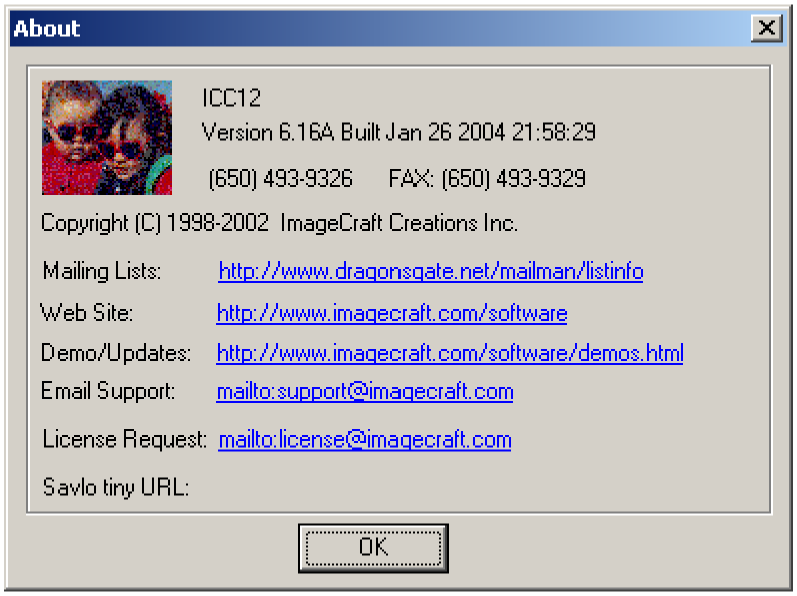
How to use ICC12 with Adapt912DT60 and FLASH Loader

This document will show and demonstrate the use of ImageCraft ICC12 Latest ***Version 6*** with Technological Arts' Adapt912DT60 module.

The FLASH Loader written by Technological Arts and can be found inside the starter package disk. The file can also be downloaded from http://www.interlog.com/~techart/myfiles/files/12disk4.zip website. The FLASHLoader will be used here 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 FLASH Loader.

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

ImageCraft Links:



ICC12
Version 6.16A Built Jan 26 2004 21:58:29
(650) 493-9326 FAX: (650) 493-9329
Copyright (C) 1998-2002 ImageCraft Creations Inc.

Mailing Lists:	http://www.dragonsgate.net/mailman/listinfo
Web Site:	http://www.imagecraft.com/software
Demo/Updates:	http://www.imagecraft.com/software/demos.html
Email Support:	mailto:support@imagecraft.com
License Request:	mailto:license@imagecraft.com
Savlo tiny URL:	

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:

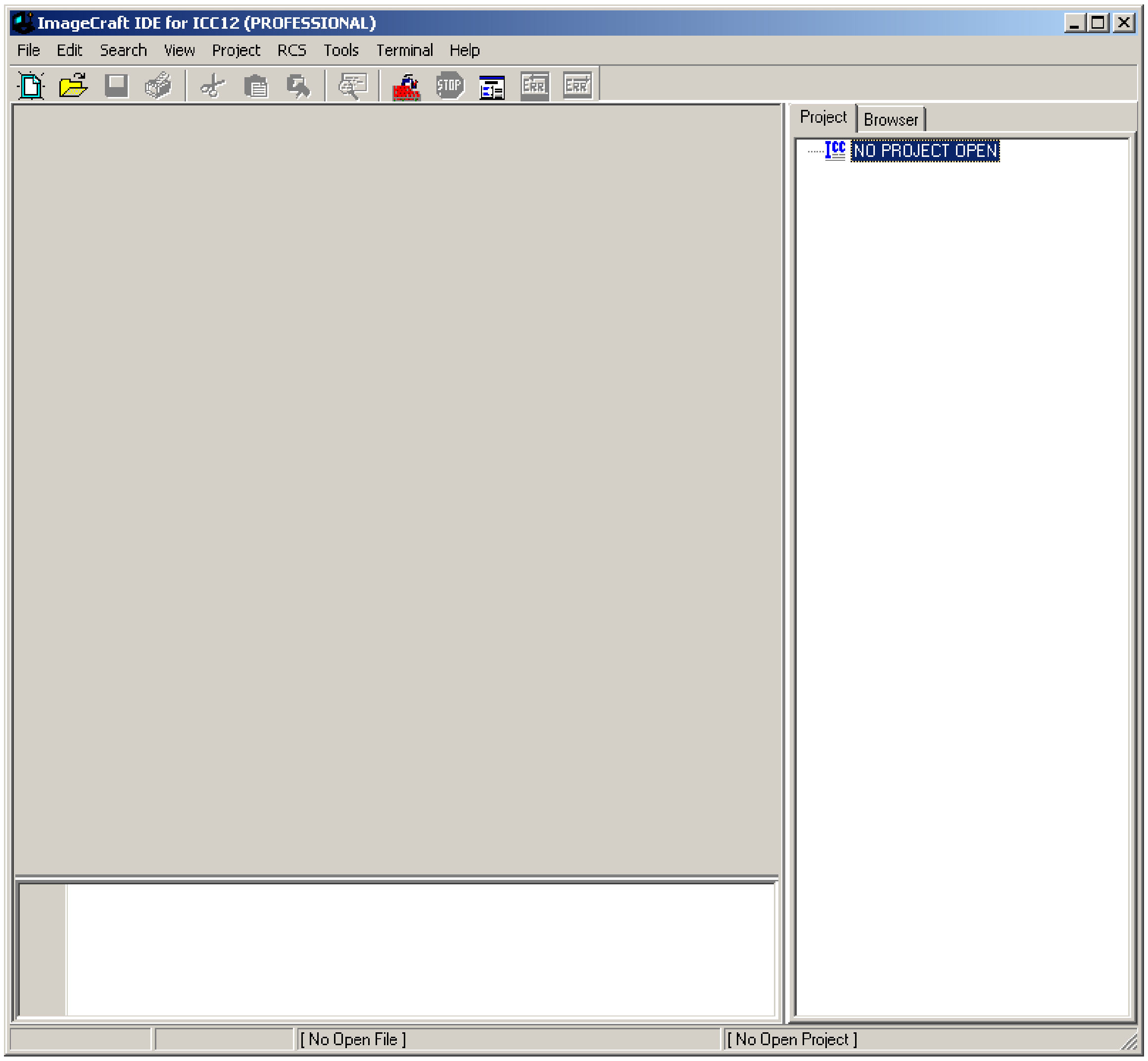
http://www.interlog.com/~techart/myfiles/files/12disk4.zip

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.

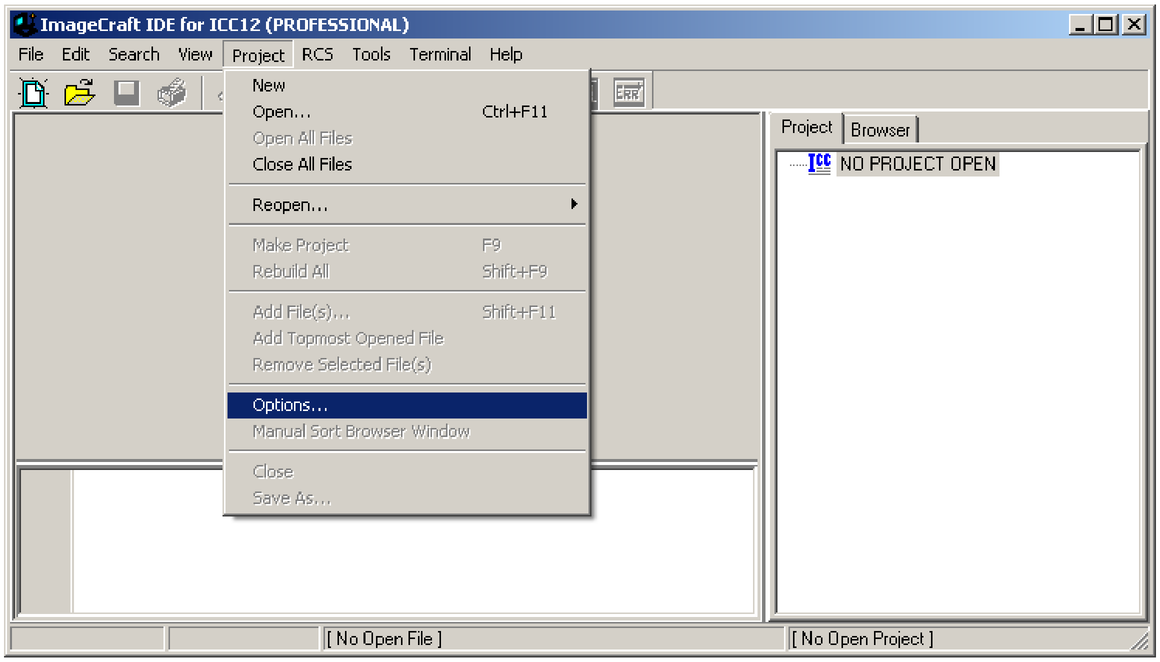
Note the 3 window panes. The top left most is greyed out and the right is the project window. The left bottom pane is where the error messages are displayed during compilation.

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 Adapt912DT60.

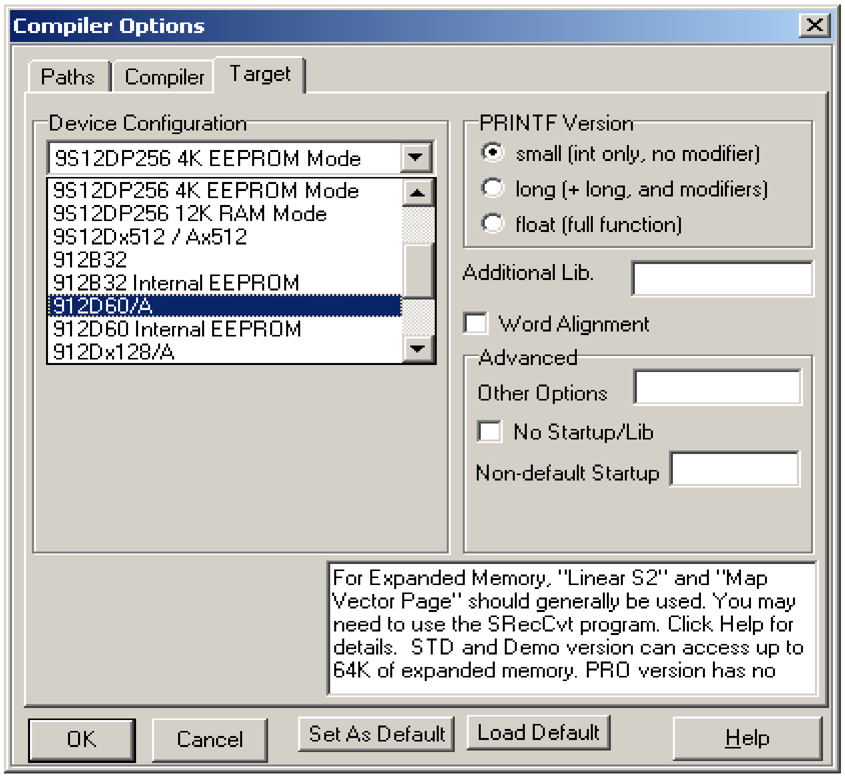

ImageCraft IDE for ICC12 (PROFESSIONAL)
File Edit Search View Project RCS Tools Terminal Help
Project
Browser
NO PROJECT OPEN
[No Open File]
[No Open Project]

Compiler Setup:

Click on Project Menu – Options – Target Tab.

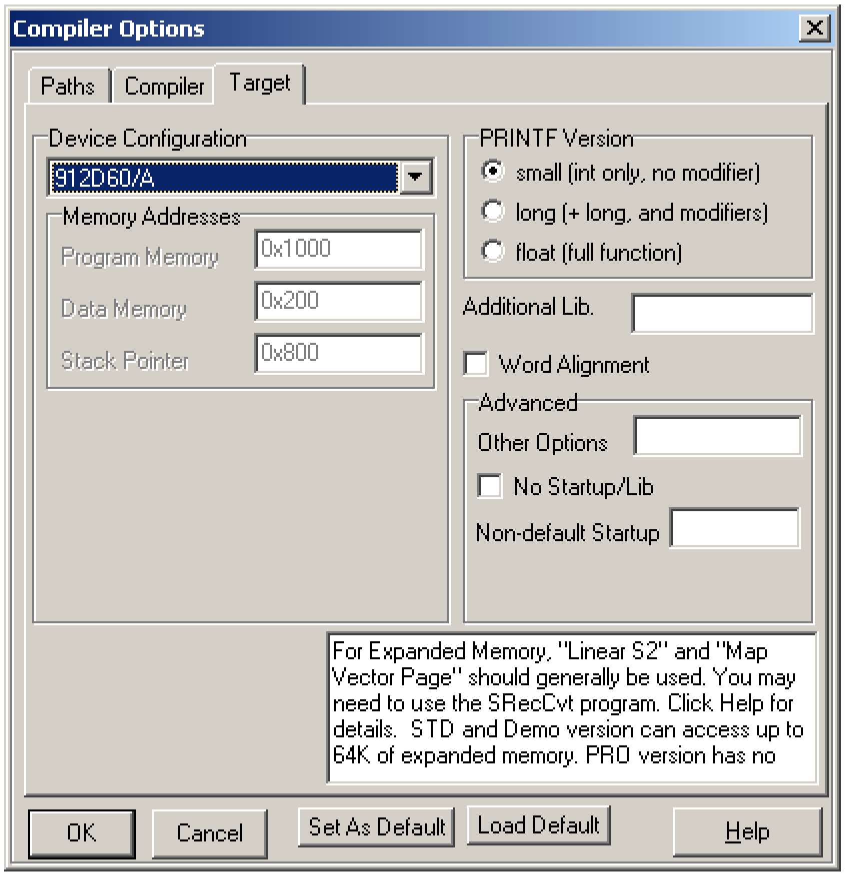


Please note the Device Configuration. Click on the pull down arrow to change the device type.



For Expanded Memory, "Linear S2" and "Map Vector Page" should generally be used. You may need to use the SRecCvt program. Click Help for details. STD and Demo version can access up to 64K of expanded memory. PRO version has no

Scroll up or down to select 912D60/A as shown.


