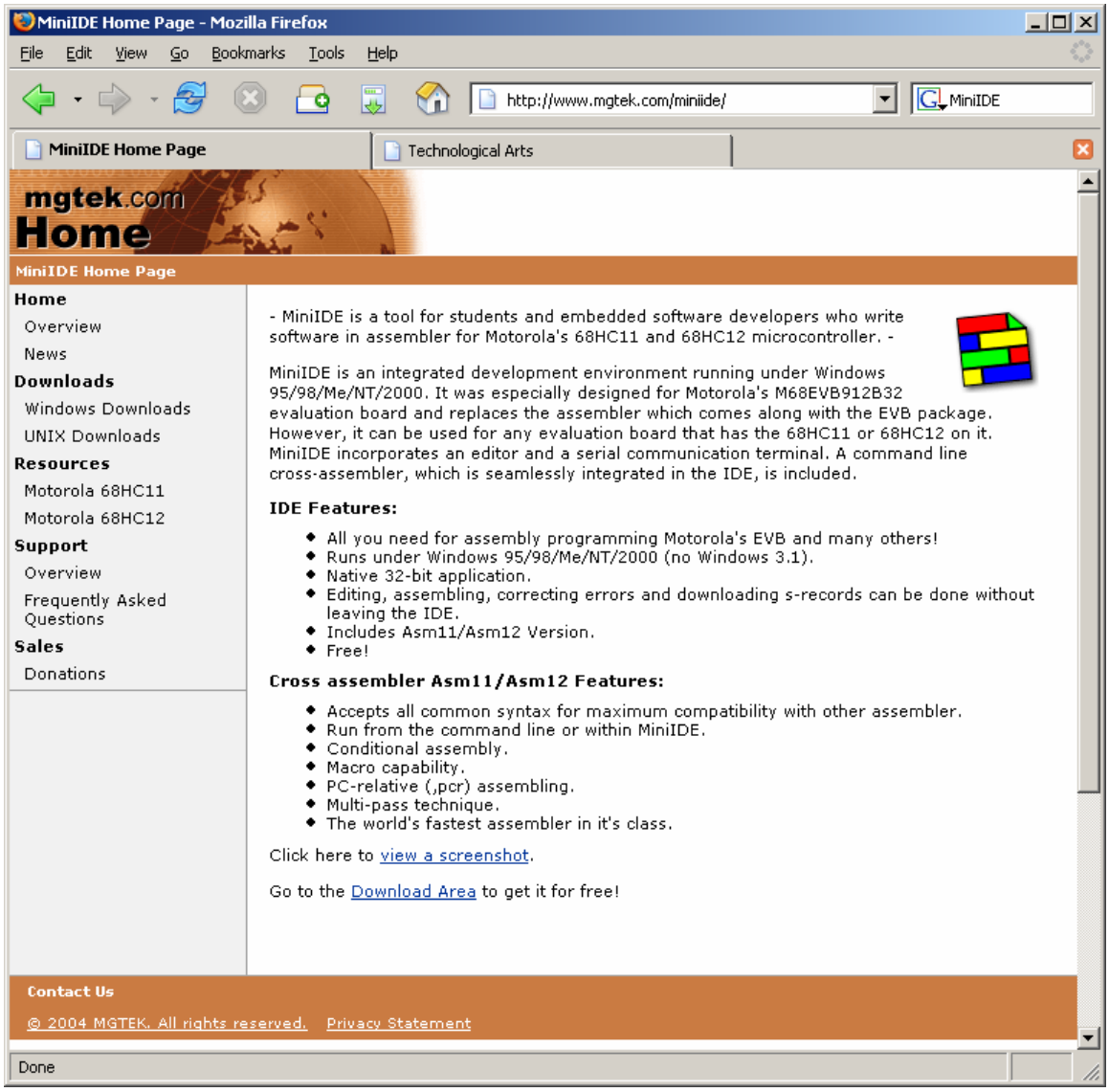
How to use MiniIDE with Adapt11 and MicroLoad

Download MiniIDE from http://www.mgtek.com/miniide/

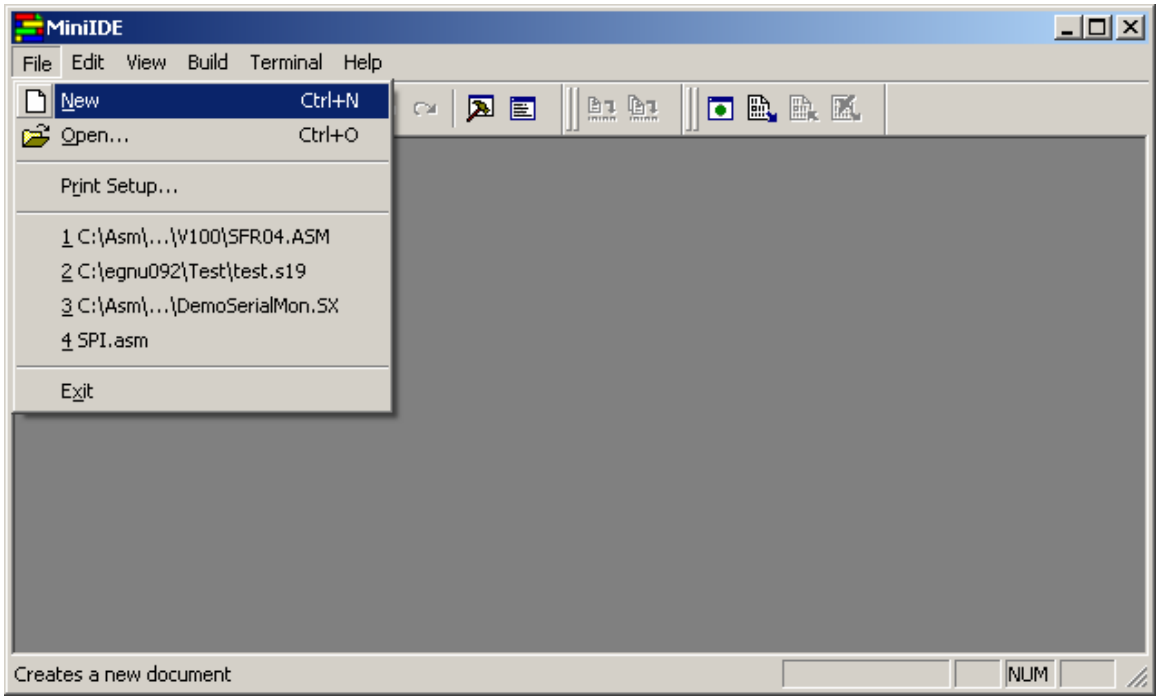


Click on the Download area to select your Operating System (OS) and download MiniIDE.

