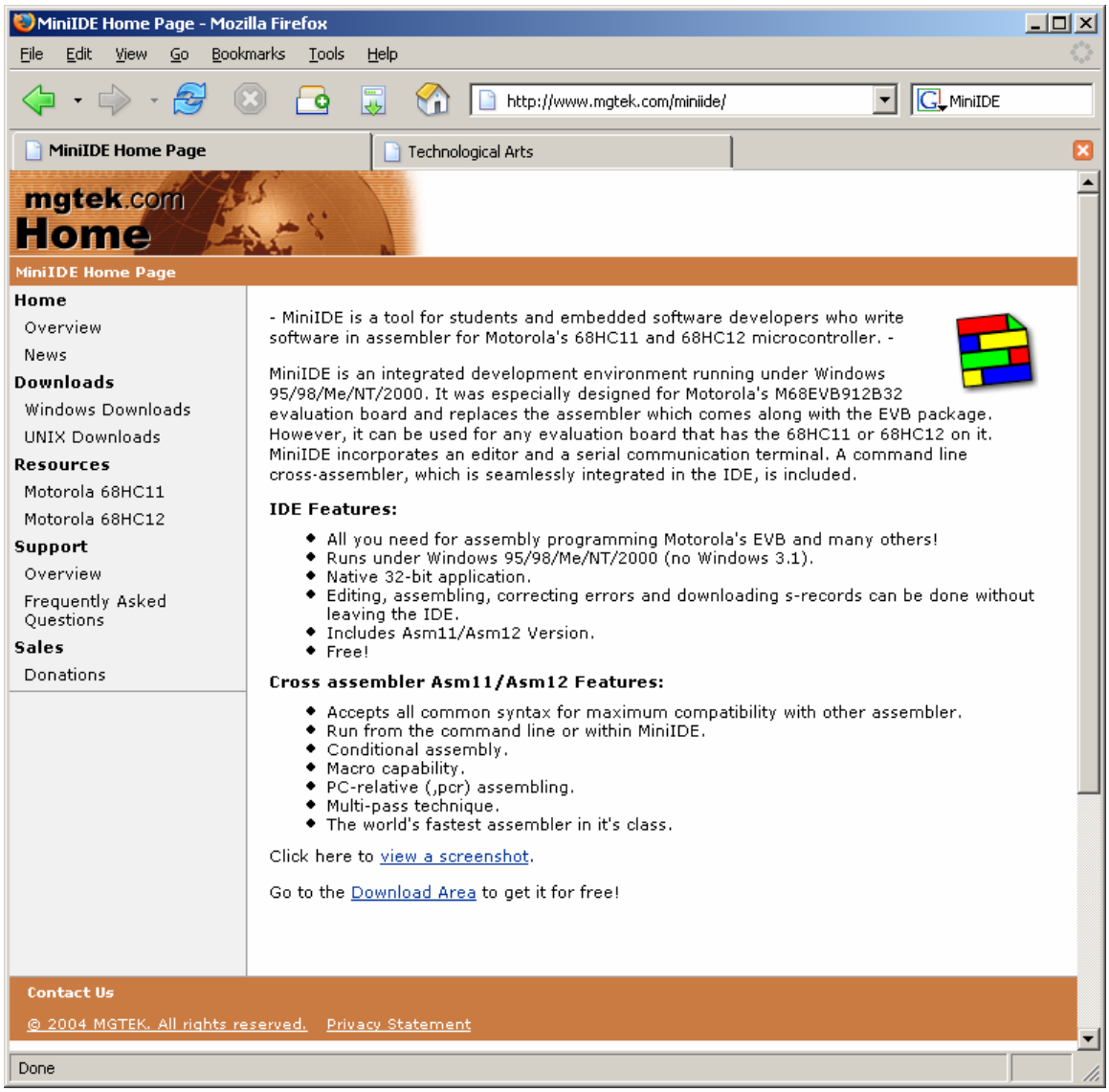
How to use MiniIDE with uBUG12

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

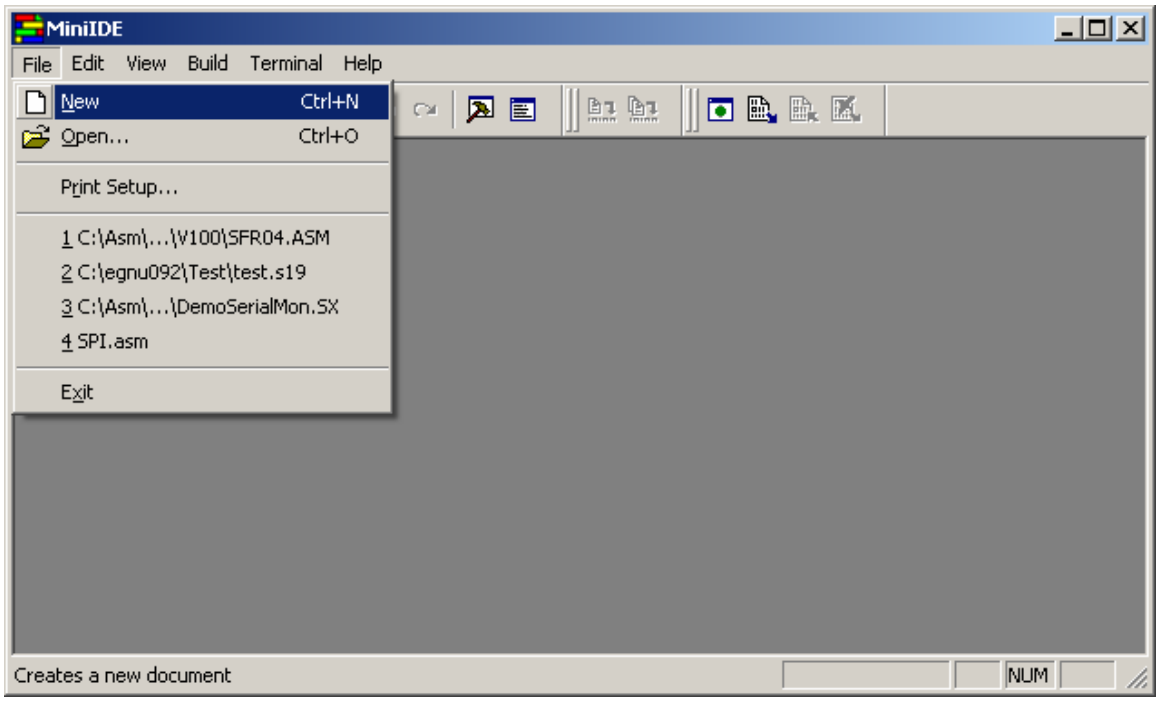


Click on the Download area to select your Operating System (OS) and download MiniIDE.

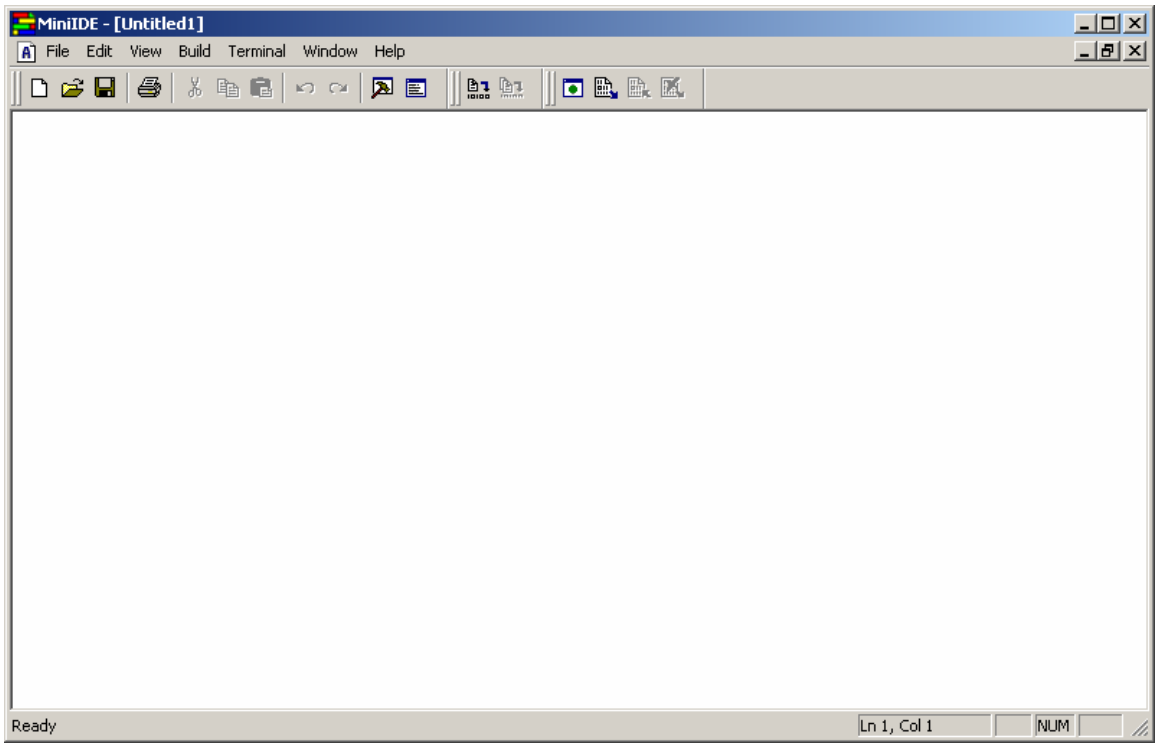
This document assumes that MiniIDE has been installed in your computer.

