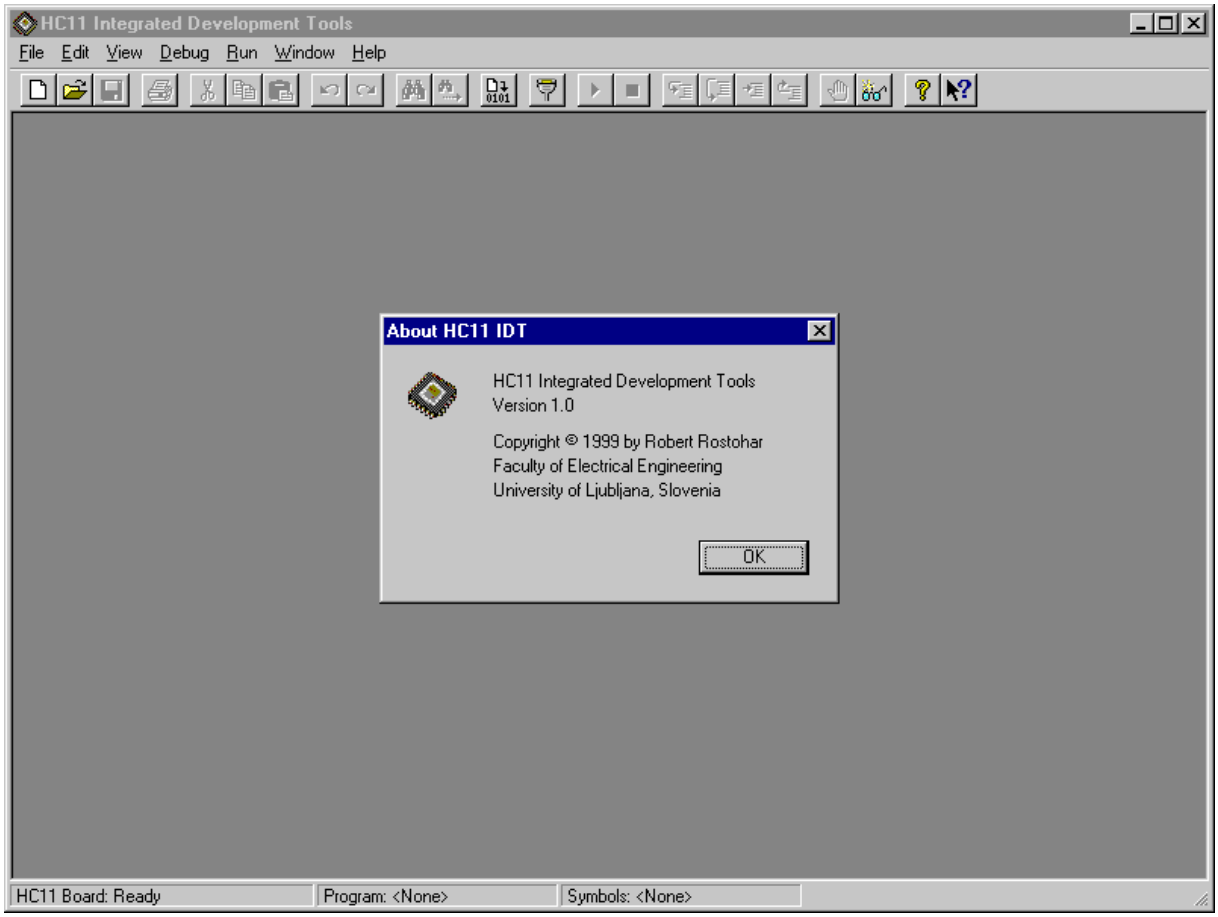
Glavne značilnosti razvojnega orodja "HC11 IDT":

- vgrajen urejevalnik teksta (Editor)
- vgrajen makro-zbirniški prevajalnik (Macro Assembler)
- vgrajen inverzni prevajalnik (Disassembler)
- podpora simbolov (Symbols)
- nalaganje programov (Download)
- zagon programov (Run)
- izvajanje programov po korakih (Single Stepping)
- prekinitvene točke (Breakpoints)
- opazovalni elementi (Watches)
- pregled in urejanje vsebine vseh registrov
- pregled in urejanje celotnega naslovnega prostora (64 kBytes Memory Map)
- podpora osnovnih številskih sestavov (DEC, BIN, OCT in HEX)
- vgrajen serijski terminal
- sprotna pomoč (Online Help)

Programsko razvojno orodje nudi celotno podporo za razvoj aplikacij, ki vsebuje načrtovanje programov in v povezavi z mikrokrmilniškim sistemom preverjanje delovanja programov.

5.2 Grafični vmesnik

Grafični vmesnik omogoča uporabniku, da na enostaven način uporablja vse funkcije, ki jih nudi programsko razvojno orodje. Izdelan je bil s programskim orodjem “Visual Basic 5.0”. Izgled grafičnega vmesnika je prikazan na sliki 5.1.



Slika 5.1: Izgled grafičnega vmesnika

Na vrhu vmesnika se nahajajo menuji, ki omogočajo izbiranje ukazov. Za večino ukazov so na razpolago tudi kombinacije tipk kot bližnjice (Shortcuts). Pomembnejše in pogostejše ukaze pa lahko izbiramo tudi preko ikon v orodni vrstici, ki se nahaja pod menuji. Spodaj pa se nahaja statusna vrstica, ki prikazuje trenutno stanje mikrokrmilniškega sistema in naloženi program ter naložene simbole. Možna stanja mikrokrmilniškega sistema so: ni priključen (Not Connected), se ne odziva (Not Responding), izvaja se program (Running), pripravljen (Ready), v resetu (Ready - Reset), prekinjen (Ready - Break), nepravilna koda (Ready - Illegal Opcode). Na delovni površini pa se lahko nahajajo razna okna, kot so: Editor, Disassembler, Memory Window, CPU Registers, Stack, Watch Window, Registers, Serial Terminal …

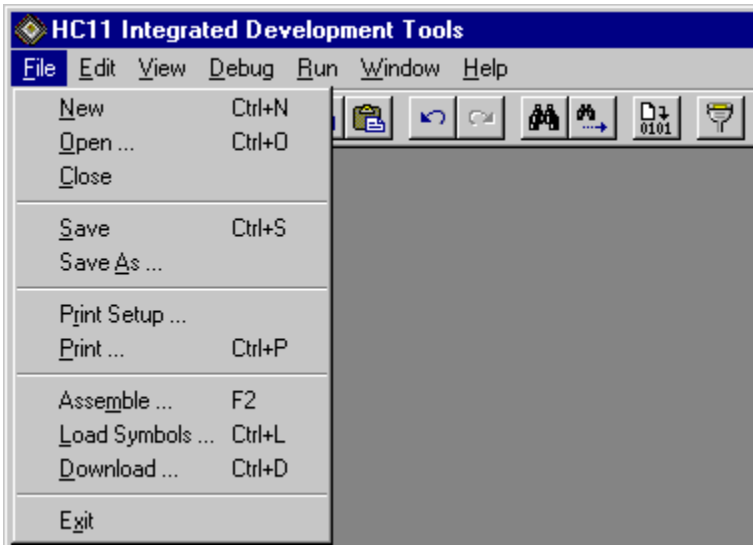
5.2.1 Datoteke

Programsko razvojno orodje uporablja različne vrste datotek (files), od katerih ima vsaka svojo funkcijo. Za ustrezno ločitev so uporabljene različne končnice pri imenih datotek. Na sliki 5.2 so zbrane vse uporabljene vrste datotek in njihove končnice.

Končnica	Vrsta datoteke
*.asm	zbirniški program (Assembly Source)
*.lst	izpis prevedenega programa (Listing File)
*.err	spisek napak (Error Listing)
*.txt	tekstovna datoteka (Text File)
*.s19	strojni program (Machine Code)

Slika 5.2: Datoteke in končnice

Za delo z datotekami so namenjeni ukazi iz menuja “File”. Ti ukazi so prikazani na sliki 5.3.



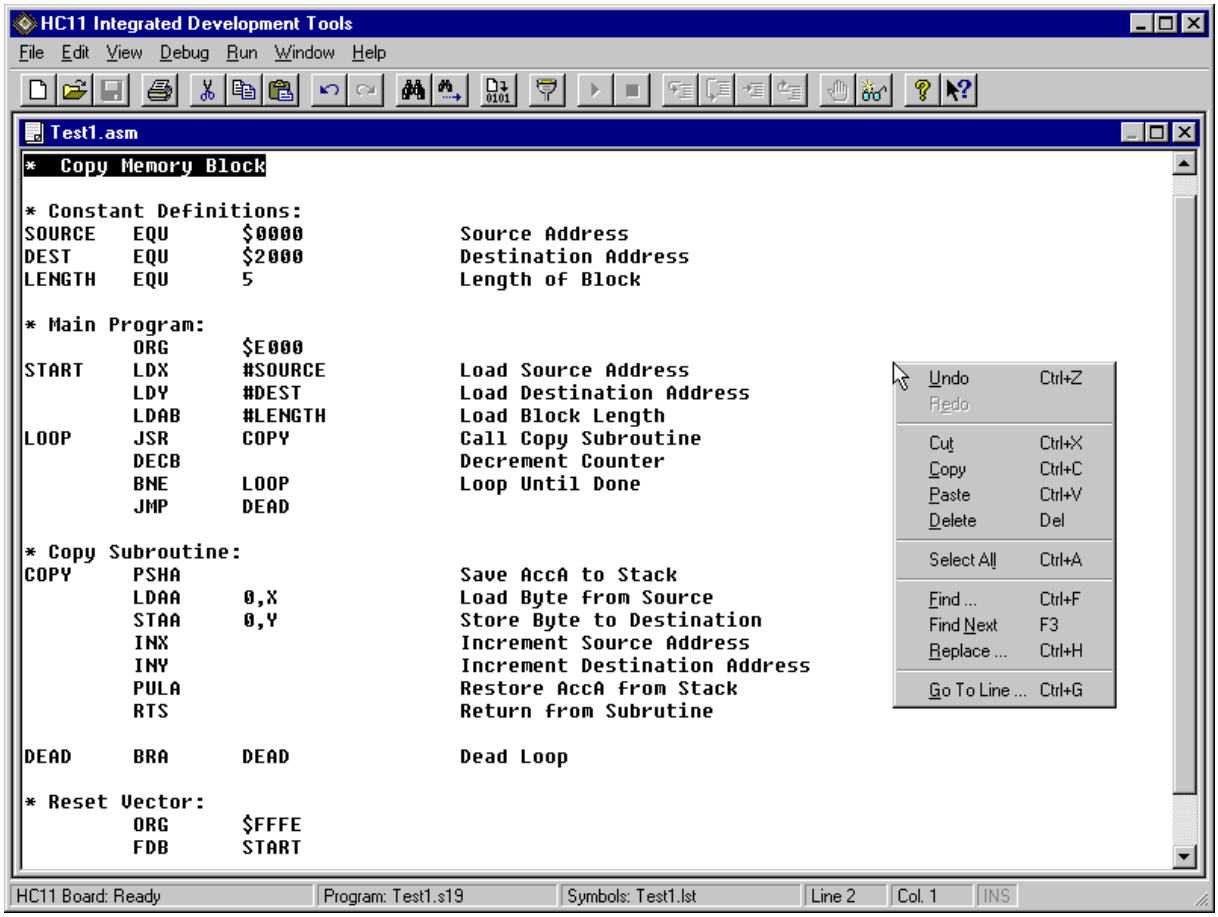
Slika 5.3: Menu “File”

Večina ukazov se nanaša na urejevalnik teksta oz. Editor. To so ukazi: New - kreiranje nove datoteke, Open - odpiranje obstoječe datoteke, Close - zapiranje odprte datoteke, Save in Save As - shranjevanje datoteke, Print - tiskanje datoteke. Ukaz Assemble sproži prevajanje v strojni program, ukaz Download pa naloži prevedeni program v mikrokrmilniški sistem. Ukaz Load Symbols omogoča dodatno nalaganje simbolov (izvede se avtomatsko z Assemble).

