SoftConsole FAQ

May 2013

Table of Contents

Frequently asked Questions on SoftConsole

1.What is SoftConsole?

SoftConsole is a free software development environment that enables the rapid production of C and C++ executables for ARM® Cortex™-M1, Cortex™-M3 processor, CoreMP7, and Core8051s. Creating a project inside SoftConsole allows you to write software that is immediately compiled into a usable binary code. SoftConsole includes a fully integrated debugger that offers easy access to memory contents, registers, and single-instruction execution.

2. Where do I install SoftConsole from?

You can download the latest release of SoftConsole from Microsemi website at:
www.microsemi.com/soc/download/software/softconsole/files.aspx

3. What are the system requirements for installing SoftConsole?

SoftConsole can run on the following Microsoft® Windows® operating systems:

- Microsoft® Windows® 7 Professional (U.S. Version) (32-bit and 64-bit)
- Microsoft® Windows® Vista Business (U.S. Version) (32-bit and 64-bit)
- Microsoft® Windows® XP Professional with SP3 (U.S. Version) (32-bit and 64-bit)

Note: SoftConsole might run on other XP/Vista variants, but it is not supported on any platforms other than those listed above. SoftConsole is not supported on any non-U.S. versions of Windows.

Minimum system requirements needed to support SoftConsole:

- Pentium 1.0 GHz processor
- NTFS, FAT32 file system
- 400 MB free disk space
- 128 MB RAM
- 1024 x 768 resolution monitor

4. How do I install SoftConsole?

To install SoftConsole follow the steps given below:

1. Download SoftConsole from Microsemi website:
   www.microsemi.com/soc/download/software/softconsole/files.aspx
2. Double-click the SoftConsole_vX_X_setup.exe file. This opens the installation wizard. Click **Next**.
3. Read through the license agreement, check the **I accept the agreement** radio button, and click **Next**.
4. If the default installation folder is acceptable, click **Next**. If you want to install SoftConsole in a different folder, browse to required location and click **Next**.
5. By default, all recommended SoftConsole components are selected for installation. If you do not want to install a component, clear the selection for it. Click **Next**. This opens the Core8051s and SDCC tools page. The Core8051s and SDCC tools page is for information purposes only. After reading it, click **Next**.
6. By default, SoftConsole creates a Start Menu folder named Microsemi SoftConsole vX.X. You can either accept the default name or use your own name. If you want to rename the SoftConsole Start Menu folder, type in the new name or browse to a folder that you would like to use. Click **Next**.
7. To have the installation program create a desktop or Quick Launch icon, select the appropriate check box. If no additional icons are required, clear the check box for both items. Click **Next**.

8. Click **Install** to complete the installation. During the installation, an information page appears.
9. Review the information and click **Next** to proceed.
10. Click **Finish** to complete the installation.

5. Is it necessary to have Administrator privileges to install SoftConsole?

Yes, Administrator privileges are required to install SoftConsole. If you are not logged in as an administrator and run the installer, the installation uses the administrator user's profile (found in the Documents and Settings folder). This creates a desktop shortcut on the administrative user's desktop and not on yours. Similarly, if you allow the installer to run SoftConsole automatically, on completion that instance of the application runs only for the administrative user's login; certain saved settings and data are accessible to the administrative user and not to you.

6. Is it advisable to create a workspace within an existing workspace or a project within an existing project folder?

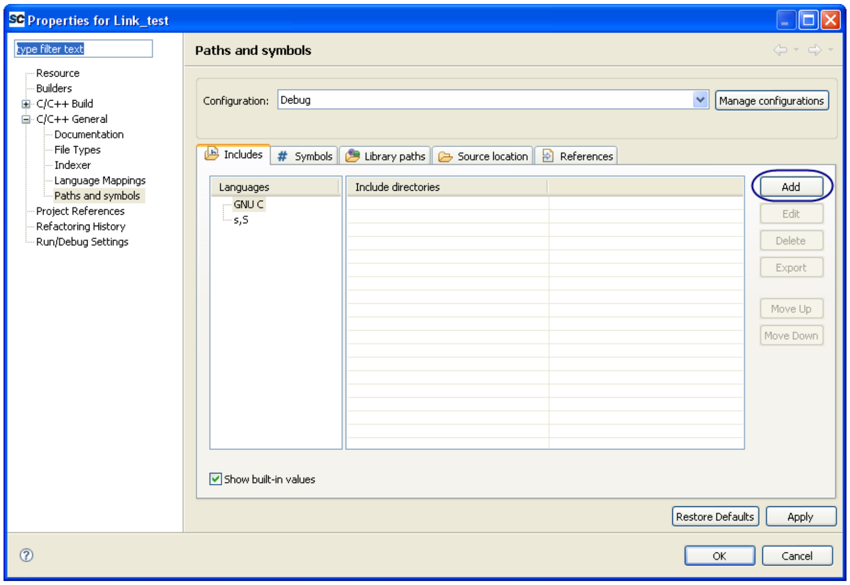
No, do not create a workspace within an existing workspace. Eclipse/CDT does not protect the user against the creation of a workspace inside a workspace or project folder and the recursive creation of deeply nested folder structures may result while importing files in such a context.