Getting Started:

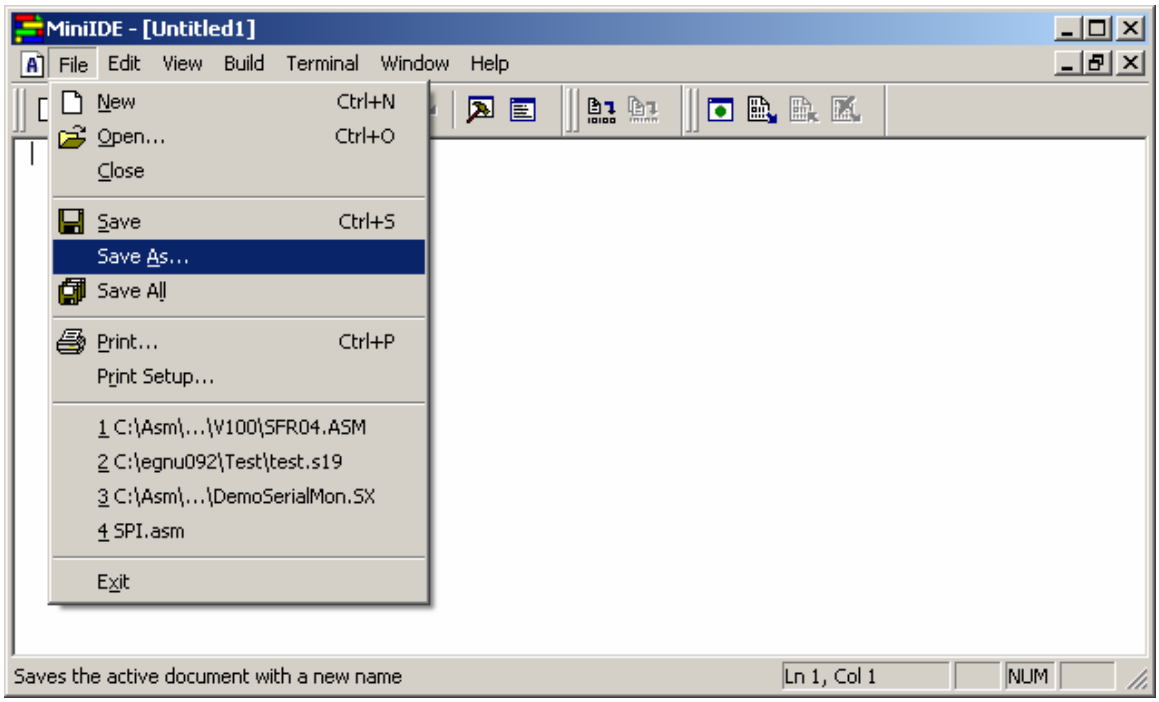
To create a new document click Menu – File - New

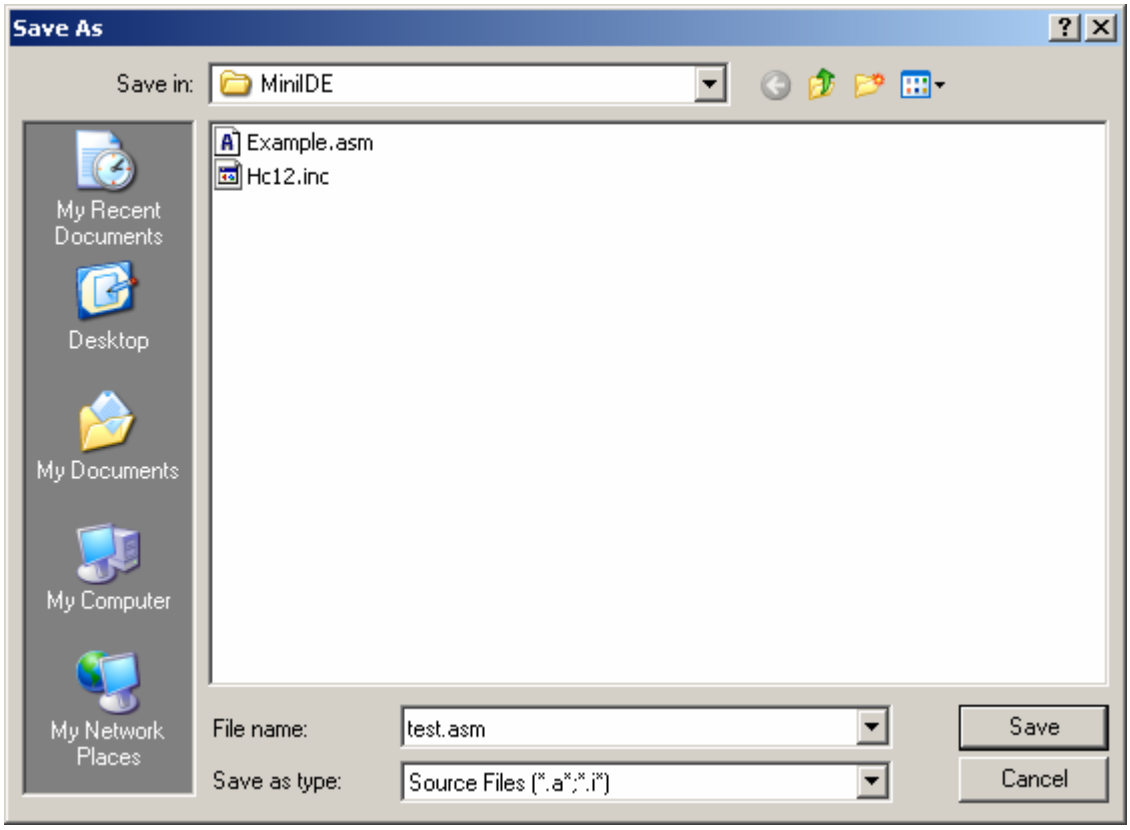


A new ***untitled*** file is created.

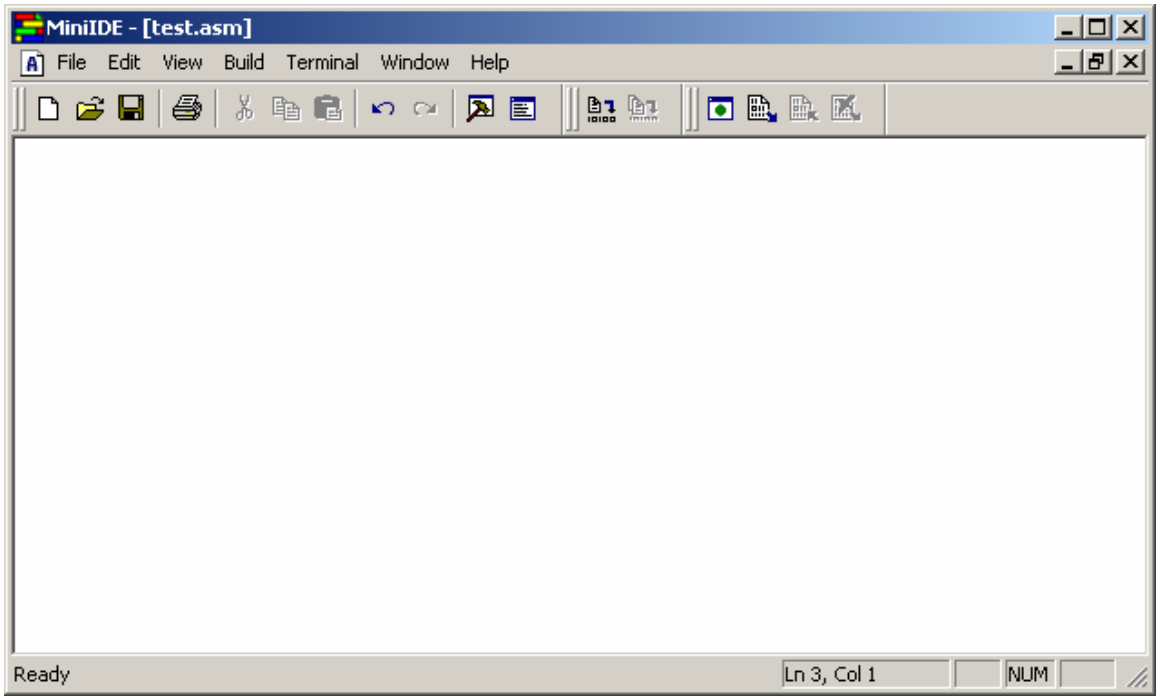


In this example the LEDs connected to PT0 and PT1 are toggled. The file is Save As ***test.asm.***

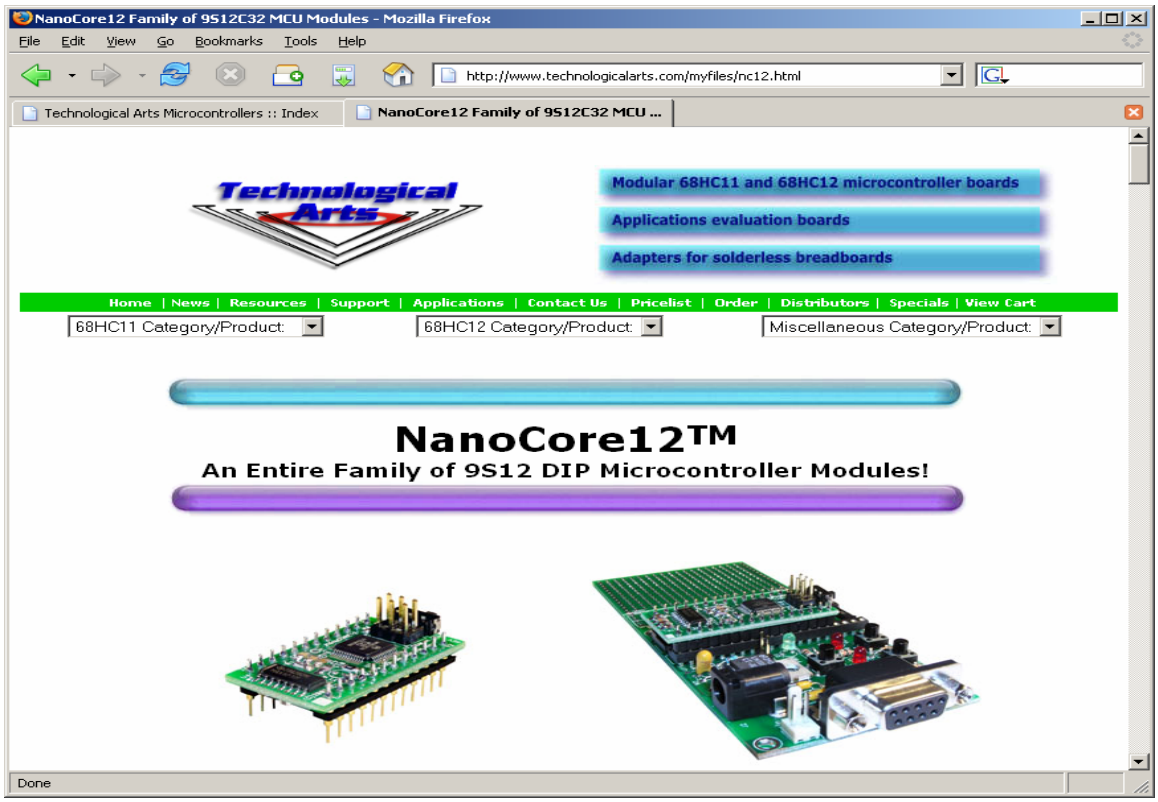




MiniIDE has changed the untitled file to ***test.asm***.