5.2.2 Okno “Editor”

Editor predstavlja urejevalnik teksta, ki je namenjen predvsem za pisanje in urejanje izvornih zbirniških programov ter za pregledovanje izpisov prevedenih programov ali spiskov napak. Vgrajene ima standardne funkcije za urejanje, kot so: Razveljavi (Undo), Ponovi (Redo), Izreži (Cut), Kopiraj (Copy), Prilepi (Paste), Izbriši (Delete), Izberi vse (Select All), Najdi (Find), Najdi ponovno (Find Next), Zamenjaj (Replace), Pojdi v vrstico (Go To Line). Vsi ti ukazi so zbrani v menuju “Edit” (dostopen tudi z desnim klikom miške v oknu Editorja).

Na sliki 5.4 je prikazano aktivno okno urejevalnika teksta oz. Editorja z naloženo datoteko “Test1.asm”. Prav tako je prikazan tudi aktiviran menu “Edit” znotraj okna.



Slika 5.4: Okno “Editor”

Kadar je Editor aktivno okno, se v statusni vrstici pojavita še številka vrstice (Line) in številka stolpca (Col), ki kažeta na trenutni položaj kurzorja ter stanje tipke Insert (INS).

5.2.3 Modul “Assembler”

Assembler je zbirniški prevajalnik, ki izvorni zbirniški program (*.asm) prevede v strojno kodo z Motorolinim S19 formatom (*.s19). Hkrati generira izpis prevedenega programa (*.lst), če pa se pojavi napaka v sintaksi, pa generira datoteko s spiskom napak (*.err). Uporabljen je zunanji prevajalnik z izvršnim programom AS11.EXE.

Vrstica zbirniškega programa ima sledeči format:

```
Label  Instruction  Comment
```

Na začetku je oznaka vrstice (Label), ki je lahko tudi izpuščena. Sestavljena je lahko iz črk in številk, vendar mora biti prvi znak črka. Uporablja se kot referenca pri skokih in podprogramih ter kot kazalec na pomnilniško lokacijo. Sledi ukaz (Instruction), ki je lahko mnemonik za ukaz mikrokrmilnika ali pa psevdo ukaz prevajalnika. Nazadnje pa je lahko dodan še komentar (Comment) ali pa je izpuščen. Kot ločilo med posameznimi polji se uporablja en ali več presledkov ali en ali več tabulatorjev. Če se vrstica začne z znakom “*”, predstavlja to komentar in je prevajalnik ne upošteva.

Prevajalnik prepozna psevdo ukaze, ki so navedeni v sliki 5.5.

Psevdo ukaz	**Pomen**	**Primer**
Label EQU <Word>	Equal - definiranje label	YEAR EQU 1974
ORG <Word>	Origin - nastavitev 16-bitnega lokacijskega števca	ORG $E000
FCB <Byte>, …	Form Constant Byte - definiranje 8-bitnih konstant	FCB $00,%01010011
FDB <Word>, …	Form Double Byte - definiranje 16-bitnih konstant	FDB $ABCD,$2702
FCC "<Text>"	Form Constant Char - definiranje tekstovnih konstant	FCC "Simple Text"
RMB <Number>	Reserve Memory Bytes - rezerviranje 8-bitnih lokacij	RMB 27
FILL <Byte>,<Count>	Fill Memory - polnjenje bloka pomnilnika	FILL $00,5
END	End Program - konec programa	END

Slika 5.5: Psevdo ukazi prevajalnika

Prevajalnik podpira zapis števil v štirih osnovnih številskih formatih (DEC, HEX, BIN, OCT). Številko zapišemo v ustreznem formatu z uporabo posebnih znakov za predpono. Vsi številski formati s pripadajočimi predponami so prikazani na sliki 5.6. Števila, ki so zapisana brez predpone, predstavljajo običajna desetiška števila.

Predpona	Številski format	Primer
brez	DEC - decimalni	1974
$	HEX - heksadecimalni	$B25E
%	BIN - binarni	%01010001
@	OCT - oktalni	@12345

Slika 5.6: Številski formati in predpone

Hkrati s prevajanjem se naložijo tudi simboli v računalniški pomnilnik. To je del generiranega izpisa prevedenega programa, ki vključuje tudi izvorni zbirniški program. Omogoča uporabo label namesto konstantnih številk in ostalih simbolov ter komentarjev, kar še dodatno olajša delo pri preverjanju izvajanja programov.

5.2.4 Modul "Download"

Modul "Download" skrbi za nalaganje programske kode in podatkov, ki so prevedeni v strojno kodo (format S19) v mikrokrmilniški sistem. Nalaganje se izvaja s pomočjo ukaza "Output", ki ga nudi BIOS in je namenjen za zapis podatkov v mikrokrmilniški sistem.

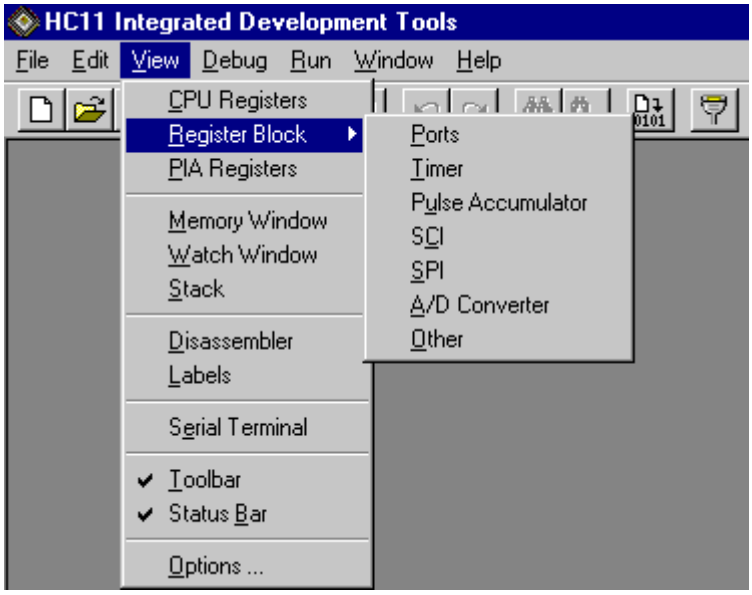
Nalaganju strojne kode je dodano še preverjanje, ki vsak zapisani podatek prebere nazaj in primerja s prvotno poslanim podatkom. Branje podatkov iz mikrokrmilniškega sistema je izvedeno s pomočjo ukaza "Input", ki ga omogoča BIOS.

Ker imamo na razpolago hiter asinhroni serijski vmesnik za prenos podatkov, traja prenos strojne kode dolžine 1 kByte le dobro sekundo (vključno s preverjanjem).

5.2.5 Modul "Debugger"

Debugger (razhroščevalnik) programov predstavlja orodje, ki se uporablja za preverjanje delovanja programov in iskanje ter odpravljanje morebitnih napak (Bug - hrošč). Vgrajen ima inverzni prevajalnik (Disassembler) in omogoča koračno izvajanje programov. Pri tem se lahko uporabljajo tudi prekinitvene točke (Breakpoints). Poleg tega omogoča pregledovanje in urejanje vsebine vseh registrov in celotnega pomnilniškega naslovnega prostora.

Modul “Debugger” je sestavljen iz več oken: Disassembler, Labels, Registers, Stack, Watch Window, Memory Window in posredno tudi Serial Terminal. Do teh oken dostopamo preko ukazov, ki jih najdemo v menuju “View”. Ti ukazi so prikazani na sliki 5.7.



Slika 5.7: Menu “View”

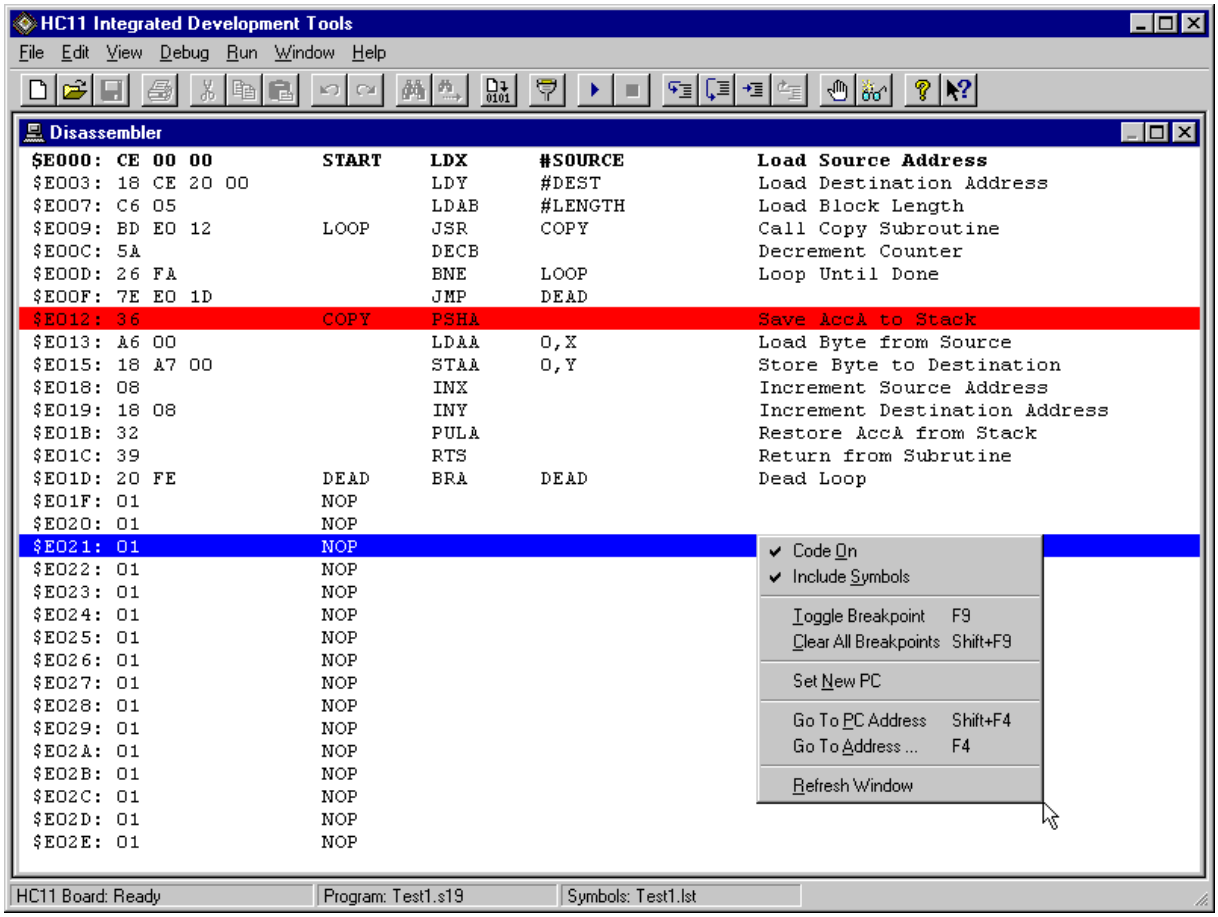
Večina ukazov iz menuja “View” se nanaša na modul “Debugger”. Z njimi lahko odpremo ali zapremo izbrano okno. Ukazi so smiselno grupirani, da omogočajo večjo preglednost.