Compiler Options

Device Configuration: 912D60/A

Memory Addresses
Program Memory: 0x1000
Data Memory: 0x200
Stack Pointer: 0x800

PRINTF Version
[✓ small (int only, no modifier)] long (+ long, and modifiers) float (full function)

Additional Lib.

☐ Word Alignment

Advanced
Other Options
☐ No Startup/Lib
Non-default Startup

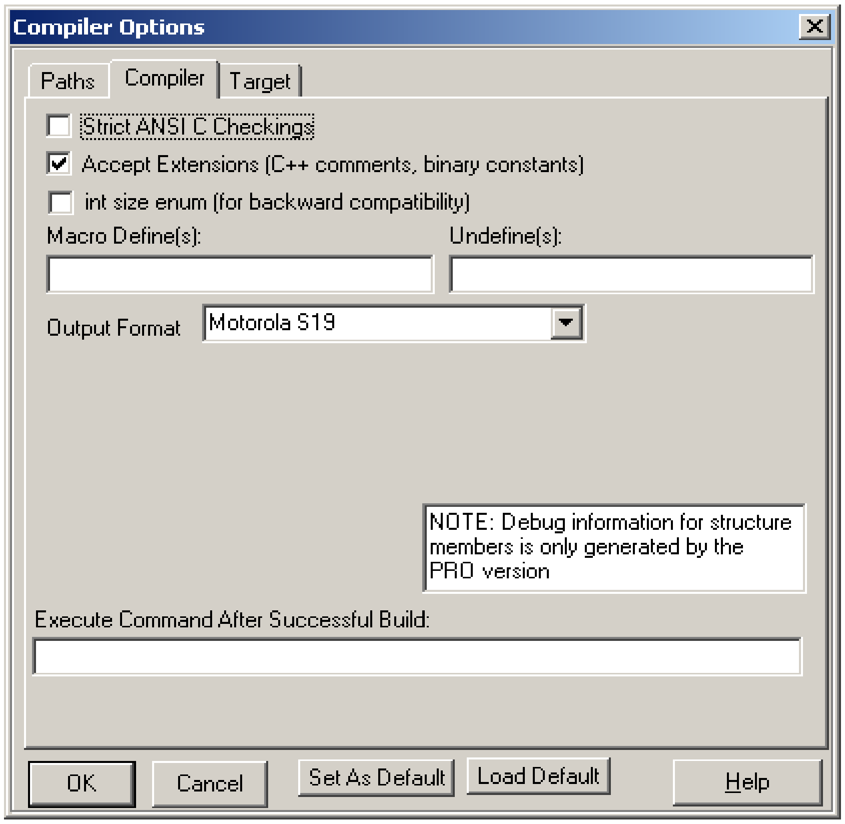
For Expanded Memory, "Linear S2" and "Map Vector Page" should generally be used. You may need to use the SRecCvt program. Click Help for details. STD and Demo version can access up to 64K of expanded memory. PRO version has no

Device Configuration:

Program Memory: ***0x1000***
Data Memory: ***0x200***
Stack Pointer: ***0x800***

The program code is allocated to start from 0x1000. The internal RAM is to start from 0x200 and the stack is to start from 0x800 and work downward.

On the compiler tab there are several choices of S-record output as shown.



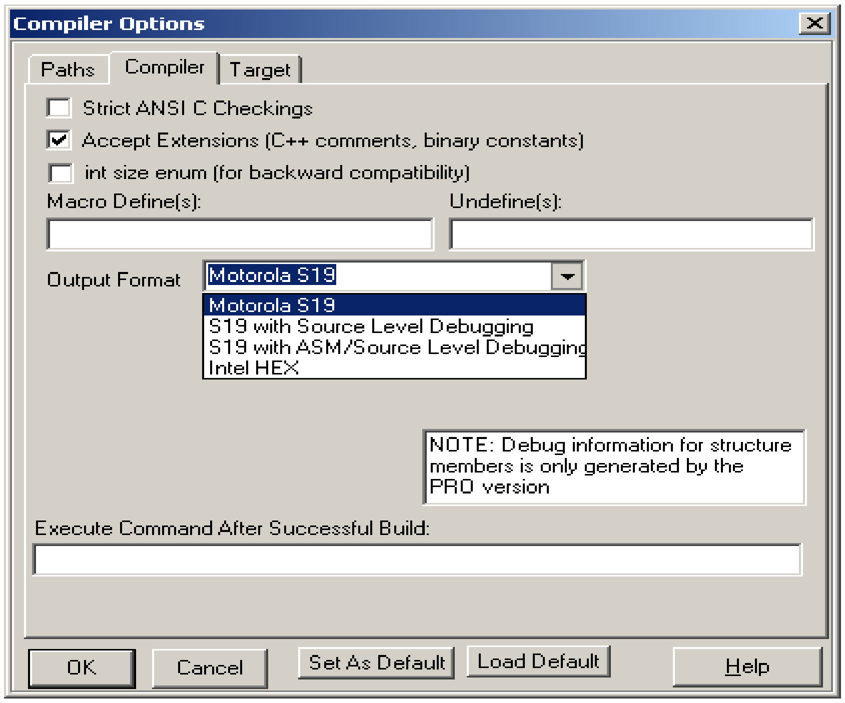
[ ] Strict ANSI C Checkings
[✓] Accept Extensions (C++ comments, binary constants)
[ ] int size enum (for backward compatibility)

Macro Define(s):
Undefine(s):
Output Format: Motorola S19

NOTE: Debug information for structure members is only generated by the PRO version

Execute Command After Successful Build:

Select which one that suits you.



[ ] Strict ANSI C Checkings
[✓] Accept Extensions (C++ comments, binary constants)
[ ] int size enum (for backward compatibility)

Macro Define(s):
Undefine(s):
Output Format: Motorola S19

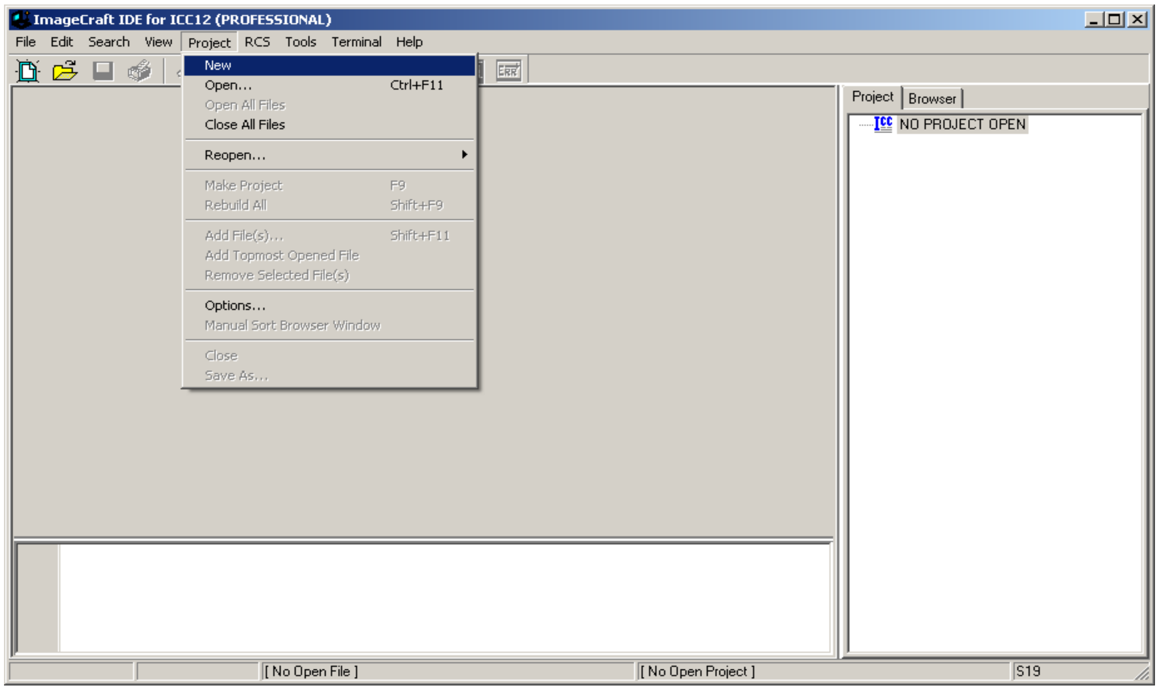
Motorola S19
S19 with Source Level Debugging
S19 with ASM/Source Level Debugging
Intel HEX

NOTE: Debug information for structure members is only generated by the PRO version

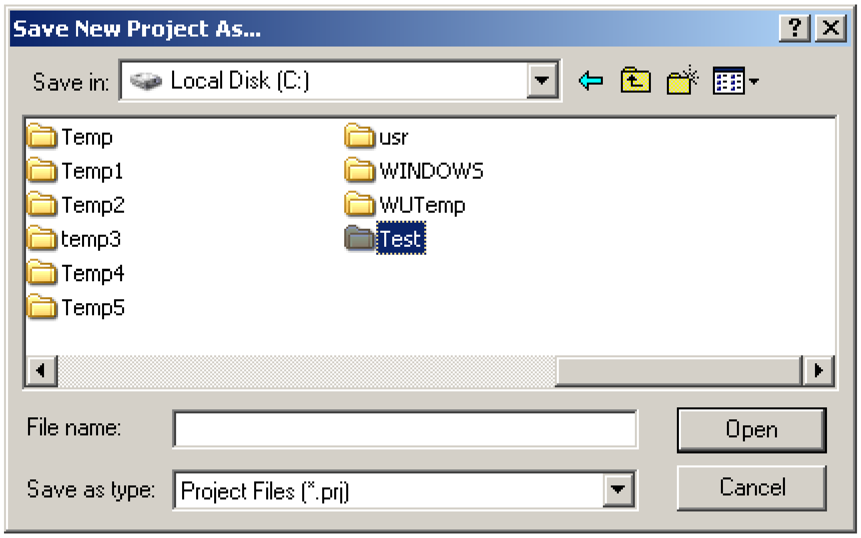
Execute Command After Successful Build:

Starting a new Project:

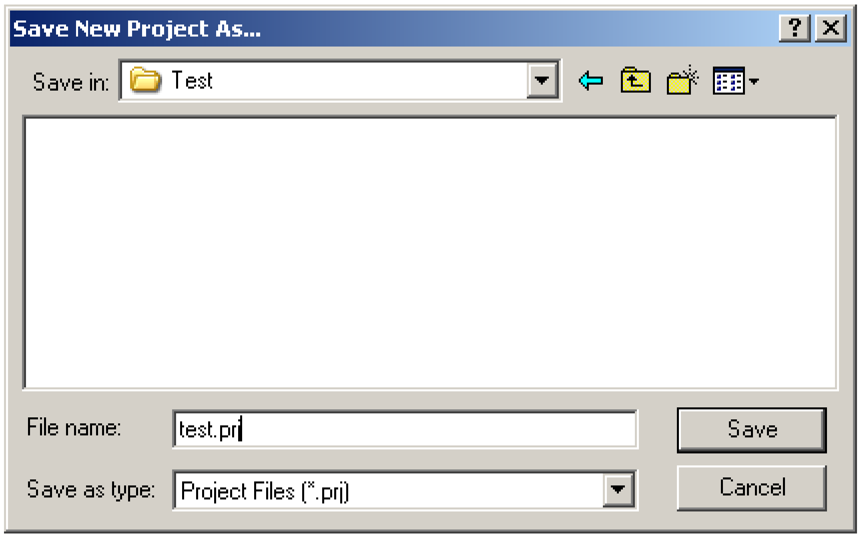
Once the compiler options are setup, a new project can be created. Click Project menu – New.



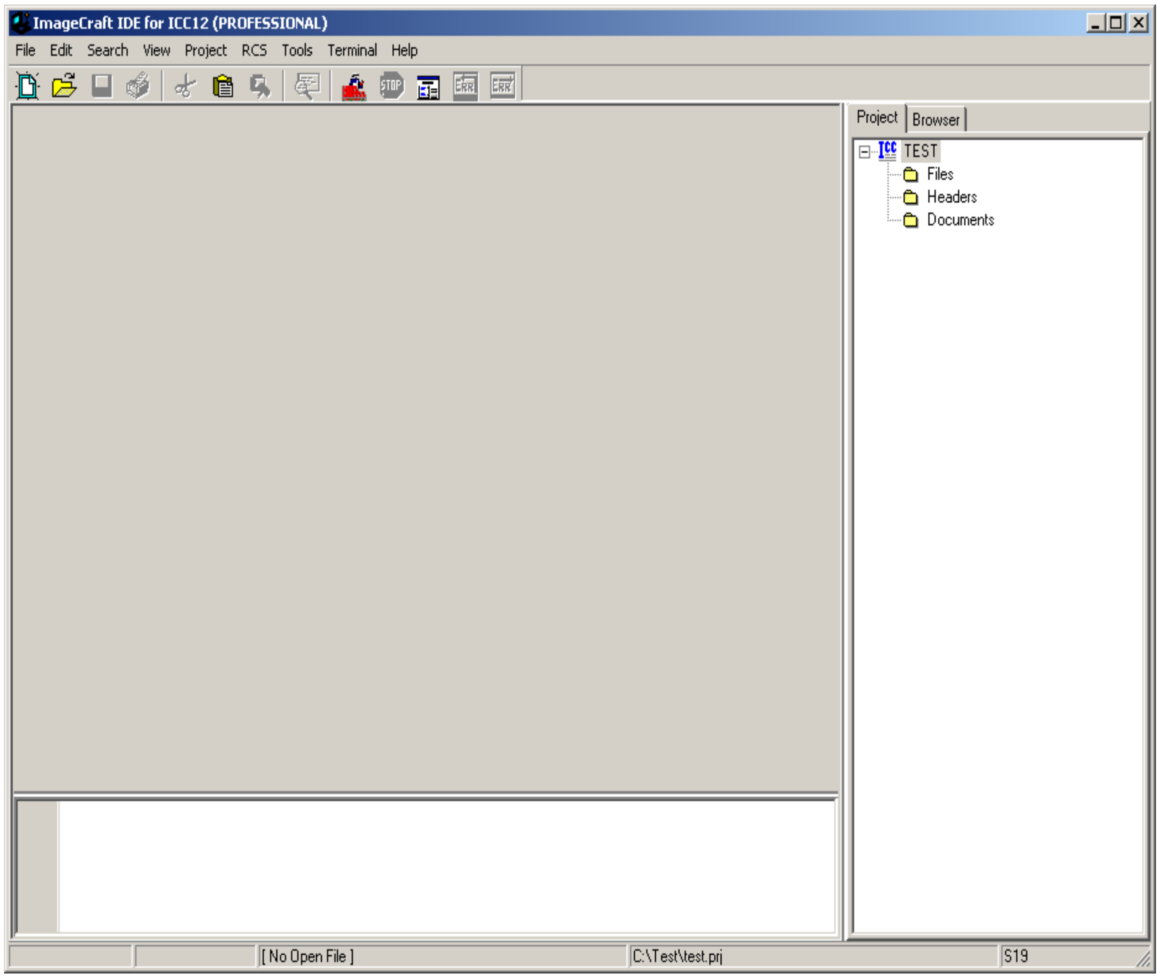
The ICC12 will prompt to save the new project. The user should decide whether to create a new directory to save the new project. In this example a new directory called ***Test*** is created and the file is saved as file ***test.prj***.



