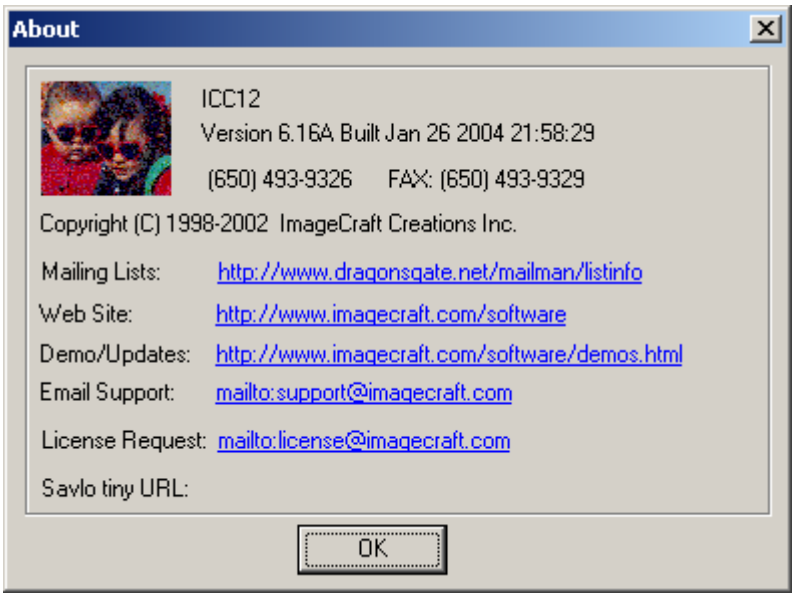
How to use ICC12 with NC12 and uBUG12

This document will show and demonstrate the use of ImageCraft ICC12 Latest ***Version 6*** with Technological Arts' NC12 module. uBUG12 GUI is used to erase and program FLASH after the compilation of a test program. Other method can be used to also erase and program the FLASH but in this example it will be the uBUG12.

This document assumes that the user is familiar with C and so will not teach how to program C here.

This document further assumes that the Serial Monitor has not been erased and is presently in the 9S12C32 MCU.

ImageCraft Links:



http://www.imagecraft.com/software/
http://www.ece.utexas.edu/%7Evalvano
http://www.dragonsgate.net/FAQ/cache/20.html
http://www.imagecraft.com/software/mdevtools.html
http://www.dragonsgate.net/mailman/listinfo

Technological Arts Links:

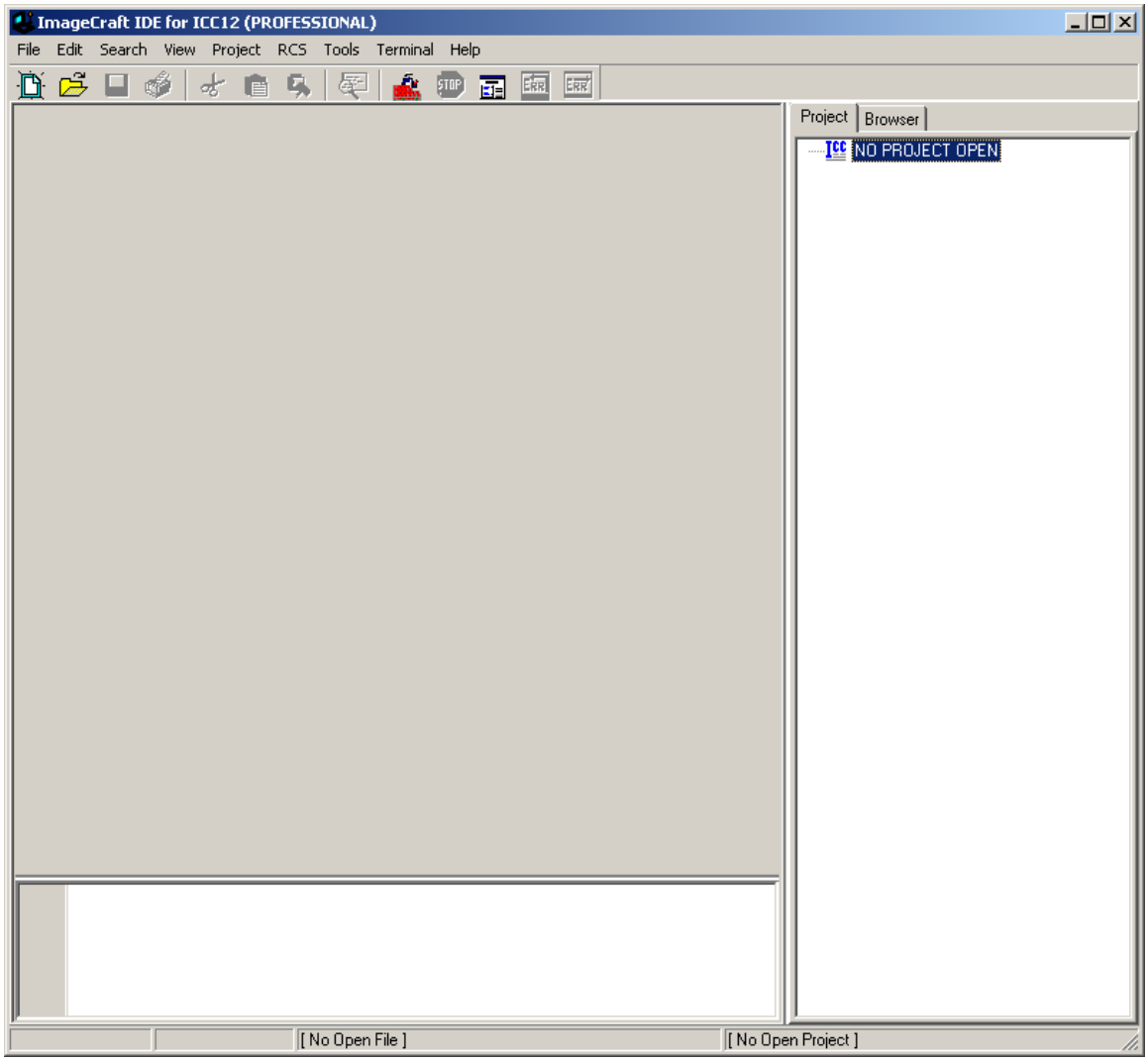
http://www.technologicalarts.com/myfiles/nc12.html
http://support.technologicalarts.ca/files/uBug12.zip

Getting Started:

Double click on the ICC12 icon. If a user has not read the ICC12 manual and just open the IDE one will wonder what to do next. Well wonder no more.

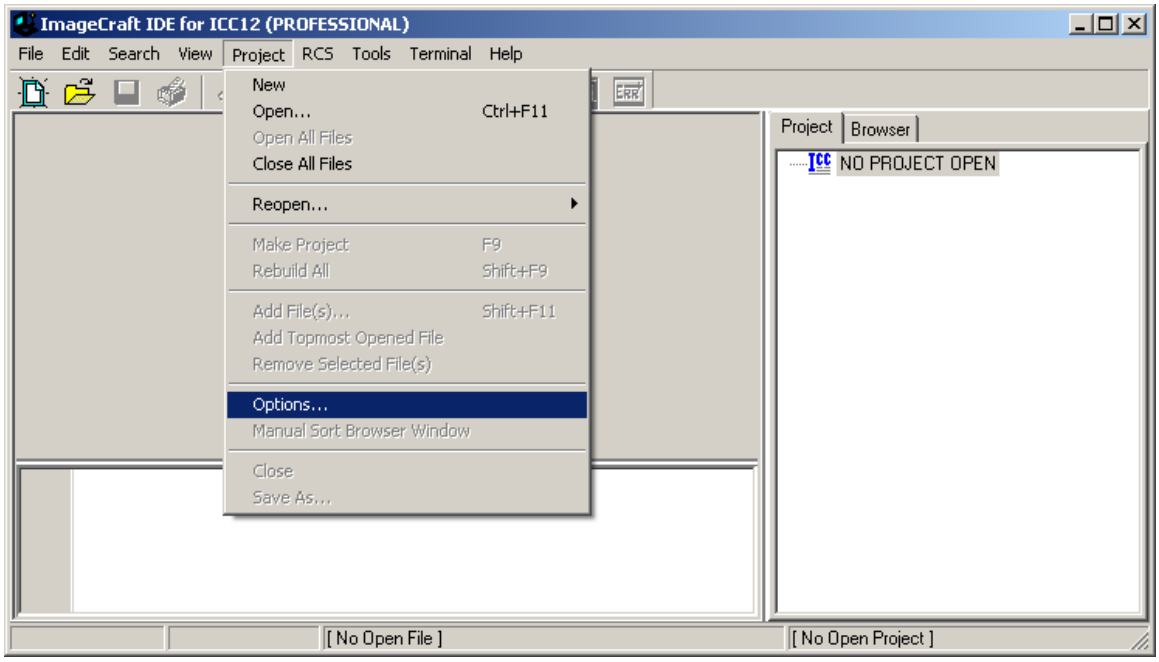
Note the 3 window panes. The top left most is greyed out and the right is the project window. The left bottom pane is where the error messages are displayed during compilation.

Before creating a new Project, the hardware target in the Compiler Options must be setup properly for the target MCU. This is to ensure that the compiler will setup the type of MCU the C program will compile for. In this example it is the NC12.

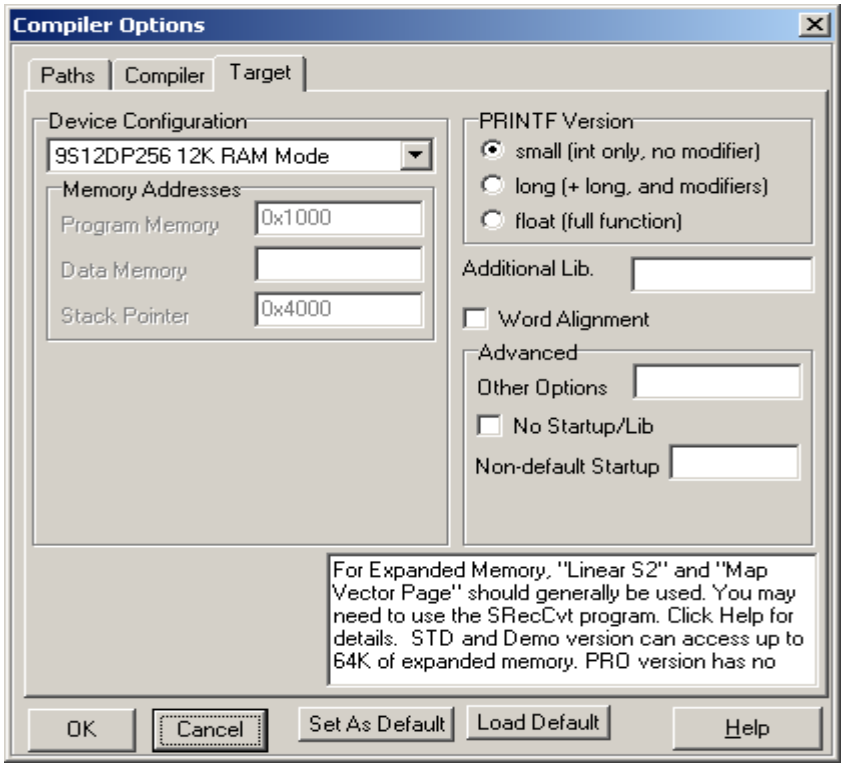


Compiler Setup:

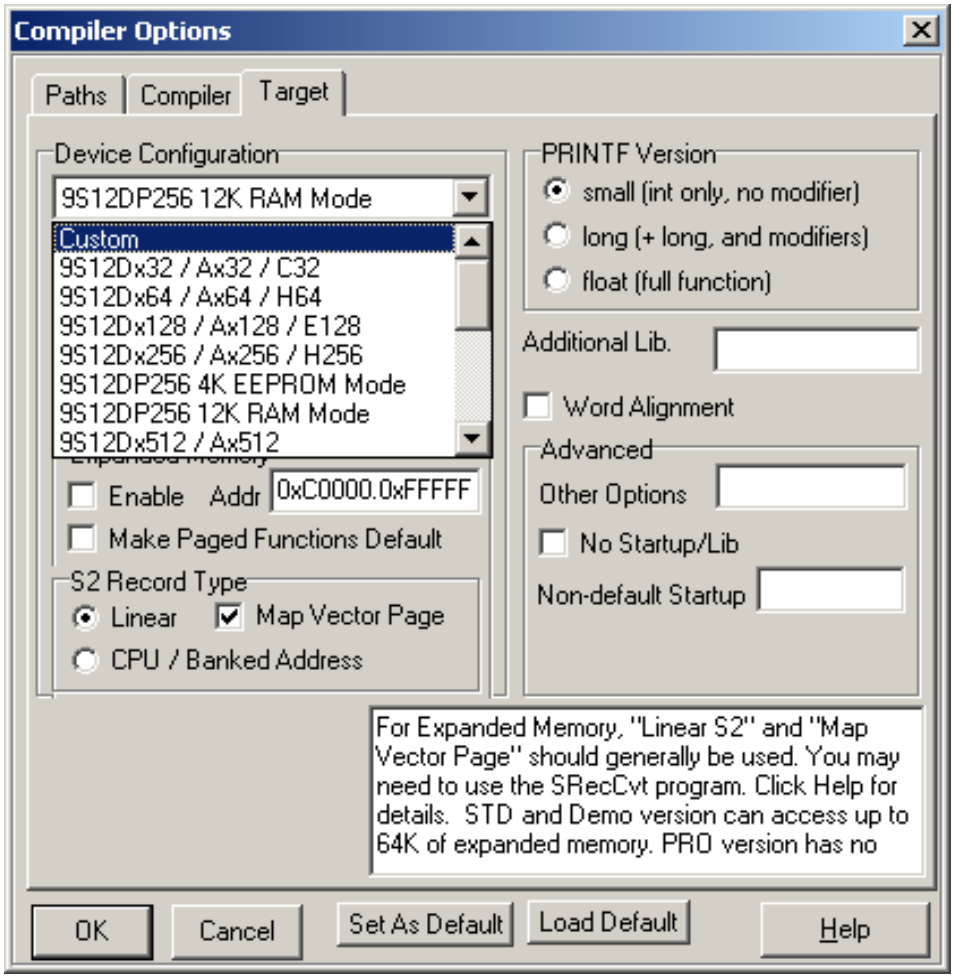
Click on Project Menu – Options – Target Tab.