5.2.5.1 Okno “Disassembler”

Disassembler je inverzni prevajalnik, ki iz strojne kode generira mnemonične ukaze in jih prikaže na zaslonu. S tem nam omogoča pregled programa v razumljivi obliki. Poleg tega nam omogoča tudi koračno izvajanje programov. Zato predstavlja Disassembler glavno okno in ključni del Debuggerja, ki ga uporabljamo predvsem pri analiziranju delovanja programov.

Okno “Disassembler” sestavljajo vrstice, kjer so zapisani: naslov, strojna koda (če je vključena opcija “Code On”) in pripadajoči mnemonični ukaz mikrokrmilnika. Poleg tega lahko vrstica vključuje tudi izvorni zbirniški program, če so naloženi simboli in če je vključena opcija “Include Symbols”. Ukazi za delo s tem oknom (razen za koračno izvajanje - menu “Run”) so zbrani v menuju “Debug” (dostopen tudi z desnim klikom miške v oknu “Disasembler”).

Slika 5.8 prikazuje okno “Disassembler” z aktiviranim menujem “Debug” znotraj okna.



Slika 5.8: Okno “Disassembler”

Z ukazom “Go to Address” lahko izberemo začetni naslov, od katerega naprej bo izpisan program. Če pa želimo izpis od naslova, na katerega kaže programski števec, lahko uporabimo ukaz “Go To PC Address”. Z uporabo ukaza “Set New PC” lahko nastavimo novo vrednost programskega števca, da kaže na ukaz trenutno izbrane vrstice programa. Vrstica, ki ima enak naslov kot programski števec, je izpisana s krepko pisavo.

Pri generiranju izpisa disasembliranega programa se naprej iz mikrokrmilniškega sistema prebere strojna koda s pomočjo ukaza “Input”, ki je vgrajen v BIOS-u. Prebrana strojna koda se nato dekodira v mnemonične ukaze, ki se prikažejo. Če imamo naložene tudi simbole (izpis prevedenega zbirniškega programa) in so vključeni v prikaz, se namesto dekodiranega mnemoničnega ukaza prikaže celotna vrstica prevedenega zbirniškega programa, ki vsebuje tudi labele in komentarje. To velja seveda za primer, ko se strojna koda prevedene vrstice programa ujema z dejansko strojno kodo, prebrano iz mikrokrmilniškega sistema na istem naslovu. V nasprotnem primeru pa se vrstica prevedenega zbirniškega programa ne upošteva in se izpiše le mnemonik ukaza dekodirane dejanske strojne kode. Na ta način je omogočena tudi vključitev simbolov v izpis, kar še dodatno olajša delo pri analiziranju programov.

Poglavitni del pri analiziranju izvajanja programske kode predstavljajo vgrajeni ukazi, ki omogočajo koračno izvajanje programov. To so ukazi: Run, Step Into, Step Over, Step Out, Run To Cursor in Stop, ki so zbrani v menuju "Run". Menu in ukazi so prikazani na sliki 5.9.

Slika 5.9: Menu "Run"

Ukaz "Run" sproži izvajanje programa od trenutnega programskega števca naprej, kar se izvede s pomočjo ukaza "Execute", ki ga omogoča BIOS. Izvajanje programa se prekine, če naleti program na prekinitveno točko (Breakpoint). Izvajanje pa lahko tudi zaustavimo s pomočjo ukaza "Stop", ki sproži nemaskirano prekinitev NMI preko CTS linije. Z ukazom "Run" tudi nadaljujemo izvajanje prekinjenega oz. zaustavljenega programa.

Ukaza "Step Into" in "Step Over" omogočata izvršitev trenutnega ukaza, na katerega kaže programski števec. Razlikujeta se edino pri izvrševanju ukazov za klic podprogramov, kjer ukaz "Step Into" izvrši le skok na podprogram (vstopi v podprogram), ukaz "Step Over" pa izvrši celotni podprogram kot en sam ukaz. Pri koračnem izvajanju v podprogramih se uporablja še ukaz "Step Out", ki izvrši vse ukaze do konca trenutnega podprograma in se ustavi za ukazom, ki je klical podprogram. Ta ukaz namreč omogoča, da se na eleganten način vrnemo iz podprograma, če smo pomotoma vstopili vanj ali pa nas njegovo izvajanje več ne zanima. Ukaz "Run To Cursor" pa sproži izvajanje programa, ki se ustavi, ko program pride do ukaza trenutno izbrane vrstice (Cursor).

Vsi ukazi za omejeno izvajanje programa (Step Into, Step Out, Step Over in Run To Cursor) so realizirani tako, da se najprej ugotovi, kje se mora program ustaviti, in se tam postavi prekinitvena točka. Nato pa se sproži izvajanje programa preko ukaza "Execute". Ko se program ustavi, se postavljena prekinitvena točka odstrani. Na ta način dosežemo, da se vsi ukazi dejansko izvajajo v mikrokrmilniku, in sicer v realnem času. Ko pa program naleti na postavljeno prekinitveno točko, se program zaustavi in mikrokrmilniški sistem preko BIOS-a generira ustrezno sporočilo, ki ga pošlje razvojnemu orodju. Zato ima programsko razvojno orodje le vlogo krmiljenja in prikazovanja koračnega izvajanja programov, medtem ko se programi dejansko izvajajo v mikrokrmilniku.

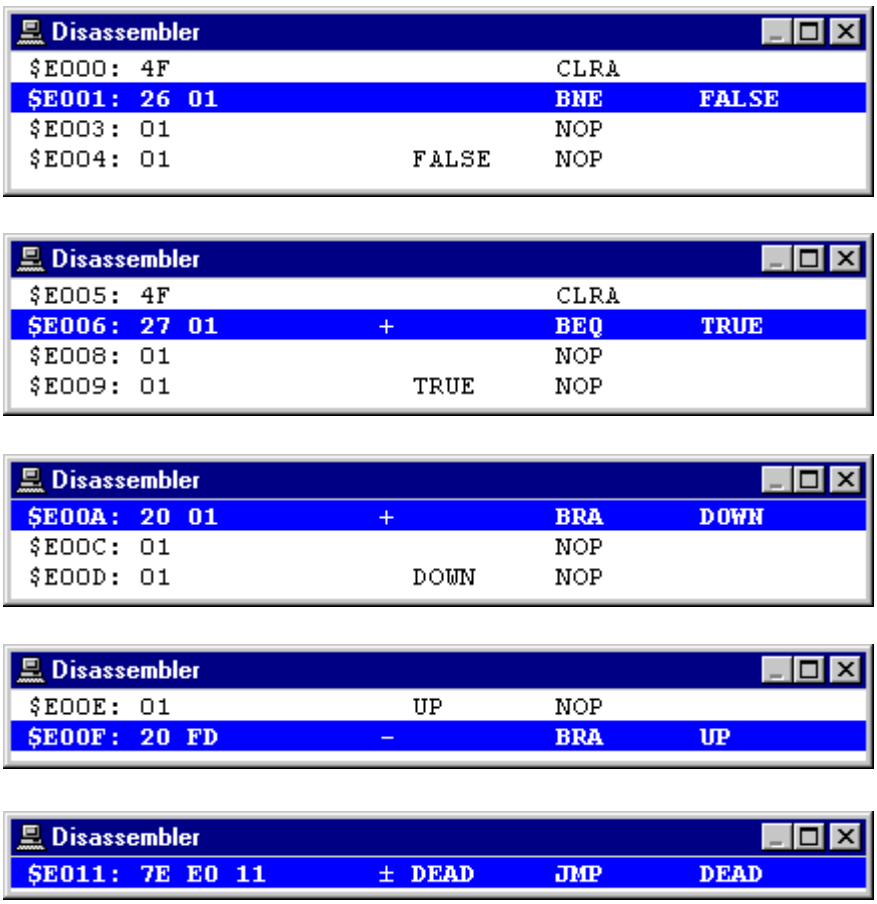
Prekinitvena točka (Breakpoint) se vstavlja tako, da se na mestu, kjer želimo prekinitev, pripadajoči ukaz programa zamenja z ukazom za programsko prekinitev - SWI. Prvotni ukaz se predhodno shrani v pomnilnik osebnega računalnika. Pri odstranjevanju prekinitvene točke pa se ukaz SWI zamenja nazaj s prvotnim ukazom. Vse to se realizira z uporabo ukaza "Output", ki je vgrajen v BIOS-u.

Prekinitvene točke lahko doda ali odstrani uporabnik s pomočjo ukaza "Toggle Breakpoint" za trenutno izbrano vrstico ali kar z dvojnim klikom miške nad ustrezno vrstico programa.

Okno "Disassembler" nam prikazuje tudi potek nadaljevanja programa pri ukazih za brezpogojne in pogojne skoke (pogoji se preverjajo, ali so izpolnjeni ali ne).