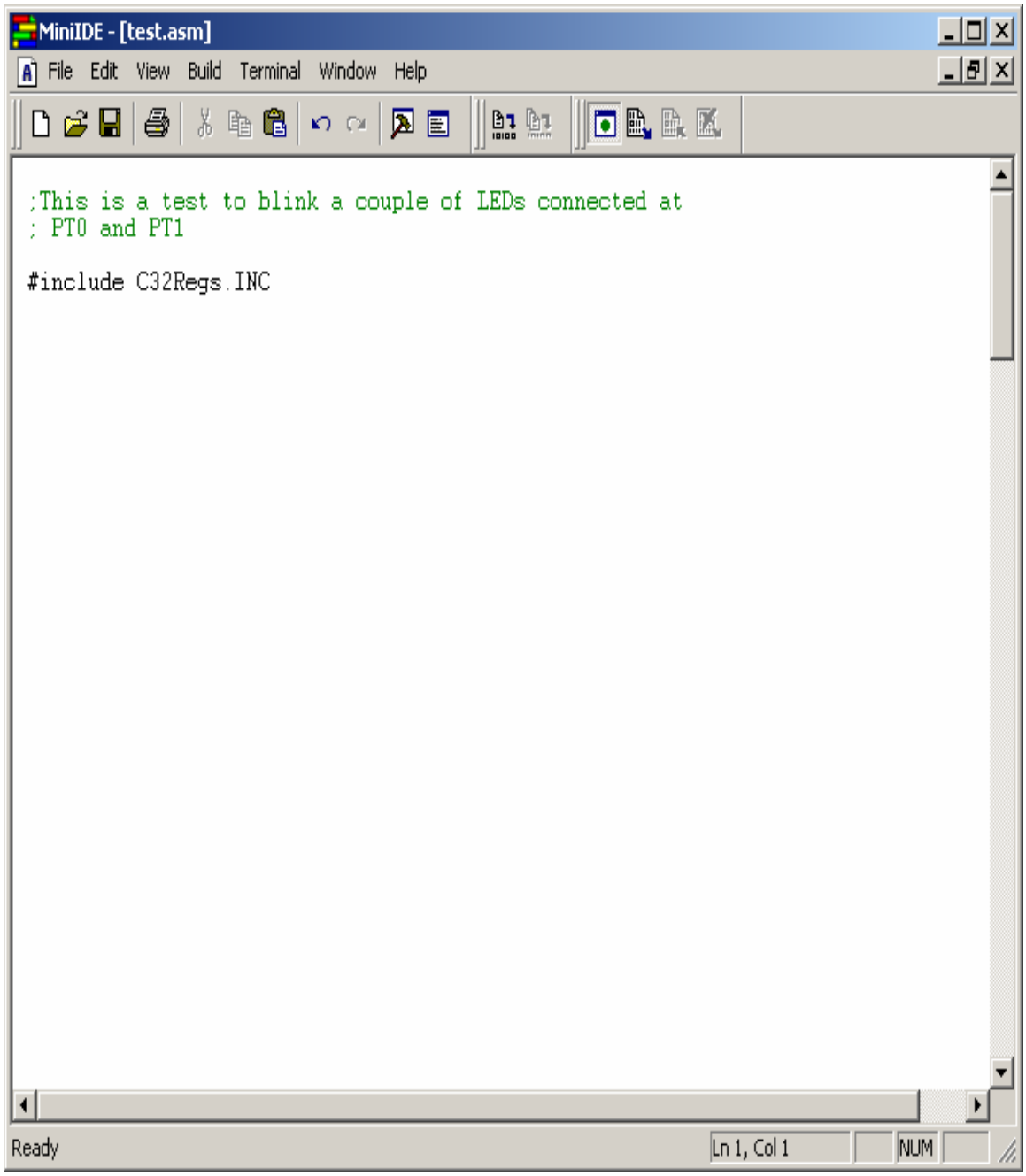


This document will use the NC12DX with Docking Module from Technological Arts. http://www.technologicalarts.com/myfiles/nc12.html



In this example, various Register definitions of 9S12C32 are in the include file called C32Regs.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 uBUG12 in erasing and programming the NC12DX Flash.



Parameters:

For compatibility with other 9S12 MCU the RAM gets move from default location to $3800 to $3FFF.

```
* Operational Parameters
RAM:            equ     $3800           ;Ram got move from default to $3800 - $3FFF
STACK:          equ     $3F80           ;At end of RAM
FLASH:          equ     $4000           ;Fixed FLASH or PPAGE = $3E
VectorTable:    equ     $FF80           ;Beginning of Vector Table interrupt

OscFreq:        equ     8000            ;Enter Osc speed
initSYNR:       equ     $02             ; mult by synr + 1 = 3 (24MHz)
initREFDV:      equ     $00             ;
PLLSEL:         equ     %10000000       ;PLL select bit
LOCK:           equ     %00001000       ;lock status bit
PLLON:          equ     %01000000       ;phase lock loop on bit
```

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     $3F80           ;At end of RAM
```

Means that STACK = $3F80

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 $3800

```
        Org     RAM

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

Meaning dum = $3800 and temp = $3801.

Below assigns the start of code. For example,

```
        Org     FLASH           ;Start of CODE

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

The is assigned to start at $4000 as defined by

```
FLASH:          equ     $4000           ;Fixed FLASH or PPAGE = $3E
```

In this example the PLL is enabled. One maynot want the PLL enabled so it is a matter of not including the codes below.

Enabling PLL:

```
; Initialize clock generator and PLL
        bclr    CLKSEL,PLLSEL           ;disengage PLL to system
        bset    PLLCTL,PLLON            ;turn on PLL

        movb    #initSYNR,SYNR          ;set PLL multiplier
        movb    #initREFDV,REFDV        ;set PLL divider

        nop
        nop
        nop

        nop
        nop
        nop
        nop

        brclr   CRGFLG,LOCK,*+0         ;while (!(crg.crgflg.bit.lock==1))
        bset    CLKSEL,PLLSEL           ;engage PLL to system
```

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

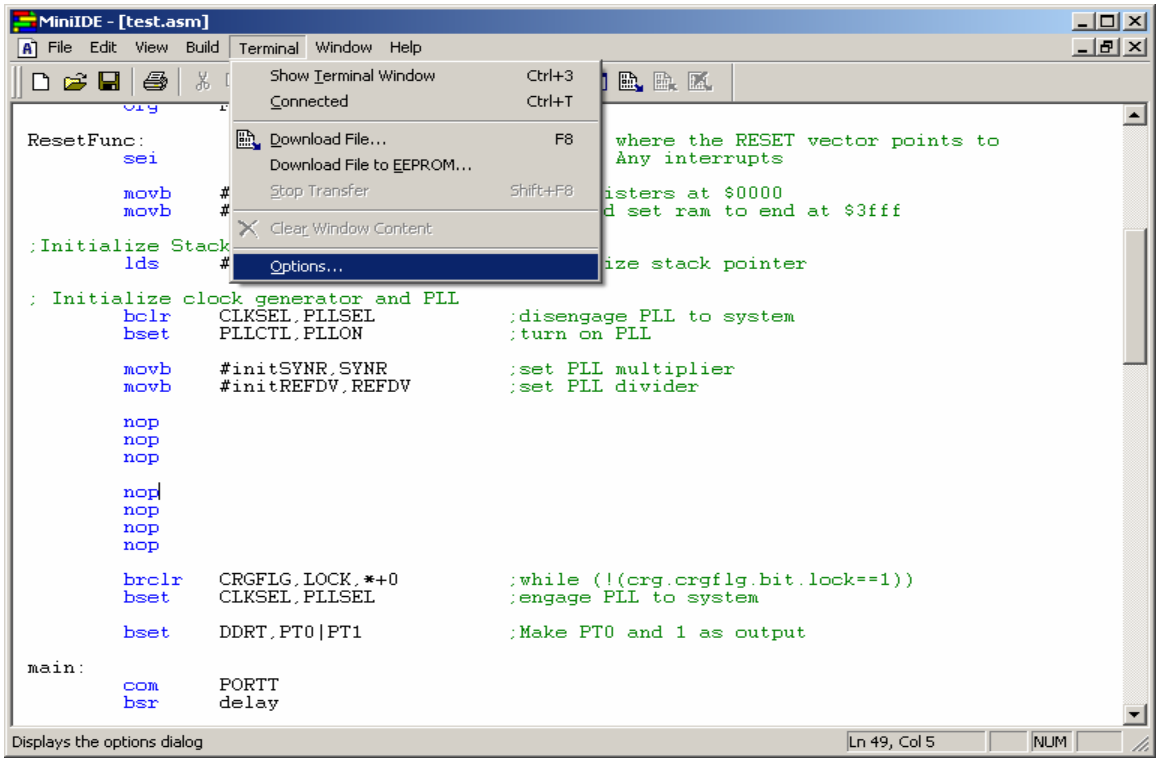
```
;This is a test to blink a couple of LEDs connected at
; PT0 and PT1

#include C32Regs.INC

*  Operational Parameters
RAM:            equ     $3800   ;Ram got move from default to $3800 - $3FFF
STACK:          equ     $3F80   ;At end of RAM
FLASH:          equ     $4000   ;Fixed FLASH or PPAGE = $3E
VectorTable:    equ     $FF80   ;Beginning of Vector Table interrupt

OscFreq:        equ     8000            ;Enter Osc speed
initSYNR:       equ     $02             ; mult by synr + 1 = 3 (24MHz)
initREFDV:      equ     $00             ;
PLLSEL:         equ     %10000000       ;PLL select bit
LOCK:           equ     %00001000       ;lock status bit
PLLON:          equ     %01000000       ;phase lock loop on bit

LED1            equ     1               ;Port T bit 0
LED2            equ     2               ;Port T bit 1


        Org     RAM

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

        Org     FLASH           ;Start of CODE

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

        movb    #$00,INITRG             ;set registers at $0000
        movb    #$39,INITRM             ;move and set ram to end at $3fff

;Initialize Stack
        lds     #STACK                  ;initialize stack pointer