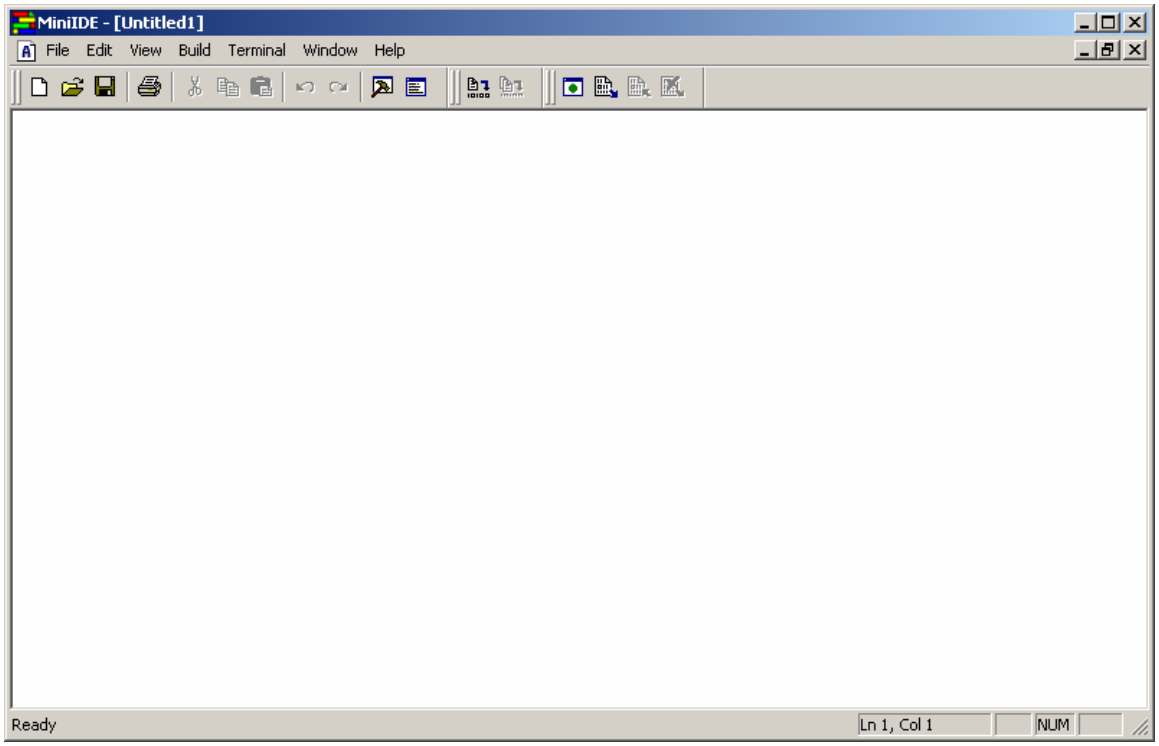
This document assumes that MiniIDE has been installed in your computer.

Getting Started:

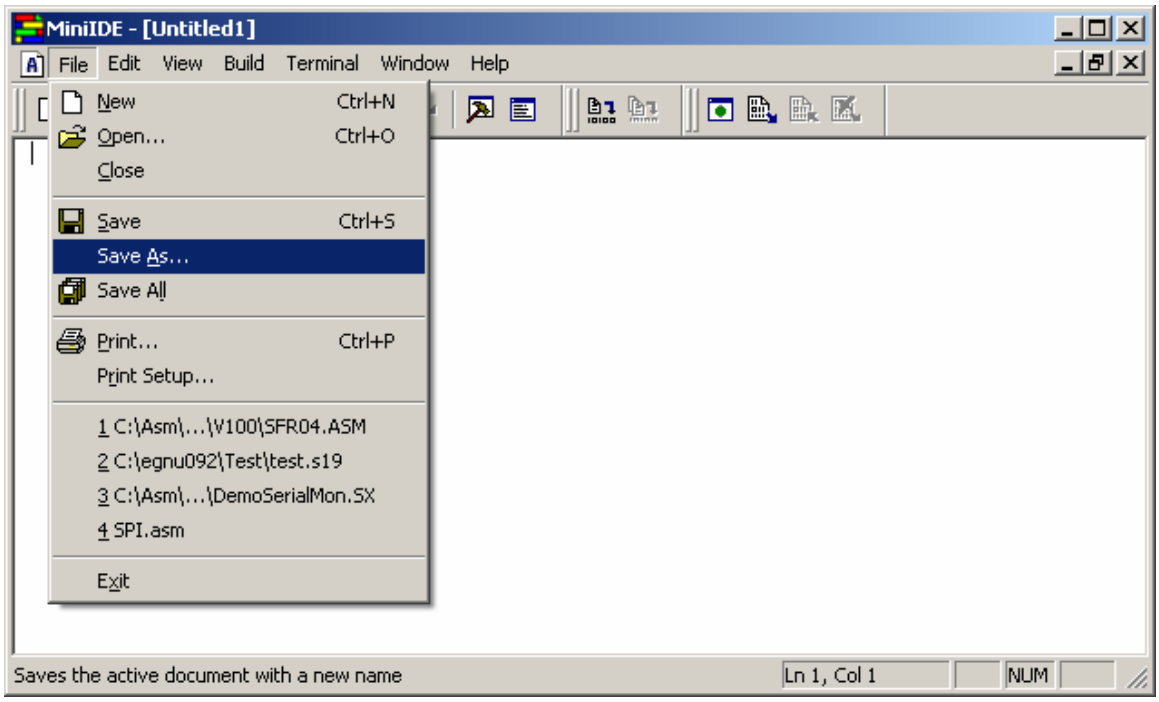
To create a new document click Menu – File - New