Uporabljajo se posebni znaki, ki so postavljeni pred labelami:

- + skok navzdol (proti višjim naslovom)
- – skok navzgor (proti nižjim naslovom)
- ± skok na isti naslov
  skok se ne izvrši (brez oznake)



Skok se ne izvrši (neizpolnjen pogoj)

Skok navzdol (izpolnjen pogoj)

Skok navzdol

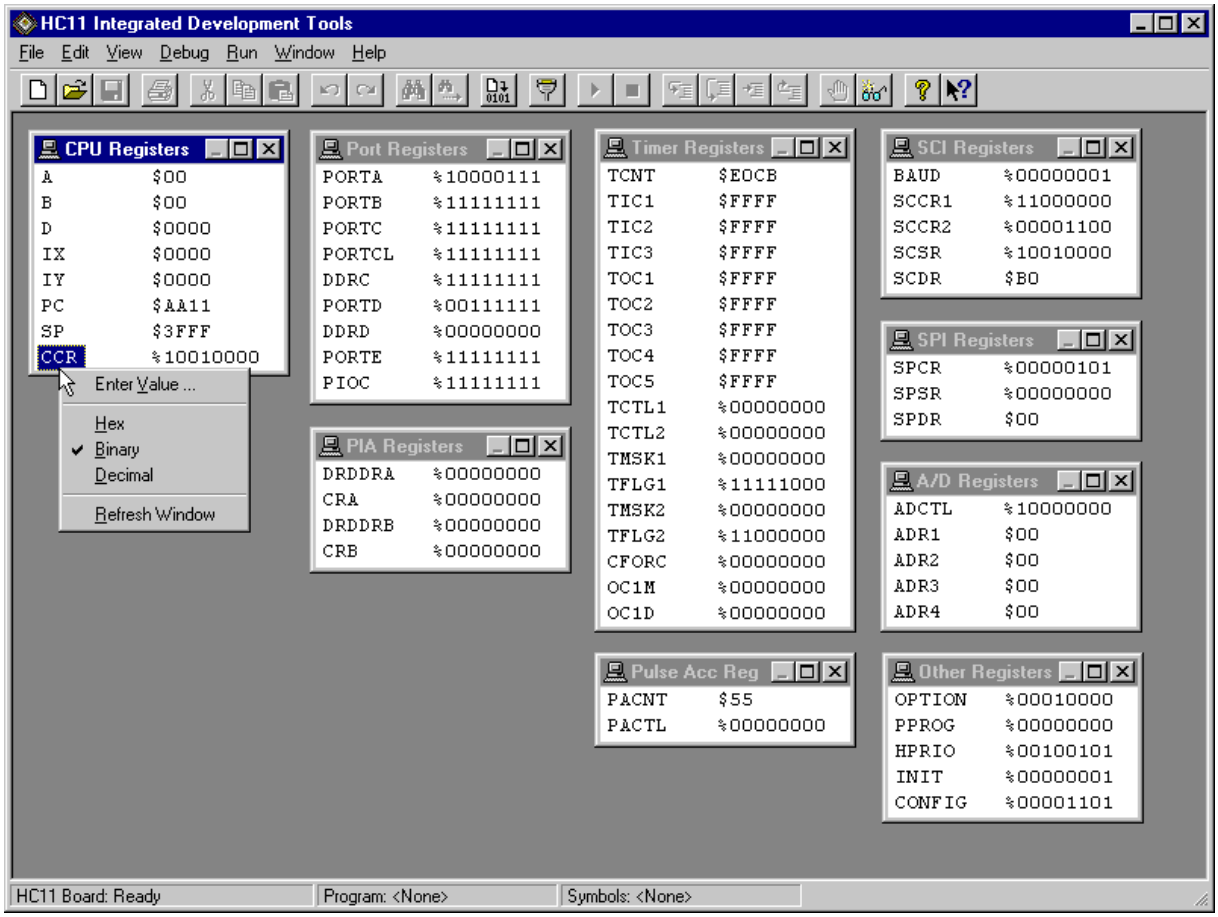
Skok navzgor

Skok na isti naslov

Slika 5.10: Prikaz poteka nadaljevanja programa pri skokih

5.2.5.2 Okna “Registers”

Mikrokrmilniški sistem vsebuje naslednje skupine registrov: registri centralno procesne enote (CPU Registers), notranji registri mikrokrmilnika (Port, Timer, Pulse Accumulator, SCI, SPI, A/D in Other Registers) ter registri paralelnega vmesnika (PIA Registers). Za vsako skupino je izdelano okno, kjer so razvrščeni posamezni registri. Na sliki 5.11 so prikazani vsi registri, razvrščeni v omenjenih skupinah. Prikazan je tudi aktiviran menu za registre, ki je dostopen z desnim klikom miške znotraj okna registrov.



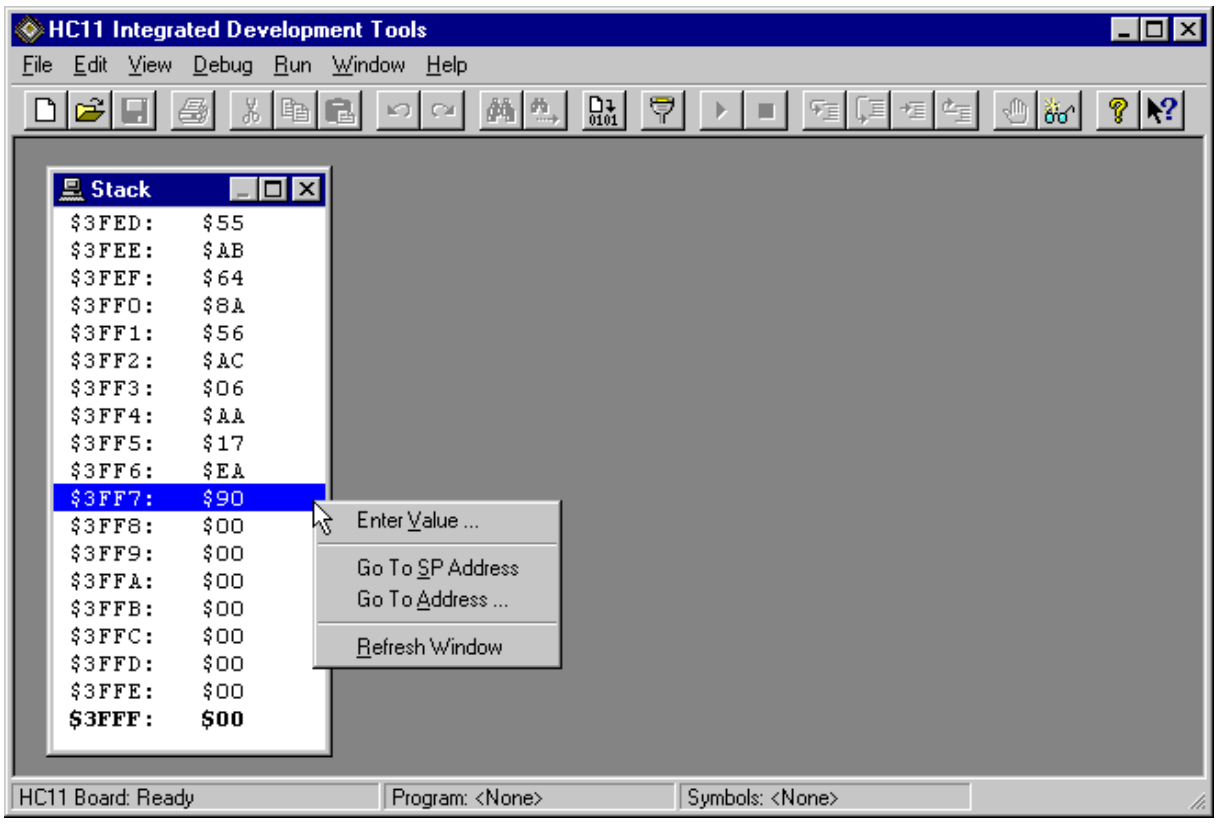
Slika 5.11: Okna “Registers”

Vsebino registrov lahko prikazujemo v osnovnih številskih sistemih (Decimal, Binary, Hex). S pomočjo ukaza “Enter Value” lahko vpišemo novo vrednost v registre, ki omogočajo vpis.

Vsi registri so mapirani v naslovnem prostoru mikrokrmilniškga sistema (registri centralno procesne enote so preslikani v pomnilnik). Za bralno / pisalni dostop se zato uporabljajo ukazi, ki ji nudi BIOS. Branje vrednosti registrov je izvedeno z ukazom “Input”, zapis novih vrednosti v registre pa z ukazom “Ouput”.

5.2.5.3 Okno “Stack”

“Stack” predstavlja okno, v katerem je prikazan del pomnilnika, ki se uporablja kot sklad in na katerega kaže kazalec sklada. Slika 5.12 prikazuje okno “Stack” z aktiviranim menujem.



Slika 5.12: Okno “Stack”

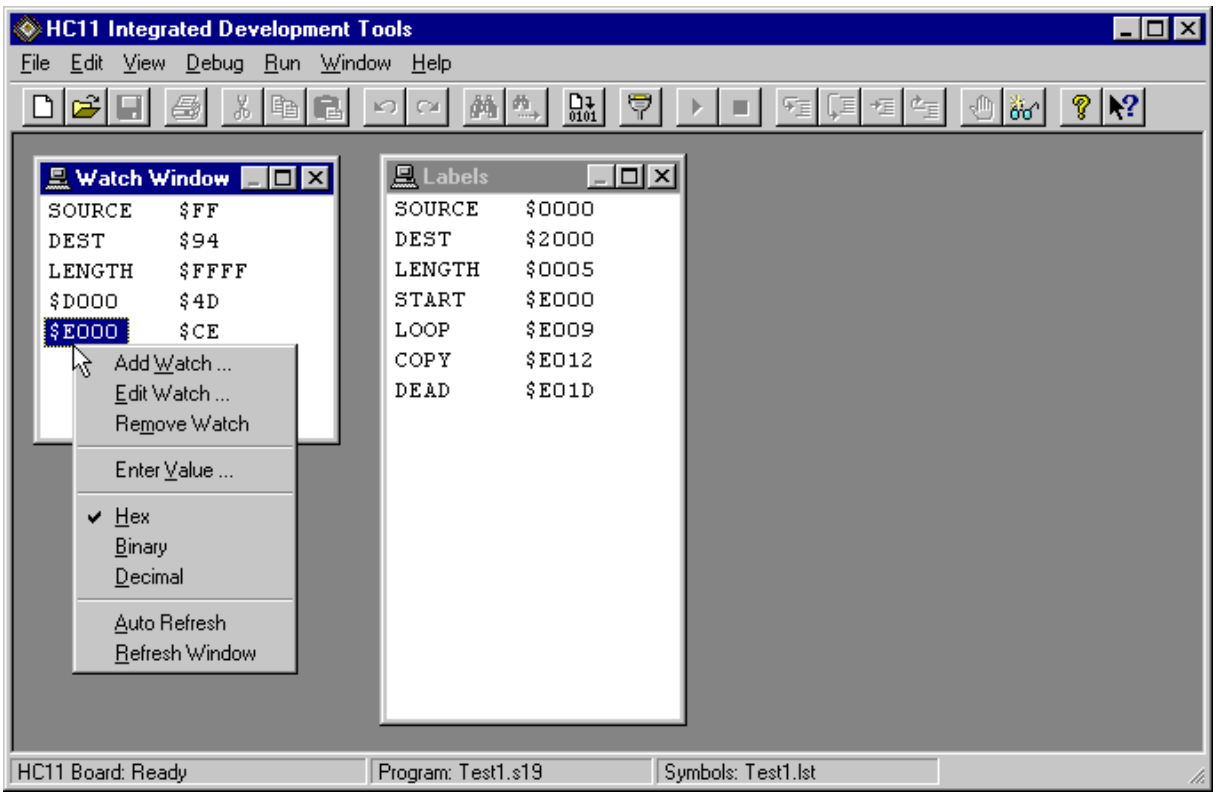
Vrstični prikaz je sestavljen iz naslovov in 8-bitnih vrednosti zaporednih pomnilniških lokacij. Vse številke so zapisane v heksadecimalnem formatu (Hex). Vrstica, ki kaže na isti naslov kot kazalec sklada, je izpisana s krepko pisavo.