; Initialize clock generator and PLL
        bclr    CLKSEL,PLLSEL           ;disengage PLL to system
        bset    PLLCTL,PLLON            ;turn on PLL

        movb    #initSYNR,SYNR          ;set PLL multiplier
        movb    #initREFDV,REFDV        ;set PLL divider

        nop
        nop
```

```
        nop

        nop
        nop
        nop
        nop

        brclr   CRGFLG,LOCK,*+0         ;while (!(crg.crgflg.bit.lock==1))
        bset    CLKSEL,PLLSEL           ;engage PLL to system

        bset    DDRT,PT0|PT1            ;Make PT0 and 1 as output

main:
        com     PORTT
        bsr     delay
        bsr     delay
        bsr     delay
        bsr     delay

        bra     main


delay:
        ldy     #0000

dlyloop:
        dbne    y,dlyloop
        rts



;-------------------------------------------------------------------------
        ORG     VectorTable             ;Definition of Vector tables

        dc.w    ResetFunc               ;Reserve
        dc.w    ResetFunc               ;Reserve
        dc.w    ResetFunc               ;Reserve
        dc.w    ResetFunc               ;Reserve

        dc.w    ResetFunc               ;PWM Emergency Shutdown
        dc.w    ResetFunc               ;VREG LVI
        dc.w    ResetFunc               ;Port P
        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;Reserved

        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;CAN transmit
        dc.w    ResetFunc               ;CAN receive
        dc.w    ResetFunc               ;CAN errors
        dc.w    ResetFunc               ;CAN wake-up
        dc.w    ResetFunc               ;FLASH

        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;Reserve
        dc.w    ResetFunc               ;Reserve

        dc.w    ResetFunc               ;Reserved
        dc.w    ResetFunc               ;Reserved
```

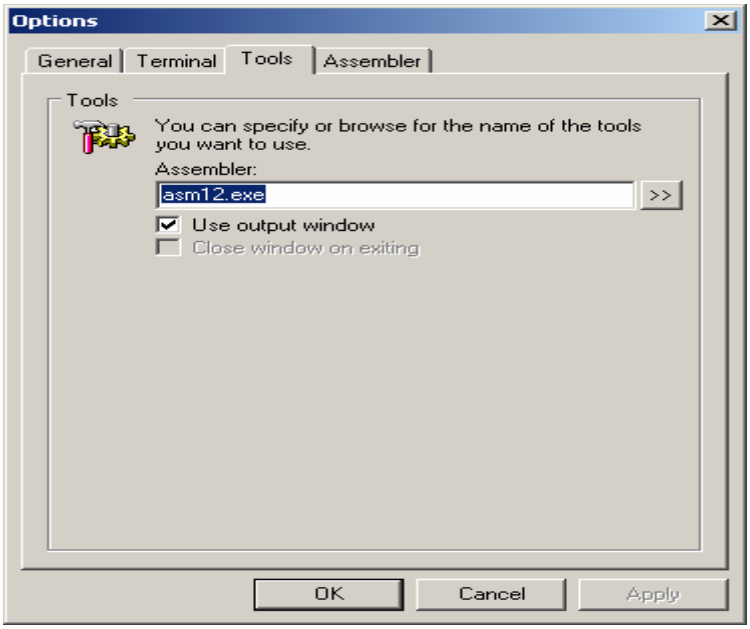
```
dc.w    ResetFunc               ;CRG Self Clock Mode
dc.w    ResetFunc               ;CRG PLL lock
dc.w    ResetFunc               ;Reserved
dc.w    ResetFunc               ;Reserved
dc.w    ResetFunc               ;Reserved

dc.w    ResetFunc               ;Port J (PIEP)
dc.w    ResetFunc               ;Reserved
dc.w    ResetFunc               ;ATD (ATDCTL2 - ASCIE)
dc.w    ResetFunc               ;Reserved
dc.w    ResetFunc               ;SCI
dc.w    ResetFunc               ;SPI
dc.w    ResetFunc               ;Pulse Accumulator 0 input edge
dc.w    ResetFunc               ;Pulse Accumulator 0 overflow
dc.w    ResetFunc               ;Standard Timer 0 Overflow
dc.w    ResetFunc               ;Timer 0 Channel 7
dc.w    ResetFunc               ;Timer 0 Channel 6
dc.w    ResetFunc               ;Timer 0 Channel 5
dc.w    ResetFunc               ;Timer 0 Channel 4

dc.w    ResetFunc               ;Timer 0 Channel 3
dc.w    ResetFunc               ;Timer 0 Channel 2
dc.w    ResetFunc               ;Timer 0 Channel 1
dc.w    ResetFunc               ;Timer 0 Channel 0