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

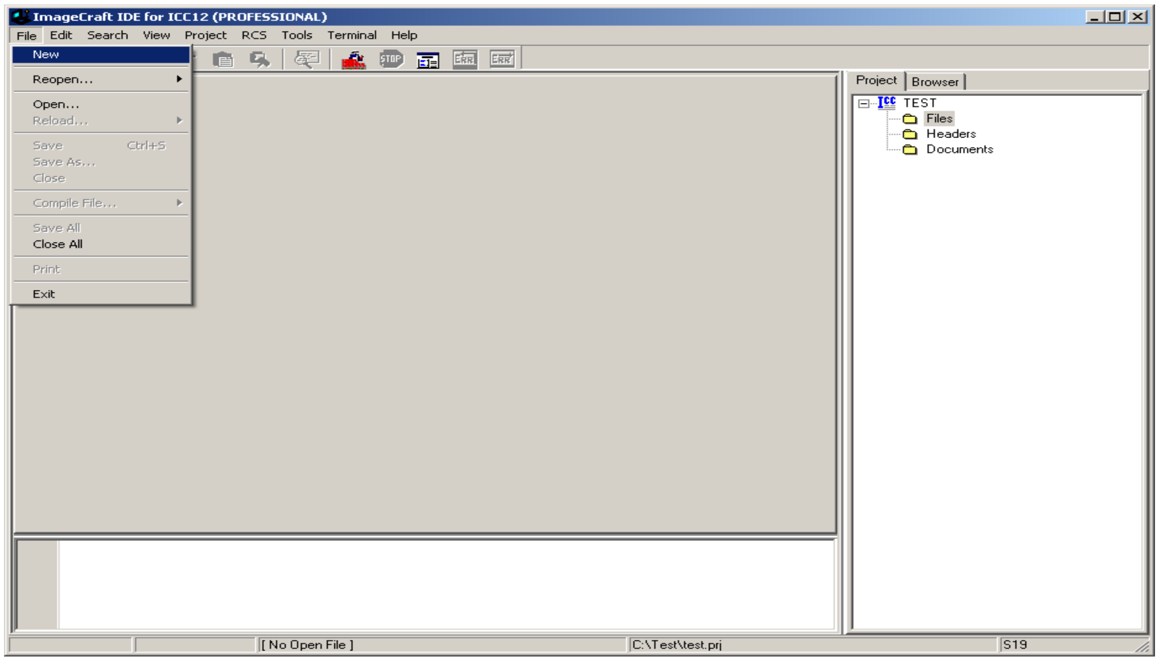


Note that the project window has changed to add Files, Headers and Documents.

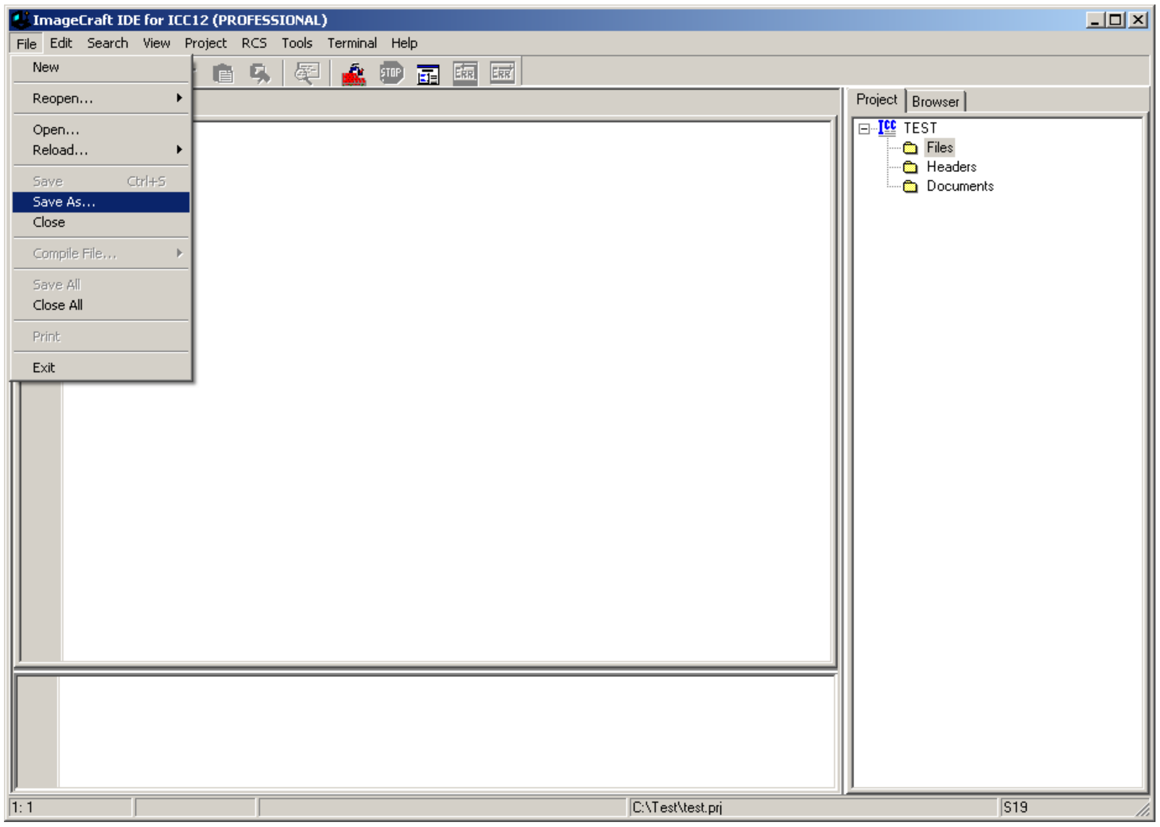

ImageCraft IDE for ICC12 (PROFESSIONAL)
File Edit Search View Project RCS Tools Terminal Help
Project Browser
TEST
Files
Headers
Documents
[No Open File]
C:\Test\test.prj
S19

Creating a new file to the project:

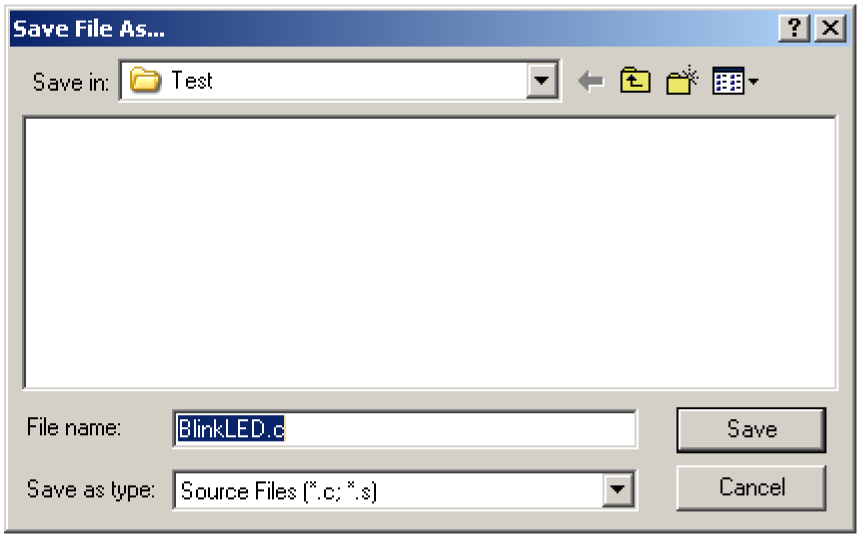
To add files to the project, click on the File menu – new as shown.



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



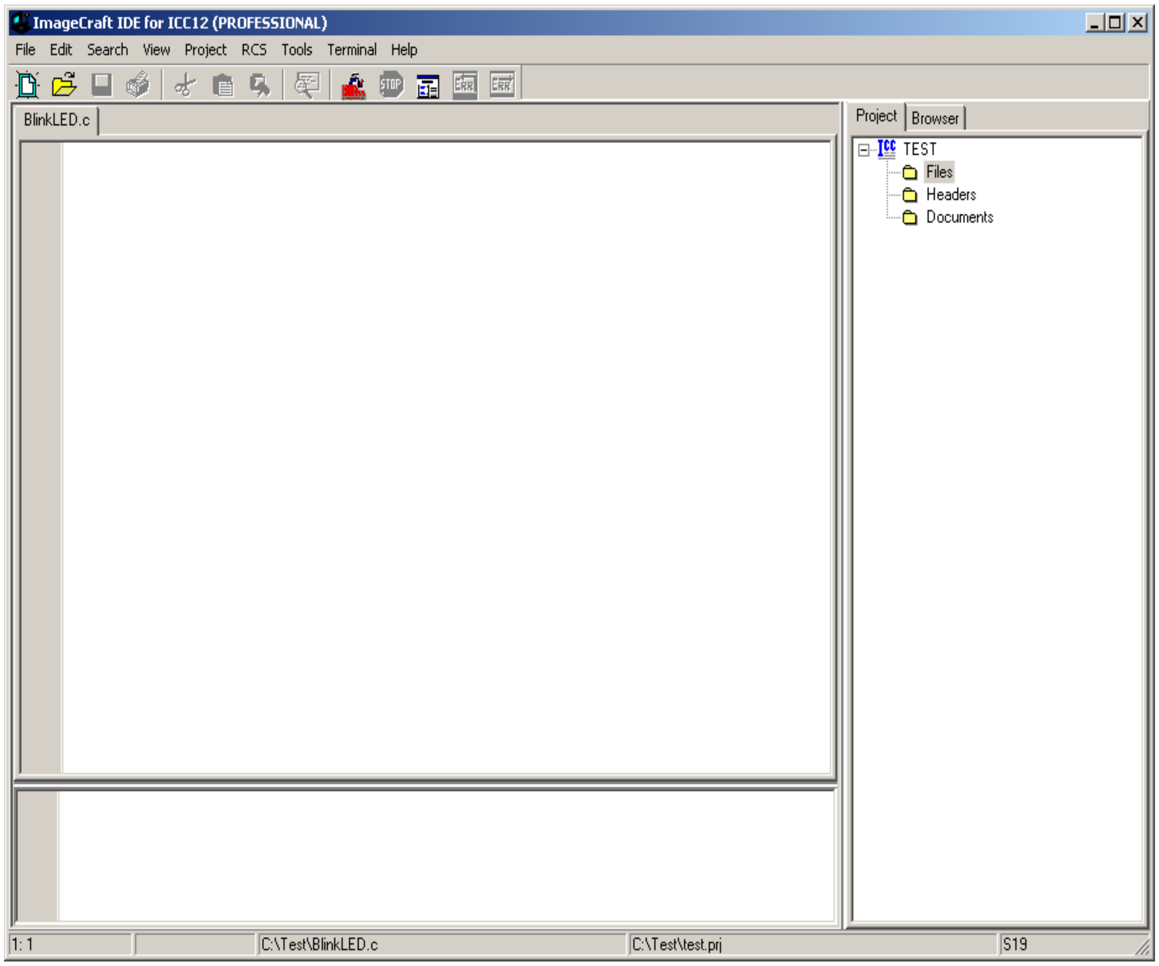
ICC12 will open an explorer window to help save the file. Type BlinkLED.c then press the save button.