7. How to link files to a SoftConsole project that are not in SoftConsole workspace?

Files can be linked rather than importing them into the workspace. This can be useful if you wish to keep firmware in a specific folder which is outside the SoftConsole workspace. This works well for the SmartFusion device to compile firmware drivers, and if the SmartFusion MSS project is not within the SoftConsole project.

Follow the steps given below to link files to SoftConsole project:

1. Open the project properties dialog and select **Paths and symbols** under **C/C++ General** then click **Add** on the Includes tab.



Properties for Link_test

Paths and symbols

Configuration: Debug

Figure 1. Properties for Link Test

2. Click “**File System..**” in the Add directory path dialog box.

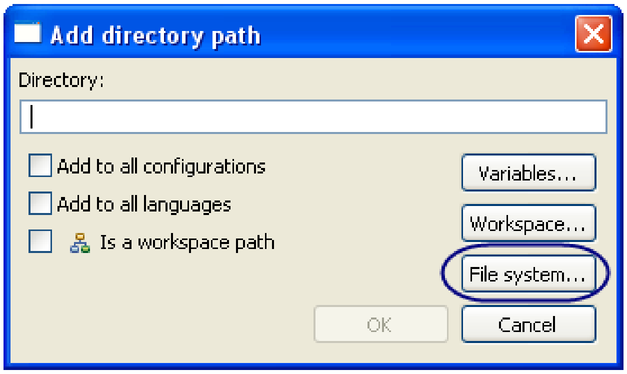
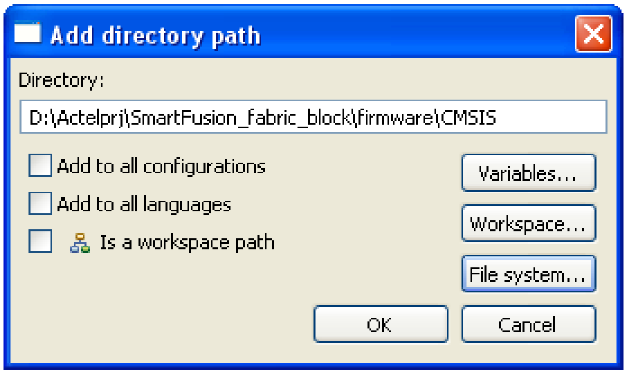


Figure 2. Add Directory Path

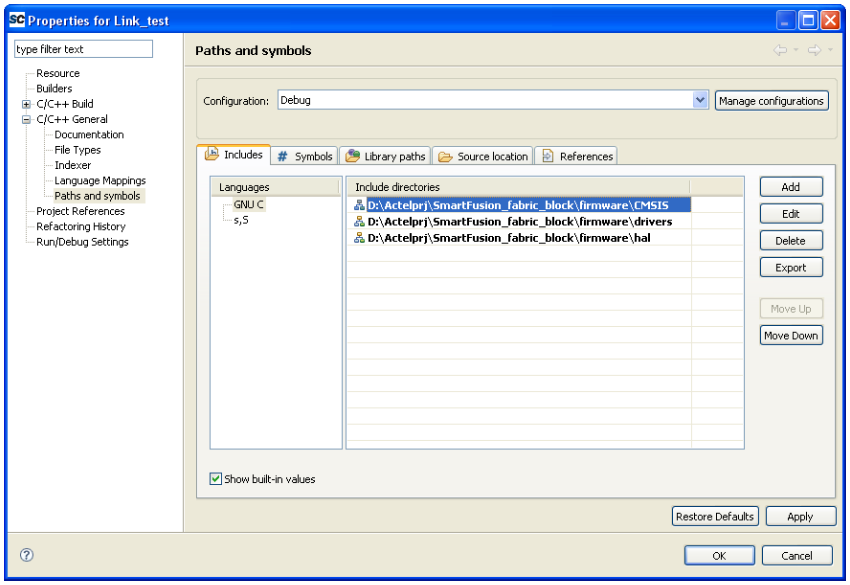
3. Navigate to the firmware folder and click **OK**.