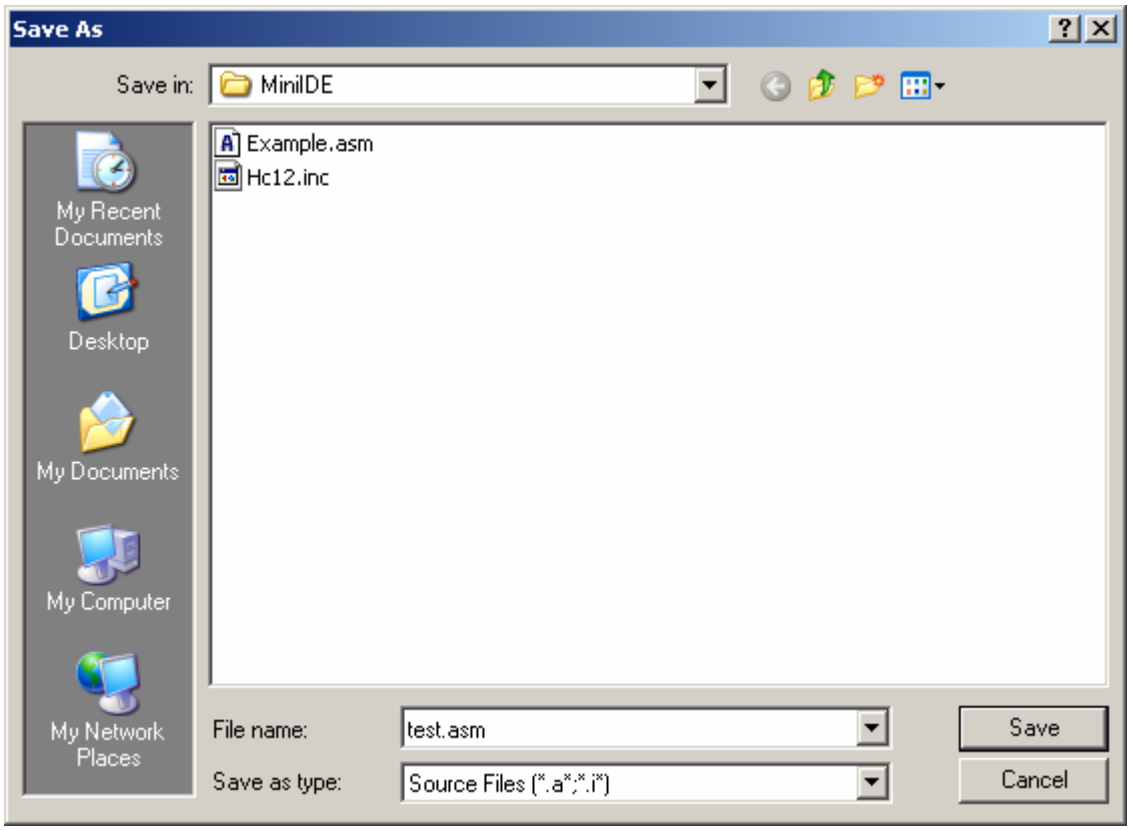


A new ***untitled*** file is created.

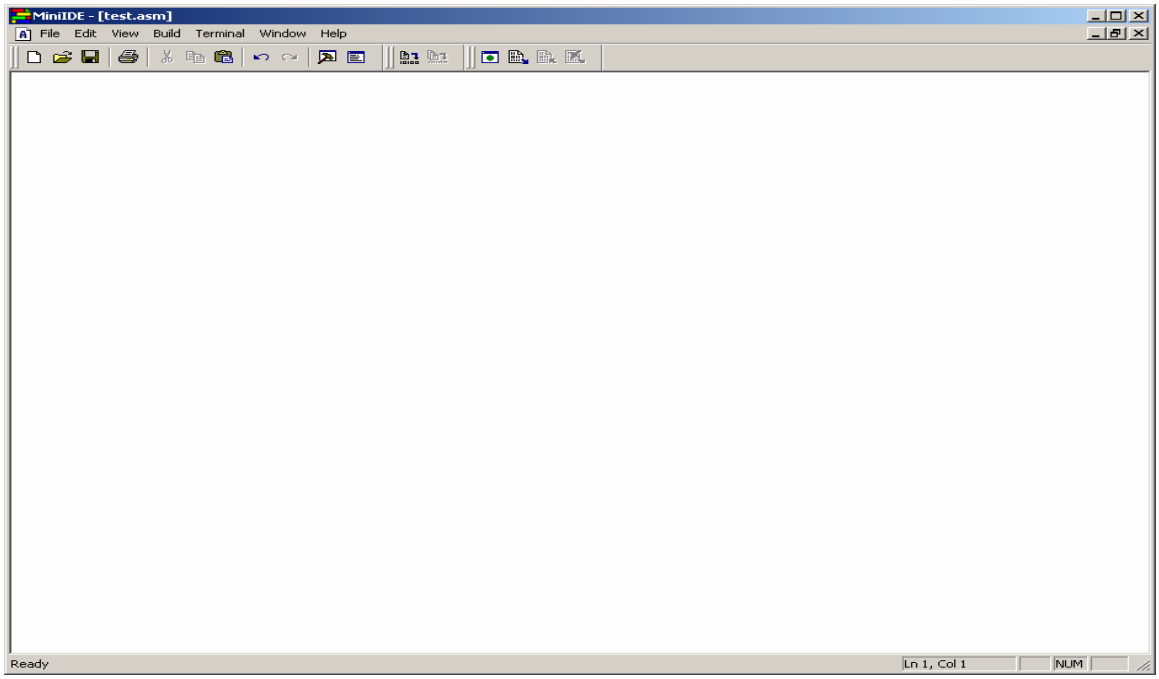


In this example the LED connected to PORTA bit 6 is toggled. The file is **Save As *test.asm.***

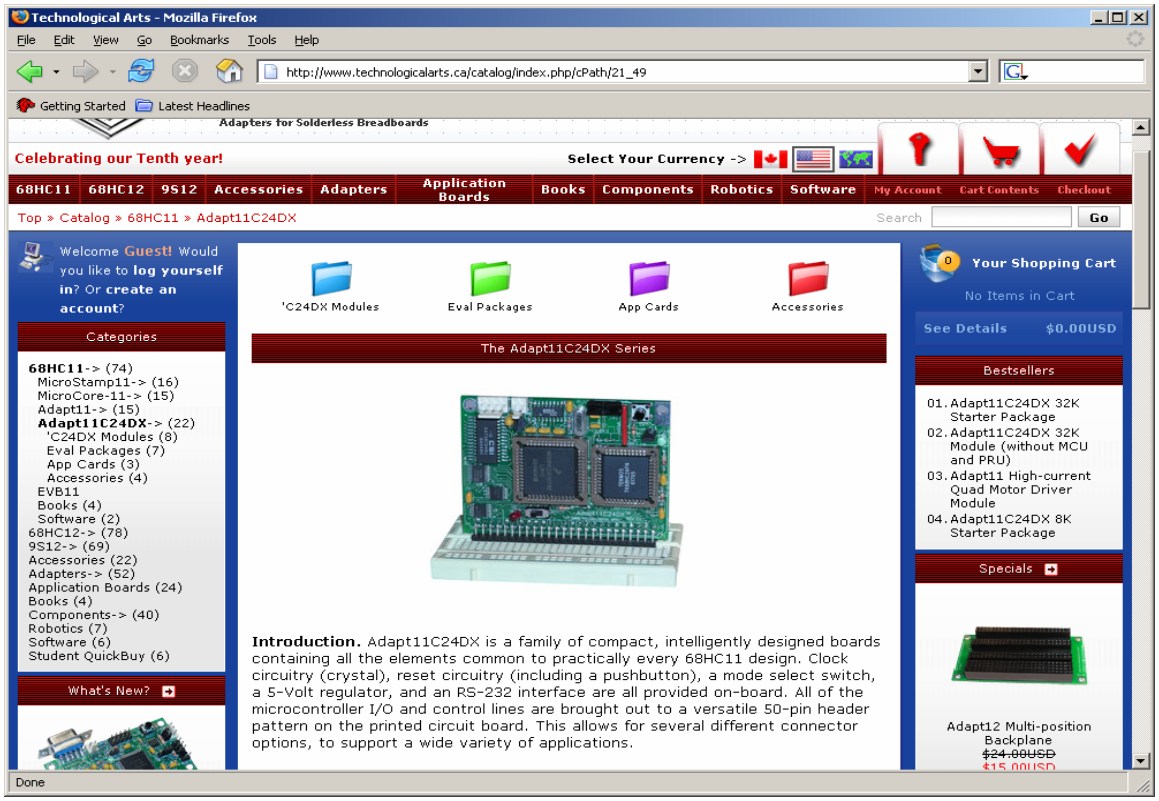




MiniIDE has changed the untitled file to ***test.asm***.