File name: BlinkLED.c

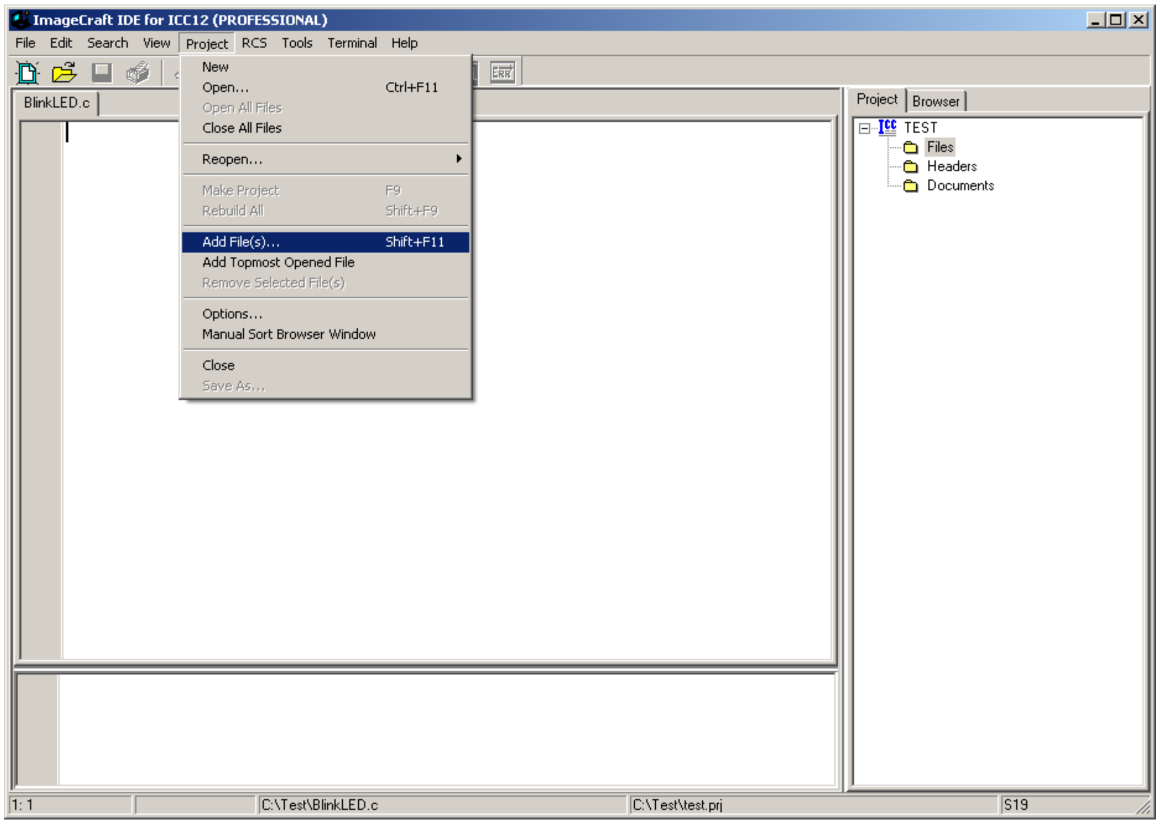
Save as type: Source Files (*.c; *.s)

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

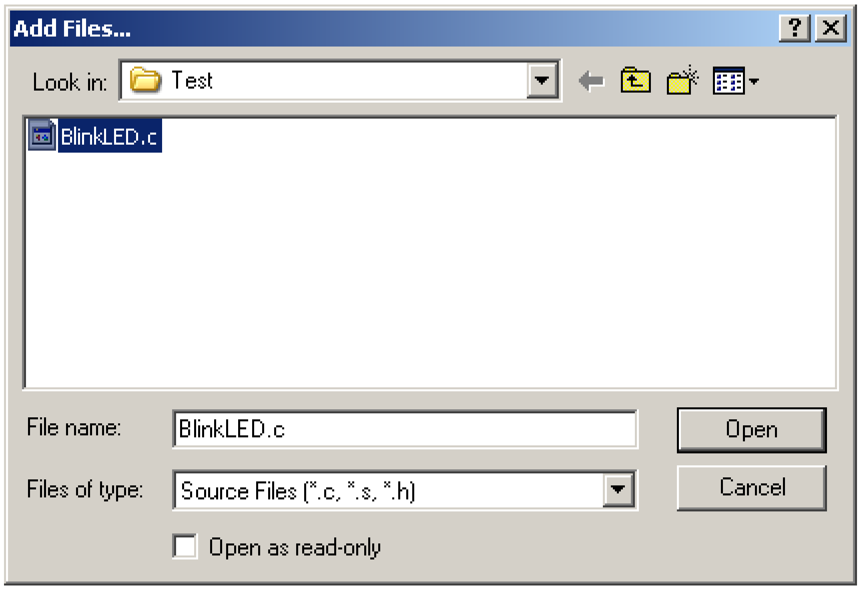

ImageCraft IDE for ICC12 (PROFESSIONAL)
File Edit Search View Project RCS Tools Terminal Help
BlinkLED.c
Project
Browser
TEST
Files
Headers
Documents

```
1: 1    C:\Test\BlinkLED.c    C:\Test\test.prj    S19
```

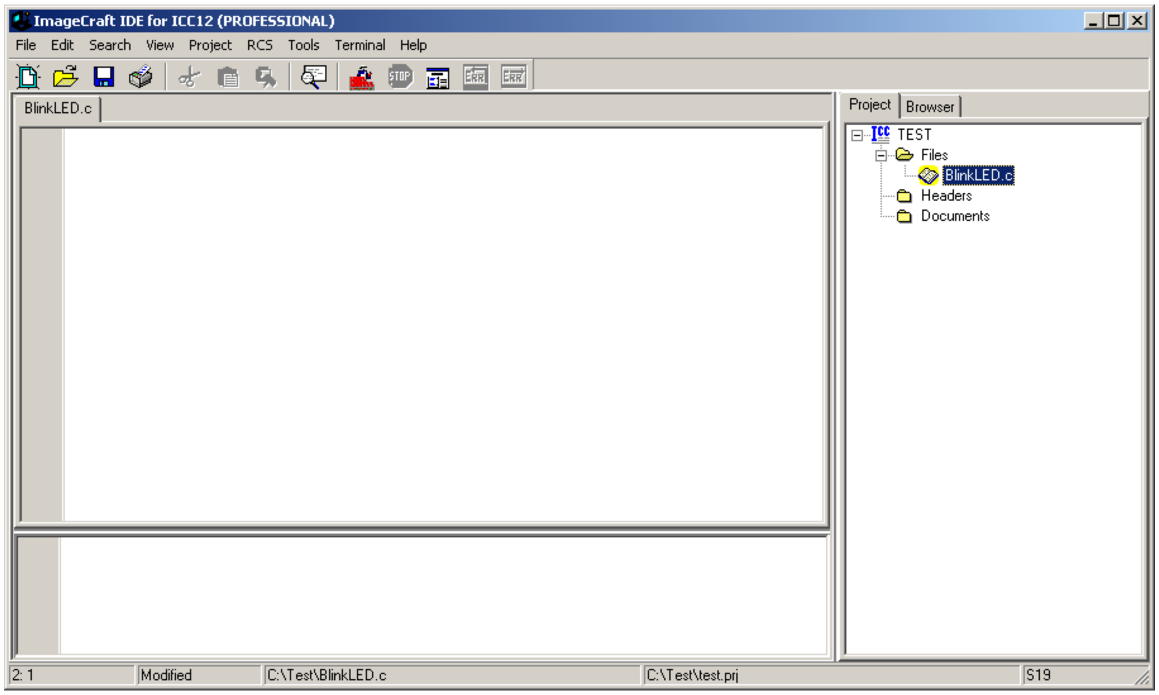
To add BlinkLED.c to the Project, click on the Project menu – Add File(s)



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

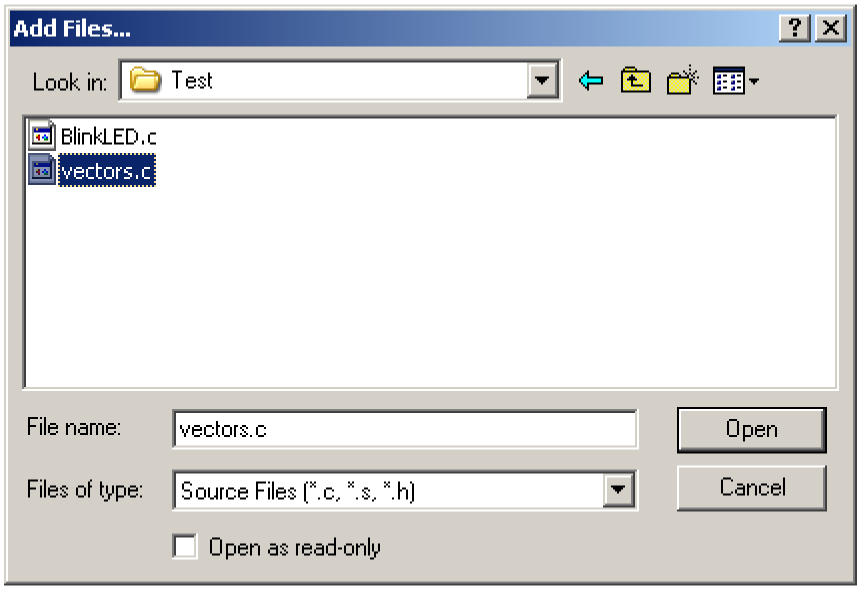


Note that the right window pane has changed to include BlinkLED.c under the Files Project.



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)

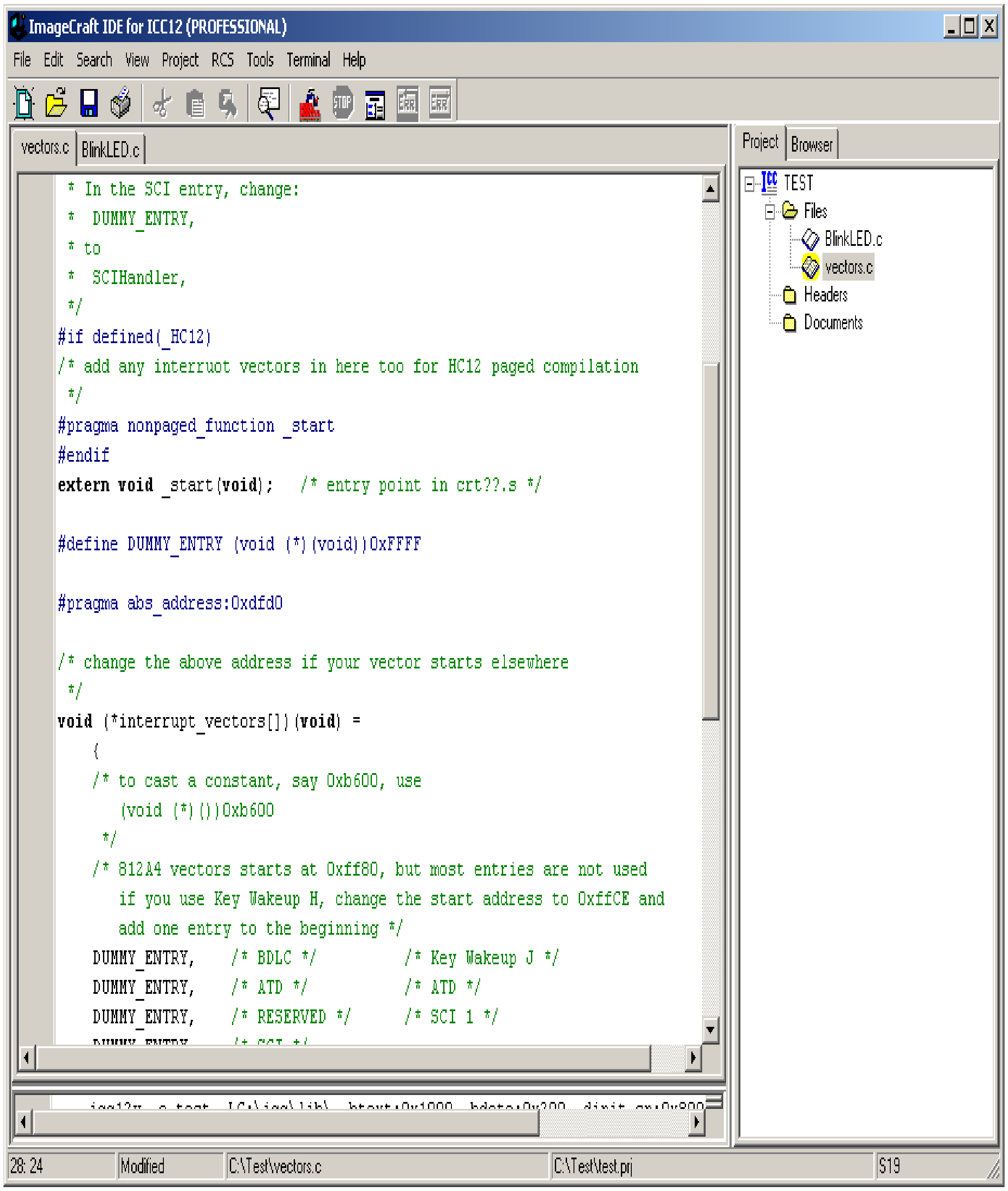


File name: vectors.c

Files of type: Source Files (*.c, *.s, *.h)

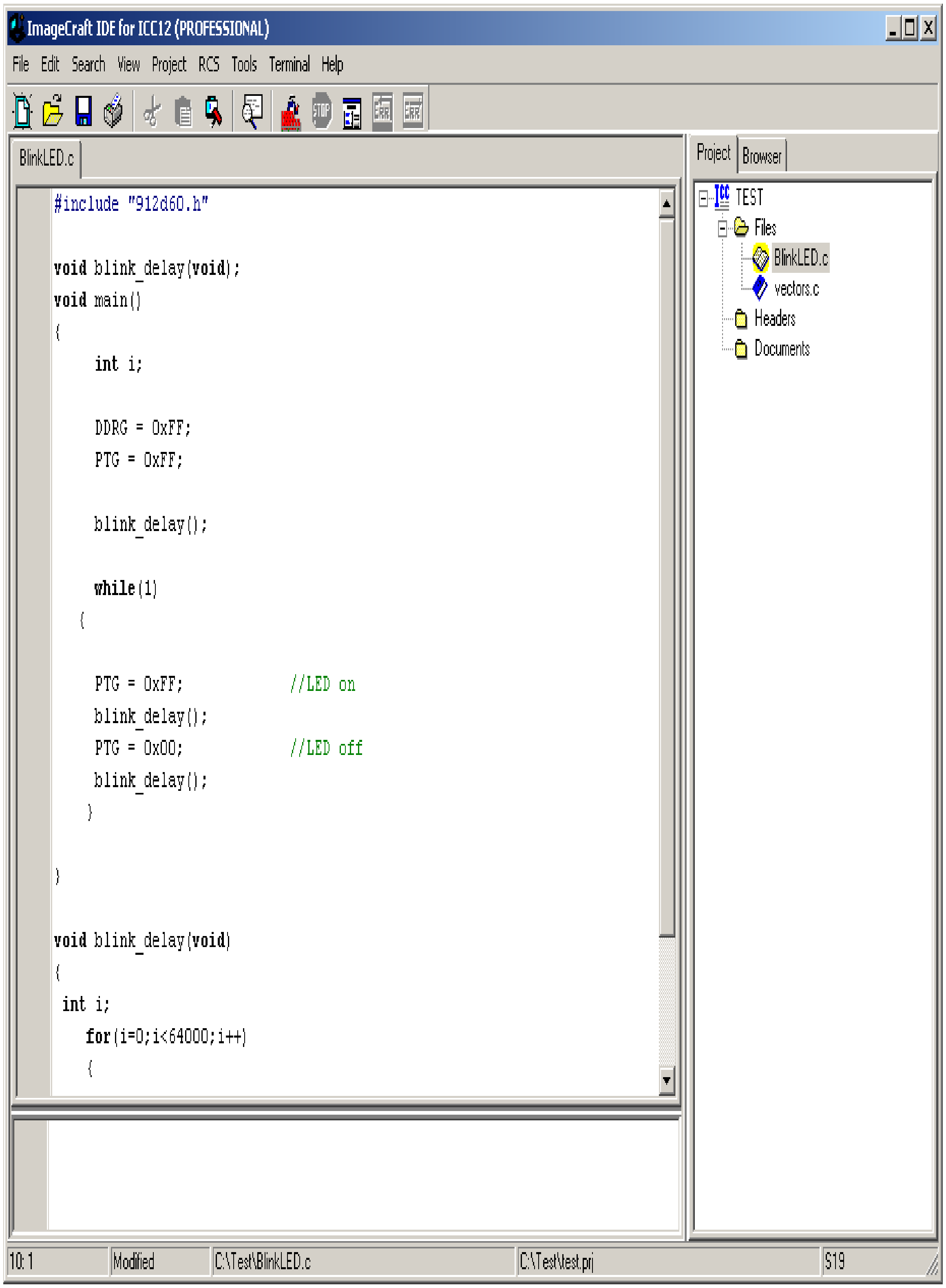
Note that ICC12 has changed to include ***vectors.c*** It is important to note that the **vectors.c** was written for the 68HC912B32 and 812A4 MCUs. One should edit the file to include other ISR addresses for the 912D60. This example edits the line ***#pragma abs_address:0xffd0*** to ***#pragma abs_address:0xdfd0***