Please note the Device Configuration. Click on the pull down arrow to change the device type.



Scroll up or down to select Custom as shown. Note that an 9S12C32 device Configuration does already exist. Unfortunately, the addresses are not setup properly with using uBUG12. Therefore the Custom configuration must be selected and the memory parameters are changed to reflect uBUG12 usage.



Custom Device Configuration:

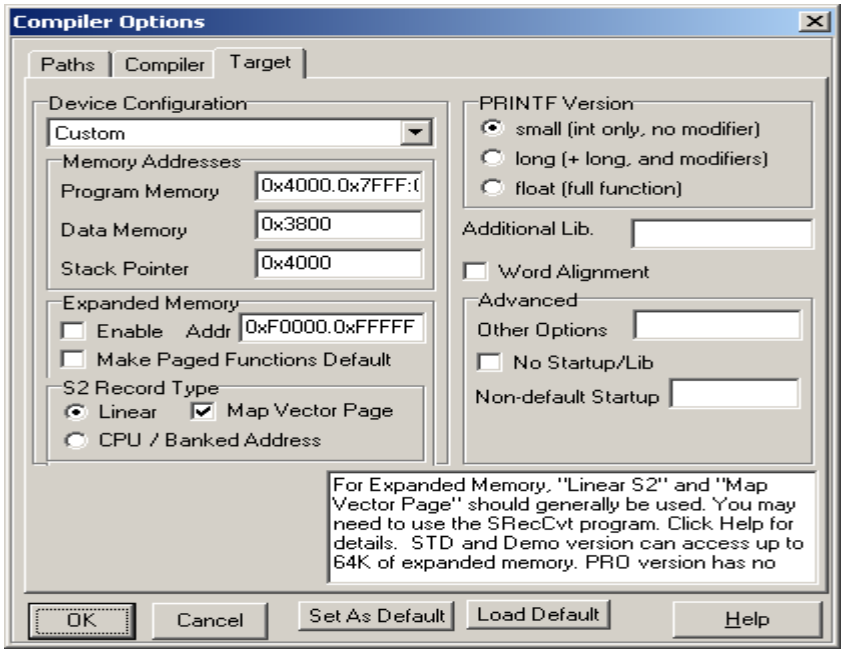
Program Memory: ***0x4000.0x7FFF:0xC000.0xFFFF***
Data Memory: ***0x3800***
Stack Pointer: ***0x4000***

Expanded Memory:

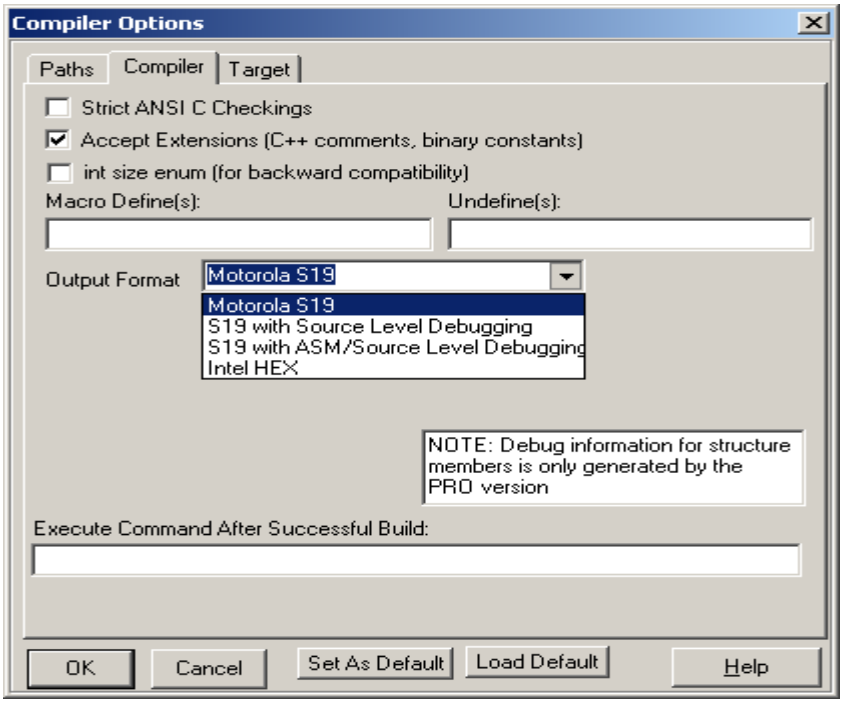
PPAGE $3E and $3F are the fixed memory and are allocated for
0x4000.0x7FFF:0xC000.0xFFFF

S2 Record Type:

Make sure to select Linear and the Map Vector Page is check marked.

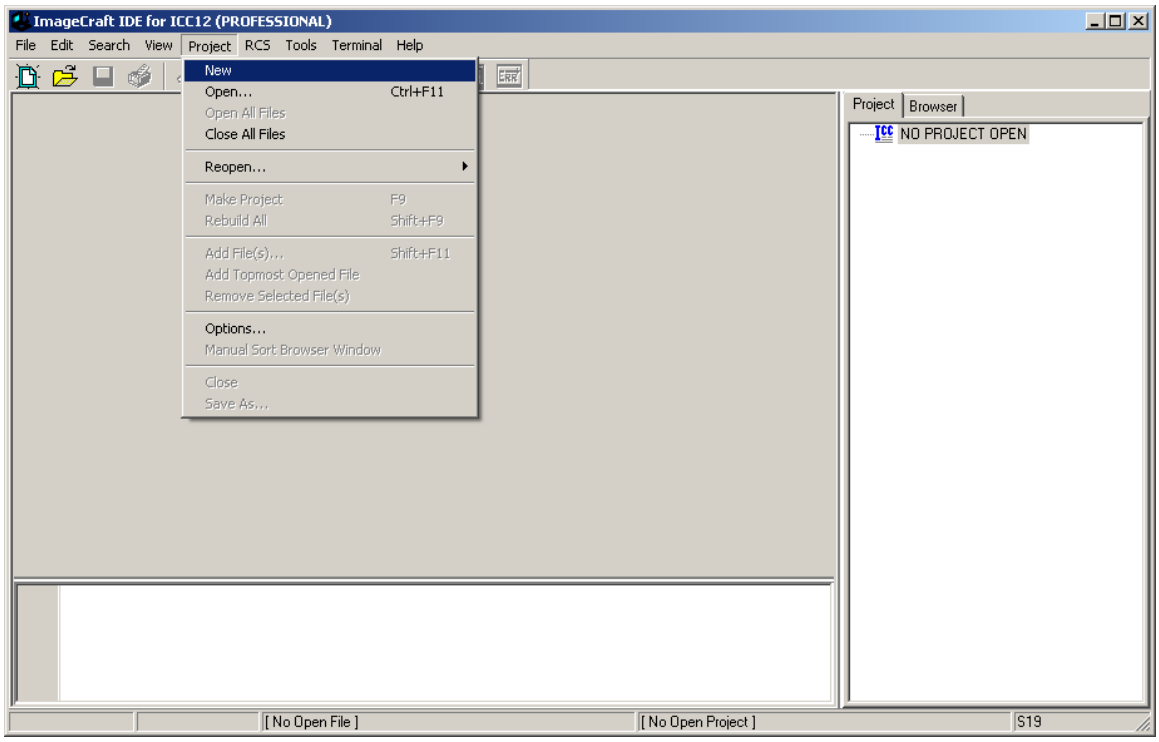


On the compiler tab there are several choices of S-record output as shown. Select which one that suits you.

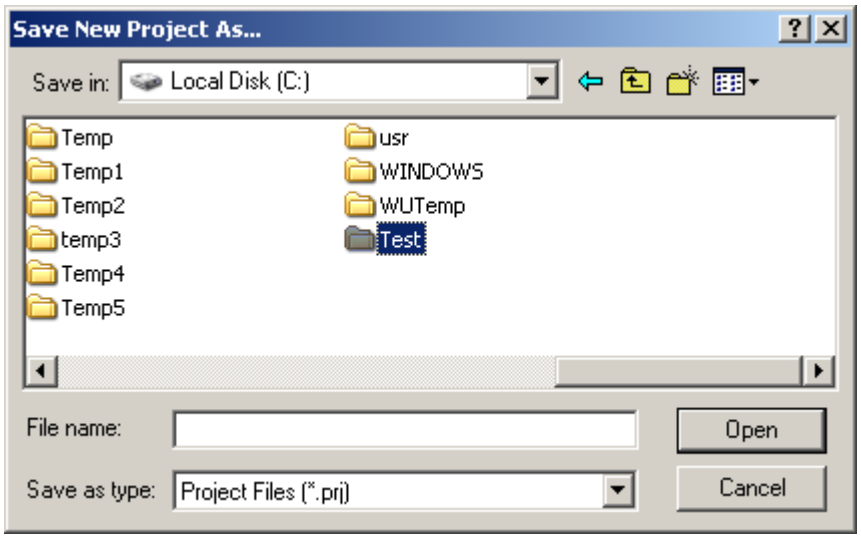


Starting a new Project:

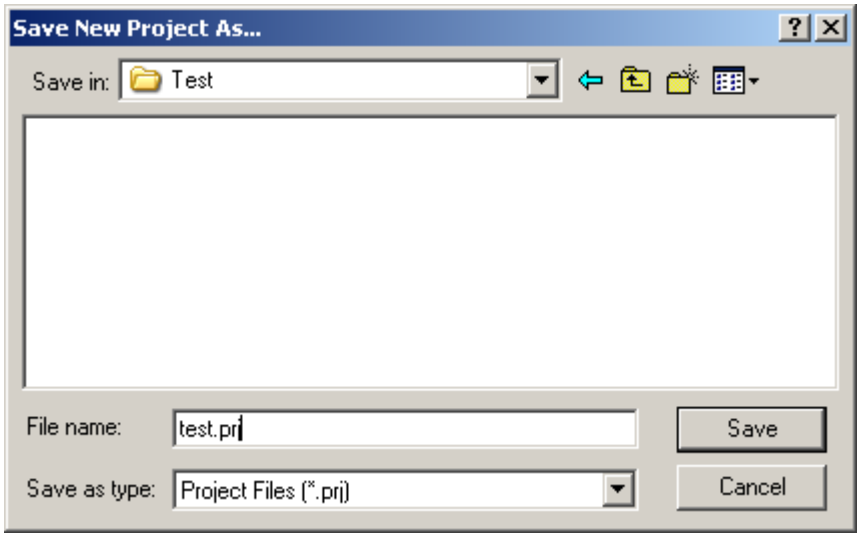
Once the compiler options are setup, a new project can be created. Click Project menu – New.



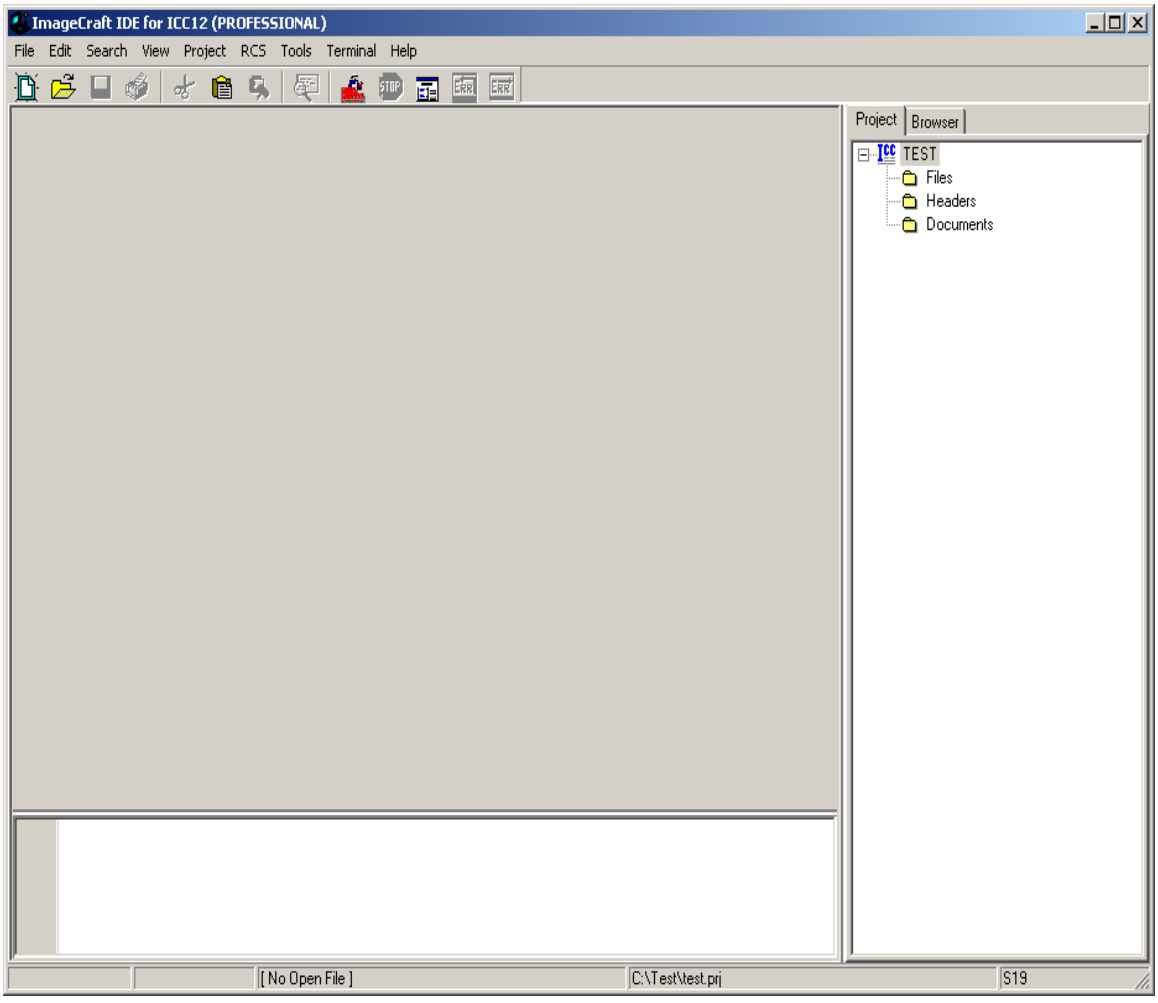
The ICC12 will prompt to save the new project. The user should decide whether to create a new directory to save the new project. In this example a new directory called ***Test*** is created and the file is saved as file ***test.prj***.