dc.w    ResetFunc               ;Real Time Interrupt
dc.w    ResetFunc               ;IRQ
dc.w    ResetFunc               ;XIRQ
dc.w    ResetFunc               ;SWI
dc.w    ResetFunc               ;Instruction Trap
dc.w    ResetFunc               ;COP failure
dc.w    ResetFunc               ;Clock Monitor
dc.w    ResetFunc               ;Power On Reset
```

Assemble or Build a file:

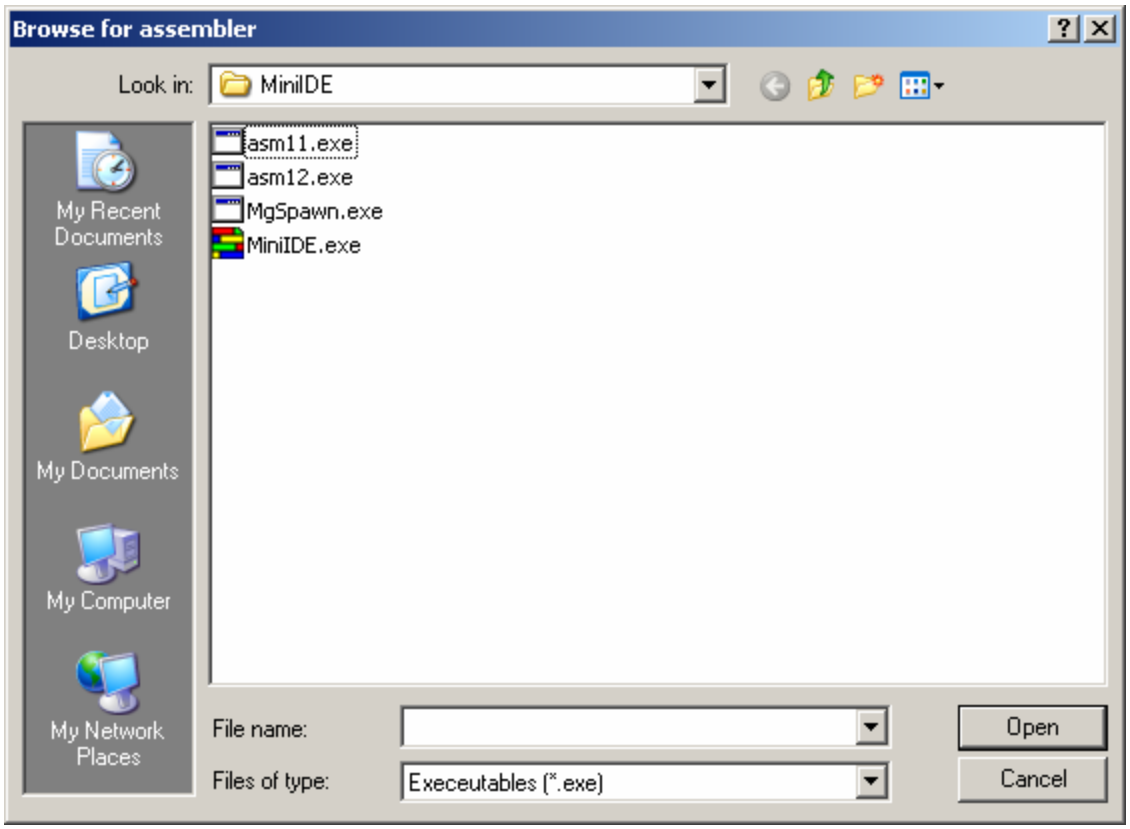
The first thing to do is check the options to make sure it is set for HC12 assembler. Click on Terminal Menu – Options and then Tools tab.



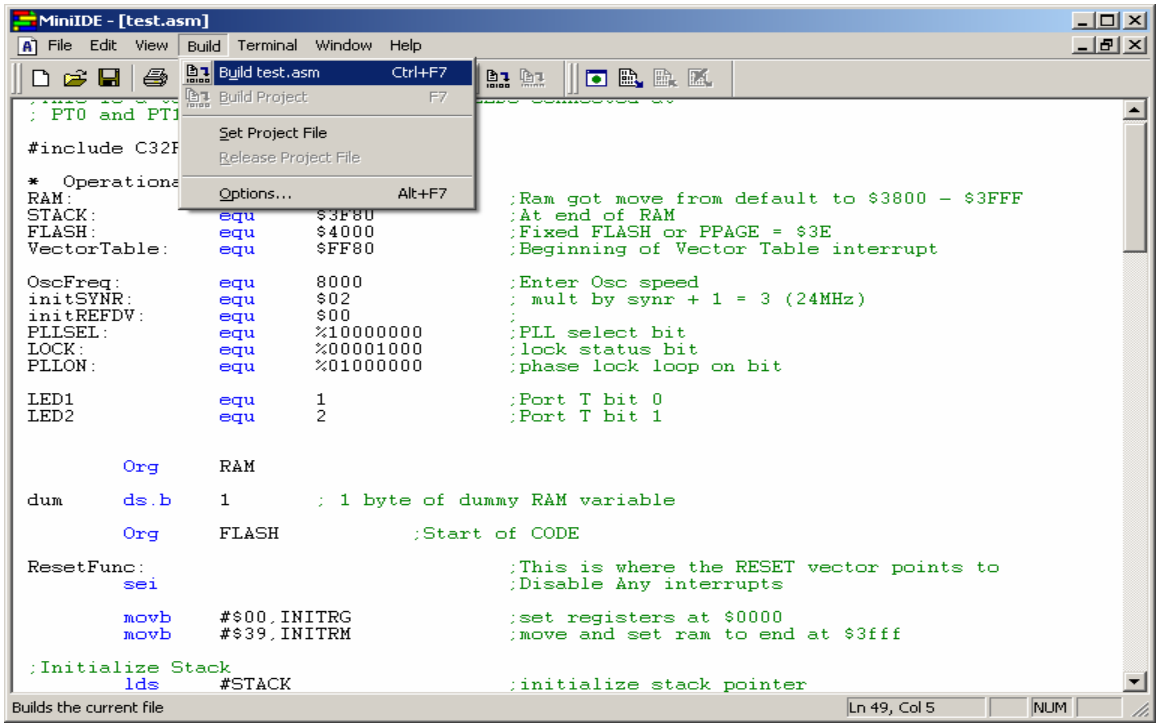
Make sure to select and use the ***asm12.exe*** as the assembler for HC12 and 9S12 MCUs.



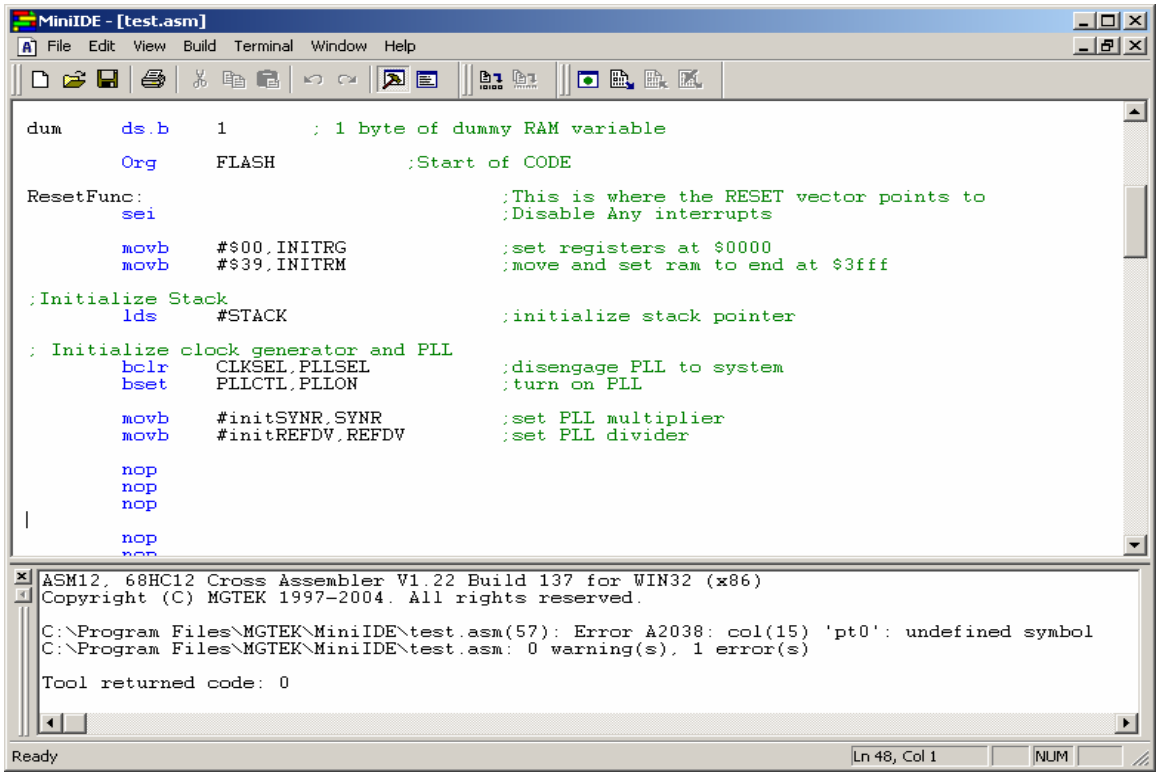
As one can note, the ***asm11.exe*** are for HC11 MCUs.



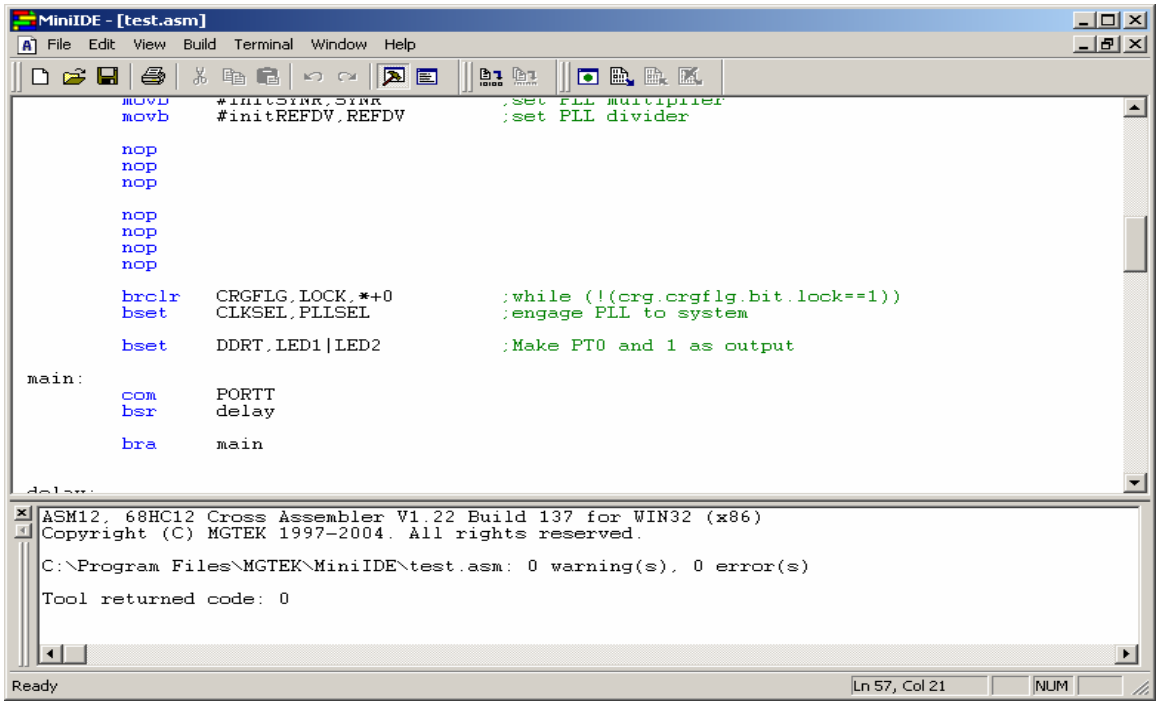
To build the file, select Build menu – Build ***test.asm*** as shown.



After the build – Please note the error(s).



The error is at ***line 57*** showing the string ***pt0*** is an undefined symbol. Go to line 57 and replace both PT0 and PT1 as LED1 and LED2. Save the revised file and re-build. Note now the build is error free.



Using uBUG12 to ERASE and program FLASH:

uBUG12 is a GUI to interface with Freescale's Serial Monitor that are pre-programmed into the NC12s and Adapt9S12E128 families. It has some similarities with Gordon Doughman's DBUG12.

uBUG12 can be downloaded from Technological Arts website