D:\Actelprj\SmartFusion_fabric_block\firmware\CMSIS

Figure 3. Directory

4. Repeat as needed to link to all the firmware. Include paths will be visible in the **Properties** dialog box.



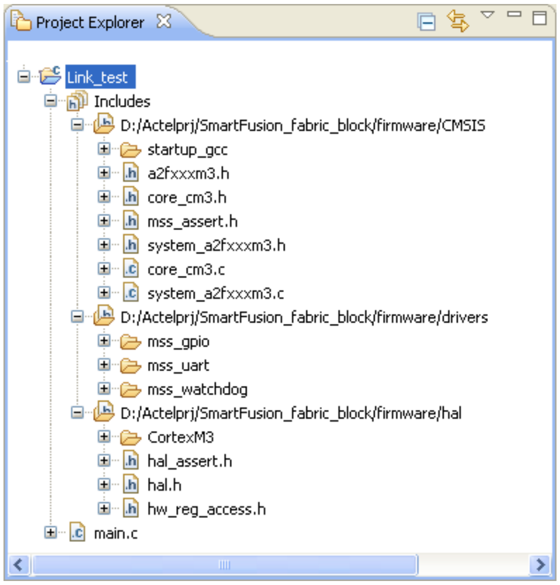
Configuration: Debug

Languages	Include directories
GNU C	D:\Actelprj\SmartFusion_fabric_block\firmware\CMSIS
s,S	D:\Actelprj\SmartFusion_fabric_block\firmware\drivers
	D:\Actelprj\SmartFusion_fabric_block\firmware\hal

☑ Show built-in values

Figure 4. Include Paths

5. When finished, click **OK**. The include paths will be shown in the SoftConsole Project Explorer (Figure 5).



```
Link_test
  Includes
    D:/Actelprj/SmartFusion_fabric_block/firmware/CMSIS
      startup_gcc
      a2fxxxm3.h
      core_cm3.h
      mss_assert.h
      system_a2fxxxm3.h
      core_cm3.c
      system_a2fxxxm3.c
    D:/Actelprj/SmartFusion_fabric_block/firmware/drivers
      mss_gpio
      mss_uart
      mss_watchdog
    D:/Actelprj/SmartFusion_fabric_block/firmware/hal
      CortexM3
      hal_assert.h
      hal.h
      hw_reg_access.h
  main.c
```

Figure 5. Project Explorer

8. How to exclude files present in a SoftConsole project from build?

To exclude files present in the SoftConsole project, follow the steps given below:

1. Right-click the required file and select “**Exclude from build…”.**

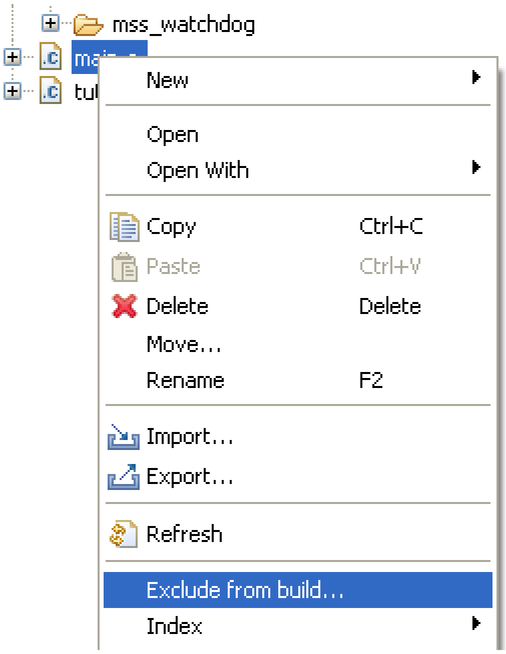
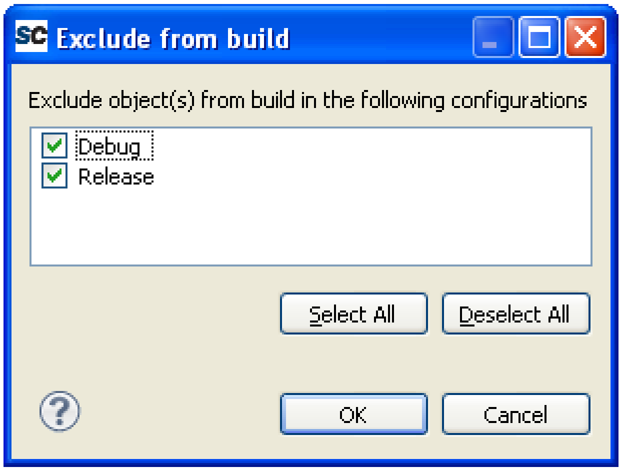