Type the filename as ***test.prj*** and click on the Save button.

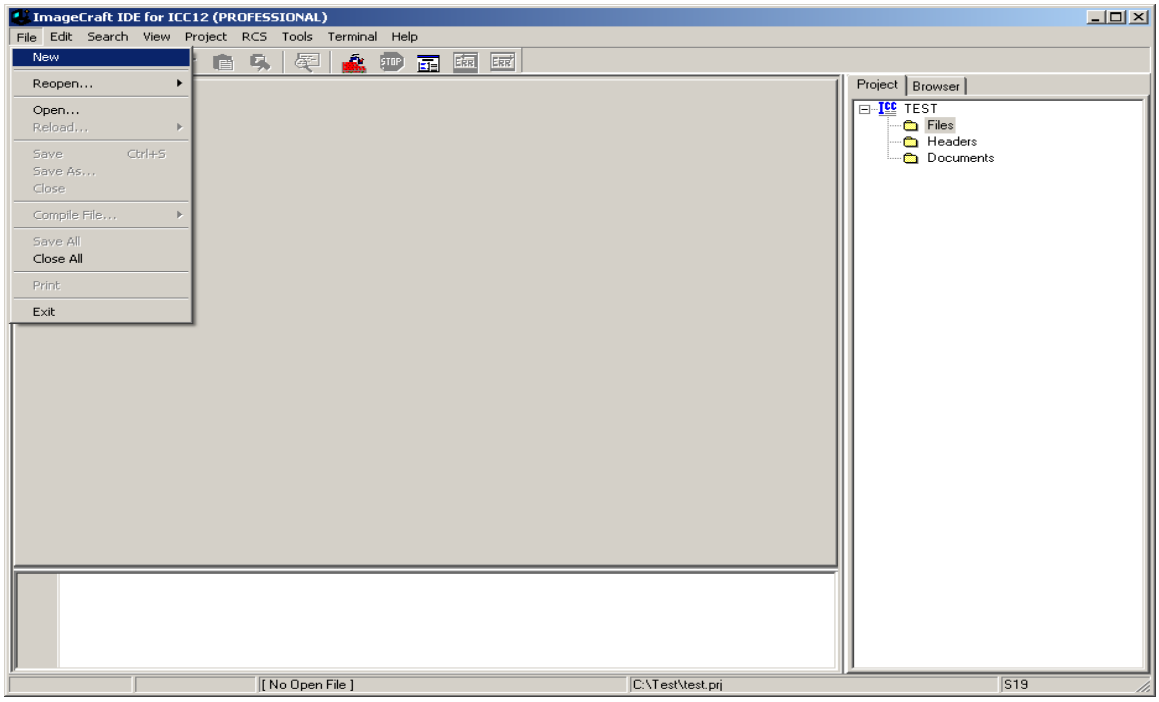


Note that the project window has changed to add Files, Headers and Documents.

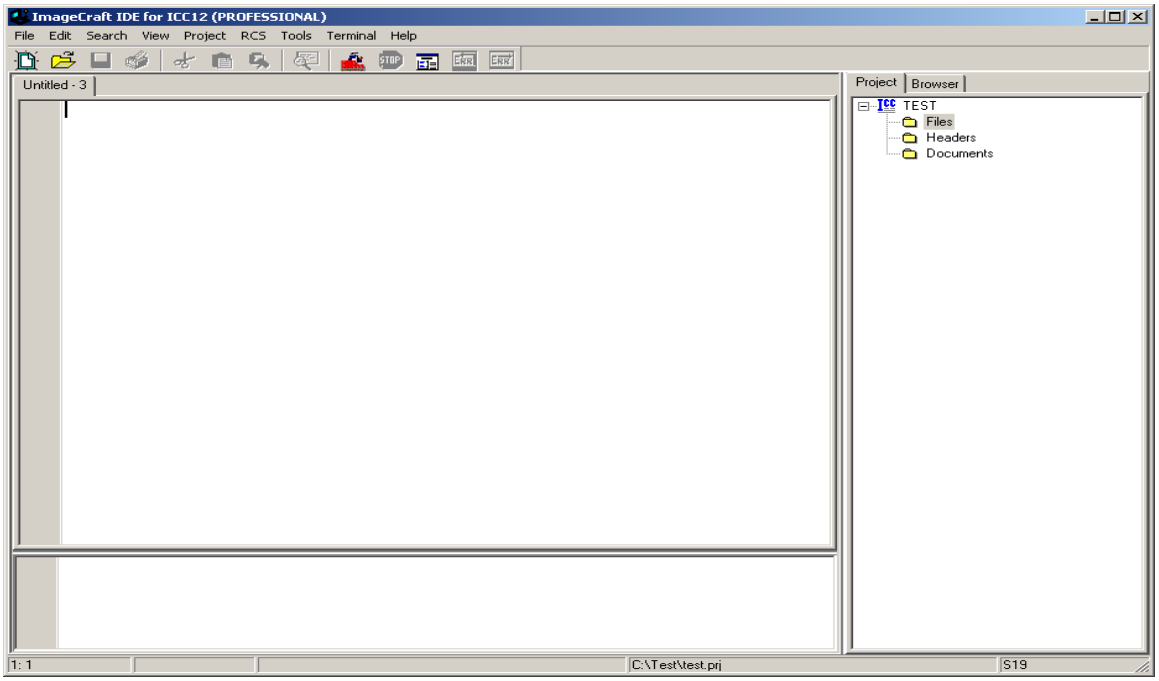


Creating a new file to the project:

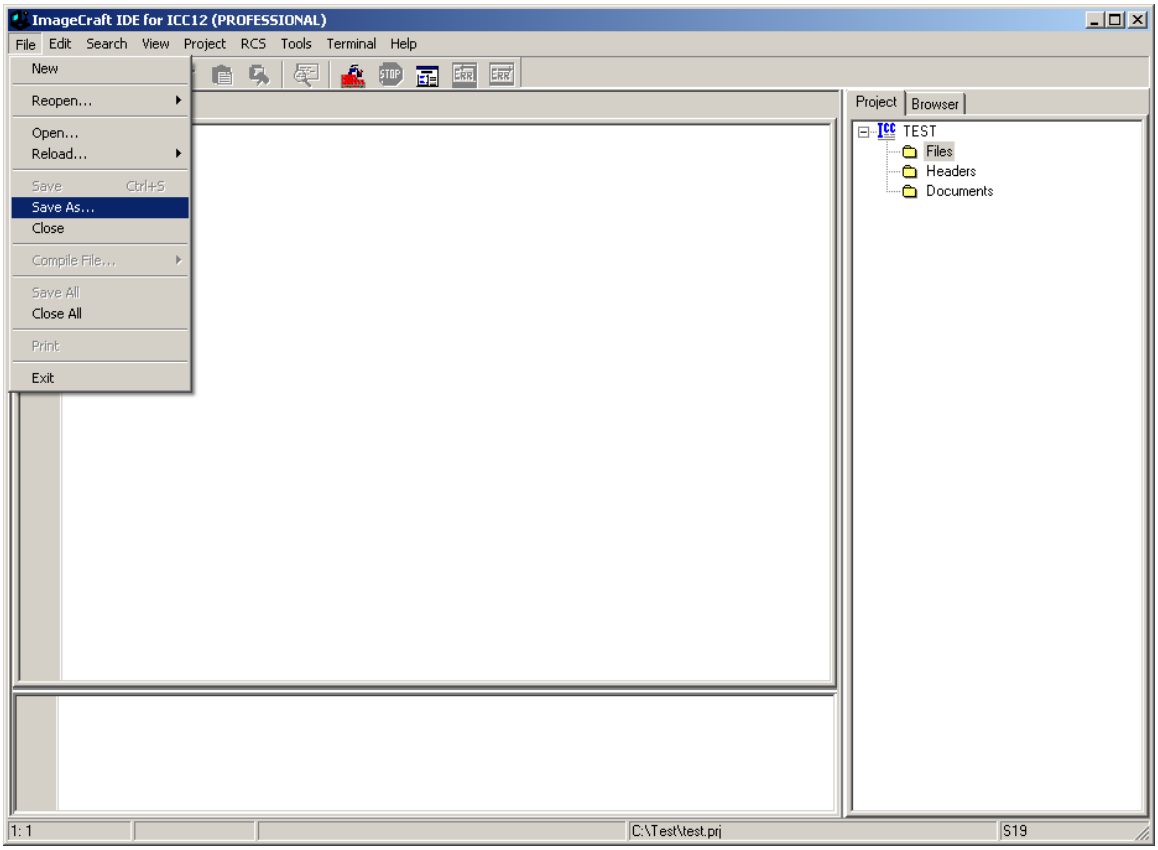
To add files to the project, click on the File menu – new as shown.



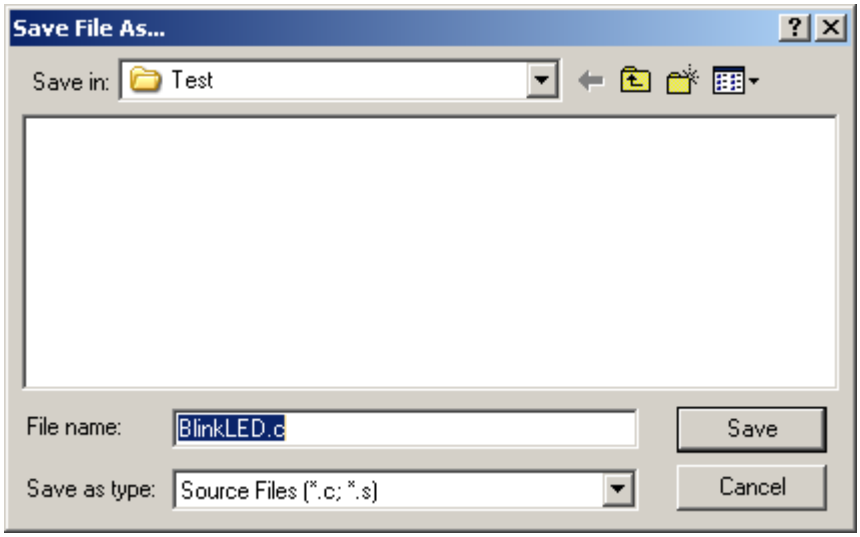
Note that ICC12 created an untitled file. Save the file as ***BlinkLED.C***.