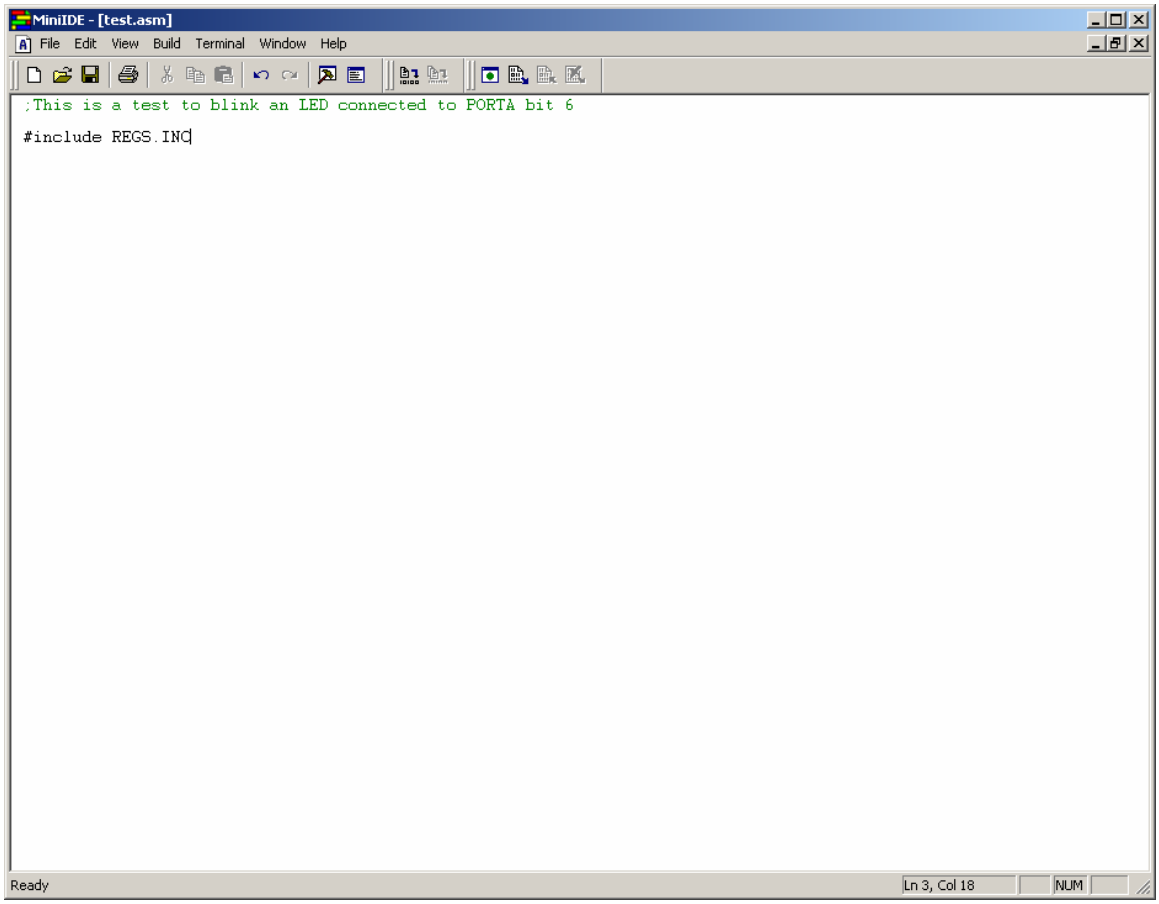


This document will use the Adapt11C24DX from Technological Arts.
http://www.technologicalarts.ca/catalog/index.php/cPath/21_49



In this example, various Register definitions of HC11 are in the include file called ***REGS.INC***. Generally, these types of files are to be found at www.Freescale.com website. If the file does not exist then make one by looking at the Datasheets of the MCUs.

Also, this document assumes that one is familiar with what are PORTs and Registers. This document will only show how to use ***MiniIDE*** all the way to using ***MicroLoad*** in programming the Adapt11 EEPROM. Erasing is done automatically and is transparent to the user. This is assumed to be case if the user is not familiar with the EEPROM state machine .



Parameters:

```
* Operational Parameters
RAM:            equ     $0000           ;Internal Ram start at $0000
STACK:          equ     $00FF           ;At end of RAM
EEPROM:         equ     $F800           ;
VectorTable:    equ     $FFD0           ;Beginning of Vector Table interrupt
```

Please note the use of ***equ.*** It simply means a string is equal to a value to connect both the meaning of the string and the value assigned to it. For example,

```
STACK:          equ     $00FF           ;At end of RAM
```

Means that STACK = $00FF

To define RAM variables by the use of ***ds*** as define segment of a variable. For example below, please note the start of RAM is defined to begin at $0000

```
        Org     RAM

dum     ds.b    1       ; 1 byte of dummy RAM variable
temp    ds.b    1       ; another byte of dummy RAM variable
```

Meaning dum = $0000 and temp = $0001.

Below assigns the start of code. For example,

```
        Org     EEPROM          ;Start of CODE

ResetFunc:                      ;This is where the RESET vector points to
        sei                     ;Disable Any interrupts
```

The is assigned to start at $8000 as defined by

```
EEPROM:         equ     $8000           ;for 32K EEPROM
```