The original vector address 0xFFD0 is changed to 0xDFD0. This is because the FLASH Loader resides from $E000 to $FFFF.



```
 * In the SCI entry, change:
 *  DUMMY_ENTRY,
 * to
 *  SCIHandler,
 */
#if defined(_HC12)
/* add any interruot vectors in here too for HC12 paged compilation
 */
#pragma nonpaged_function _start
#endif
extern void _start(void);   /* entry point in crt??.s */

#define DUMMY_ENTRY (void (*)(void))0xFFFF

#pragma abs_address:0xdfd0

/* change the above address if your vector starts elsewhere
 */
void (*interrupt_vectors[])(void) =
    {
    /* to cast a constant, say 0xb600, use
       (void (*)())0xb600
     */
    /* 812A4 vectors starts at 0xff80, but most entries are not used
       if you use Key Wakeup H, change the start address to 0xffCE and
       add one entry to the beginning */
    DUMMY_ENTRY,    /* BDLC */          /* Key Wakeup J */
    DUMMY_ENTRY,    /* ATD */           /* ATD */
    DUMMY_ENTRY,    /* RESERVED */      /* SCI 1 */
```

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

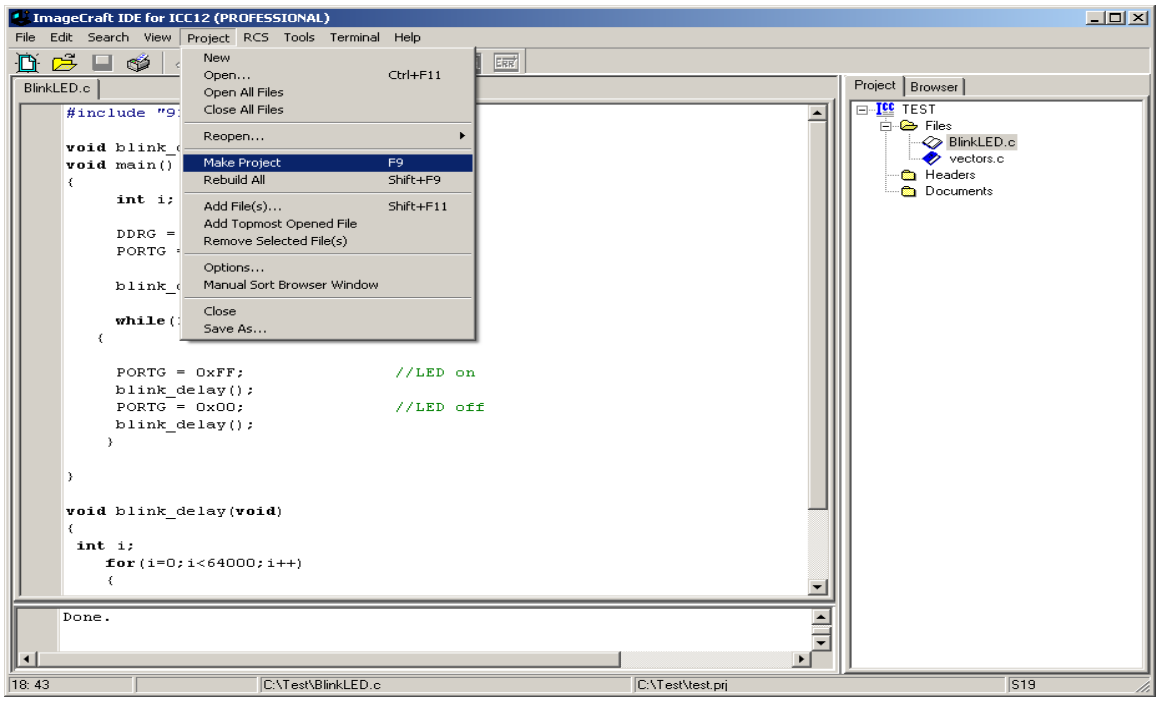


```c
#include "912d60.h"

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

     DDRG = 0xFF;
     PTG = 0xFF;

     blink_delay();

     while(1)
   {

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


}

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

```
#include "912d60.h"

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

       DDRG = 0xFF;
       PORTG = 0xFF;

       blink_delay();

       while(1)
   {

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

}

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

Compiling/Build/Make the file:

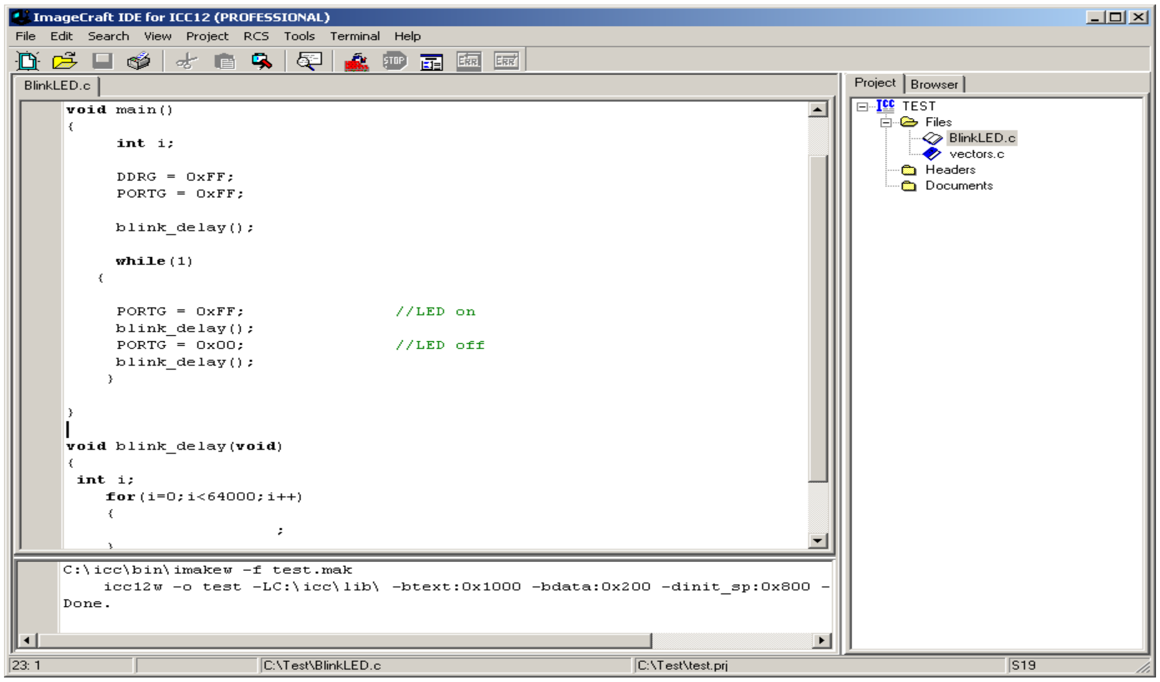
To make the file click Project menu – make project as shown.



```
#include "9
void blink_
void main()
{
    int i;

    DDRG =
    PORTG =

    blink_

    while(
  {

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

}

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

```
Done.
```

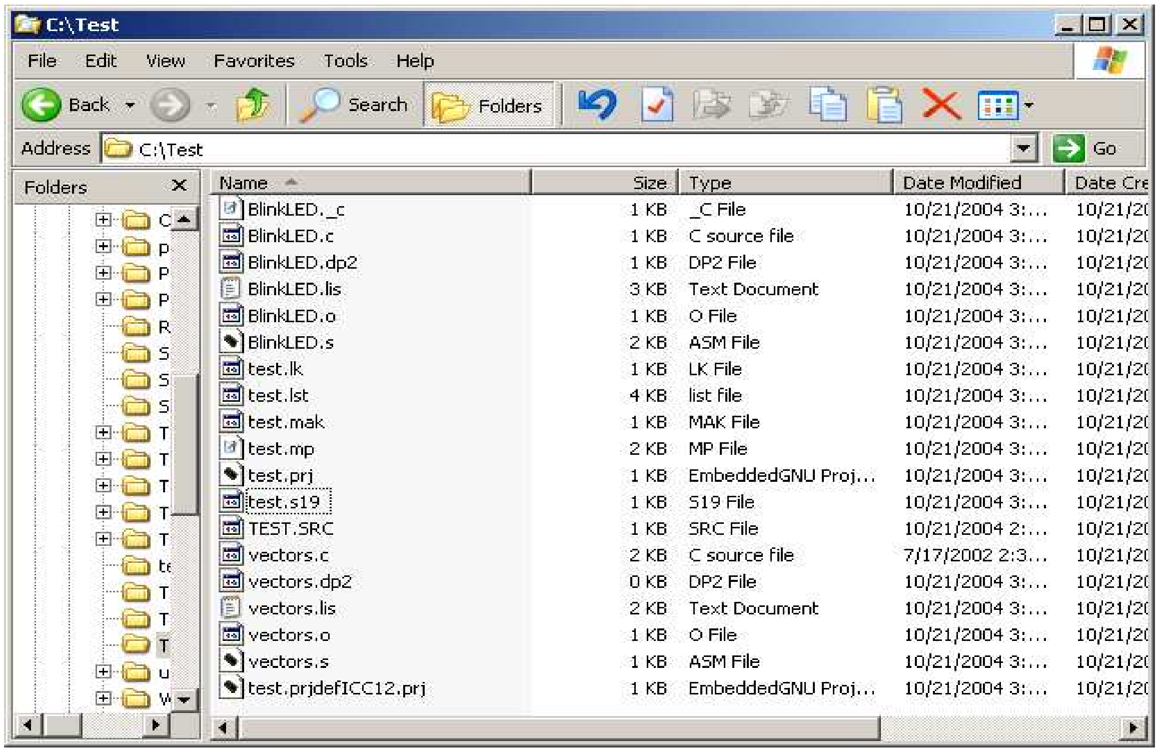
Note the bottom window pane will show messages to display how the build progressed. Any errors, if any, are shown in this window. The build was without error so we can progress to erasing and programming the 912D60/A.



```
void main()
{
    int i;

    DDRG = 0xFF;
    PORTG = 0xFF;

    blink_delay();

    while(1)
  {

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

}

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

```
C:\icc\bin\imakew -f test.mak
    icc12w -o test -LC:\icc\lib\ -btext:0x1000 -bdata:0x200 -dinit_sp:0x800 -
Done.
```

Note the other extraneous files are created after a make.



Name	Size	Type	Date Modified	Date Cre
BlinkLED._c	1 KB	_C File	10/21/2004 3:...	10/21/20
BlinkLED.c	1 KB	C source file	10/21/2004 3:...	10/21/20
BlinkLED.dp2	1 KB	DP2 File	10/21/2004 3:...	10/21/20
BlinkLED.lis	3 KB	Text Document	10/21/2004 3:...	10/21/20
BlinkLED.o	1 KB	O File	10/21/2004 3:...	10/21/20
BlinkLED.s	2 KB	ASM File	10/21/2004 3:...	10/21/20
test.lk	1 KB	LK File	10/21/2004 3:...	10/21/20
test.lst	4 KB	list file	10/21/2004 3:...	10/21/20
test.mak	1 KB	MAK File	10/21/2004 3:...	10/21/20
test.mp	2 KB	MP File	10/21/2004 3:...	10/21/20
test.prj	1 KB	EmbeddedGNU Proj...	10/21/2004 3:...	10/21/20
test.s19	1 KB	S19 File	10/21/2004 3:...	10/21/20
TEST.SRC	1 KB	SRC File	10/21/2004 2:...	10/21/20
vectors.c	2 KB	C source file	7/17/2002 2:3...	10/21/20
vectors.dp2	0 KB	DP2 File	10/21/2004 3:...	10/21/20
vectors.lis	2 KB	Text Document	10/21/2004 3:...	10/21/20
vectors.o	1 KB	O File	10/21/2004 3:...	10/21/20
vectors.s	1 KB	ASM File	10/21/2004 3:...	10/21/20
test.prjdefICC12.prj	1 KB	EmbeddedGNU Proj...	10/21/2004 3:...	10/21/20

Using WordPad to check the content of ***test.s19*** file. Note that the S-records are of different lengths.

```
S10E1000CF080016107087CE02008E8F
S110100B020027056A000820F6CE1075CDFE
S111101802008E10752706180A307020F51697
S1071026102A20FE6A
S110102A34B7751B9EC6FF7B002AC6FF7BF2
S11010370028161052200EC6FF7B0028165C
S11110441052790028161052 20F0B757303D94
S111105234B7751B9ECC00006C1E2007EC1EEC
S1111060C300016C1EEC1E8CFA0025F2B7577B
S105106E303D0F
S111DFD0FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF4D
S111DFDEFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF3F
S111DFECFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF31
S109DFFAFFFFFFFF100011
S10810701D0016073D00
S9031000EC
```

Examining S-record:

If one looks closely at the S-record one can see S1 to be of different lengths. This is atypical S-record generated by ICC12. S1 records are programmed in the ***$1000 - $DFFF*** memory blocks.