http://support.technologicalarts.ca/files/uBug12.zip

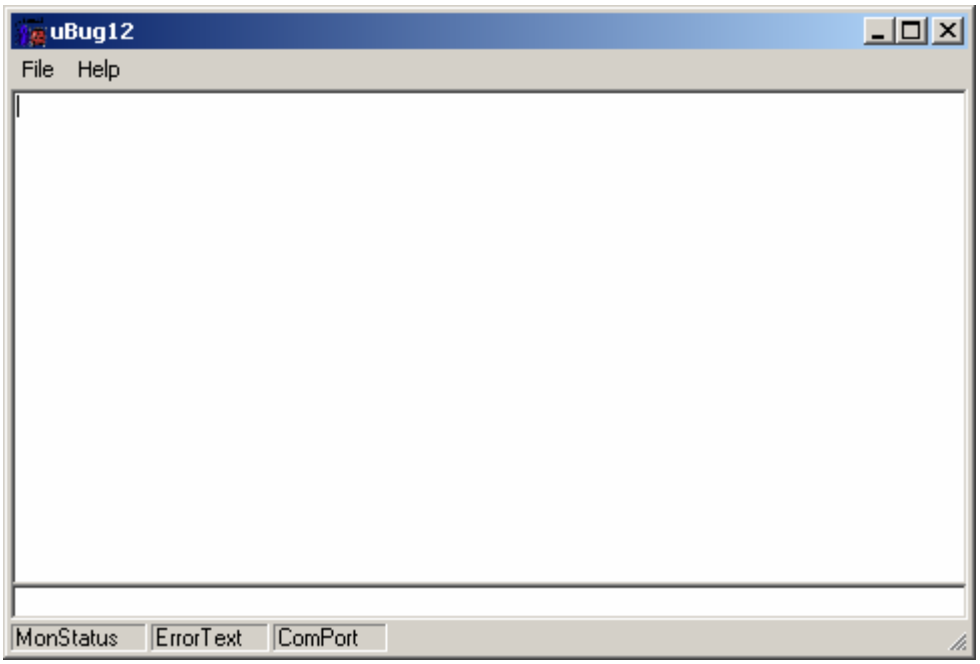
For PCs with Windows98SE the .net framework must be installed in order for uBUG12 to run. WinXP, 2K the .net framework is (usually) already installed. The .net framework can be found at MS website

http://www.microsoft.com/downloads/details.aspx?FamilyID=d7158dee-a83f-4e21-b05a-009d06457787&displaylang=en

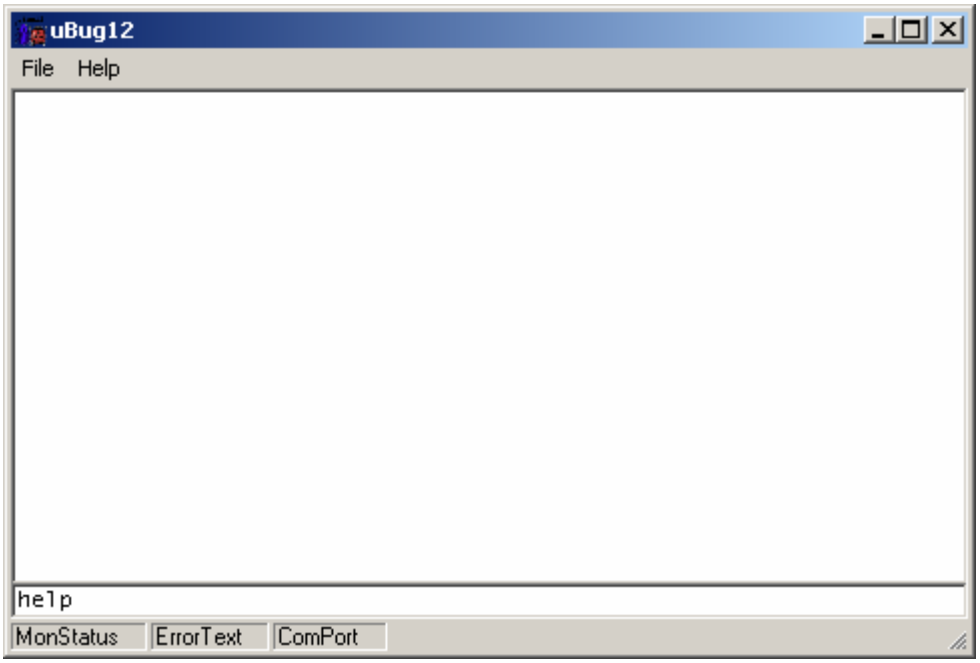
Getting Started:

Double click on the uBUG12 icon to start GUI. Below is what uBUG12 started.

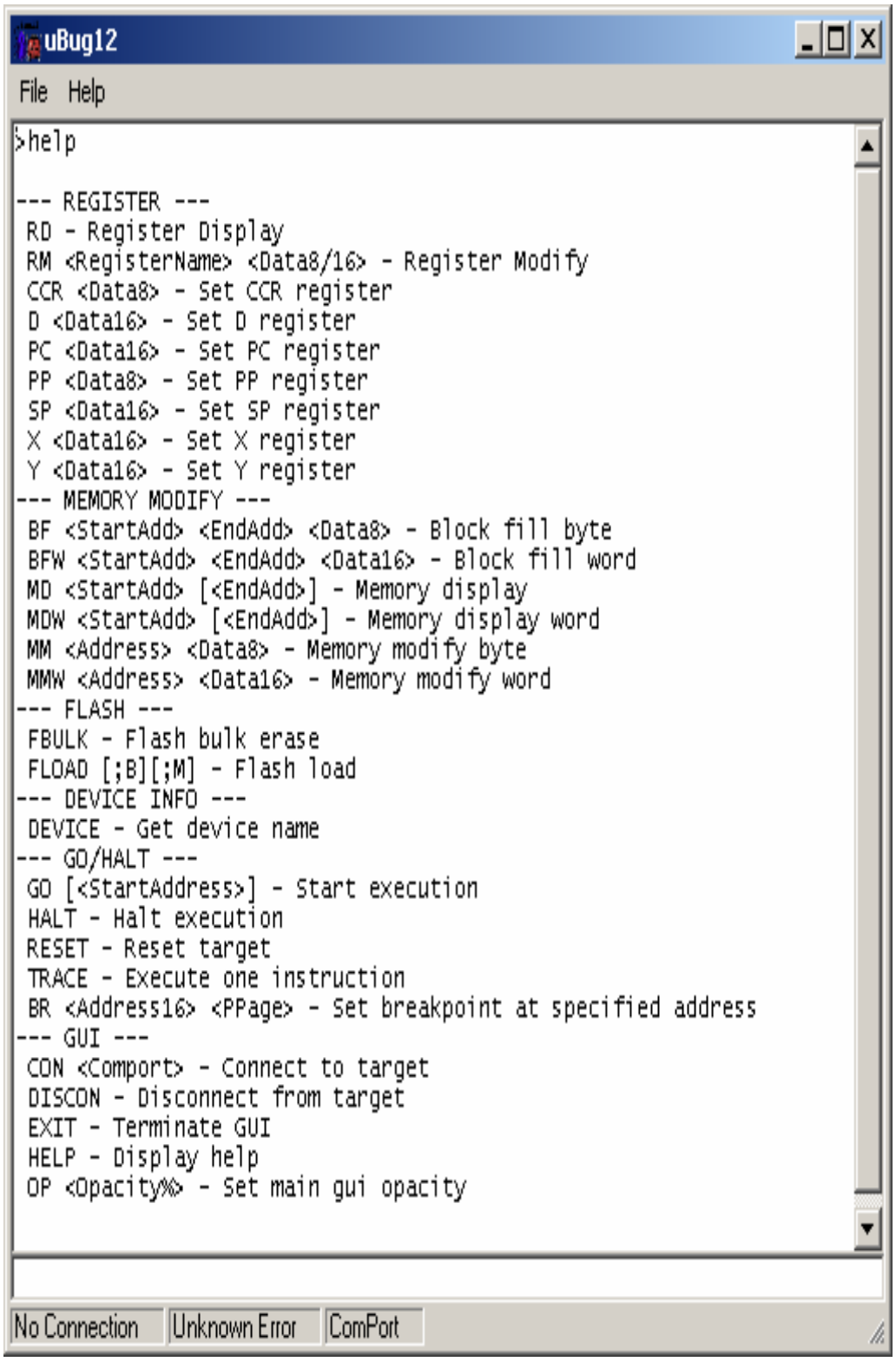
Here uBUG12 is waiting for commands. By typing ***help*** one can see different commands that can be used.



Type the ***help*** command



Once the help command is invoked, uBUG12 will list the different commands as shown.

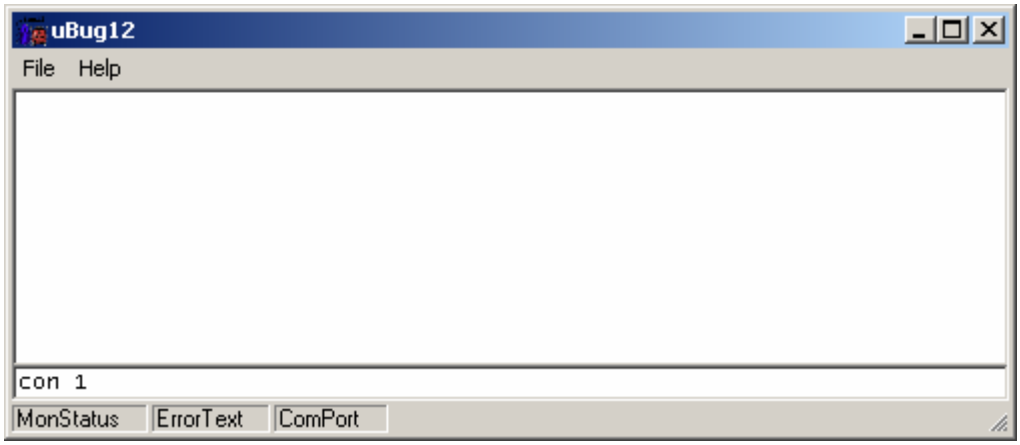


Connecting:

This document will use COM 1 of the PC to connect to the target as an example. For PC without serial port, a USB to COM can be purchase from any computer store.

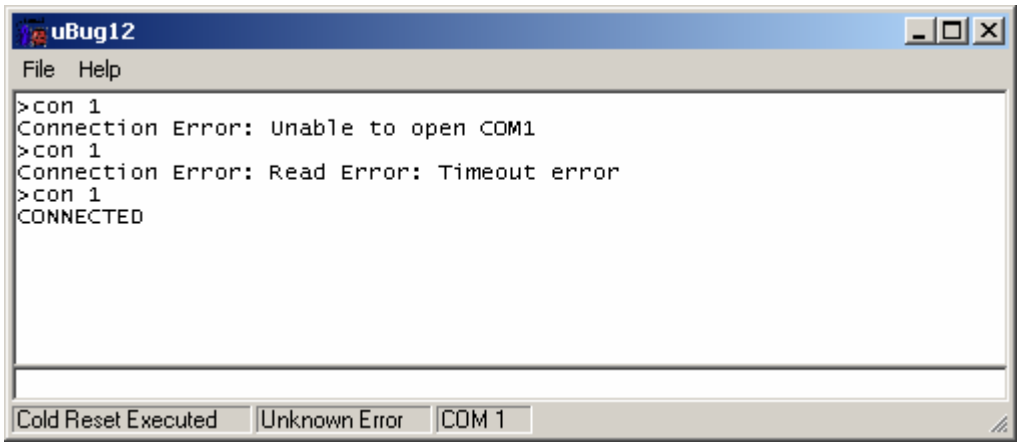
Connect a Serial cable from COM 1 to the Docking Module. Slide the Run/Load switch to Load or Boot then power up the board. Make sure the power LED is on.

The command to connect is ***CON 1*** for COM 1 and ***CON 2*** for COM 2.



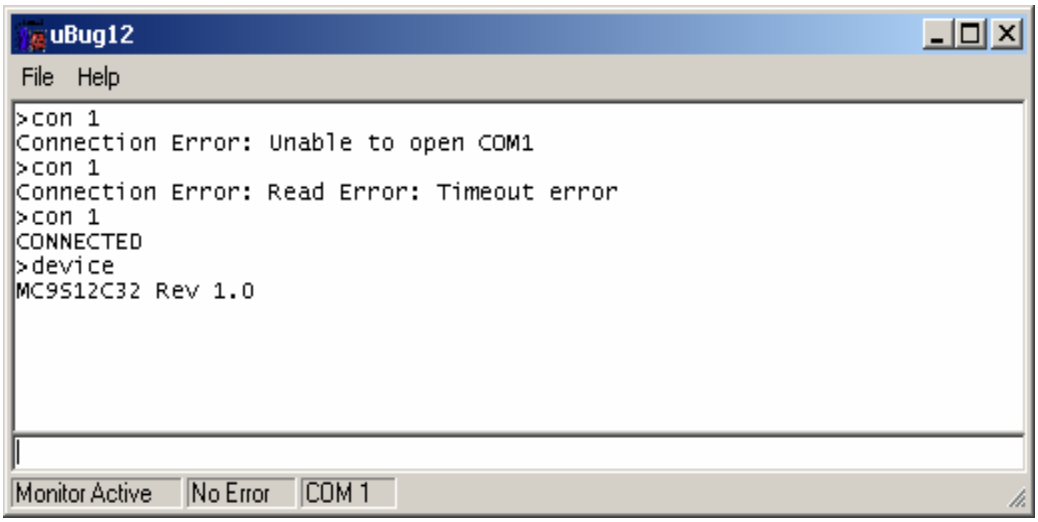
2 possible errors can occur:
Connection Error: Unable to open COM1 <- Another application is using the COM port