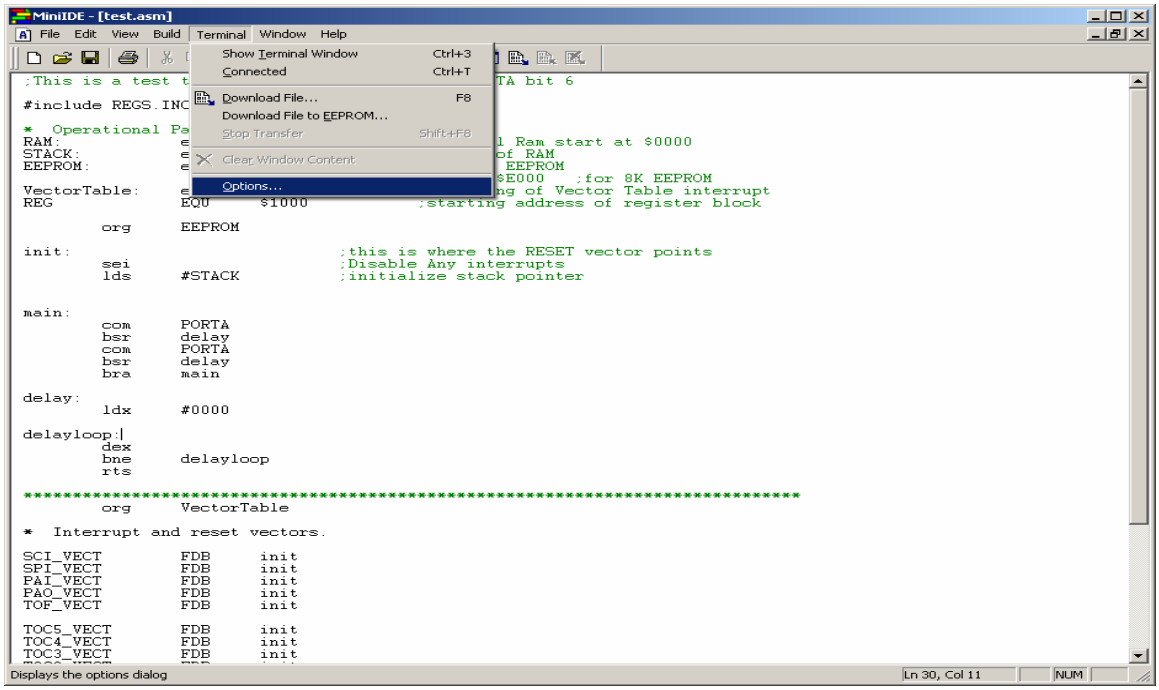
Type the rest of the codes below and once that is done, the code can be assembled or build.

```
;This is a test to blink an LED connected to PORTA bit 6

#include REGS.INC

*  Operational Parameters
RAM:            equ     $0000           ;Internal Ram start at $0000
STACK:          equ     $00FF           ;At end of RAM
EEPROM:         equ     $8000           ;for 32K EEPROM
                                        ;equ    $E000   ;for 8K EEPROM
VectorTable:    equ     $FFD6           ;Beginning of Vector Table interrupt
REG             EQU     $1000           ;starting address of register block

        org     RAM

dum     ds.b    1       ; 1 byte of dummy RAM variable
temp    ds.b    1       ; another byte of dummy RAM variable

        org     EEPROM

Start:                                  ;this is where the RESET vector points
        sei                             ;Disable Any interrupts
        lds     #STACK                  ;Startialize stack pointer
```

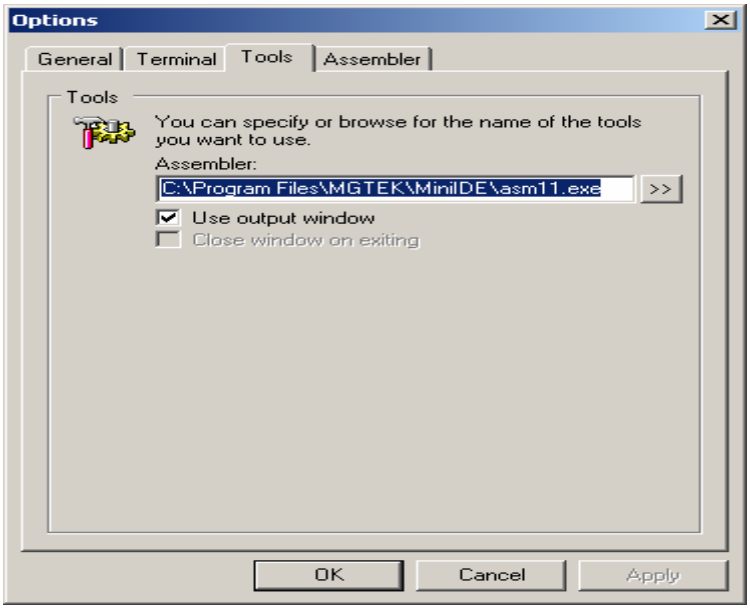
```
main:
            com         PORTA
            bsr         delay
            com         PORTA
            bsr         delay
            bra         main

delay:
            ldx         #0000

delayloop:
            dex
            bne         delayloop
            rts

*******************************************************************************
            org         VectorTable

*  Interrupt and reset vectors.

SCI_VECT            FDB         Start
SPI_VECT            FDB         Start
PAI_VECT            FDB         Start
PAO_VECT            FDB         Start
TOF_VECT            FDB         Start

TOC5_VECT           FDB         Start
TOC4_VECT           FDB         Start
TOC3_VECT           FDB         Start
TOC2_VECT           FDB         Start
TOC1_VECT           FDB         Start
TIC3_VECT           FDB         Start
TIC2_VECT           FDB         Start
TIC1_VECT           FDB         Start

RTI_VECT            FDB         Start
IRQ_VECT            FDB         Start
XIRQ_VECT           FDB         Start
SWI_VECT            FDB         Start
TRAP_VECT           FDB         Start
COP_FAIL_VECT       FDB         Start
COP_CMF_VECT        FDB         Start
RESET_VECT          FDB         Start
```

Assemble or Build a file:

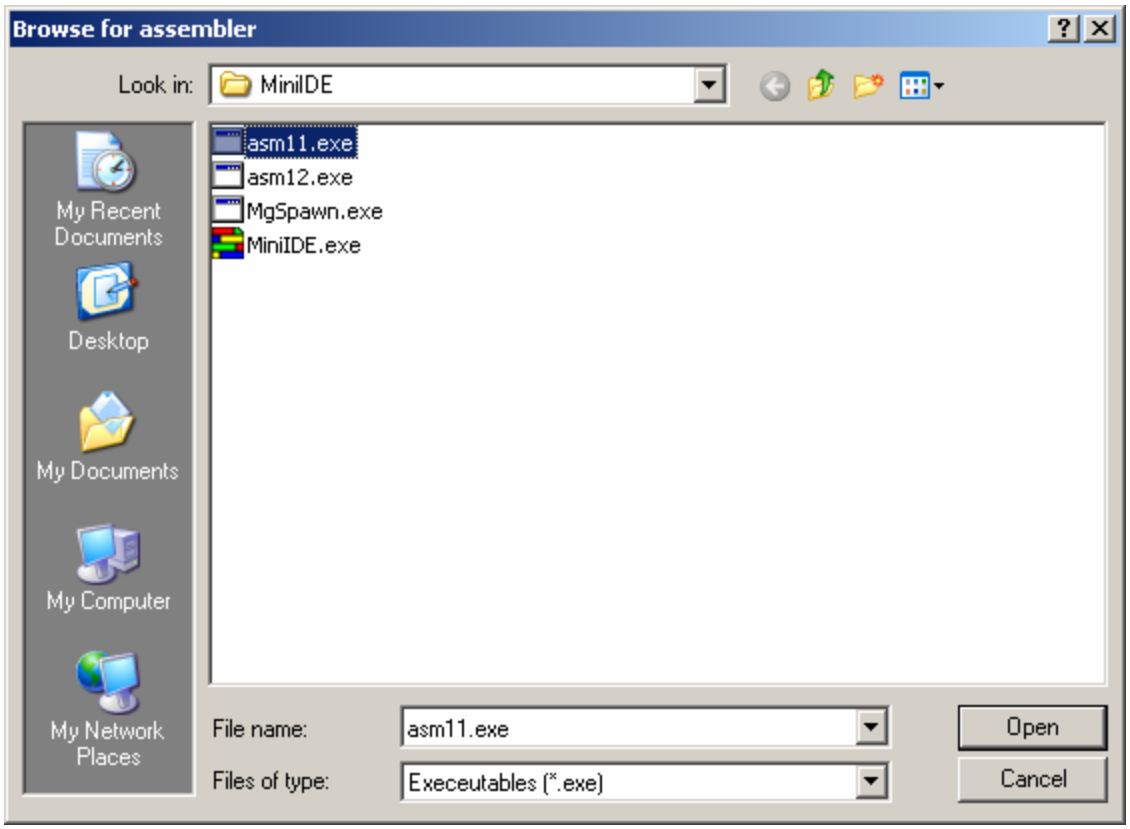
The first thing to do is check the options to make sure it is set for HC11 assembler. Click on Terminal Menu – Options and then Tools tab.