As stated previously, FLASH Loader occupies $E000 to $FFFF therefore the vector address at below $E000.

Note the content of the memory address at $DFFE:$DFFF is $1000, the RESET vector.

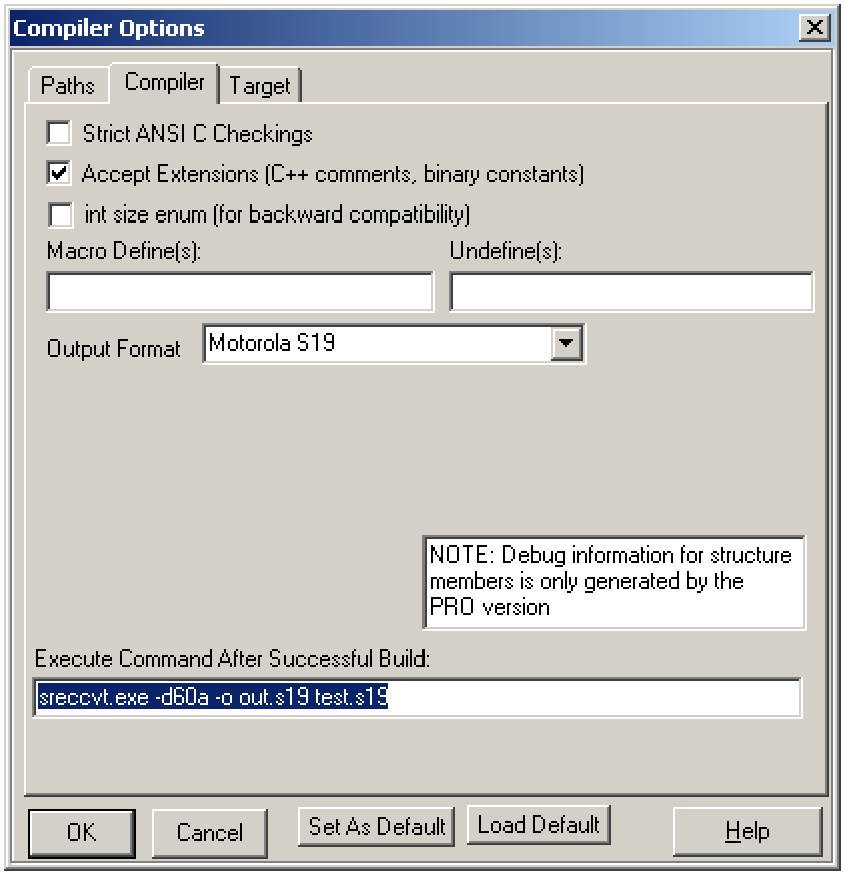
```
S111DFD0FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF4D
S111DFDEFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF3F
S111DFECFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF31
S109DFFAFFFFFFFF100011
```

The S-record below is the start of code. The content of address beginning at $1000 to $1050

```
S10E1000CF080016107087CE02008E8F
S110100B020027056A000820F6CE1075CDFE
S111101802008E10752706180A307020F51697
S1071026102A20FE6A
S110102A34B7751B9EC6FF7B002AC6FF7BF2
S11010370028161052200EC6FF7B0028165C
S1111044105279002816105220F0B757303D94
S111105234B7751B9ECC00006C1E2007EC1EEC
S1111060C300016C1EEC1E8CFA0025F2B7577B
S105106E303D0F
```

For the 912D60A, the S- records needs to be formatted for 64 bytes lengths. Therefore SrecCVT program needs to be invoked to be used to reformat the S-record. Under Project-Options and Compiler Tab add the following line at the Execute Command after Successful Build line. Press Ok button then re-make the file.

sreccvt.exe -d60a -o out.s19 test.s19


Compiler Options

Paths | Compiler | Target

- [ ] Strict ANSI C Checkings
- [x] Accept Extensions (C++ comments, binary constants)
- [ ] int size enum (for backward compatibility)

Macro Define(s):

Undefine(s):

Output Format: Motorola S19

NOTE: Debug information for structure members is only generated by the PRO version

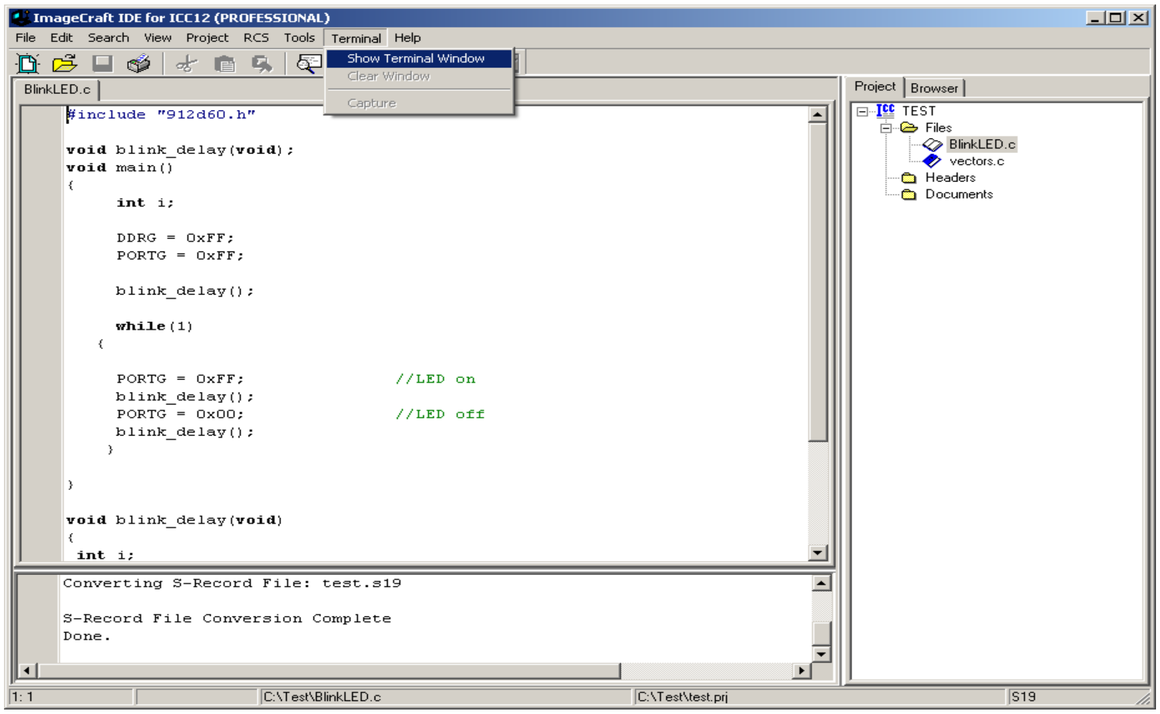
Execute Command After Successful Build:

```
sreccvt.exe -d60a -o out.s19 test.s19
```

Note that ***test.19*** is the original S-record and ***out.s19*** is the file to be uploaded to the MCU.

Programming the Adapt912DT60:

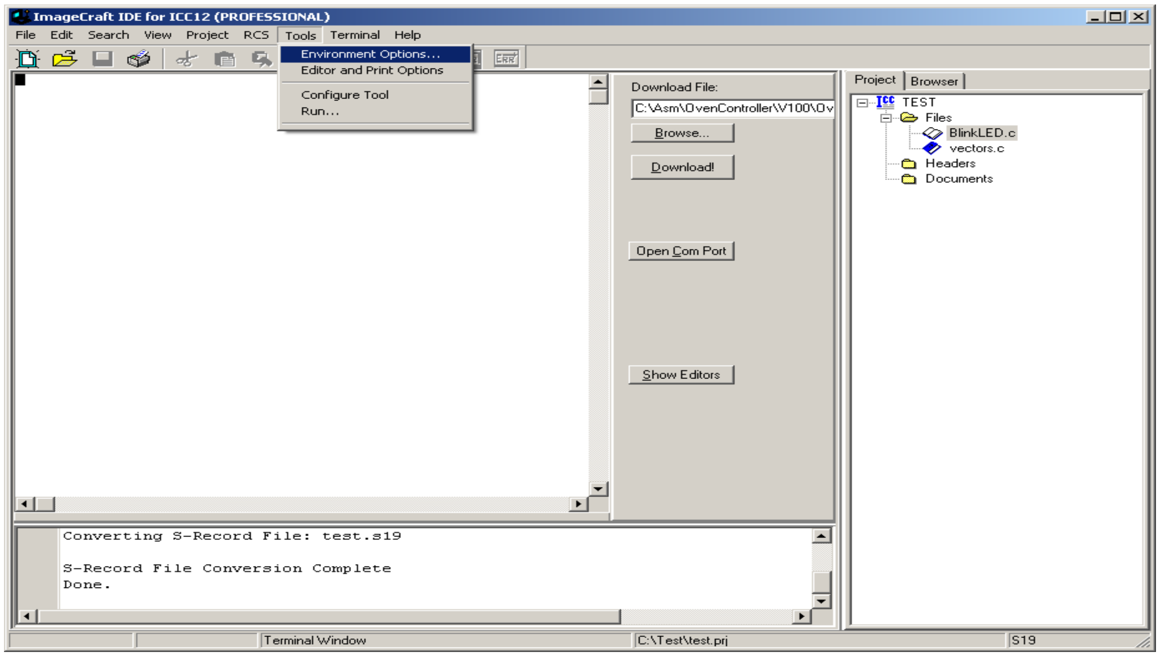
Open ICC12 terminal window by Selecting Show Terminal Window. Connect Serial cable to any available PC COM port and the other end to Adapt912DT60.



```
#include "912d60.h"

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

     DDRG = 0xFF;
     PORTG = 0xFF;

     blink_delay();

     while(1)
   {

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

}

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

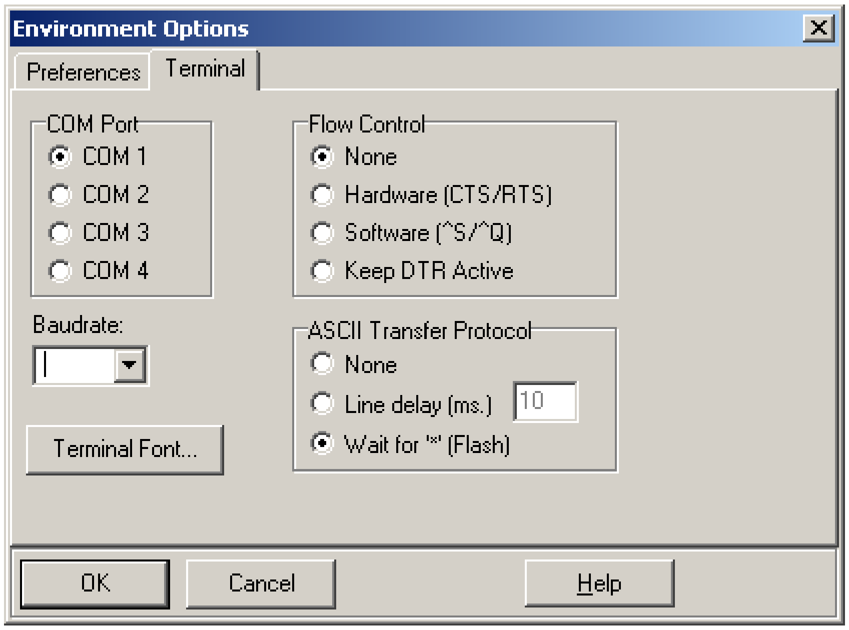
```
Converting S-Record File: test.s19

S-Record File Conversion Complete
Done.
```

Set the COM port parameters under Tools-Environment Options and then Terminal tab.

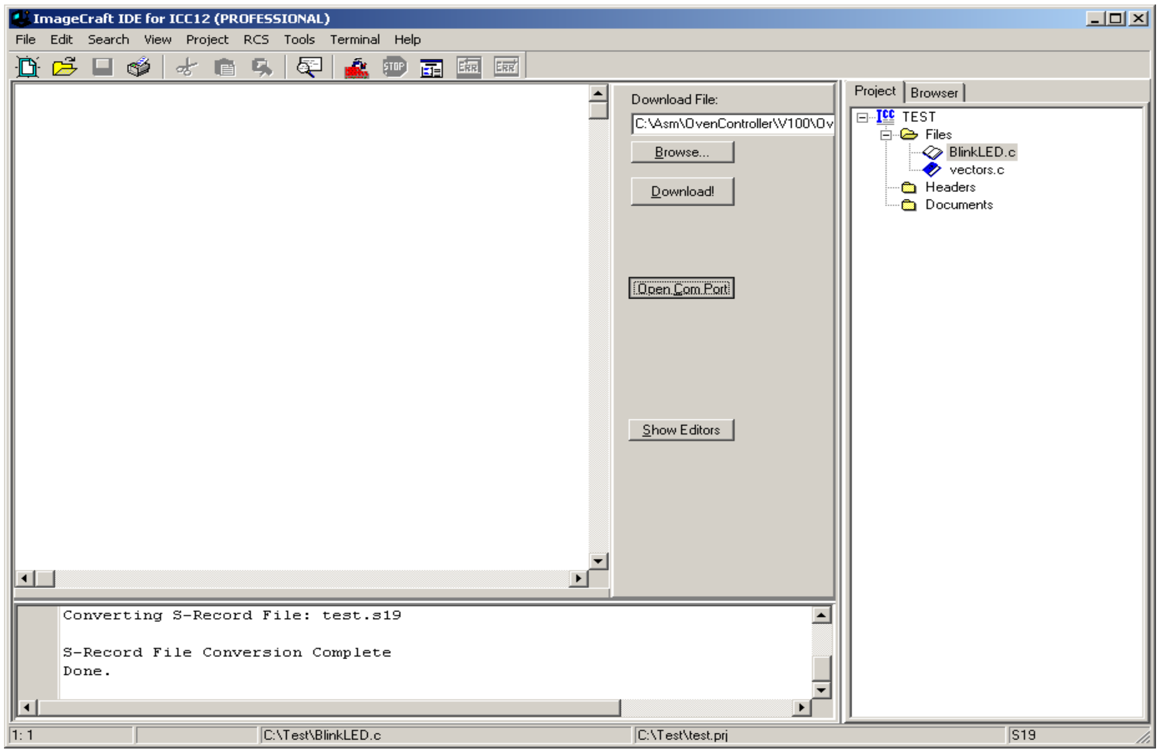