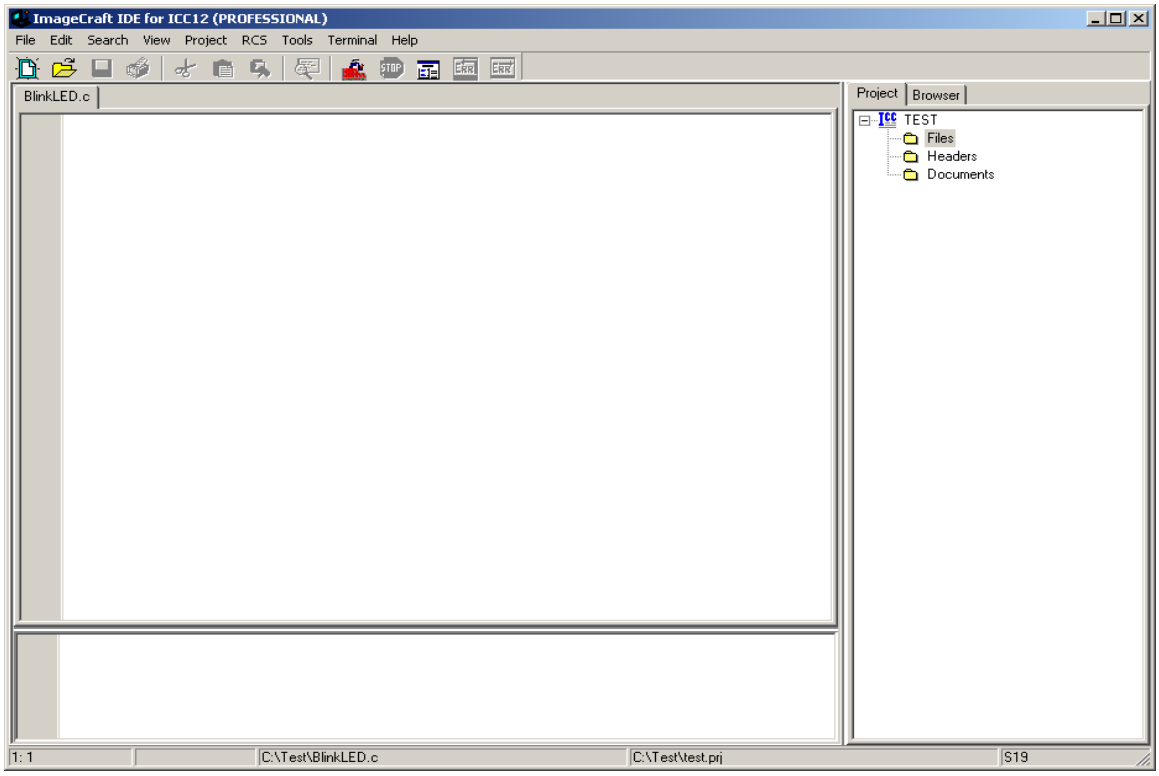
To save, click on File menu – Save As



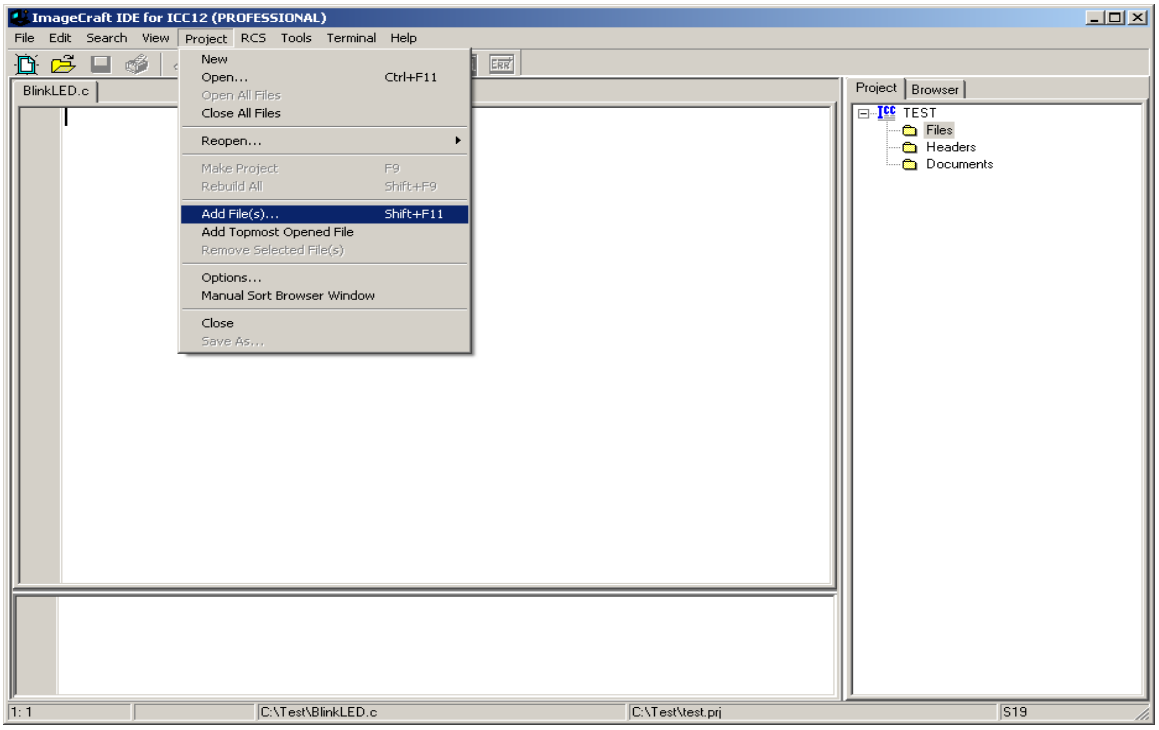
ICC12 will open an explorer window to help save the file. Type BlinkLED.c then press the save button.



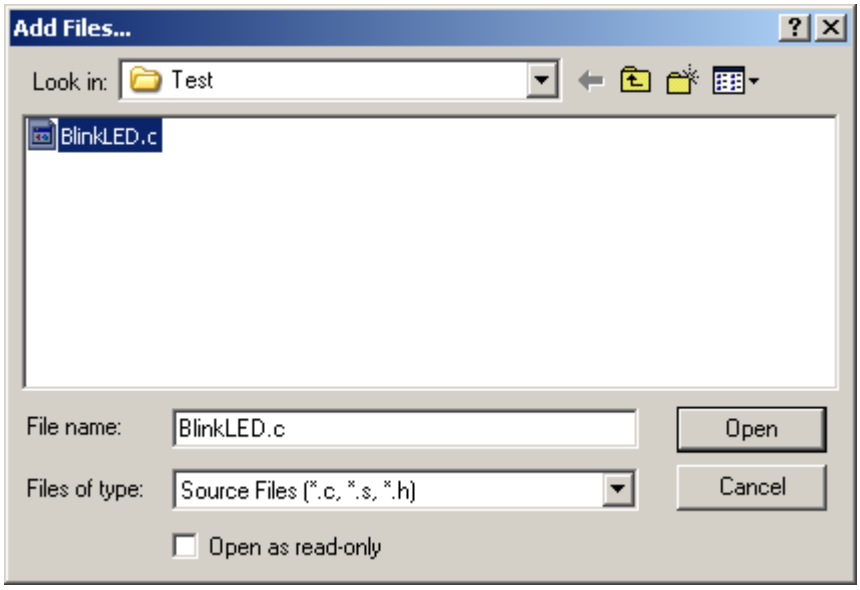
Note that ICC12 has renamed the file to BlinkLED.c.



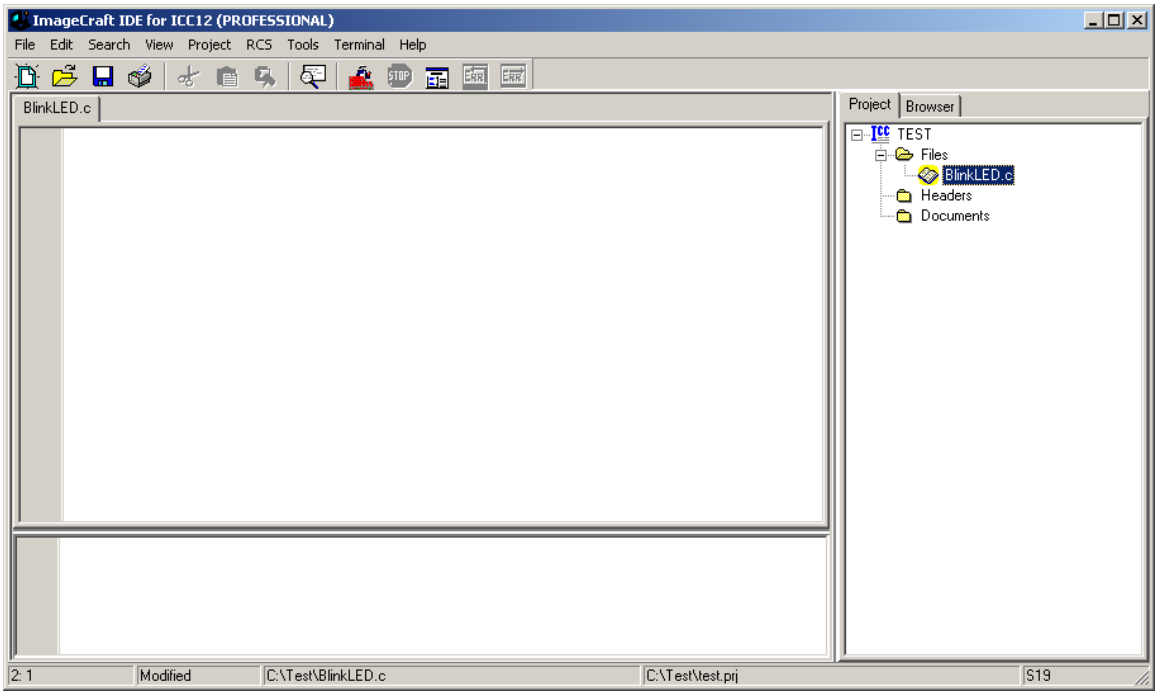
To add BlinkLED.c to the Project, click on the Project menu – Add File(s)



ICC12 will open an explorer window to help and locate the file of interest.

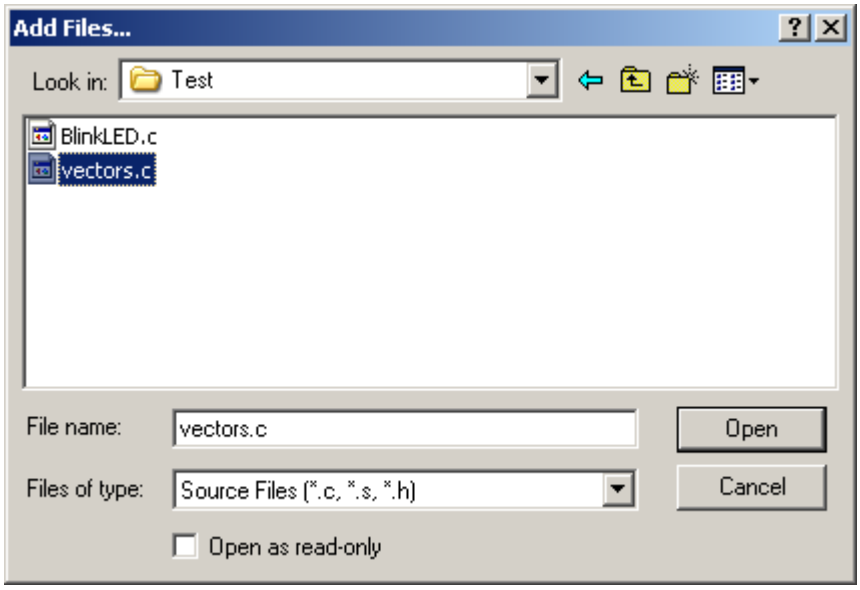


Note that the right window pane has changed to include BlinkLED.c under the Files Project.

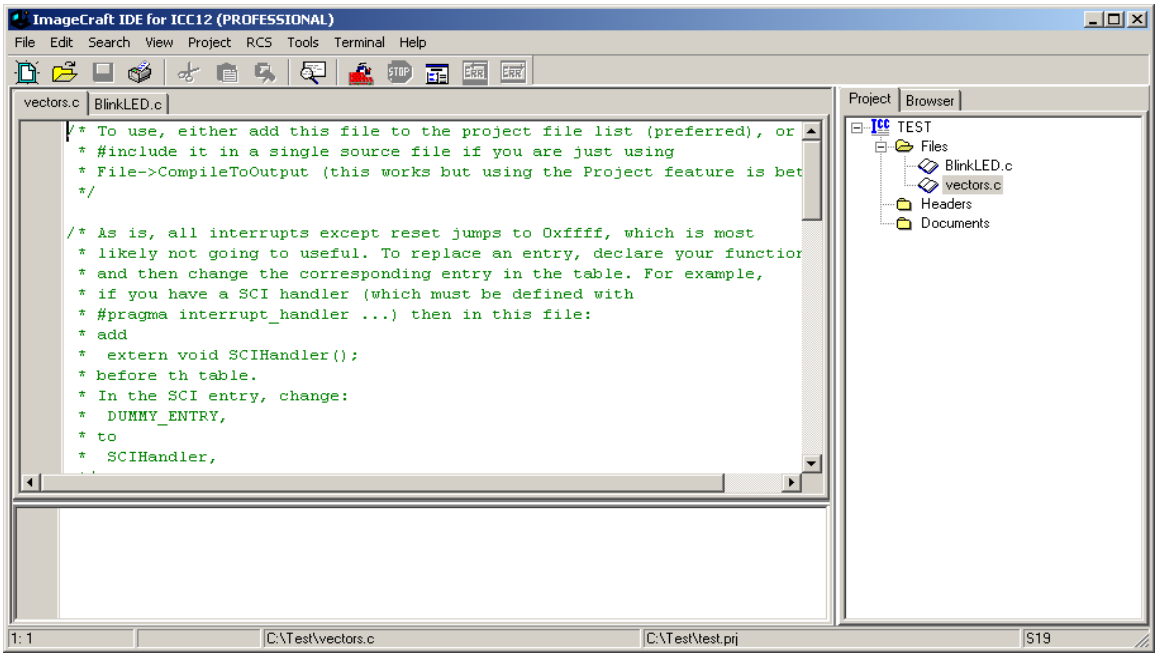


Locate ***vectors.c*** and copy file to Test directory. The major reason why this must be done is because of project to project dependency. It is not good to keep editing a single ***vectors.c*** if other projects are using this same file. It becomes a problem to keep track of the changes made to the different projects.