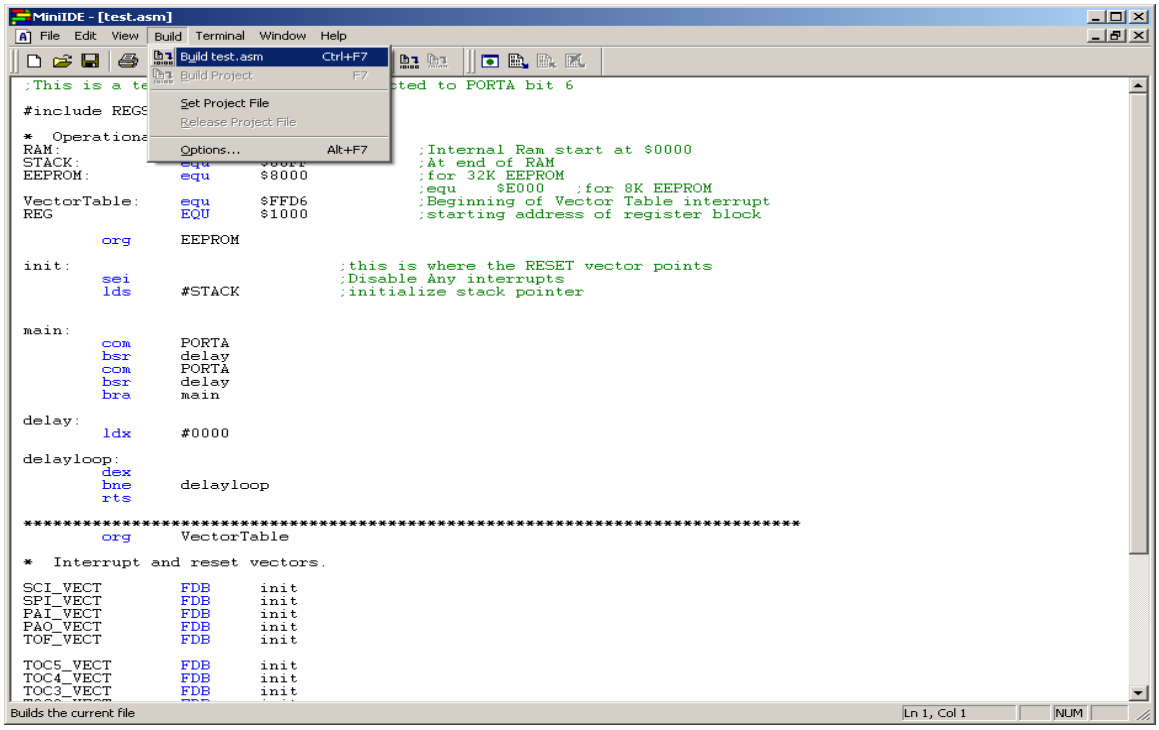
Make sure to select and use the ***asm11.exe*** as the assembler for HC11MCUs.



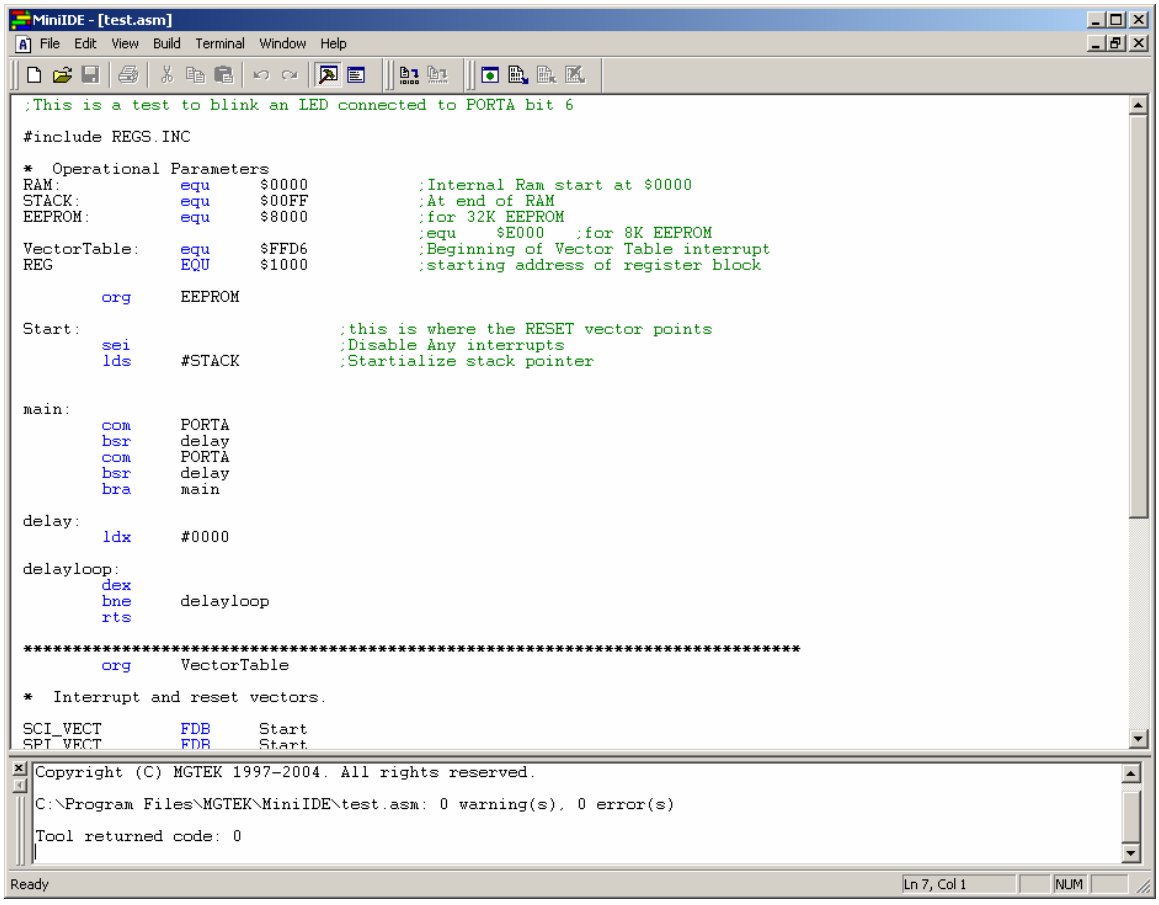
As one can note, the ***asm11.exe*** are for HC11 MCUs.