Z ukazom “Go To Address” lahko izberemo začetni naslov, od katerega bo prikazan sklad. S pomočjo ukaza “Go To SP Address” pa bo prikazan sklad do vključno naslova, na katerega kaže kazalec sklada. Ukaz “Enter Value” je namenjen za zapis nove vrednosti na izbrano pomnilniško lokacijo sklada. Vsi ti ukazi so zbrani v menuju, ki je dostopen z desnim klikom miške znotraj okna “Stack”.

Za bralno / pisalni dostop do vrednosti pomnilniških lokacij sta uporabljena ukaza “Input” in “Output”, ki sta vgrajena v BIOS-u.

5.2.5.4 Okno “Watch”

Okno “Watch” omogoča pregled in urejanje vsebine za posebej izbrane 8-bitne ali 16-bitne pomnilniške lokacije. Pri izbiri naslovov lahko uporabljamo tudi labele, če so naloženi simboli. Pregled vseh label in pripadajočih vrednosti nam omogoča okno “Labels”. Slika 5.13 prikazuje primer za okno “Watch” z aktiviranim menujem in primer za okno “Labels”.



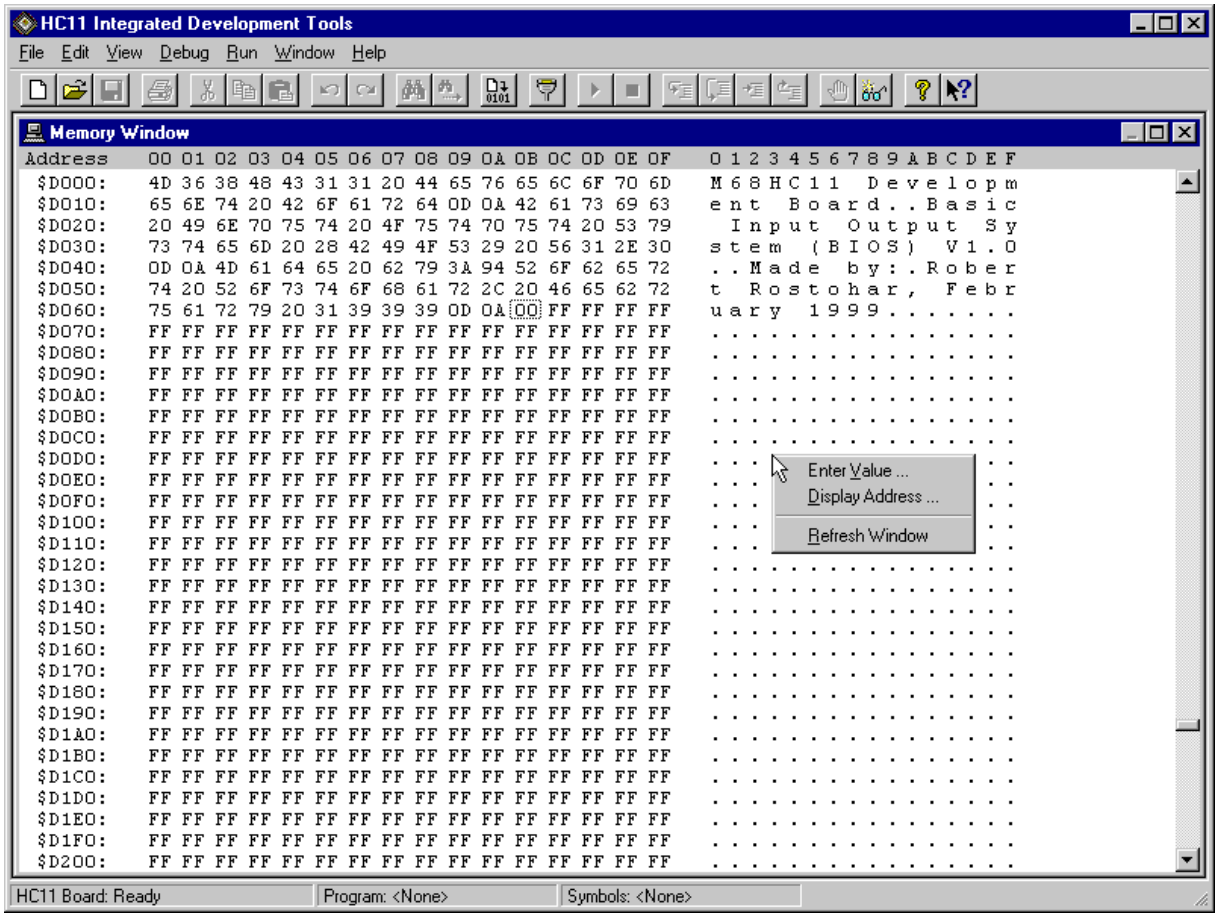
Slika 5.13: Okno “Watch” in okno “Labels”

Okno “Watch” sestavljajo vrstice, v katerih so prikazani opazovalni elementi (Watches). Ti vsebujejo naslov izbrane pomnilniške lokacije in njeno vrednost. Opazovalne elemente lahko dodamo z ukazom “Add Watch”, jih odstranimo z ukazom “Remove Watch” ali pa uredimo njihov naslov z ukazom “Edit Watch”. Vsebino opazovalnih elementov lahko prikazujemo v enem izmed osnovnih številskih formatov (Decimal, Binary, Hex). Z ukazom “Enter Value” lahko vpišemo novo vrednost v izbrani opazovalni element. Vse ukaze, ki so potrebni za delo z oknom “Watch”, najdemo v menuju, ki ga aktiviramo z desnim klikom miške znotraj okna.

Ker so opazovalni elementi izbrane pomnilniške lokacije, uporabljamo ukaze BIOS-a za branje njihovih vrednosti (ukaz “Input”) in za zapis novih vrednosti (ukaz “Output”).

5.2.5.5 Okno “Memory”

Okno “Memory” predstavlja posebno okno, ki je namenjeno za pregled in urejanje vsebine vseh pomnilniških lokacij iz celotnega naslovnega prostora mikrokrmilniškega sistema. Izgled okna z aktiviranim menujem je prikazan na sliki 5.14.



Slika 5.14: Okno “Memory”

Okno je sestavljeno iz vrstic, ki vsebujejo 16-bitni heksadecimalni naslov, ki mu sledi prikaz vsebine pripadajočih šestnajstih zaporednih lokacij, najprej kot heksadecimalna števila, nato pa še kot ASCII znaki. Začetek izpisa določimo s pomočjo ukaza “Display Address” ali pa z uporabo miške (Scroll Bar) oz. tipkovnice. Nove vrednosti v pomnilniške lokacije zapisujemo z uporabo ukaza “Enter Value”. Ukazi za delo z oknom “Memory” so dostopni z desnim klikom miške znotraj okna.

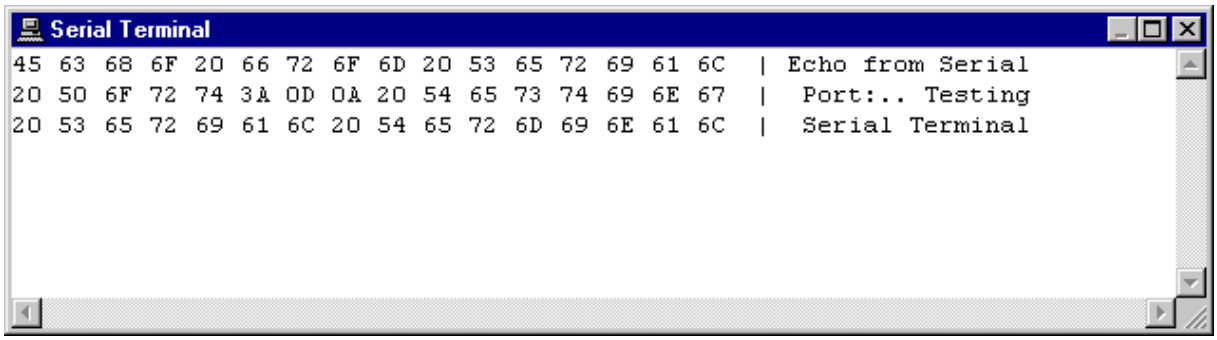
Kakor pri vseh oknih, ki omogočajo bralno / pisalni dostop do pomnilniških lokacij, je tudi pri oknu “Memory” pregled in urejanje vsebine pomnilniških lokacij izveden s pomočjo ukazov “Input” in “Output”, ki jih nudi BIOS.

5.2.6 Serijski terminal

Serijski terminal je okno, ki omogoča uporabo osebnega računalnika kot običajnega serijskega terminala. Ko se izvaja uporabniški program in je aktivno okno terminala, se vsi znaki, ki so vneseni preko tipkovnice, pošljejo preko serijskega vmesnika mikrokrmilniškemu sistemu (brez izpisa na zaslonu terminala - without Echo). Istočasno pa se vsi sprejeti znaki preko serijskega vmesnika iz mikrokrmilniškega sistema izpišejo v oknu serijskega terminala.

Serijski terminal običajno izpisuje sprejete ASCII znake (primer prikazuje slika 5.15a). Lahko pa izpisuje tudi heksadecimalne vrednosti sprejetih znakov v kombinaciji s pripadajočimi ASCII znaki (primer prikazuje slika 5.15b).

Slika 5.15a: Serijski terminal (ASCII)



Slika 5.15b: Serijski terminal (Hex & ASCII)

Serijski terminal lahko uporabljamo le, če se izvaja uporabniški program. Ko je ta zaustavljen in ima nadzor nad izvajanjem program BIOS-a, serijski terminal ni aktiven. V tem primeru je serijska komunikacija rezervirana za ukaze, ki so vgrajeni v BIOS-u.

5.3 Instalacija

Za namestitev programskega razvojnega orodja na osebni računalnik z Windows 95/98/NT operacijskim sistemom oz. instalacijo je izdelan posebni namestitveni program (Setup). Ta prekopira in razpakira vse potrebne datoteke za delovanje celotnega programskega paketa. Tudi namestitveni program je bil izdelan s programskim orodjem “Visual Basic 5.0”.