```
Converting S-Record File: test.s19

S-Record File Conversion Complete
Done.
```

Select the COM of your choice, BAUD = 9600, Flow Control = None and ASCII Transfer Protocol to Wait for * (Flash). Press OK button to continue.

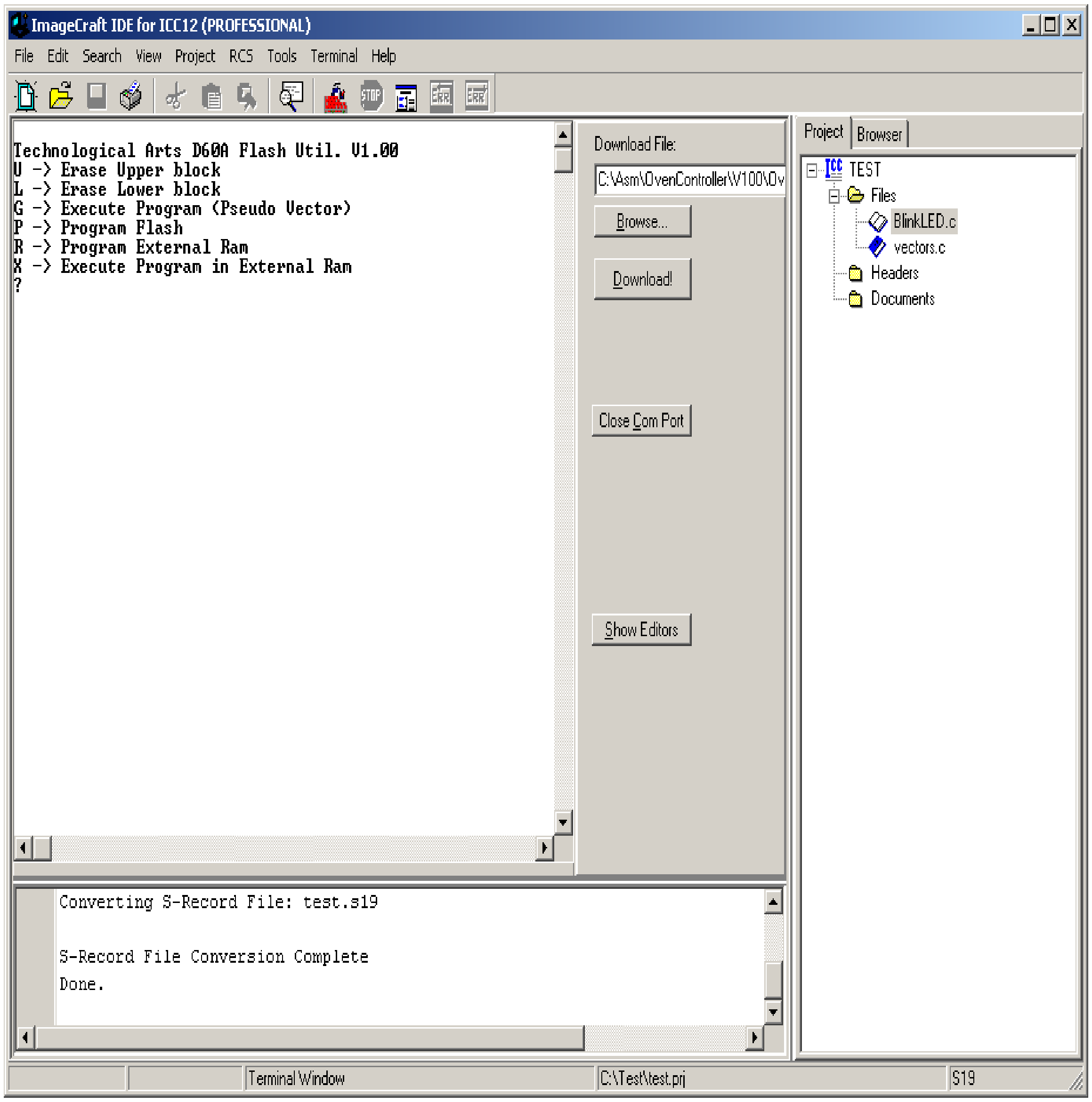


In the middle is the Open Com Port button. Click to connect.



```
Converting S-Record File: test.s19

S-Record File Conversion Complete
Done.
```

Move SW1 Load/Run switch to Load.   Power up unit or press the RESET button if already powered up.



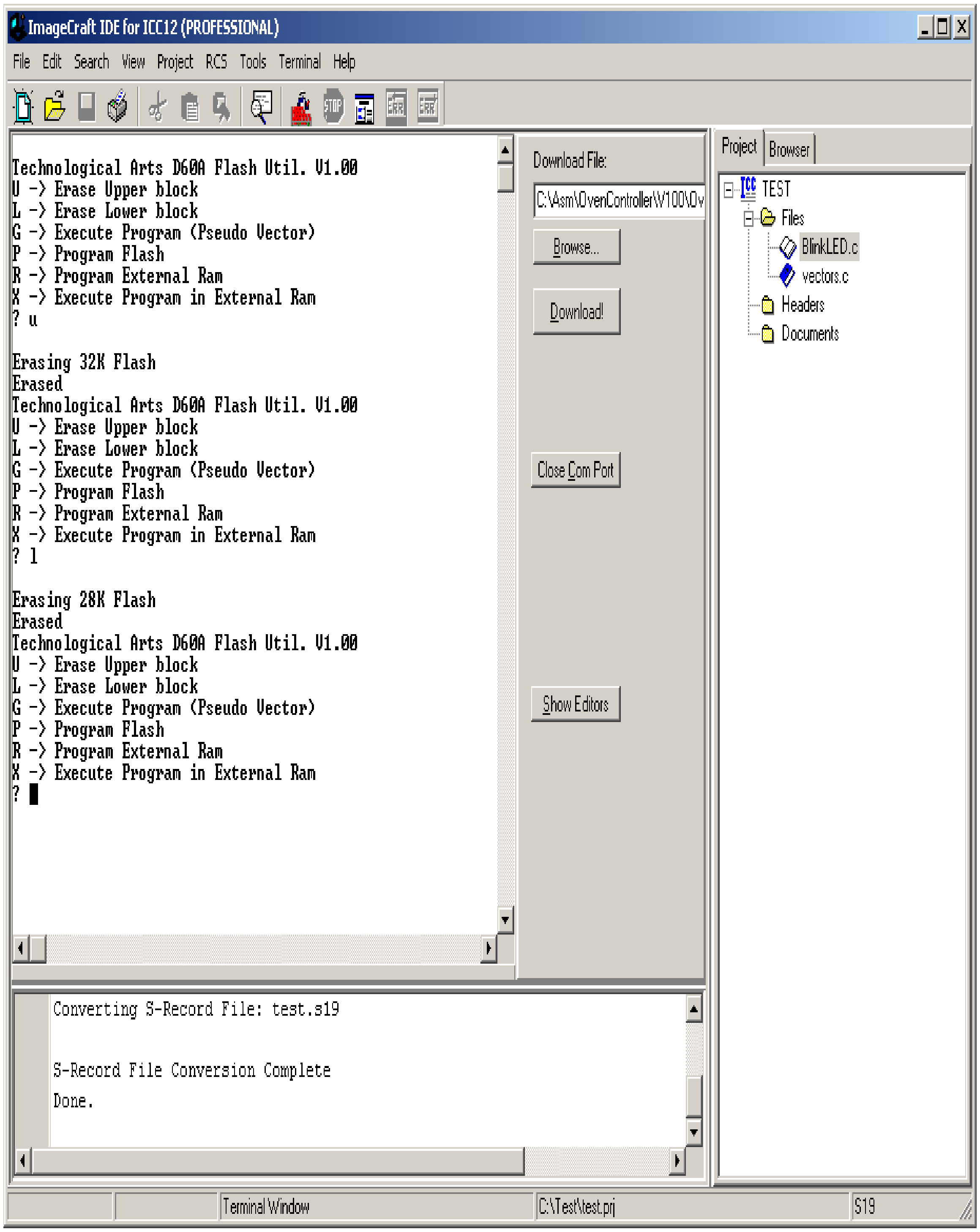
```
Technological Arts D60A Flash Util. V1.00
U -> Erase Upper block
L -> Erase Lower block
G -> Execute Program (Pseudo Vector)
P -> Program Flash
R -> Program External Ram
X -> Execute Program in External Ram
?
```

```
Converting S-Record File: test.s19

S-Record File Conversion Complete
Done.
```

Please note the in the 912D60/A the Flash modules are in 2 location. One is below $8000, beginning at $1000 to $7FFF (Lower block). The other is at $8000 to $FFFF (upper block). The FlashLoader allows one to erase either Lower or upper memory blocks. Since the code starts from $1000 and the Psuedo vector at $DFD0, it is important that both blocks be erased.

Erase command:

The command to erase upper memory block is the letter ***U*** and ***L*** for lower the memory block as shown.

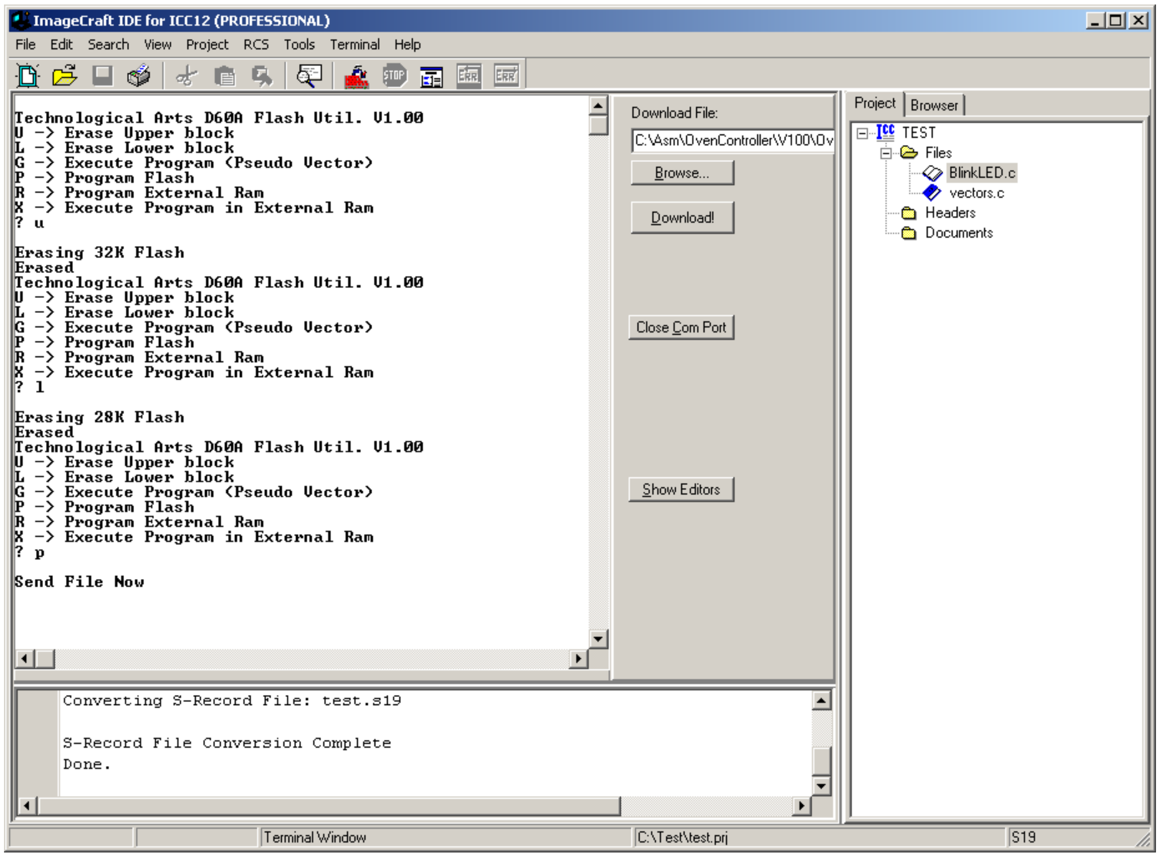


```
Technological Arts D60A Flash Util. V1.00
U -> Erase Upper block
L -> Erase Lower block
G -> Execute Program (Pseudo Vector)
P -> Program Flash
R -> Program External Ram
X -> Execute Program in External Ram
? u

Erasing 32K Flash
Erased
Technological Arts D60A Flash Util. V1.00
U -> Erase Upper block
L -> Erase Lower block
G -> Execute Program (Pseudo Vector)
P -> Program Flash
R -> Program External Ram
X -> Execute Program in External Ram
? l

Erasing 28K Flash
Erased
Technological Arts D60A Flash Util. V1.00
U -> Erase Upper block
L -> Erase Lower block
G -> Execute Program (Pseudo Vector)
P -> Program Flash
R -> Program External Ram
X -> Execute Program in External Ram
? █
```