To build the file, select Build menu – Build ***test.asm*** as shown.



After the build – Please check for any error(s).



Using MicroLoad to ERASE and program EEPROM:

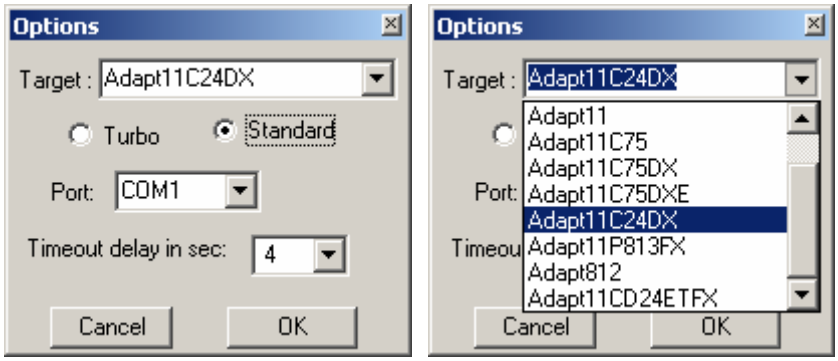
MicroLoad is an easy to use GUI to to erase and program the Adapt11 families. ***MicroLoad*** can be found at Technological Arts website http://www.technologicalarts.com/myfiles/microload.html and can be downloaded at http://pages.interlog.com/~techart/myfiles/files/ml131.zip

Getting Programming started:

Double click on the ***MicroLoad*** icon to start GUI. Below is what ***MicroLoad*** started.

Selecting Targets:

To select a Target click on the Option menu. Click on the Target Pulldown to choose the target of interest as shown. In this example the Adapt11C24DX is selected.

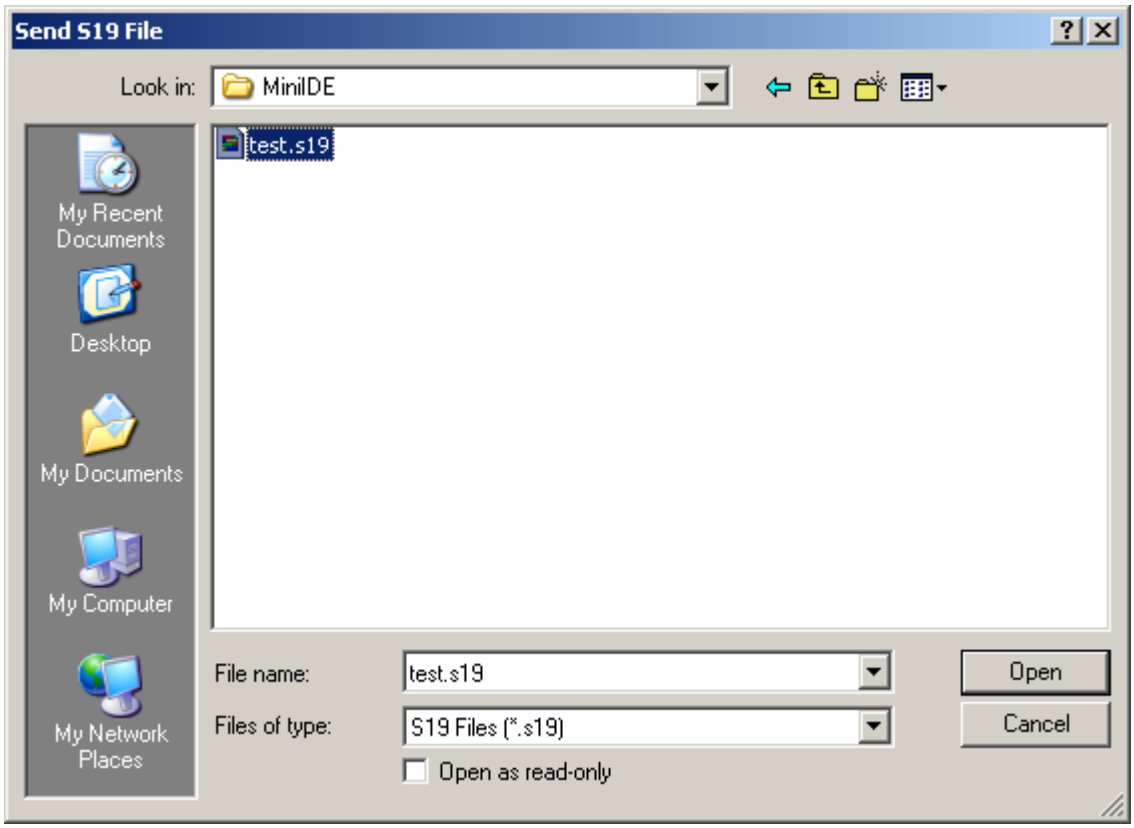


Once a target is selected press the OK button.

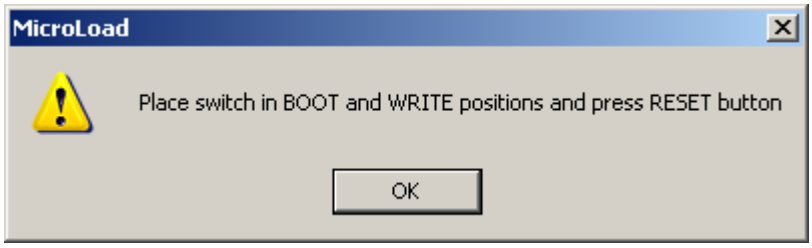
Programming:

To program press the Load button as shown.

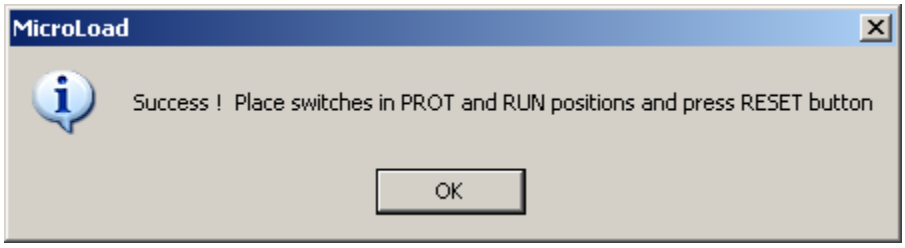
Microload will open an explorer window to help locate the file of interest. Click on test.s19 then press the ***Open*** button.