6. Zaključek

Izdelano razvojno mikrokrmilniško okolje za M68HC11 predstavlja zelo obsežno magistrsko nalogo. Ta vsebuje načrtovanje in izdelavo strojne opreme za večnamenski mikrokrmilniški sistem ter izdelavo zmogljive programske opreme, ki predstavlja celotno razvojno podporo. Izpolnjene in celo presežene so bile vse postavljene zahteve, ki so navedene v uradnem opisu teme magistrske naloge.

V celoti predstavlja izdelana naloga integrirano razvojno okolje za mikrokrmilnik M68HC11, ki je zaradi nizke cene in relativno enostavne izdelave dostopno širokemu krogu uporabnikov. Še posebej je primerno za uporabo v pedagoške namene oz. za laboratorijske vaje, ker nam omogoča koračno izvajanje programov in hkrati analiziranje posameznih izvršenih ukazov. Študentom omogoča, da si z minimalnimi stroški zgradijo lasten razvojni sistem in tako nadgrajujejo pridobljena znanja tudi po končanih laboratorijskih vajah.

Izdelava sistema ne zahteva posebnega znanja, saj je potrebno izdelati le tiskano ploščico in na njo prispajkati ustrezne elemente. Za izdelavo tiskane ploščice je na razpolago načrt v programskem paketu “Tango” (PCB datoteka). Stroški izdelave mikrokrmilniškega sistema so majhni, saj vsebujejo le stroške za izdelavo tiskane ploščice in stroške za nabavo elementov.

Izdelano razvojno mikrokrmilniško okolje predstavlja osnovo, na kateri bodo lahko narejene različne aplikacije, kar bo še dodatno povečalo uporabnost in priljubljenost. Zato uporaba ne bo omejena le na laboratorijske vaje, ampak je načrtovana tudi postavitev domače strani na internetu. Tam bo predstavljeno to razvojno mikrokrmilniško okolje s celotno pripadajočo dokumentacijo in navodili za izdelavo sistema. Ravno tako bo možno dobiti tudi najnovejšo verzijo programskega razvojnega orodja. Načrtovana je tudi ustrezna tehnična podpora, ki bo uporabnikom pomagala pri odpravljanju morebitnih težav.

V izdelavo te magistrske naloge je bilo vloženo ogromno truda in časa ter pridobljenega predhodnega znanja. V veliko pomoč so mi bile tudi dolgoletne izkušnje iz programiranja najrazličnejših mikrokrmilnikov in izdelave raznih mikrokrmilniških sistemov ter zelo dobro poznavanje programiranja s programskim orodjem “Visual Basic”. Vse to mi je omogočilo predvsem dobro zasnovo naloge in seveda ustrezno realizacijo vseh zastavljenih ciljev.

Priloga: Izpis programa BIOS-a

Priloga vsebuje izpis prevedenega zbirniškega programa BIOS-a verzije 1.0 za razvojni mikrokrmilniški sistem. Program vsebuje tudi komentarje v angleškem jeziku.

M68HC11 Development Board Basic Input Output System (BIOS) V1.0 - Listing File

```
0001                    ************************************************************************
0002                    * M68HC11 Development Board Basic Input Output System (BIOS) V1.0      *
0003                    *  Made by: Robert Rostohar, February 1999                             *
0004                    ************************************************************************
0005
0006                    ************************************************************************
0007                    * Definitions                                                          *
0008                    ************************************************************************
0009 0006               ACK      EQU   6           Acknowledge Character
0010 0015               NAK      EQU   21          Not Acknowledge Character
0011 3fff               SPINIT   EQU   $3FFF       Stack Pointer Initial Value (Top of XRAM)
0012
0013                    ************************************************************************
0014                    * HC11 Registers and Control Bits                                       *
0015                    ************************************************************************
0016 102b               BAUD     EQU   $102B       SCI Baud Rate Register
0017 102d               SCCR2    EQU   $102D       SCI Control Register 2
0018 102e               SCSR     EQU   $102E       SCI Status Register
0019 102f               SCDR     EQU   $102F       SCI Data Register
0020
0021                    * SCSR ($102E) SCI Status Register Control Bits
0022 0080               TDRE     EQU   $80         Transmit Data Register Empty Flag
0023 0020               RDRF     EQU   $20         Receive Data Register Full Flag
0024
0025                    ************************************************************************
0026                    * System Reset Vectors                                                  *
0027                    *  System EPROM: $E000..$FFFF at Reset (Active)                         *
0028                    *  System EPROM: $C000..$DFFF after Reading Reset Vector                *
0029                    ************************************************************************
0030 dfd6                        ORG   $DFD6
0031
0032 dfd6 c0 3e                  FDB   I_SCI       SCI Interrupt Vector
0033 dfd8 c0 42                  FDB   I_SPI       SPI Interrupt Vector
0034 dfda c0 46                  FDB   I_PAIE      Pulse Accumulator Input Edge Interrupt Vector
0035 dfdc c0 4a                  FDB   I_PAO       Pulse Accumulator Overflow Interrupt Vector
0036 dfde c0 4e                  FDB   I_TOF       Timer Overflow Interrupt Vector
0037 dfe0 c0 52                  FDB   I_TOC5      Timer Output Compare 5 Interrupt Vector
0038 dfe2 c0 56                  FDB   I_TOC4      Timer Output Compare 4 Interrupt Vector
0039 dfe4 c0 5a                  FDB   I_TOC3      Timer Output Compare 3 Interrupt Vector
0040 dfe6 c0 5e                  FDB   I_TOC2      Timer Output Compare 2 Interrupt Vector
0041 dfe8 c0 62                  FDB   I_TOC1      Timer Output Compare 1 Interrupt Vector
0042 dfea c0 66                  FDB   I_TIC3      Timer Input Capture 3 Interrupt Vector
0043 dfec c0 6a                  FDB   I_TIC2      Timer Input Capture 2 Interrupt Vector
0044 dfee c0 6e                  FDB   I_TIC1      Timer Input Capture 1 Interrupt Vector
0045 dff0 c0 72                  FDB   I_RTI       RTI Interrupt Vector
0046 dff2 c0 76                  FDB   I_IRQ       IRQ Interrupt Vector
0047 dff4 c0 7a                  FDB   I_XIRQ      XIRQ Interrupt Vector
0048 dff6 c0 7e                  FDB   I_SWI       SWI Interrupt Vector
0049 dff8 c0 8b                  FDB   I_ILLOP     Illegal Opcode Trap Interrupt Vector
0050
0051 dffa c0 00                  FDB   COPRES      COP Reset Vector
0052 dffc c0 04                  FDB   CLKMRES     Clock Monitor Reset Vector
0053 dffe c0 08                  FDB   RESET       Reset Vector
0054
```

```
                    ************************************************************************
                    * Reset and Interrupt Routines                                         *
                    ************************************************************************
c000                         ORG   $C000       System EPROM Base Address

c000 86 02          COPRES   LDAA  #2          COP Failure Reset Entry Point (ID=2)
c002 20 06                   BRA   INIT
c004 86 01          CLKMRES  LDAA  #1          Clock Monitor Fail Reset Entry Point (ID=1)
c006 20 02                   BRA   INIT
c008 86 00          RESET    LDAA  #0          Reset Entry Point (ID=0)
c00a 8e 3f ff       INIT     LDS   #SPINIT     Initialise SP
c00d c6 01                   LDAB  #$01
c00f f7 10 2b                STAB  BAUD        Set SCI Baud Rate = 38400 (XTAL = 4.9152 MHz)
c012 c6 0c                   LDAB  #%00001100
c014 f7 10 2d                STAB  SCCR2       Enable SCI Receive and Transmit
c017 ce c0 7a                LDX   #I_XIRQ
c01a ff ff f4                STX   _XIRQ       Set XIRQ Interrupt Vector
c01d ce c0 7e                LDX   #I_SWI
c020 ff ff f6                STX   _SWI        Set SWI Interrupt Vector
c023 ce c0 8b                LDX   #I_ILLOP
c026 ff ff f8                STX   _ILLOP      Set Illegal Opcode Trap Interrupt Vector
c029 ce ff fe                LDX   #_RESET
c02c 16                      TAB
c02d 50                      NEGB
c02e 3a                      ABX
c02f 3a                      ABX
c030 ee 00                   LDX   0,X
c032 3c                      PSHX              "PC" = Reset Vector
c033 ce 00 00                LDX   #0
c036 3c                      PSHX              "IY" = $0000
c037 3c                      PSHX              "IX" = $0000
c038 3c                      PSHX              "ACCA" = $00, "ACCB" = $00
c039 c6 90                   LDAB  #%10010000
c03b 37                      PSHB              "CCR": S=1,X=0,H=0,I=1,N=0,Z=0,V=0,C=0
c03c 20 4f                   BRA   WBOUT       Jump to WBOUT with ID in AccA

c03e 86 14          I_SCI    LDAA  #20         SCI Entry Point (ID=20)
c040 20 4b                   BRA   WBOUT
c042 86 13          I_SPI    LDAA  #19         SPI Entry Point (ID=19)
c044 20 47                   BRA   WBOUT
c046 86 12          I_PAIE   LDAA  #18         Pulse Accum. Input Edge Entry Point (ID=18)
c048 20 43                   BRA   WBOUT
c04a 86 11          I_PAO    LDAA  #17         Pulse Accum. Overflow Entry Point (ID=17)
c04c 20 3f                   BRA   WBOUT
c04e 86 10          I_TOF    LDAA  #16         Timer Overflow Entry Point (ID=16)
c050 20 3b                   BRA   WBOUT
c052 86 0f          I_TOC5   LDAA  #15         Timer Output Compare 5 Entry Point (ID=15)
c054 20 37                   BRA   WBOUT
c056 86 0e          I_TOC4   LDAA  #14         Timer Output Compare 4 Entry Point (ID=14)
c058 20 33                   BRA   WBOUT
c05a 86 0d          I_TOC3   LDAA  #13         Timer Output Compare 3 Entry Point (ID=13)
c05c 20 2f                   BRA   WBOUT
c05e 86 0c          I_TOC2   LDAA  #12         Timer Output Compare 2 Entry Point (ID=12)
c060 20 2b                   BRA   WBOUT
c062 86 0b          I_TOC1   LDAA  #11         Timer Output Compare 1 Entry Point (ID=11)
c064 20 27                   BRA   WBOUT
c066 86 0a          I_TIC3   LDAA  #10         Timer Input Capture 3 Entry Point (ID=10)
c068 20 23                   BRA   WBOUT
c06a 86 09          I_TIC2   LDAA  #9          Timer Input Capture 2 Entry Point (ID=9)
c06c 20 1f                   BRA   WBOUT
c06e 86 08          I_TIC1   LDAA  #8          Timer Input Capture 1 Entry Point (ID=8)
c070 20 1b                   BRA   WBOUT
c072 86 07          I_RTI    LDAA  #7          RTI Entry Point (ID=7)
c074 20 17                   BRA   WBOUT
c076 86 06          I_IRQ    LDAA  #6          IRQ Entry Point (ID=6)
c078 20 13                   BRA   WBOUT
c07a 86 05          I_XIRQ   LDAA  #5          XIRQ Entry Point (ID=5)
c07c 20 0f                   BRA   WBOUT
c07e 86 04          I_SWI    LDAA  #4          SWI Entry Point (ID=4)
c080 18 30                   TSY               Transfer SP+1 to IY
c082 cd ee 07                LDX   7,Y         Read Return Address for SWI
c085 09                      DEX               Set Return Address to Point at SWI Opcode
c086 cd ef 07                STX   7,Y         Store Return Address for SWI
c089 20 02                   BRA   WBOUT
c08b 86 03          I_ILLOP  LDAA  #3          Illegal Opcode Trap Entry Point (ID=3)