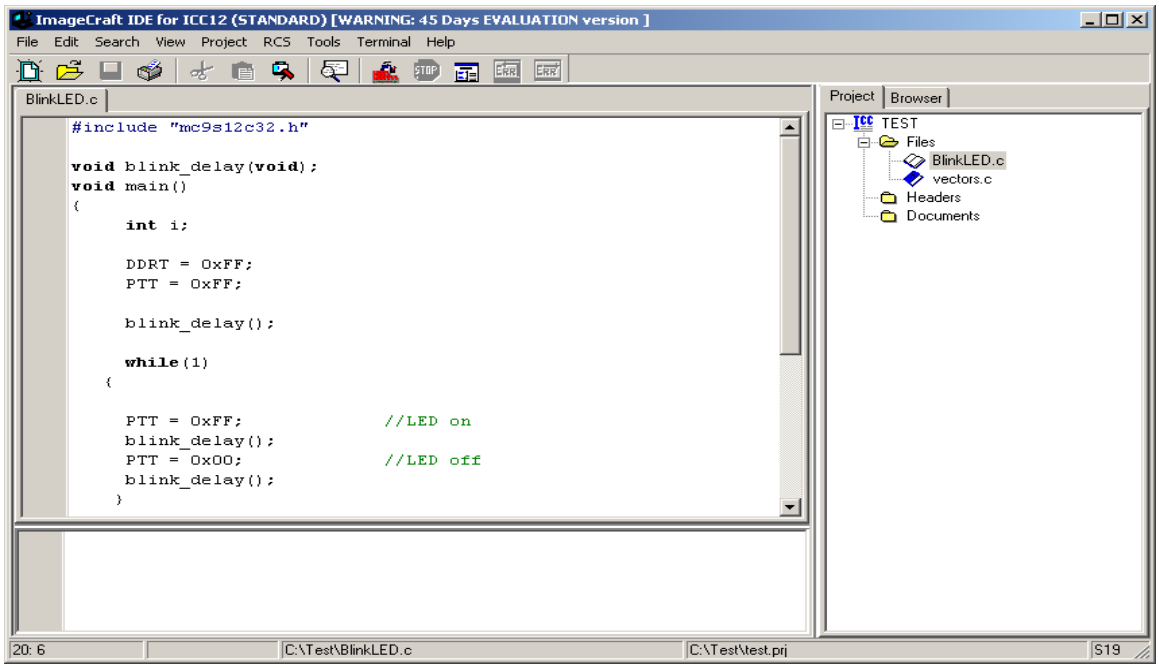
To add ***vectors.c*** to the Project, click on the Project menu – Add File(s)



Note that ICC12 has changed to include ***vectors.c*** It is important to note that the **vectors.c** was written for the 68HC912B32 and 812A4 MCUs. One should edit the file to include other ISR addresses for the 9S12C32. This example uses the file as is.



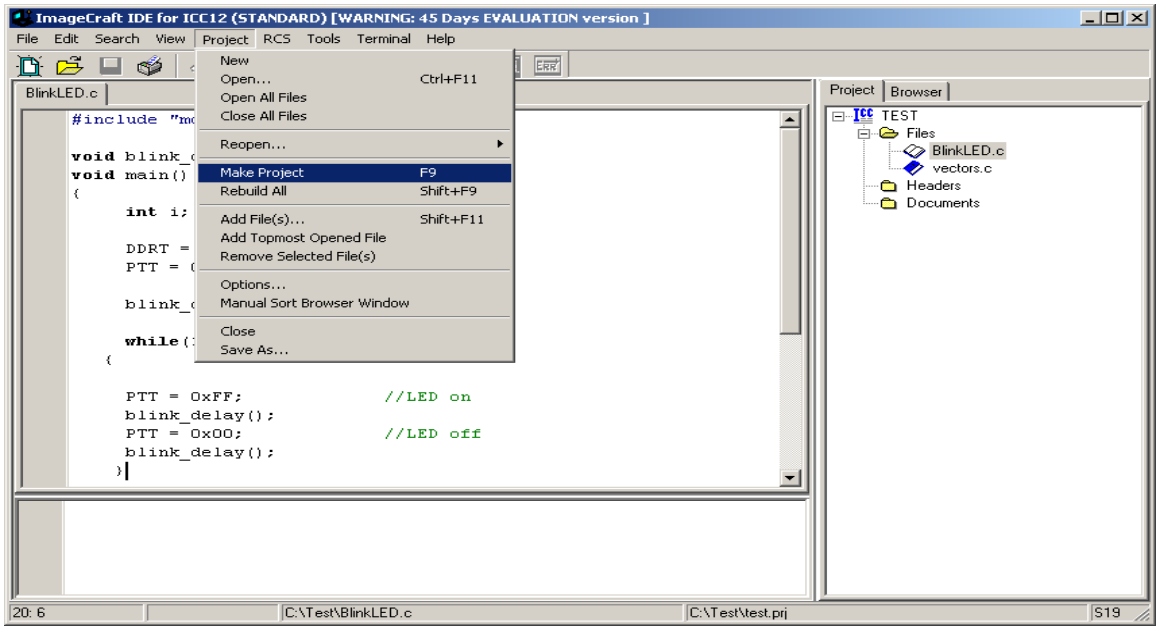
Write the codes below into BlinkLED.c file. Once it is written we can then compile/make/build the code.



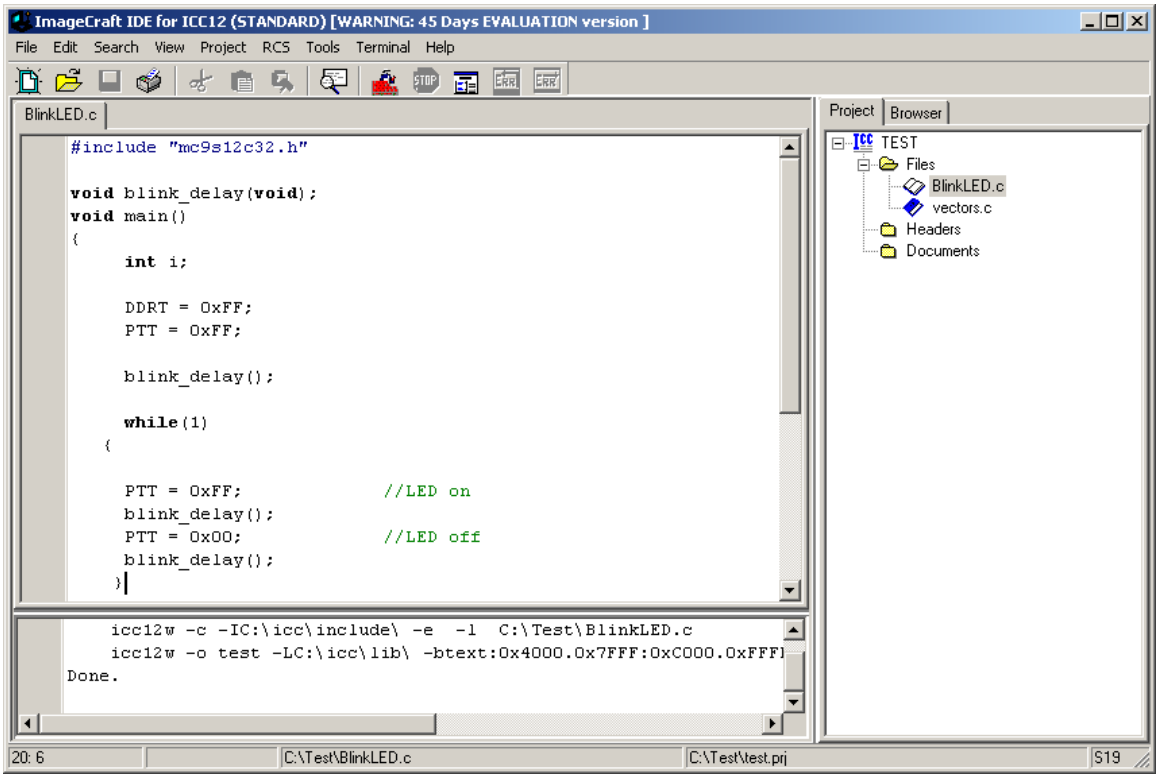

```
#include "mc9s12c32.h"

void blink_delay(void);
void main()
{
       int i;

       DDRT = 0xFF;
       PTT = 0xFF;

       blink_delay();

       while(1)
   {

       PTT = 0xFF;                    //LED on
       blink_delay();
       PTT = 0x00;                    //LED off
       blink_delay();
      }

}

void blink_delay(void)
{
 int i;
      for(i=0;i<64000;i++)
      {                               ;
      }
}
```

Compiling/Build/Make the file:

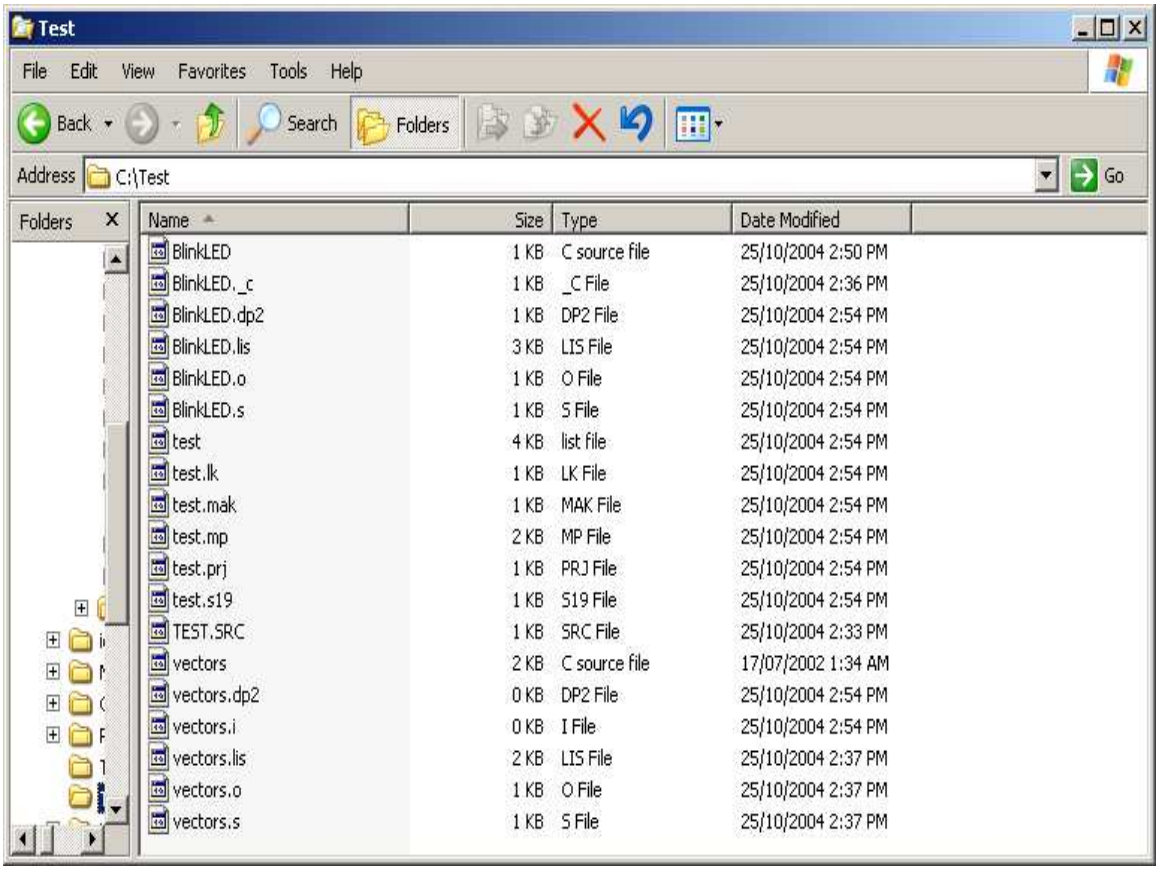
To make the file click Project menu – make project as shown.



Note the bottom window pane will show messages to display how the build progressed. Any errors, if any, are shown in this window. The build was without error so we can progress to erasing and programming the 9S12C32.



Note the other extraneous files are created after a make.



Using WordPad to check the content of test.s19

```
S10E4000CF400016407087CE38008EC1
S110400B380027056A000820F6CE4075CD68
S111401838008E40752706180A307020F51601
S1074026402A20FE0A
S110402A34B7751B9EC6FF7B0242C6FF7BA8
S1104037024016405220EC6FF7B024016C8
S11140444052790240164052220F0B757303DEA
S111405234B7751B9ECC00006C1E2007EC1EBC
S1114060C300016C1EEC1E8CFA0025F2B7574B
S105406E303DDF
S111FFD0FFFFFFFFFFFFFFFFFFFFFFFF2D
S111FFDEFFFFFFFFFFFFFFFFFFFFFFFF1F
S111FFECFFFFFFFFFFFFFFFFFFFFFFFF11
S109FFFAFFFFFFFF4000C1
S10840701D0016073DD0
S9034000BC
```

If one looks closely at the S-record one can see the S1 records are programmed into e ***$4000 - $7FFF*** and ***$C000 - $FFFF*** memory blocks. The main reason has to do with uBUG12. It adjusts the memory configuration into 2 separate blocks.

Below is the vector address as will be program into $C000 to $FFFF block.

S111FFD0FFFFFFFFFFFFFFFFFFFFFFFFFF2D
S111FFDEFFFFFFFFFFFFFFFFFFFFFFFFFF1F
S111FFECFFFFFFFFFFFFFFFFFFFFFFFFFF11
S109FFFAFFFFFFFF4000C1

Note the address at $FFFE = $4000, the start of code from power up or RESET.