```
Converting S-Record File: test.s19

S-Record File Conversion Complete
Done.
```

Programming:

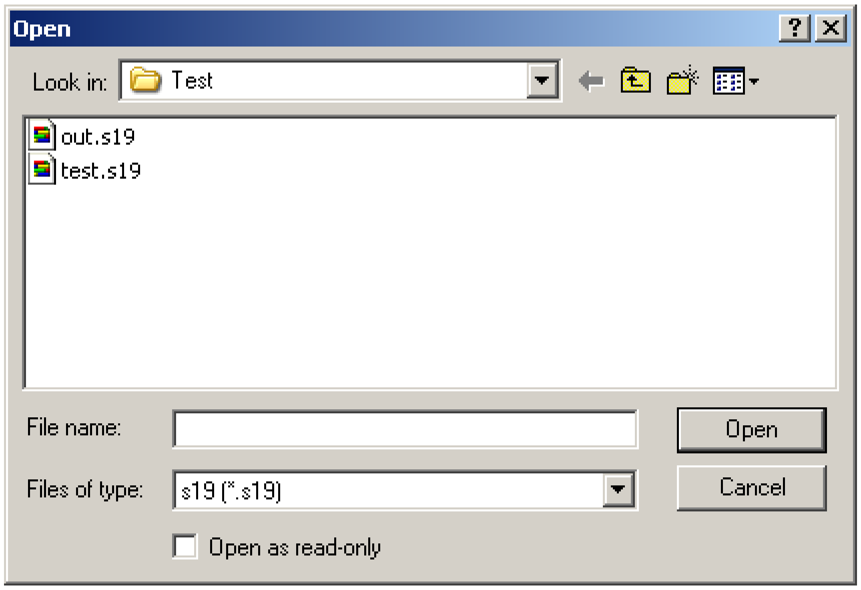
To program select P command as shown.



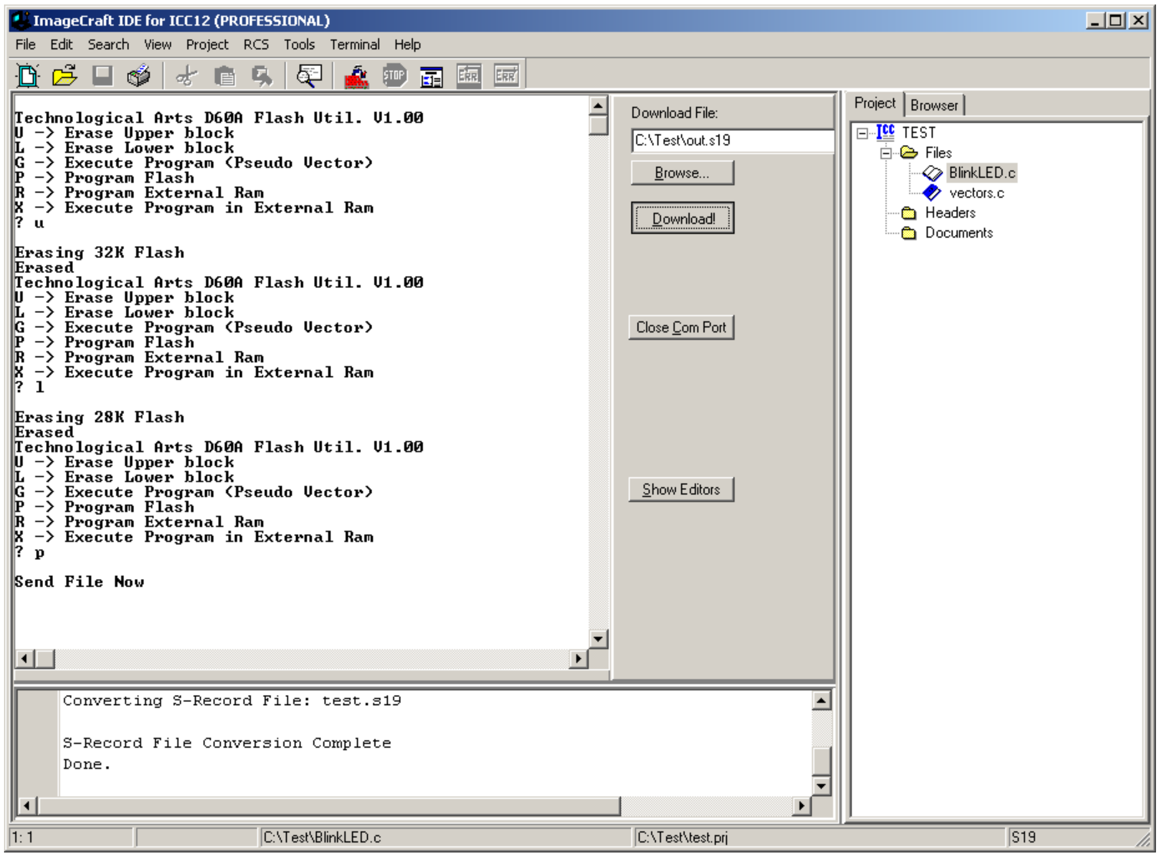
```
Technological Arts D60A Flash Util. V1.00
U -> Erase Upper block
L -> Erase Lower block
G -> Execute Program (Pseudo Vector)
P -> Program Flash
R -> Program External Ram
X -> Execute Program in External Ram
? u

Erasing 32K Flash
Erased
Technological Arts D60A Flash Util. V1.00
U -> Erase Upper block
L -> Erase Lower block
G -> Execute Program (Pseudo Vector)
P -> Program Flash
R -> Program External Ram
X -> Execute Program in External Ram
? l

Erasing 28K Flash
Erased
Technological Arts D60A Flash Util. V1.00
U -> Erase Upper block
L -> Erase Lower block
G -> Execute Program (Pseudo Vector)
P -> Program Flash
R -> Program External Ram
X -> Execute Program in External Ram
? p

Send File Now
```

```
Converting S-Record File: test.s19

S-Record File Conversion Complete
Done.
```

Under the Download File press the Browse button to help locate the ***out.s19*** file. Select ***out.s19*** then press the ***open*** button.



Once the correct file is selected, press the Download button to initiate upload to Adapt912DT60



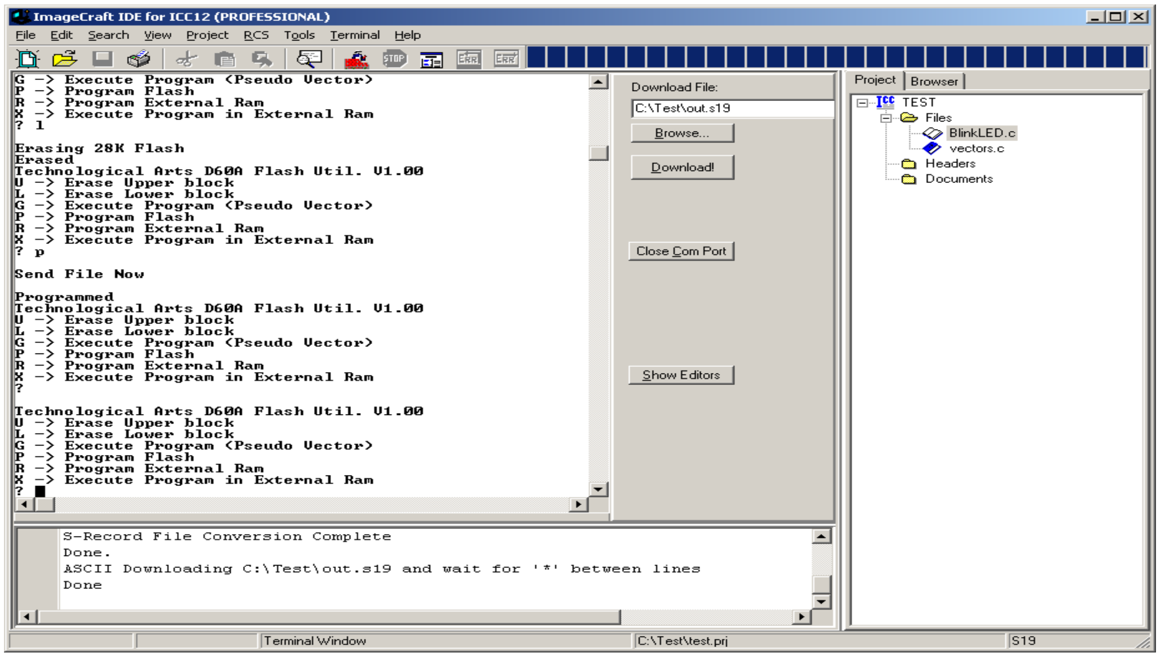
```
Technological Arts D60A Flash Util. V1.00
U -> Erase Upper block
L -> Erase Lower block
G -> Execute Program (Pseudo Vector)
P -> Program Flash
R -> Program External Ram
X -> Execute Program in External Ram
? u

Erasing 32K Flash
Erased
Technological Arts D60A Flash Util. V1.00
U -> Erase Upper block
L -> Erase Lower block
G -> Execute Program (Pseudo Vector)
P -> Program Flash
R -> Program External Ram
X -> Execute Program in External Ram
? l

Erasing 28K Flash
Erased
Technological Arts D60A Flash Util. V1.00
U -> Erase Upper block
L -> Erase Lower block
G -> Execute Program (Pseudo Vector)
P -> Program Flash
R -> Program External Ram
X -> Execute Program in External Ram
? p

Send File Now
```

Download File: C:\Test\out.s19

```
Converting S-Record File: test.s19

S-Record File Conversion Complete
Done.
```

1: 1 | C:\Test\BlinkLED.c | C:\Test\test.prj | S19

A progress Bar will indicate that the file is being transferred.



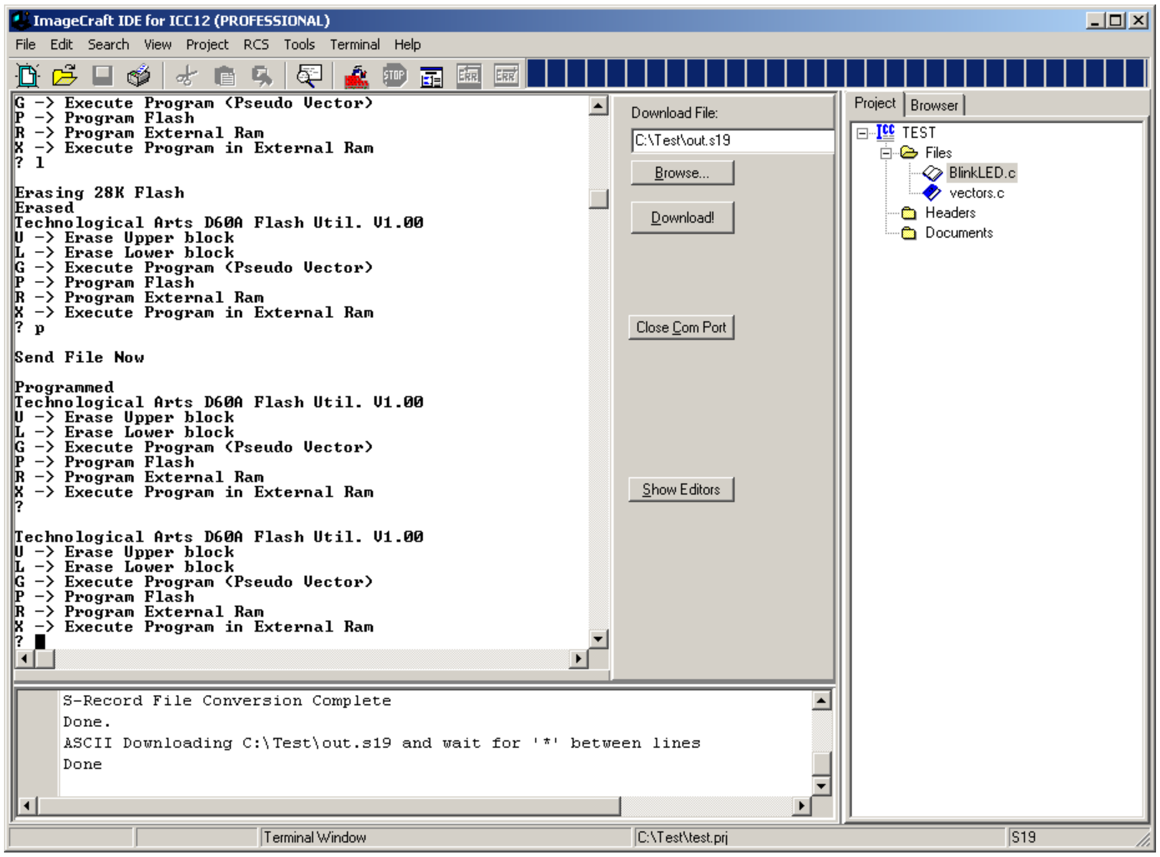
```
G -> Execute Program (Pseudo Vector)
P -> Program Flash
R -> Program External Ram
X -> Execute Program in External Ram
? l

Erasing 28K Flash
Erased
Technological Arts D60A Flash Util. V1.00
U -> Erase Upper block
L -> Erase Lower block
G -> Execute Program (Pseudo Vector)
P -> Program Flash
R -> Program External Ram
X -> Execute Program in External Ram
? p

Send File Now

Programmed
Technological Arts D60A Flash Util. V1.00
U -> Erase Upper block
L -> Erase Lower block
G -> Execute Program (Pseudo Vector)
P -> Program Flash
R -> Program External Ram
X -> Execute Program in External Ram
?

Technological Arts D60A Flash Util. V1.00
U -> Erase Upper block
L -> Erase Lower block
G -> Execute Program (Pseudo Vector)
P -> Program Flash
R -> Program External Ram
X -> Execute Program in External Ram
?
```

Download File: C:\Test\out.s19

```
S-Record File Conversion Complete
Done.
ASCII Downloading C:\Test\out.s19 and wait for '*' between lines
Done
```

Terminal Window | C:\Test\test.prj | S19

Once file transmit is completed a ***Programmed*** message will appear on the screen as shown.



```
G -> Execute Program (Pseudo Vector)
P -> Program Flash
R -> Program External Ram
X -> Execute Program in External Ram
? l

Erasing 28K Flash
Erased
Technological Arts D60A Flash Util. V1.00
U -> Erase Upper block
L -> Erase Lower block
G -> Execute Program (Pseudo Vector)
P -> Program Flash
R -> Program External Ram
X -> Execute Program in External Ram
? p

Send File Now

Programmed
Technological Arts D60A Flash Util. V1.00
U -> Erase Upper block
L -> Erase Lower block
G -> Execute Program (Pseudo Vector)
P -> Program Flash
R -> Program External Ram
X -> Execute Program in External Ram
?

Technological Arts D60A Flash Util. V1.00
U -> Erase Upper block
L -> Erase Lower block
G -> Execute Program (Pseudo Vector)
P -> Program Flash
R -> Program External Ram
X -> Execute Program in External Ram
?
```

```
S-Record File Conversion Complete
Done.
ASCII Downloading C:\Test\out.s19 and wait for '*' between lines
Done
```

Move SW1 Run/Load switch to Run and press the RESET button. The LED on the Adapt912DT60 that is connected to PG7 will begin to blink.

This concludes the use of ICC12 with Adapt912DT60 to using the FLASH Loader.