Connection Error: Read Error: Timeout error <- The MCU not currently in LOAD mode or the cable is disconnected from either PC or Docking Module, lastly the serial cable is connected on the wrong COM port.



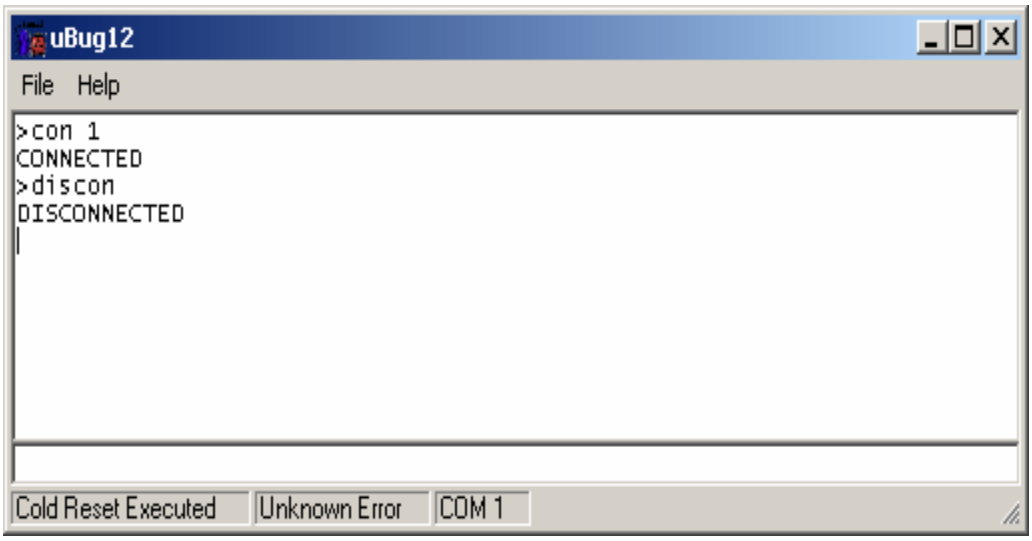
A ***CONNECTED*** message will appear to show good connection between PC and the target.

The ***device*** command will show the target type is as MC9S12C32 Rev 1.0.



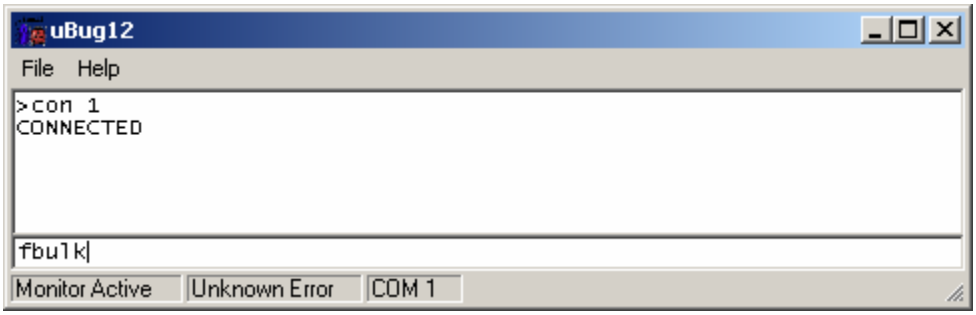
Disconnecting:

To disconnect uBUG12 from the serial port, the command **discon**. This would allow other application to use the COM 1 like MiniIDE, HyperTerm or Tera Term.

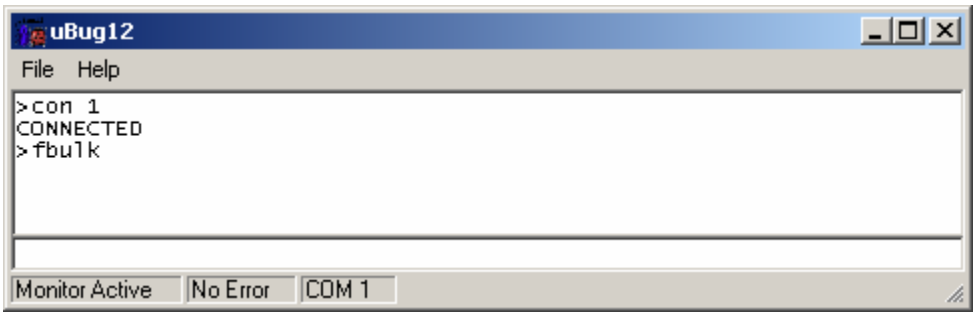


Flash erase and programming:

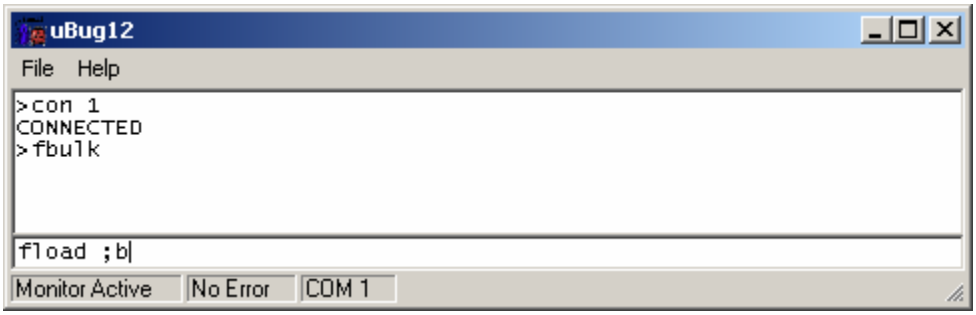
To erase the Flash memory the command is ***FBULK***.



Successful erase

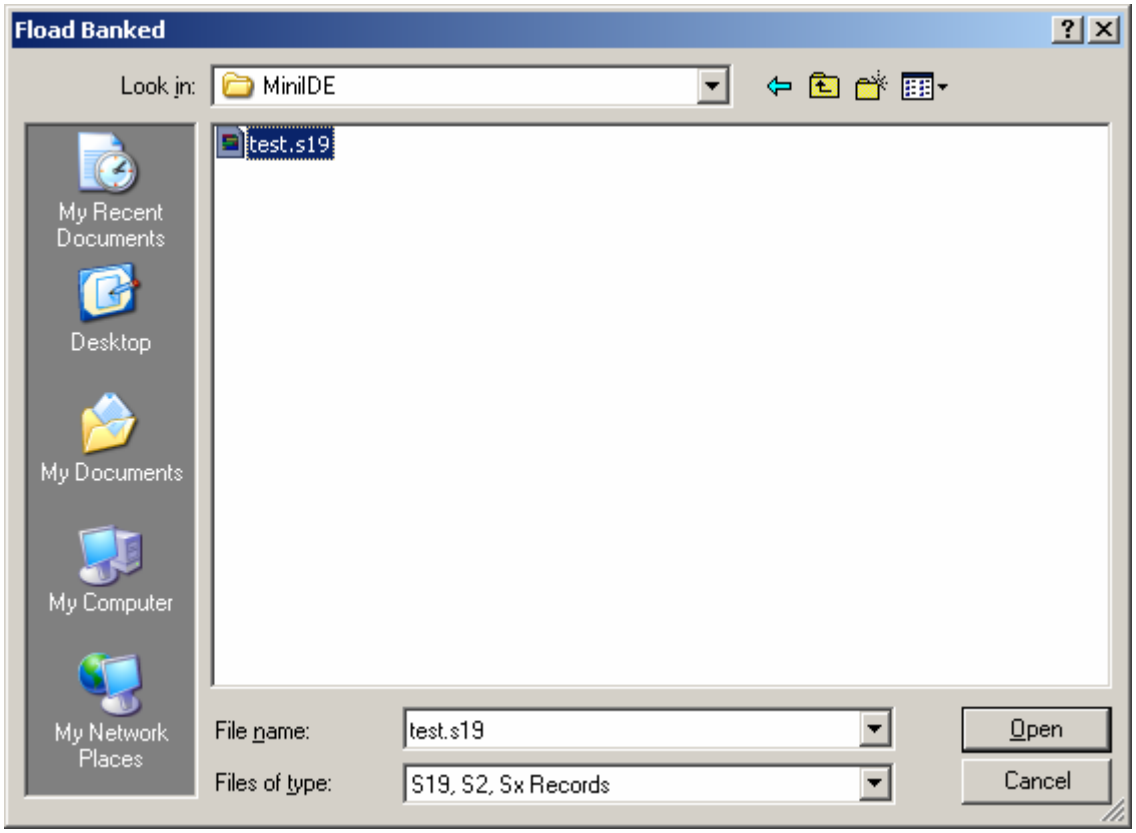


To program FLASH the command is ***Fload ;b*** for banked S19, SX, S2 records. For non banked S2 or formatted S19 (went thru SrecCVT) record the command is ***Fload***.

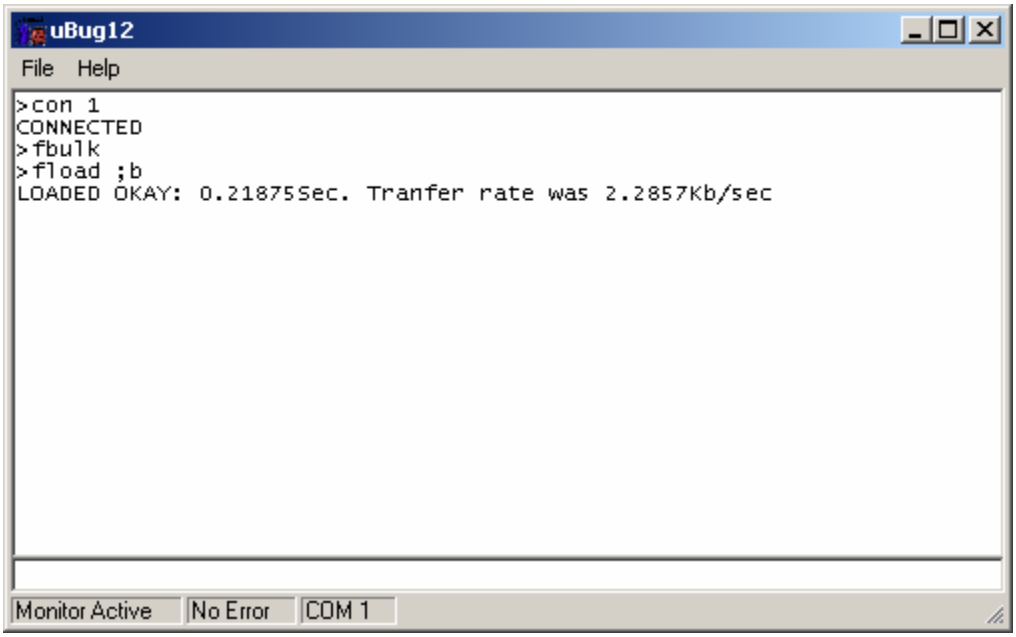


Once the Fload ;b command is invoked, uBUG12 will open an explorer window to help and locate the S-record. In this example, ***test.s19*** will be the target S-record file.

Double click on the file to initiate upload.



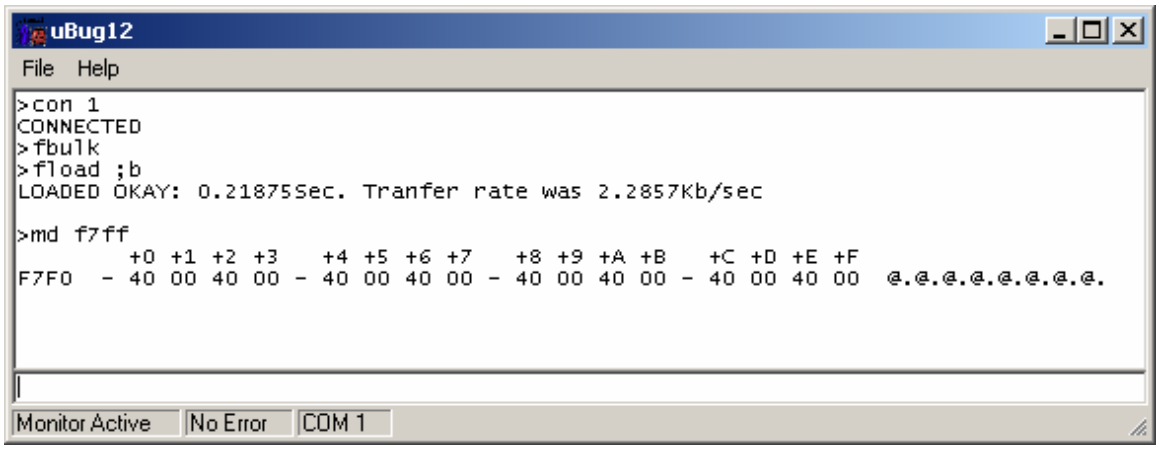
The test.s19 is programmed ok as shown.



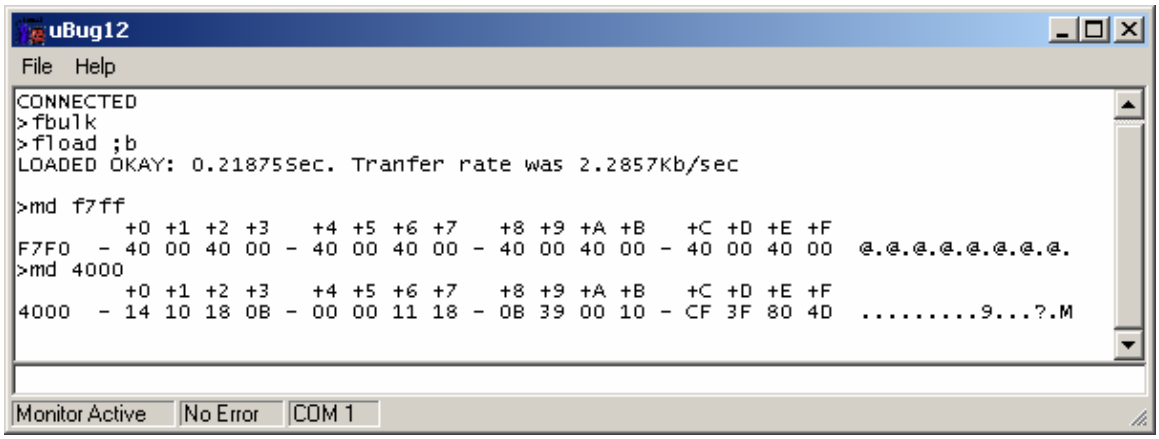