S109FFFAFFFFFFFF***4000***C1

Below is the start of code address as will be program into the $4000 to $7FFF block.

S10E4000CF400016407087CE38008EC1
S110400B380027056A000820F6CE4075CD68
S111401838008E40752706180A307020F51601
S1074026402A20FE0A
S110402A34B7751B9EC6FF7B0242C6FF7BA8
S1104037024016405220EC6FF7B024016C8
S1114044405279024016405220F0B757303DEA
S111405234B7751B9ECC00006C1E2007EC1EBC
S1114060C300016C1EEC1E8CFA0025F2B7574B
S105406E303DDF

Below is the start of code address

S10E**4000**CF400016407087CE38008EC1

Programming:

This document assumes that the Serial monitor is not erased and is present on the NC12.

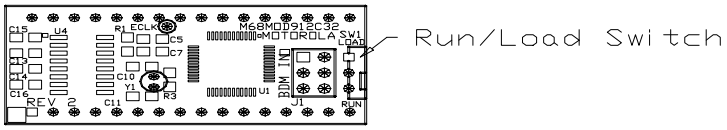
Download uBUG12 from Technological Arts which can be found at the link below
http://support.technologicalarts.ca/files/uBug12.zip

or from the CD that came with the NC12 kit.

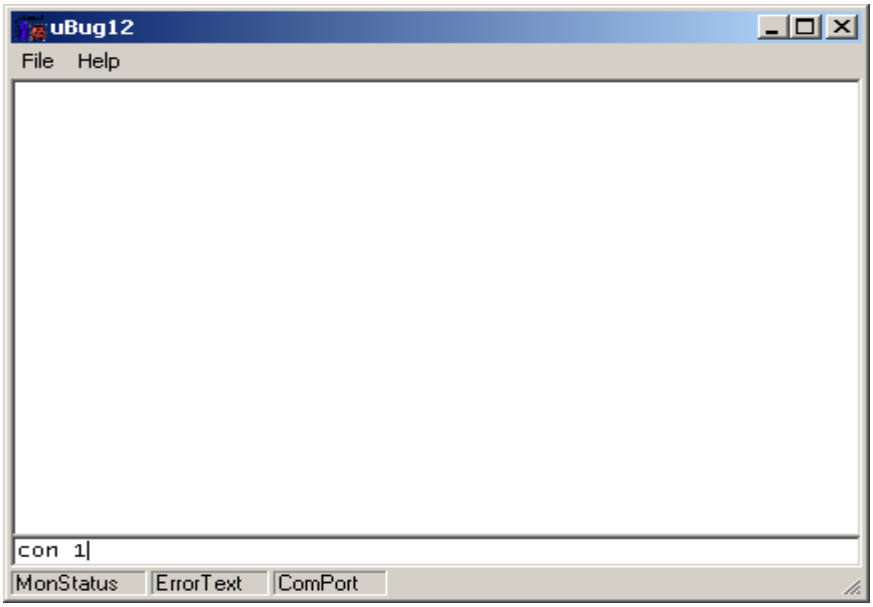
For windows 98 users the .NET framework must be installed before running uBUG12. The MS site link is

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

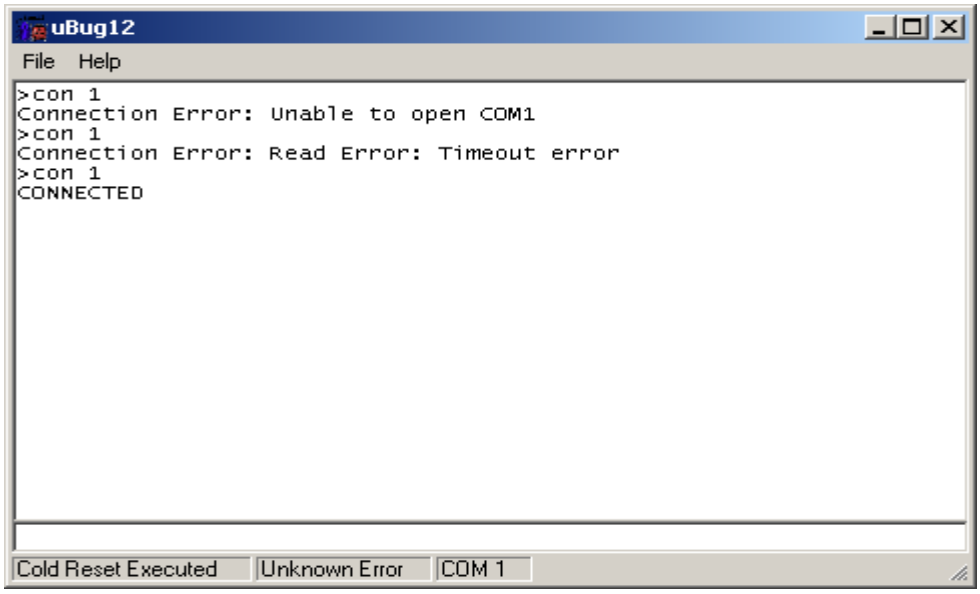
Switch the Run/Load switch to Load position and apply power to the board.



Double click on uBUG12 icon to execute program.

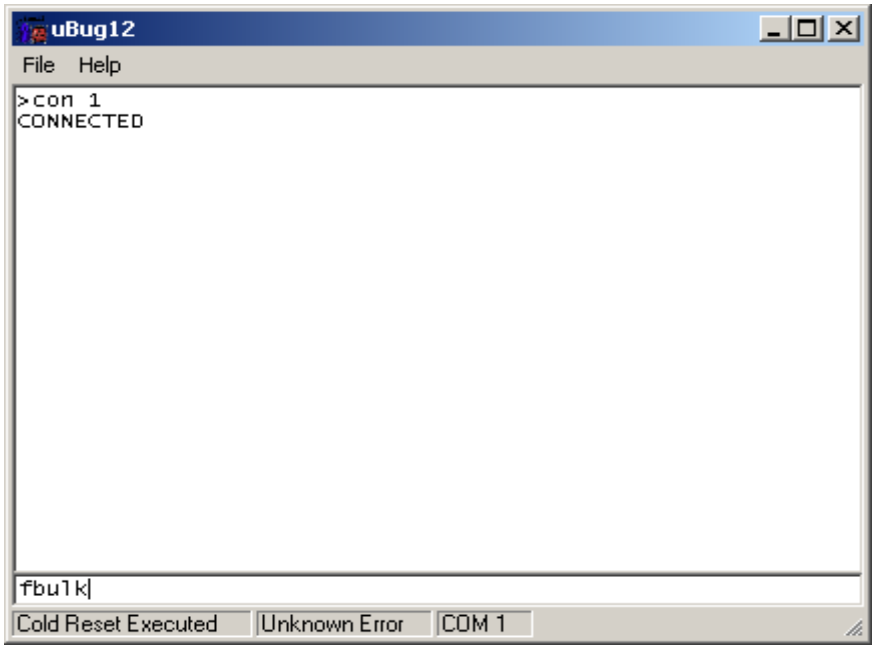


In the command bar type ***con 1*** for COM 1 or ***con 2*** for COM 2. A ***CONNECTED*** message will appear to indicate that a connection between PC and NC12 has been established.

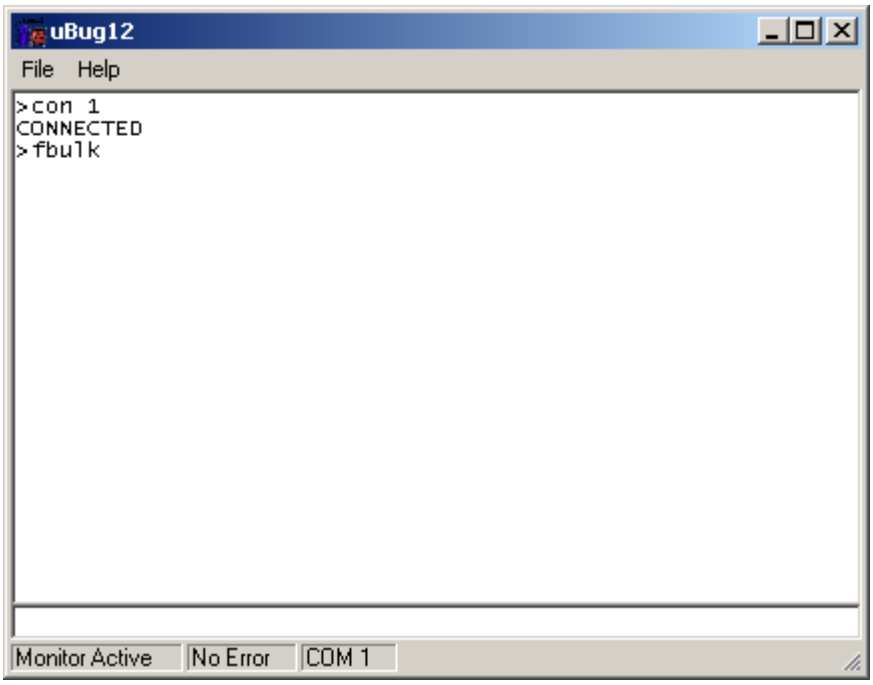


2 possible errors can occur:
Connection Error: Unable to open COM1 <- Another application is using the COM port

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 to the wrong COM port.



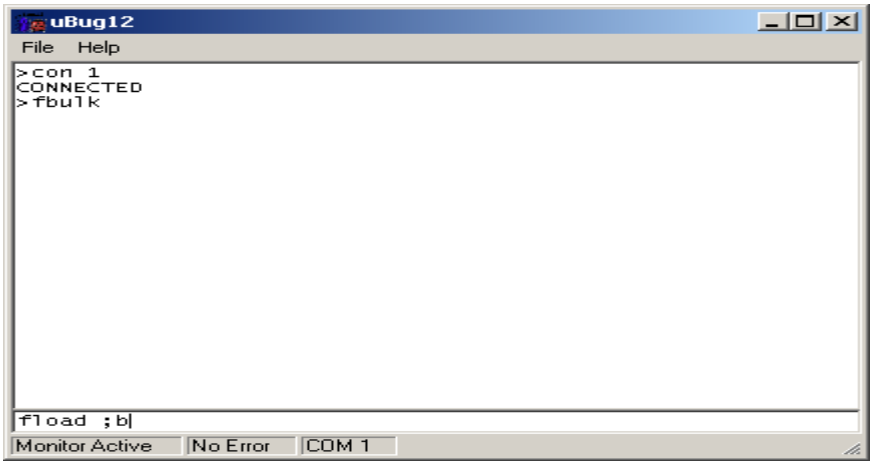
Make sure to erase the FLASH memory by typing the command ***FBULK***.

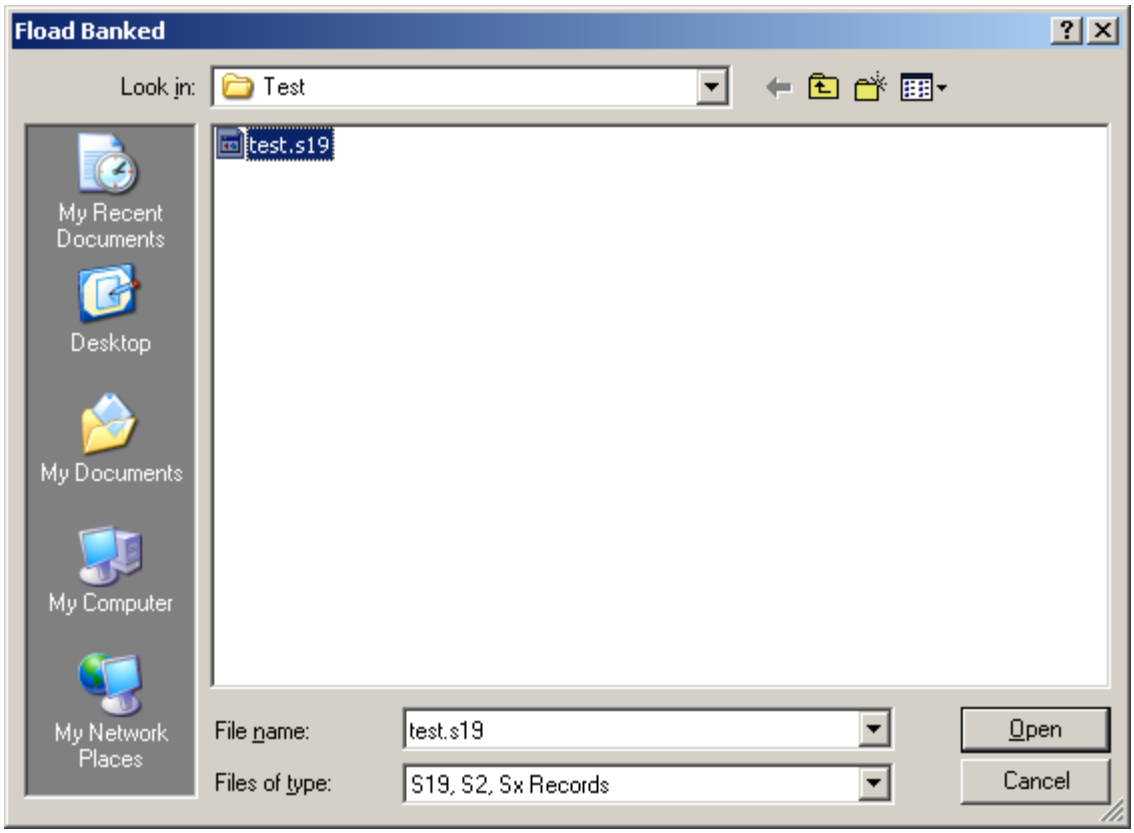


To program, type the command ***FLOAD ;B*** for banked S19, S2, SX and formatted S19 (went thru SrecCVT program) records. For non-banked S2 the command is ***FLOAD***.