```

```
0131 c08d ce 00 03    WBOUT    LDX   #3          Transmit 'B' and Break ID (3 times)
0132 c090 f6 10 2e    WBOUTB   LDAB  SCSR        Wait for Empty Data Transmit Register
0133 c093 c4 80                ANDB  #TDRE       Test TDRE
0134 c095 27 f9                BEQ   WBOUTB
0135 c097 c6 42                LDAB  #'B'
0136 c099 f7 10 2f             STAB  SCDR        Transmit 'B' Character
0137 c09c f6 10 2e    WBOUTID  LDAB  SCSR        Wait for Empty Data Transmit Register
0138 c09f c4 80                ANDB  #TDRE       Test TDRE
0139 c0a1 27 f9                BEQ   WBOUTID
0140 c0a3 b7 10 2f             STAA  SCDR        Transmit Break ID Byte
0141 c0a6 09                   DEX
0142 c0a7 26 e7                BNE   WBOUTB      Repeat 3 times
0143                  *        BRA   LOOP        Enter Main Loop
0144
0145                  ***************************************************************************
0146                  * LOOP - Main Loop                                                        *
0147                  *  Waiting for PC Commands: OUTPUT, INPUT, STACK or EXECUTE               *
0148                  *  HC11 Interrupted by: Reset, SWI, Non-Maskable or Maskable Interrupt *
0149                  ***************************************************************************
0150 c0a9 b6 10 2e    LOOP     LDAA  SCSR        Wait for Command Byte
0151 c0ac 84 20                ANDA  #RDRF       Test RDRF
0152 c0ae 27 f9                BEQ   LOOP
0153 c0b0 b6 10 2f             LDAA  SCDR        Read Command Byte
0154 c0b3 81 4f       NEXT1    CMPA  #'O'
0155 c0b5 26 03                BNE   NEXT2
0156 c0b7 7e c0 cf             JMP   OUTPUT      Go to OUTPUT Command
0157 c0ba 81 49       NEXT2    CMPA  #'I'
0158 c0bc 26 03                BNE   NEXT3
0159 c0be 7e c1 5c             JMP   INPUT       Go to INPUT Command
0160 c0c1 81 53       NEXT3    CMPA  #'S'
0161 c0c3 26 03                BNE   NEXT4
0162 c0c5 7e c1 f1             JMP   STACK       Go to STACK Command
0163 c0c8 81 58       NEXT4    CMPA  #'X'
0164 c0ca 26 dd                BNE   LOOP
0165 c0cc 7e c2 29             JMP   EXECUTE     Go to EXECUTE Command
0166
0167                  ***************************************************************************
0168                  * OUTPUT - PC Writes Data to HC11                                         *
0169                  *  PC:   'O', Address High, Address Low, Lenght, Data, CheckSum           *
0170                  *  HC11: ACK or NAK                                                        *
0171                  ***************************************************************************
0172 c0cf ce 00 00    OUTPUT   LDX   #0          Initialise CheckSum Byte
0173 c0d2 c6 4b                LDAB  #75         Initialise TimeOut Counter
0174 c0d4 b6 10 2e    WAITOAH  LDAA  SCSR        Wait for Address High Byte
0175 c0d7 84 20                ANDA  #RDRF       Test RDRF
0176 c0d9 26 07                BNE   NEXTOAH     Jump if Byte Received
0177 c0db 01                   NOP
0178 c0dc 5a                   DECB
0179 c0dd 26 f5                BNE   WAITOAH
0180 c0df 7e c0 a9             JMP   LOOP        No Character (TimeOut = 1200 Cycles)
0181 c0e2 f6 10 2f    NEXTOAH  LDAB  SCDR        Read Address High Byte
0182 c0e5 3a                   ABX               Update CheckSum Byte
0183 c0e6 17                   TBA
0184 c0e7 5f                   CLRB
0185 c0e8 18 8f                XGDY              Store Address High Byte
0186 c0ea c6 4b                LDAB  #75         Initialise TimeOut Counter
0187 c0ec b6 10 2e    WAITOAL  LDAA  SCSR        Wait for Address Low Byte
0188 c0ef 84 20                ANDA  #RDRF       Test RDRF
0189 c0f1 26 07                BNE   NEXTOAL     Jump if Byte Received
0190 c0f3 01                   NOP
0191 c0f4 5a                   DECB
0192 c0f5 26 f5                BNE   WAITOAL
0193 c0f7 7e c0 a9             JMP   LOOP        No Character (TimeOut = 1200 Cycles)
0194 c0fa f6 10 2f    NEXTOAL  LDAB  SCDR        Read Address Low Byte
0195 c0fd 3a                   ABX               Update CheckSum Byte
0196 c0fe 18 3a                ABY               Store Address Low Byte
0197 c100 c6 4b                LDAB  #75         Initialise TimeOut Counter
0198 c102 b6 10 2e    WAITOL   LDAA  SCSR        Wait for Lenght Byte
0199 c105 84 20                ANDA  #RDRF       Test RDRF
0200 c107 26 07                BNE   NEXTOL      Jump if Byte Received
0201 c109 01                   NOP
0202 c10a 5a                   DECB
0203 c10b 26 f5                BNE   WAITOL
0204 c10d 7e c0 a9             JMP   LOOP        No Character (TimeOut = 1200 Cycles)
0205 c110 f6 10 2f    NEXTOL   LDAB  SCDR        Read Lenght Byte
0206 c113 3a                   ABX               Update CheckSum Byte
0207 c114 17                   TBA               Store Lenght Byte
```

```
0208 c115 c6 3c       OUTDATA  LDAB  #60         Initialise TimeOut Counter
0209 c117 8f          WAITOD   XGDX              Wait for Data Byte
0210 c118 b6 10 2e             LDAA  SCSR
0211 c11b 84 20                ANDA  #RDRF       Test RDRF
0212 c11d 26 07                BNE   NEXTOD      Jump if Byte Received
0213 c11f 8f                   XGDX
0214 c120 5a                   DECB
0215 c121 26 f4                BNE   WAITOD
0216 c123 7e c0 a9             JMP   LOOP        No Character (TimeOut = 1200 Cycles)
0217 c126 8f          NEXTOD   XGDX
0218 c127 f6 10 2f             LDAB  SCDR        Read Data Byte
0219 c12a 3a                   ABX               Update CheckSum Byte
0220 c12b 18 e7 00             STAB  0,Y         Store Data Byte
0221 c12e 18 08                INY               Increment Address
0222 c130 4a                   DECA
0223 c131 26 e2                BNE   OUTDATA     End if All Data Bytes Received
0224 c133 c6 4b                LDAB  #75         Initialise TimeOut Counter
0225 c135 b6 10 2e    WAITOC   LDAA  SCSR        Wait for CheckSum Byte
0226 c138 84 20                ANDA  #RDRF       Test RDRF
0227 c13a 26 07                BNE   NEXTOC
0228 c13c 01                   NOP
0229 c13d 5a                   DECB
0230 c13e 26 f5                BNE   WAITOC
0231 c140 7e c0 a9             JMP   LOOP        No Character (TimeOut = 1200 Cycles)
0232 c143 f6 10 2f    NEXTOC   LDAB  SCDR        Read CheckSum Byte
0233 c146 3a                   ABX               Update CheckSum Byte
0234 c147 8f                   XGDX
0235 c148 86 06                LDAA  #ACK        Acknowledged
0236 c14a 5d                   TSTB
0237 c14b 27 02                BEQ   WAITO       Jump if CheskSum Byte is OK
0238 c14d 86 15       OUTNAK   LDAA  #NAK        Not Acknowledged
0239 c14f f6 10 2e    WAITO    LDAB  SCSR        Wait for Empty Data Transmit Register
0240 c152 c4 80                ANDB  #TDRE       Test TDRE
0241 c154 27 f9                BEQ   WAITO
0242 c156 b7 10 2f             STAA  SCDR
0243 c159 7e c0 a9             JMP   LOOP        Jump to Main Loop
0244
0245                  ***********************************************************************
0246                  * INPUT - PC Reads Data from HC11                                     *
0247                  *  PC:   'I', Address High, Address Low, Lenght, CheckSum              *
0248                  *  HC11: 'I', Data, CheckSum  or  NAK                                  *
0249                  ***********************************************************************
0250 c15c ce 00 00    INPUT    LDX   #0          Initialise CheckSum Byte
0251 c15f c6 4b                LDAB  #75         Initialise TimeOut Counter
0252 c161 b6 10 2e    WAITIAH  LDAA  SCSR        Wait for Address High Byte
0253 c164 84 20                ANDA  #RDRF       Test RDRF
0254 c166 26 07                BNE   NEXTIAH     Jump if Byte Received
0255 c168 01                   NOP
0256 c169 5a                   DECB
0257 c16a 26 f5                BNE   WAITIAH
0258 c16c 7e c0 a9             JMP   LOOP        No Character (TimeOut = 1200 Cycles)
0259 c16f f6 10 2f    NEXTIAH  LDAB  SCDR        Read Address High Byte
0260 c172 3a                   ABX               Update CheckSum Byte
0261 c173 17                   TBA
0262 c174 5f                   CLRB
0263 c175 18 8f                XGDY              Store Address High Byte
0264 c177 c6 4b                LDAB  #75         Initialise TimeOut Counter
0265 c179 b6 10 2e    WAITIAL  LDAA  SCSR        Wait for Address Low Byte
0266 c17c 84 20                ANDA  #RDRF       Test RDRF
0267 c17e 26 07                BNE   NEXTIAL     Jump if Byte Received
0268 c180 01                   NOP
0269 c181 5a                   DECB
0270 c182 26 f5                BNE   WAITIAL
0271 c184 7e c0 a9             JMP   LOOP        No Character (TimeOut = 1200 Cycles)
0272 c187 f6 10 2f    NEXTIAL  LDAB  SCDR        Read Address Low Byte
0273 c18a 3a                   ABX               Update CheckSum Byte
0274 c18b 18 3a                ABY               Store Address Low Byte
0275 c18d c6 4b                LDAB  #75         Initialise TimeOut Counter
0276 c18f b6 10 2e    WAITIL   LDAA  SCSR        Wait for Lenght Byte
0277 c192 84 20                ANDA  #RDRF       Test RDRF
0278 c194 26 07                BNE   NEXTIL      Jump if Byte Received
0279 c196 01                   NOP
0280 c197 5a                   DECB
0281 c198 26 f5                BNE   WAITIL
0282 c19a 7e c0 a9             JMP   LOOP        No Character (TimeOut = 1200 Cycles)