Note that the Serial Monitor resides at $F800 - $FFFF. Therefore uBUG12 will automatically re-locate the vector addresses at below $F800.

Briefly look at the Pseudo Vector address to check where the start of the program. The command is ***md f7ff*** to show a memory dump of the Pseudo Vector address at power up or reset.

The power up reset value at $F7FE is $4000. Therefore the program will start at $4000

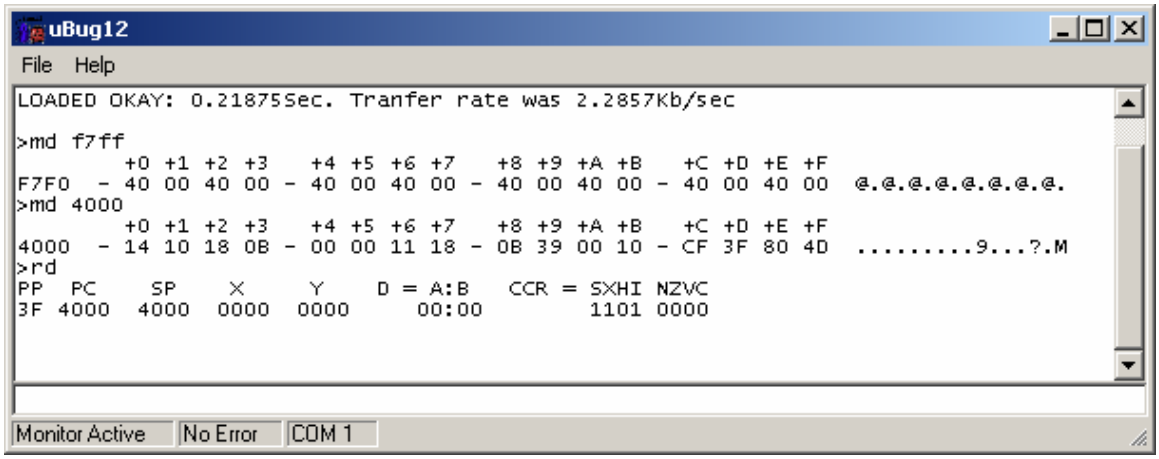


Memory dump at $4000

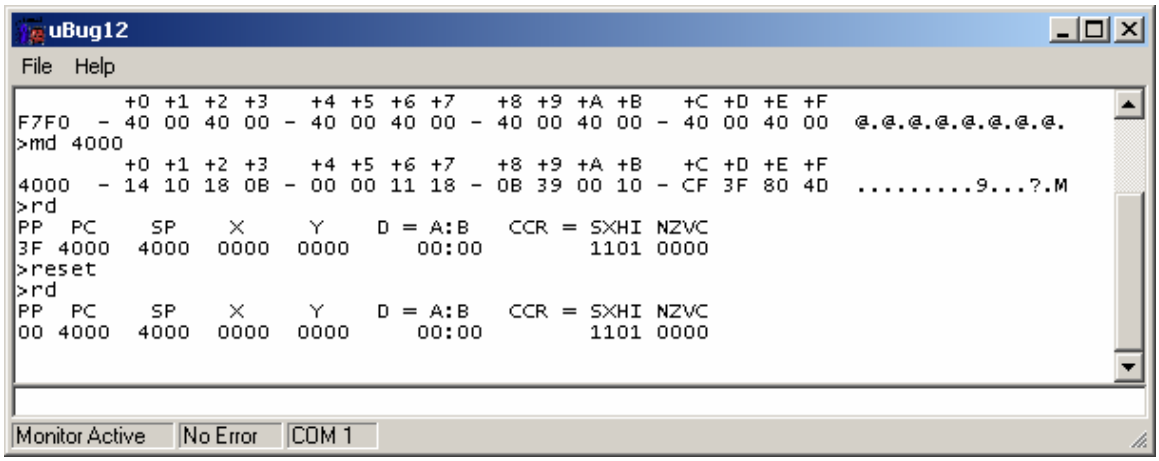


One can see that there are Data at $4000. To execute the program using uBUG12, several registers needs to be initialized. Firstly, look at the registers by the ***RD*** command then invoke the ***RESET*** command to initialize the registers if necessary.

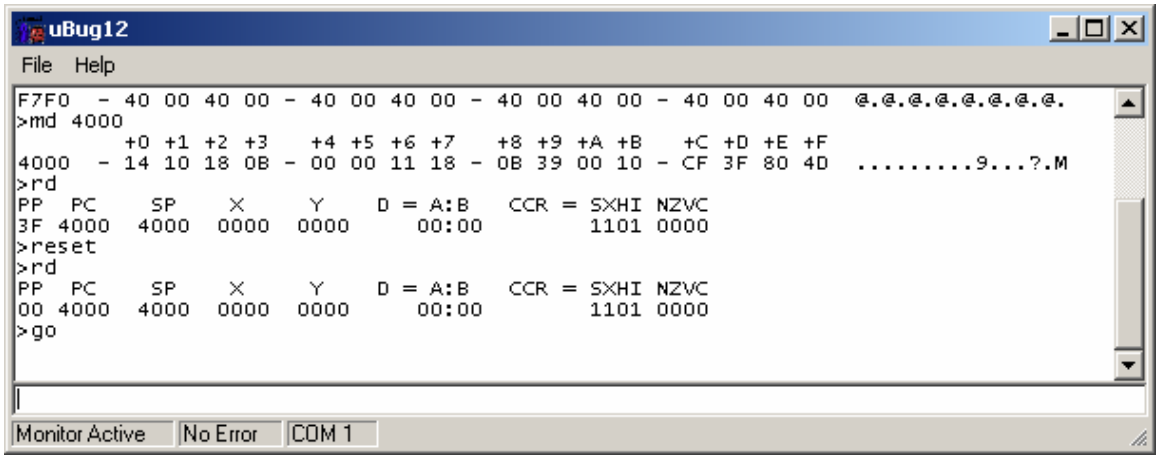
Memory dumps of the registers before ***RESET*** command is invoked.



After ***RESET*** command please note the changes with various registers. In this example only PPAGE (PP) is changed.



To execute program the command is simply type ***go*** (after RESET is invoked) or ***go 4000***.



There are 2 methods to run the code. First using uBUG12 ***go*** command, the second is by sliding the Run/Load or boot switch to Run then press the RESET button on the docking module.

The test.asm will blink the LEDs rapidly.

This concludes using MiniIDE from writing to assembling to erasing and programming the FLASH.