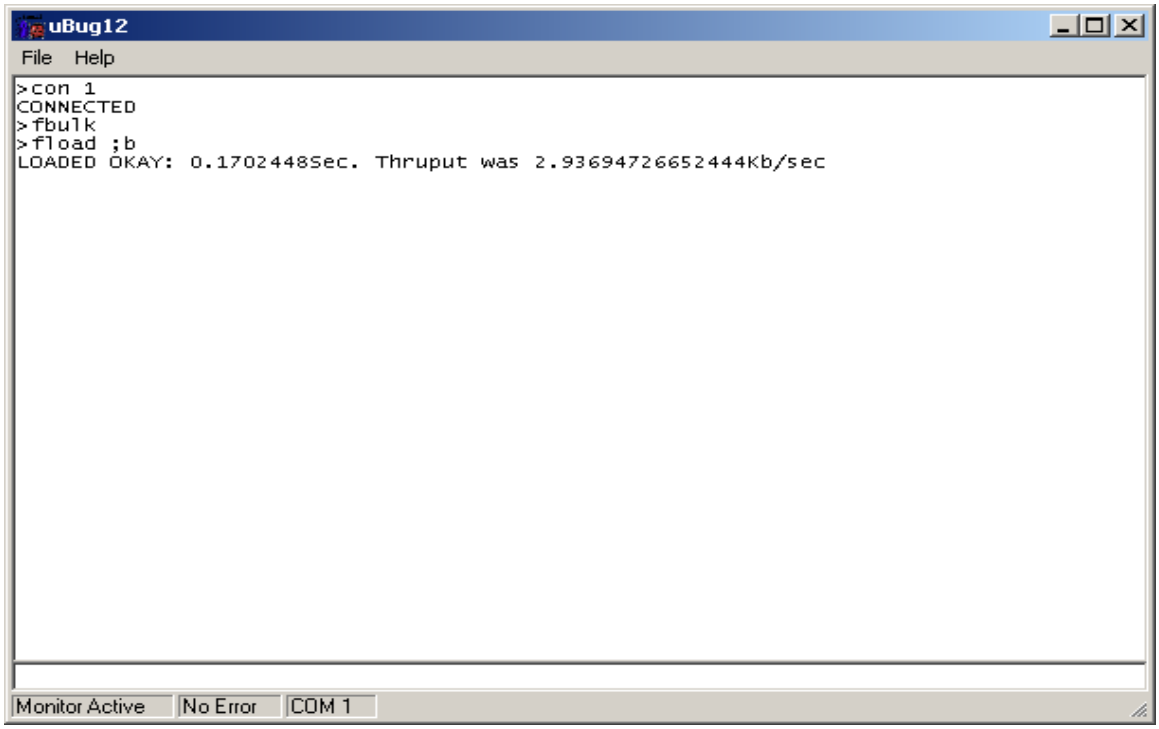
Uploading Banked S-record:

The command to upload banked S-record is ***FLOAD ;B***. Note to include the ***;B*** option to let uBUG12 know that the S-record is banked. Users should be familiar with the differences between S19, SX, S2. See Appendix A for S-record explanation.





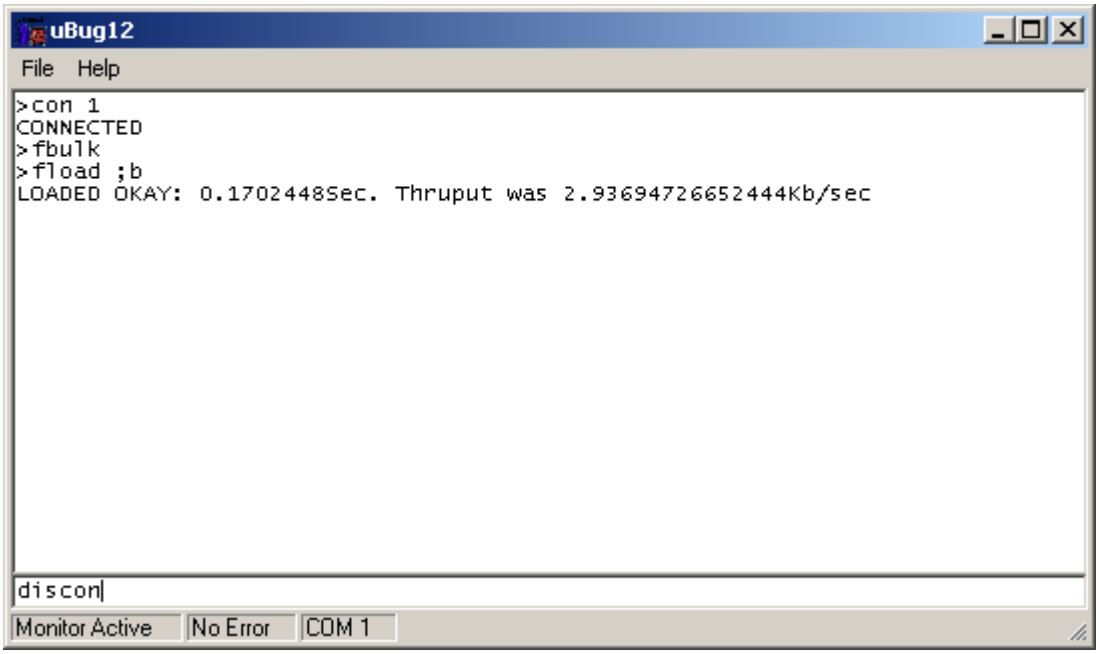
Double click on the file to initiate upload.



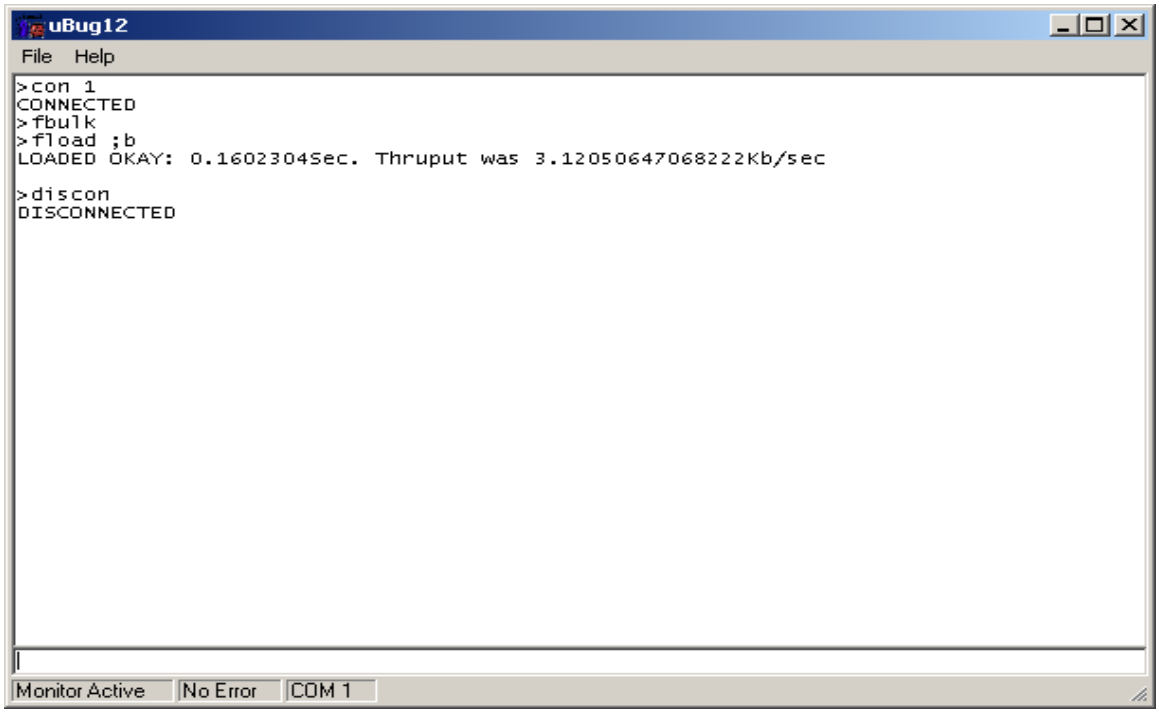
A good upload will show ***LOADED OKAY*** messages.

After successful programming slide the Run/Load switch to Run and press the reset button. The application is blinking the LED connected to at PP0

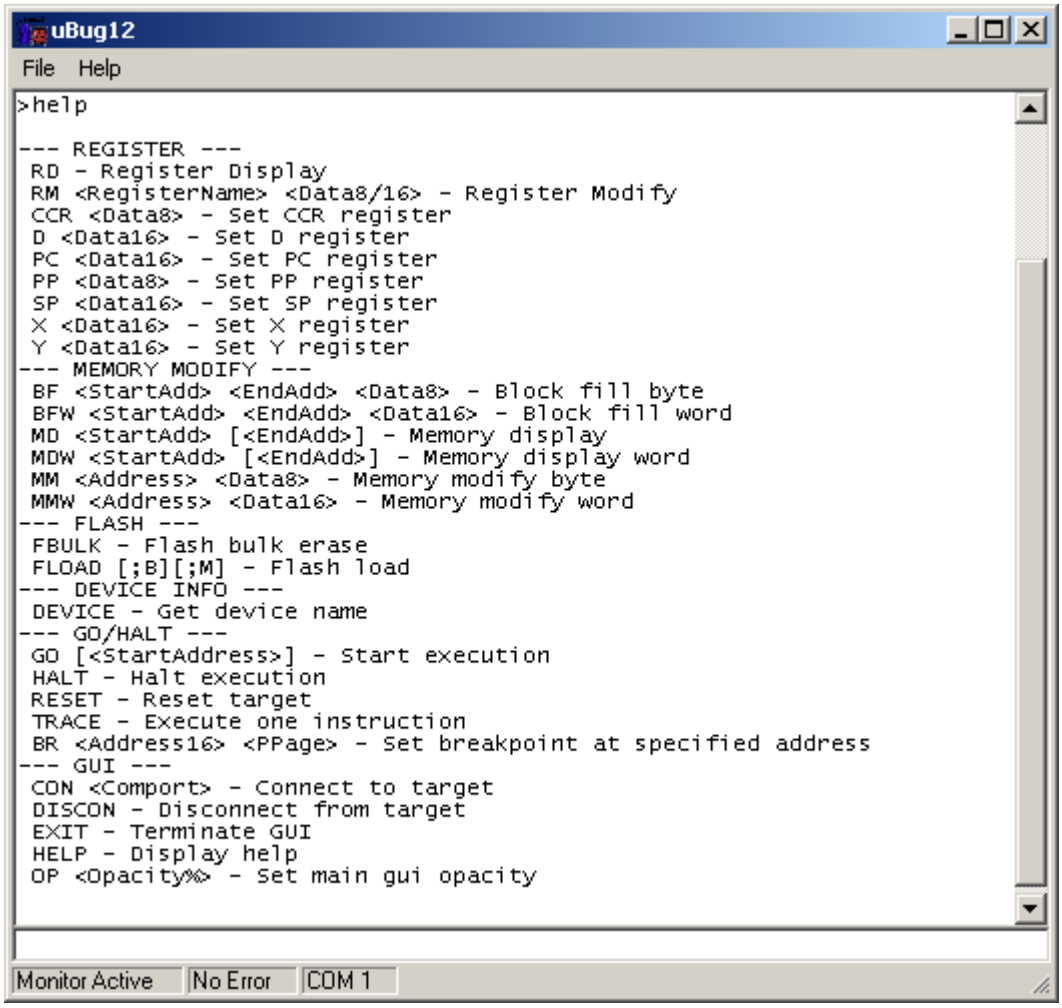
To disconnect uBUG12 GUI to serial port type the command is ***discon***.



Disconnected message will appear to indicate that the serial is ready to be use by another application like ***HyperTerm*** or ***Tera Term***.



Other uBUG12 commands are available by typing the ***help*** command.



The command help are self explanatory but one should try them out to be familiar with their usage and capability.

Note: For NC12 families

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

SerialMon moves the internal RAM to $3800 - $3FFF. Make sure your code stack begins at $4000 or at $3F80 if you intend to use uBUG12 as limited de-bugger. To make sure this is done you can add the code below to your code.

```
STACK           equ   $3F80        ;Stack at below Ubug12

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

The last thing to note is that uBUG12 enabled the PLL during Load mode. In Run mode the PLL is NOT enabled as the user maynot want this feature enabled. The code below shows how to enable the PLL.

```
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

; 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

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

Examining the content of the MCU:

Below is the S-record (test.s19) that was programmed into the NC12.