A warning message will pop to prompt in making sure the Run/Boot switch is in Boot mode and the write protect switch is in write position. For Adapt11C24DX SW2 should be positioned to boot and SW4 to write position. Press the RESET button once these are configured. It is assumed that the module is powered up with known good power supply. Press the OK button to initiate program of EEPROM.



Microload will initiate immediately. After successful programming as shown.

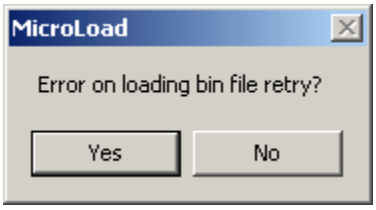


Slide SW4 to protect and SW2 to Run. Press the RESET button and the LED should now blink.

This concludes the use of MiniIDE and MicroLoad.

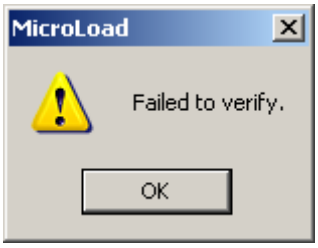
Diagnosis of encountered problems:

Microload error below is shown if PC COM and target are not properly connected.



Symptom 1:

- Wrong COM, do select another one
- Bad cable or wrong cable, do check and verify cable
- Bad power supply, do measure the voltage (5V – 9V)
- Do make MCU in Boot mode. SW2 = boot and press RESET button



Symptom 2:

- S19 record do not match EEPROM. Make sure S-record start at $8000 for 32Kbyte EEPROM and at $E000 for 8Kbyte EEPROM.
- SW4 in not in Write position. Do check and verify SW4 position for write.

REGS.INC

```
; Define Register Block Addresses

_PORTA          EQU     00h             ;I/O port A
_PORTB          EQU     04h             ;output port B
_PORTC          EQU     03h             ;I/O port C
_DDRC           EQU     07h             ;data direction for port C (1=output,
0=input)
_PORTD          EQU     08h             ;I/O port D
_DDRD           EQU     09h             ;data direction for port D
_PORTE          EQU     0ah             ;I/O port E

_TCNT           EQU     0eh             ;timer counter register

_TOC1           EQU     16h             ;output compare 1 register
_TOC1_HI        EQU     _TOC1           ;  high byte
_TOC1_LO        EQU     _TOC1+1         ;  low byte
_TOC2           EQU     18h             ;output compare 2 register
_TOC2_HI        EQU     _TOC2           ;  high byte
_TOC2_LO        EQU     _TOC2+1         ;  low byte
_TOC3           EQU     1ah             ;output compare 3 register
_TOC3_HI        EQU     _TOC3           ;  high byte
_TOC3_LO        EQU     _TOC3+1         ;  low byte
_TOC4           EQU     1ch             ;output compare 4 register
_TOC4_HI        EQU     _TOC4           ;  high byte
_TOC4_LO        EQU     _TOC4+1         ;  low byte
_TOC5           EQU     1eh             ;output compare 5 register
_TOC5_HI        EQU     _TOC5           ;  high byte
_TOC5_LO        EQU     _TOC5+1         ;  low byte

_TCTL1          EQU     20h             ;timer control register 1
_TCTL2          EQU     21h             ;timer control register 2
_TMSK1          EQU     22h             ;main timer interrupt mask register 1
_TFLG1          EQU     23h             ;main timer interrupt flag register 1
_TMSK2          EQU     24h             ;main timer interrupt mask register 2
_TFLG2          EQU     25h             ;main timer interrupt flag register 2

_PACTL          EQU     26h             ;port A control port

_SPCR           EQU     28h         ;SPI Control Register
_SPSR           EQU     29h         ;SPI Status Register
_SPDR           EQU     2Ah         ;SPI Data Register

_BAUD           EQU     2bh             ;baud rate register
_SCCR1          EQU     2ch             ;SCI control register 1
_SCCR2          EQU     2dh             ;SCI control register 2
_SCSR           EQU     2eh             ;SCI status register
_SCDR           EQU     2fh             ;SCI data register

_ADCTL          EQU     30h             ;A/D control/status register
_ADR1           EQU     31h             ;A/D result register 1
_ADR2           EQU     32h             ;A/D result register 2
_ADR3           EQU     33h             ;A/D result register 3
_ADR4           EQU     34h             ;A/D result register 4
```

```
_BPROT          EQU     35h             ;block protect register

_OPTION         EQU     39h             ;system configuration options
_COPRST         EQU     3ah             ;arm/reset COP timer
_PPROG          EQU     3bh             ;EEPROM programming register
_HPRIO          EQU     3ch             ;highest priority interrupt
_INIT           EQU     3dh             ;ram - I/O mapping register
_CONFIG         EQU     3fh             ;configuration control register


PORTA           EQU     REG+_PORTA
PORTB           EQU     REG+_PORTB
PORTC           EQU     REG+_PORTC
DDRC            EQU     REG+_DDRC
PORTD           EQU     REG+_PORTD
DDRD            EQU     REG+_DDRD
PORTE           EQU     REG+_PORTE

TCNT            EQU     REG+_TCNT

TOC1            EQU     REG+_TOC1
TOC1_HI         EQU     REG+_TOC1_HI
TOC1_LO         EQU     REG+_TOC1_LO
TOC2            EQU     REG+_TOC2
TOC2_HI         EQU     REG+_TOC2_HI
TOC2_LO         EQU     REG+_TOC2_LO
TOC3            EQU     REG+_TOC3
TOC3_HI         EQU     REG+_TOC3_HI
TOC3_LO         EQU     REG+_TOC3_LO
TOC4            EQU     REG+_TOC4
TOC4_HI         EQU     REG+_TOC4_HI
TOC4_LO         EQU     REG+_TOC4_LO
TOC5            EQU     REG+_TOC5
TOC5_HI         EQU     REG+_TOC5_HI
TOC5_LO         EQU     REG+_TOC5_LO

TCTL1           EQU     REG+_TCTL1
TCTL2           EQU     REG+_TCTL2
TMSK1           EQU     REG+_TMSK1
TFLG1           EQU     REG+_TFLG1
TMSK2           EQU     REG+_TMSK2
TFLG2           EQU     REG+_TFLG2

PACTL           EQU     REG+_PACTL

BAUD            EQU     REG+_BAUD
SCCR1           EQU     REG+_SCCR1
SCCR2           EQU     REG+_SCCR2
SCSR            EQU     REG+_SCSR
SCDR            EQU     REG+_SCDR

ADCTL           EQU     REG+_ADCTL
ADR1            EQU     REG+_ADR1
ADR2            EQU     REG+_ADR2
ADR3            EQU     REG+_ADR3
ADR4            EQU     REG+_ADR4
```

```
BPROT           EQU     REG+_BPROT

OPTION          EQU     REG+_OPTION
COPRST          EQU     REG+_COPRST
PPROG           EQU     REG+_PPROG
HPRIO           EQU     REG+_HPRIO
INIT            EQU     REG+_INIT
CONFIG          EQU     REG+_CONFIG
```