```

```
0283 c19d f6 10 2f    NEXTIL   LDAB  SCDR        Read Lenght Byte
0284 c1a0 3a                   ABX               Update CheckSum Byte
0285 c1a1 17                   TBA               Store Lenght Byte
0286 c1a2 c6 3c                LDAB  #60         Initialise TimeOut Counter
0287 c1a4 8f          WAITIC   XGDX              Wait for CheckSum Byte
0288 c1a5 b6 10 2e             LDAA  SCSR
0289 c1a8 84 20                ANDA  #RDRF       Test RDRF
0290 c1aa 26 07                BNE   NEXTIC
0291 c1ac 8f                   XGDX
0292 c1ad 5a                   DECB
0293 c1ae 26 f4                BNE   WAITIC
0294 c1b0 7e c0 a9             JMP   LOOP        No Character (TimeOut = 1200 Cycles)
0295 c1b3 8f          NEXTIC   XGDX
0296 c1b4 f6 10 2f             LDAB  SCDR        Read CheckSum Byte
0297 c1b7 3a                   ABX               Update CheckSum Byte
0298 c1b8 8f                   XGDX
0299 c1b9 5d                   TSTB
0300 c1ba 27 03                BEQ   NEXTI       Jump if CheskSum Byte is OK
0301 c1bc 7e c1 4d             JMP   OUTNAK      Not Acknowledged
0302 c1bf 8f          NEXTI    XGDX
0303 c1c0 ce 00 00             LDX   #0          Initialise CheckSum Byte
0304 c1c3 f6 10 2e    WIOUTI   LDAB  SCSR        Wait for Empty Data Transmit Register
0305 c1c6 c4 80                ANDB  #TDRE       Test TDRE
0306 c1c8 27 f9                BEQ   WIOUTI
0307 c1ca c6 49                LDAB  #'I'
0308 c1cc f7 10 2f             STAB  SCDR        Transmit 'I' Character
0309 c1cf f6 10 2e    WIOUTD   LDAB  SCSR        Wait for Empty Data Transmit Register
0310 c1d2 c4 80                ANDB  #TDRE       Test TDRE
0311 c1d4 27 f9                BEQ   WIOUTD
0312 c1d6 18 e6 00             LDAB  0,Y
0313 c1d9 f7 10 2f             STAB  SCDR        Transmit Data Byte
0314 c1dc 3a                   ABX               Update CheckSum Byte
0315 c1dd 18 08                INY               Increment Address
0316 c1df 4a                   DECA
0317 c1e0 26 ed                BNE   WIOUTD      End if All Data Bytes Sent
0318 c1e2 f6 10 2e    WIOUTC   LDAB  SCSR        Wait for Empty Data Transmit Register
0319 c1e5 c4 80                ANDB  #TDRE       Test TDRE
0320 c1e7 27 f9                BEQ   WIOUTC
0321 c1e9 8f                   XGDX
0322 c1ea 50                   NEGB
0323 c1eb f7 10 2f             STAB  SCDR        Transmit CheckSum Byte
0324 c1ee 7e c0 a9             JMP   LOOP        Jump to Main Loop
0325
0326                  ************************************************************************
0327                  * STACK - PC Reads Stack Pointer from HC11                               *
0328                  *  PC:   'S'                                                             *
0329                  *  HC11: 'S', SPH, SPL, CheckSum                                         *
0330                  ************************************************************************
0331 c1f1 30          STACK    TSX               Transfer SP+1 to IX
0332 c1f2 8f                   XGDX
0333 c1f3 c3 00 08             ADDD  #8
0334 c1f6 8f                   XGDX              SP+9 in IX
0335 c1f7 f6 10 2e    WSOUTS   LDAB  SCSR        Wait for Empty Data Transmit Register
0336 c1fa c4 80                ANDB  #TDRE       Test TDRE
0337 c1fc 27 f9                BEQ   WSOUTS
0338 c1fe c6 53                LDAB  #'S'
0339 c200 f7 10 2f             STAB  SCDR        Transmit 'S' Character
0340 c203 f6 10 2e    WSOUTSH  LDAB  SCSR        Wait for Empty Data Transmit Register
0341 c206 c4 80                ANDB  #TDRE       Test TDRE
0342 c208 27 f9                BEQ   WSOUTSH
0343 c20a 8f                   XGDX
0344 c20b b7 10 2f             STAA  SCDR        Transmit SP High Byte
0345 c20e 8f                   XGDX
0346 c20f f6 10 2e    WSOUTSL  LDAB  SCSR        Wait for Empty Data Transmit Register
0347 c212 c4 80                ANDB  #TDRE       Test TDRE
0348 c214 27 f9                BEQ   WSOUTSL
0349 c216 8f                   XGDX
0350 c217 f7 10 2f             STAB  SCDR        Transmit SP Low Byte
0351 c21a 1b                   ABA               Calculate CheckSum
0352 c21b 40                   NEGA
0353 c21c f6 10 2e    WSOUTC   LDAB  SCSR        Wait for Empty Data Transmit Register
0354 c21f c4 80                ANDB  #TDRE       Test TDRE
0355 c221 27 f9                BEQ   WSOUTC
0356 c223 b7 10 2f             STAA  SCDR        Transmit CheckSum Byte
0357 c226 7e c0 a9             JMP   LOOP        Jump to Main Loop
0358
```

```
0359                    ************************************************************************
0360                    * EXECUTE - Start (Continue) Executing Program in HC11                 *
0361                    *  PC:   'X'                                                           *
0362                    *  HC11: ACK                                                           *
0363                    ************************************************************************
0364 c229 f6 10 2e      EXECUTE  LDAB  SCSR        Wait for Empty Data Transmit Register
0365 c22c c4 80                  ANDB  #TDRE       Test TDRE
0366 c22e 27 f9                  BEQ   EXECUTE
0367 c230 c6 06                  LDAB  #ACK
0368 c232 f7 10 2f               STAB  SCDR        Transmit Acknowledge Character
0369 c235 f6 10 2e      WAITEXE  LDAB  SCSR        Wait for Empty Data Transmit Register
0370 c238 c4 80                  ANDB  #TDRE       Test TDRE
0371 c23a 27 f9                  BEQ   WAITEXE
0372 c23c 3b                     RTI               Continue Executing with Preset CPU Registers
0373
0374                    ************************************************************************
0375                    * HC11 Interrupt Vectors                                               *
0376                    ************************************************************************
0377 ffd6               _SCI     EQU   $FFD6       Serial Communication Interface (SCI)
0378 ffd8               _SPI     EQU   $FFD8       Serial Peripherial Interface (SPI)
0379 ffda               _PAIE    EQU   $FFDA       Pulse Accumulator Input Edge
0380 ffdc               _PAO     EQU   $FFDC       Pulse Accumulator Overflow
0381 ffde               _TOF     EQU   $FFDE       Timer Overflow
0382 ffe0               _TOC5    EQU   $FFE0       Timer Output Compare 5
0383 ffe2               _TOC4    EQU   $FFE2       Timer Output Compare 4
0384 ffe4               _TOC3    EQU   $FFE4       Timer Output Compare 3
0385 ffe6               _TOC2    EQU   $FFE6       Timer Output Compare 2
0386 ffe8               _TOC1    EQU   $FFE8       Timer Output Compare 1
0387 ffea               _TIC3    EQU   $FFEA       Timer Input Capture 3
0388 ffec               _TIC2    EQU   $FFEC       Timer Input Capture 2
0389 ffee               _TIC1    EQU   $FFEE       Timer Input Capture 1
0390 fff0               _RTI     EQU   $FFF0       Real Time Interrupt (RTI)
0391 fff2               _IRQ     EQU   $FFF2       IRQ Pin
0392 fff4               _XIRQ    EQU   $FFF4       XIRQ Pin (Pseudo Non-Maskable Interrupt)
0393 fff6               _SWI     EQU   $FFF6       Software Interrupt (SWI)
0394 fff8               _ILLOP   EQU   $FFF8       Illegal Opcode Trap
0395 fffa               _COP     EQU   $FFFA       COP Failure (Reset)
0396 fffc               _CLKM    EQU   $FFFC       Clock Monitor Fail (Reset)
0397 fffe               _RESET   EQU   $FFFE       Reset
0398
0399                    ************************************************************************
0400                    * Signature                                                           *
0401                    ************************************************************************
0402 d000                        ORG   $D000
0403 d000 4d 36 38 48            FCC   "M68HC11 Development Board"
          43 31 31 20
          44 65 76 65
          6c 6f 70 6d
          65 6e 74 20
          42 6f 61 72
          64
0404 d019 0d 0a                  FCB   $0D,$0A
0405 d01b 42 61 73 69            FCC   "Basic Input Output System (BIOS) V1.0"
          63 20 49 6e
          70 75 74 20
          4f 75 74 70
          75 74 20 53
          79 73 74 65
          6d 20 28 42
          49 4f 53 29
          20 56 31 2e
          30
0406 d040 0d 0a                  FCB   $0D,$0A
0407 d042 4d 61 64 65            FCC   "Made by: Robert Rostohar, February 1999"
          20 62 79 3a
          20 52 6f 62
          65 72 74 20
          52 6f 73 74
          6f 68 61 72
          2c 20 46 65
          62 72 75 61
          72 79 20 31
          39 39 39
0408 d069 0d 0a 00               FCB   $0D,$0A,$00
0409                    ************************************************************************
0410
0411                             END
```

Seznam uporabljenih virov

1. Motorola, HC11 - M68HC11 Reference Manual, 1996
2. Motorola, M68HC11A8 - HCMOS Single-Chip Microcontroller, 1996
3. Hitachi, Handbook HD6821 Data Sheet (PIA), 1980
4. Maxim, CD Data Catalog, November 1998
5. Fujitsu, CD Semiconductor Data Book, June 1998
6. Atmel, CD Nonvolatile Memory Data Book, November 1996
7. Siemens, CD Technical Product Information for Semiconductors, November 1998

Izjava

Izjavljam, da sem magistrsko delo samostojno izdelal pod vodstvom mentorja prof. dr. Franca Bratkoviča. Izkazano pomoč drugih sodelavcev sem v celoti navedel v zahvali.

Robert Rostohar