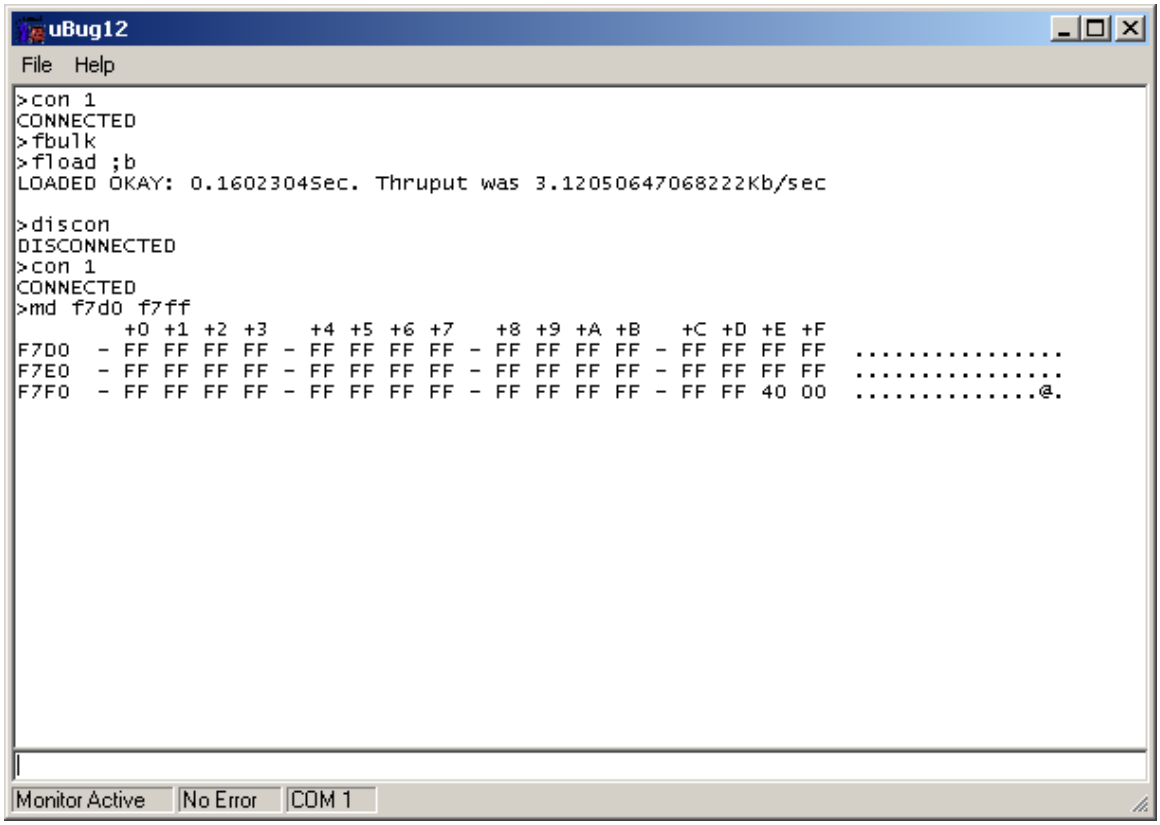
```
S10E4000CF400016407087CE38008EC1
S110400B380027056A000820F6CE4075CD68
S111401838008E40752706180A307020F51601
S1074026402A20FE0A
S110402A34B7751B9EC6FF7B0242C6FF7BA8
S1104037024016405220EC6FF7B024016C8
S11140444052790240164052220F0B757303DEA
S111405234B7751B9ECC00006C1E2007EC1EBC
S1114060C300016C1EEC1E8CFA0025F2B7574B
S105406E303DDF
S111FFD0FFFFFFFFFFFFFFFFFFFFFFFFFF2D
S111FFDEFFFFFFFFFFFFFFFFFFFFFFFFFF1F
S111FFECFFFFFFFFFFFFFFFFFFFFFFFFFF11
S109FFFAFFFFFFFF4000C1
S10840701D0016073DD0
S9034000BC
```

These are the area of interest where the S-record is programmed to. Let us start with the interrupt vector area. As stated previously, Serial Monitor re-locates the vector address at below $F800.

S111FFD0FFFFFFFFFFFFFFFFFFFFFFFFFF2D
S111FFDEFFFFFFFFFFFFFFFFFFFFFFFFFF1F
S111FFECFFFFFFFFFFFFFFFFFFFFFFFFFF11
S109FFFAFFFFFFFF4000C1

Use uBUG12 to memory dump from $F7D0 to $F7FF by the command ***md f7d0 f7ff.***

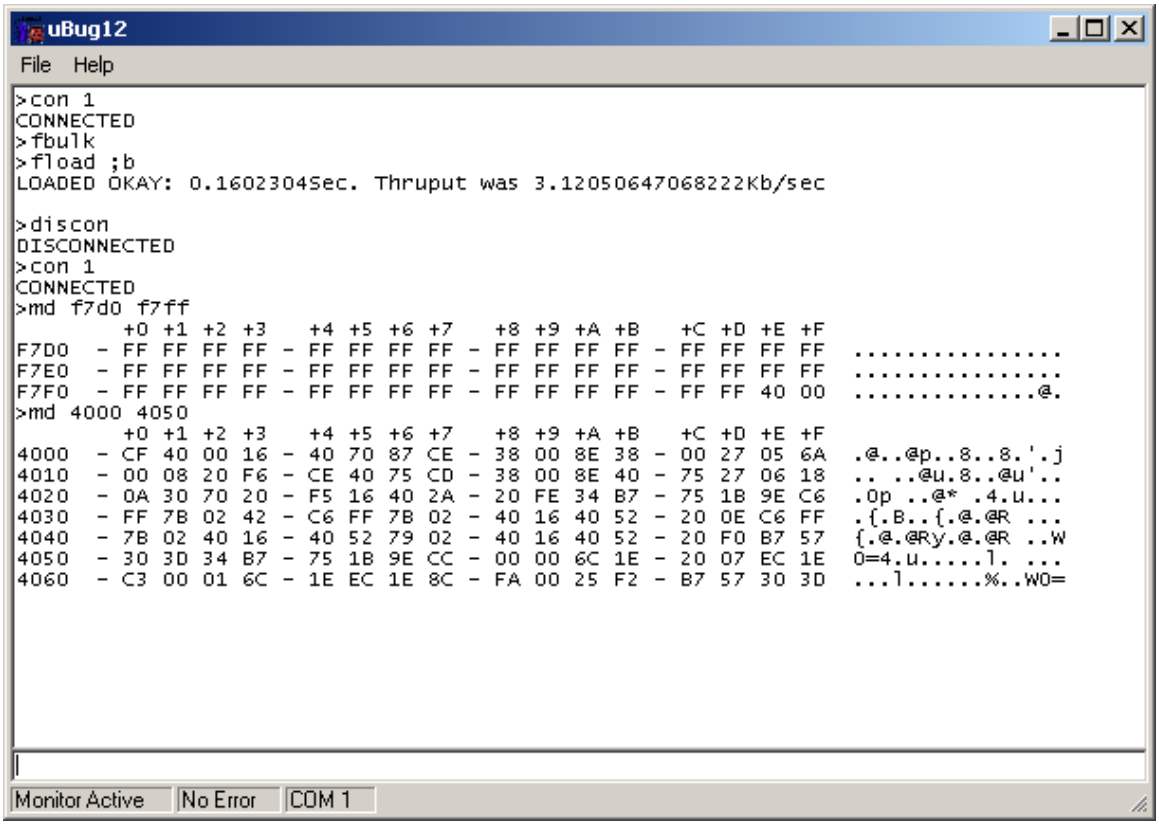
Note the content of the memory address at $F7FE:$F7FF is $4000, the RESET vector.



The content of address beginning at $4000 to $4050

```
S10E4000CF400016407087CE38008EC1
S110400B380027056A000820F6CE4075CD68
S111401838008E40752706180A307020F51601
S1074026402A20FE0A
S110402A34B7751B9EC6FF7B0242C6FF7BA8
S1104037024016405220EC6FF7B024016C8
S1114044405279024016405220F0B757303DEA
S111405234B7751B9ECC00006C1E2007EC1EBC
S1114060C300016C1EEC1E8CFA0025F2B7574B
S105406E303DDF
```

Note that the memory dump is the same as the S-record.



This concludes the use of ICC12 from erasing and programming FLASH with using uBUG12.

Appendix A

Motorola S-records

NAME
srec - S-record file and record format

DESCRIPTION

An S-record file consists of a sequence of specially formatted ASCII character strings. An S-record will be less than or equal to 78 bytes in length.

The order of S-records within a file is of no significance and no particular order may be assumed.

The general format of an S-record follows:

```
+------------------//------------------//----------------------+
| type | count | address  |            data           | checksum |
+------------------//------------------//----------------------+
```

type -- A char[2] field. These characters describe the type of record (S0, S1, S2, S3, S5, S7, S8, or S9).

count -- A char[2] field. These characters when paired and interpreted as a hexadecimal value, display the count of remaining character pairs in the record.

address -- A char[4,6, or 8] field. These characters grouped and interpreted as a hexadecimal value, display the address at which the data field is to be loaded into memory. The length of the field depends on the number of bytes necessary to hold the address. A 2-byte address uses 4 characters, a 3-byte address uses 6 characters, and a 4-byte address uses 8 characters.

data -- A char [0-64] field. These characters when paired and interpreted as hexadecimal values represent the memory loadable data or descriptive information.

checksum -- A char[2] field. These characters when paired and interpreted as a hexadecimal value display the least significant byte of the ones complement of the sum of the byte values represented by the pairs of characters making up the count, the address, and the data fields.

Each record is terminated with a line feed. If any additional or different record terminator(s) or delay characters are needed during transmission to the target system it is the responsibility of the transmitting program to provide them.

S0 Record. The type of record is 'S0' (0x5330). The address field is unused and will be filled with zeros (0x0000). The header information within the data field is divided into the following subfields.

mname is char[20] and is the module name.
ver is char[2] and is the version number.
rev is char[2] and is the revision number.
description is char[0-36] and is a text comment.
Each of the subfields is composed of ASCII bytes whose associated characters, when paired, represent one byte hexadecimal values in the case of the version and revision numbers, or represent the hexadecimal values of the ASCII characters comprising the module name and description.

S1 Record. The type of record field is 'S1' (0x5331). The address field is intrepreted as a 2-byte address. The data field is composed of memory loadable data.

S2 Record. The type of record field is 'S2' (0x5332). The address field is intrepreted as a 3-byte address. The data field is composed of memory loadable data.

S3 Record. The type of record field is 'S3' (0x5333). The address field is intrepreted as a 4-byte address. The data field is composed of memory loadable data.

S5 Record. The type of record field is 'S5' (0x5335). The address field is intrepreted as a 2-byte value and contains the count of S1, S2, and S3 records previously transmitted. There is no data field.

S7 Record. The type of record field is 'S7' (0x5337). The address field contains the starting execution address and is intrepreted as 4-byte address. There is no data field.

S8 Record. The type of record field is 'S8' (0x5338). The address field contains the starting execution address and is intrepreted as 3-byte address. There is no data field.

S9 Record. The type of record field is 'S9' (0x5339). The address field contains the starting execution address and is intrepreted as 2-byte address. There is no data field.

EXAMPLE

Shown below is a typical S-record format file.
S00600004844521B
S1130000285F245F2212226A000424290008237C2A

```
S11300100002000800082629001853812341001813
S113002041E900084E42234300182342000824A952
S107003000144ED492
S5030004F8
S9030000FC
```

The file consists of one S0 record, four S1 records, one S5 record and an S9 record.

The S0 record is comprised as follows:

- S0 S-record type S0, indicating it is a header record.
- 06 Hexadecimal 06 (decimal 6), indicating that six character pairs (or ASCII bytes) follow.
- 00 00 Four character 2-byte address field, zeroes in this example.
- 48 44 52 ASCII H, D, and R - "HDR".
- 1B The checksum.

The first S1 record is comprised as follows:

- S1 S-record type S1, indicating it is a data record to be loaded at a 2-byte address.
- 13 Hexadecimal 13 (decimal 19), indicating that nineteen character pairs, representing a 2 byte address, 16 bytes of binary data, and a 1 byte checksum, follow.
- 00 00 Four character 2-byte address field; hexidecimal address 0x0000, where the data which follows is to be loaded.
- 28 5F 24 5F 22 12 22 6A 00 04 24 29 00 08 23 7C Sixteen character pairs representing the actual binary data.
- 2A The checksum.

The second and third S1 records each contain 0x13 (19) character pairs and are ended with checksums of 13 and 52, respectively. The fourth S1 record contains 07 character pairs and has a checksum of 92.

The S5 record is comprised as follows:

- S5 S-record type S5, indicating it is a count record indicating the number of S1 records
- 03 Hexadecimal 03 (decimal 3), indicating that three character pairs follow.
- 00 04 Hexadecimal 0004 (decimal 4), indicating that there are four data records previous to this record.
- F8 The checksum.

The S9 record is comprised as follows:

- S9 S-record type S9, indicating it is a termination record.
- 03 Hexadecimal 03 (decimal 3), indicating that three character pairs follow.
- 00 00 The address field, hexadecimal 0 (decimal 0) indicating the starting execution address.
- FC The checksum.

Instructor Notes

- There isn't any evidence that Motorola ever has made use of the header information within the data field of the S0 record, as described above. This must have been used by some third party vendors.
- This is the only place that a 78-byte limit on total record length or 64-byte limit on data length is documented. These values shouldn't be trusted for the general case.
- The count field can have values in the range of 0x3 (2 bytes of address + 1 byte checksum = 3, a not very useful record) to 0xff; this is the count of remaining character pairs, including checksum.
- If you write code to convert S-Records, you should always assume that a record can be as long as 514 (decimal) characters in length (255 * 2 = 510, plus 4 characters for the type and count fields), plus any terminating character(s). That is, in establishing an input buffer in C, you would declare it to be an array of 515 chars, thus leaving room for the terminating null character.