Figure 6. Properties Window

2. You can select the configuration build for which the exclusion needs to be done.


Exclude from build

Exclude object(s) from build in the following configurations

☑ Debug
☑ Release

Figure 7. Exclude from Build Window

3. The file that is excluded from the build is displayed differently in the Project Explorer – the symbol indicating the type of file is hollow:

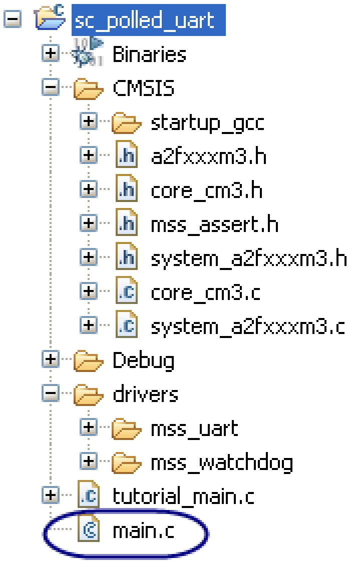


Figure 8. Main.c Window

4. When a build is performed, the file is excluded. To remove the exclusion, right-click on the file and click on "**Exclude from build**" option.

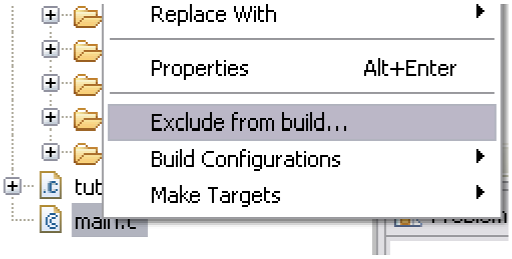
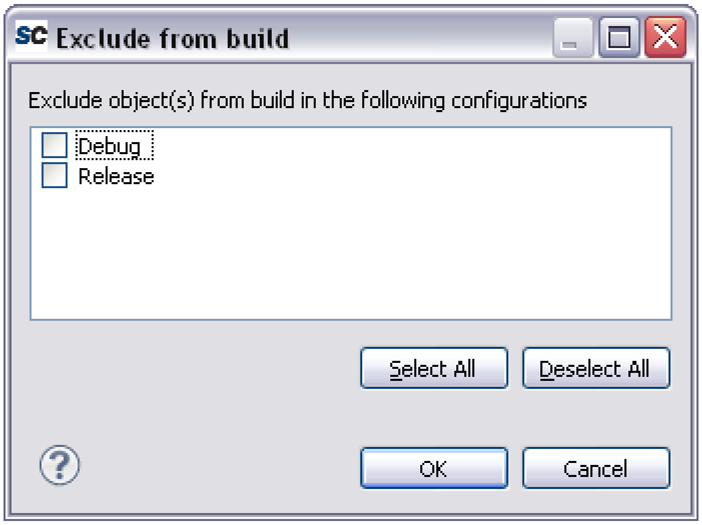


Figure 9. Options Window

5. Uncheck the options "**Debug**" and "**release**" and then click **ok**.



Exclude object(s) from build in the following configurations

□ Debug
□ Release

Figure 10. Uncheck Options

6. The file is displayed in the Project Explorer window by default and is included in the build.

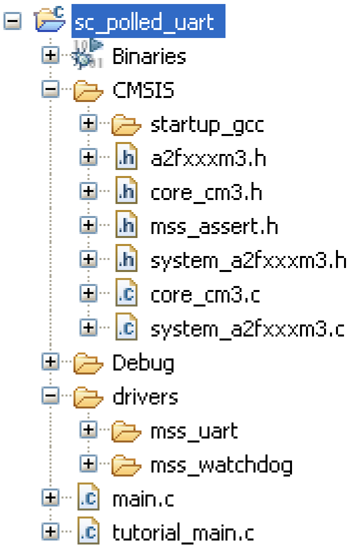
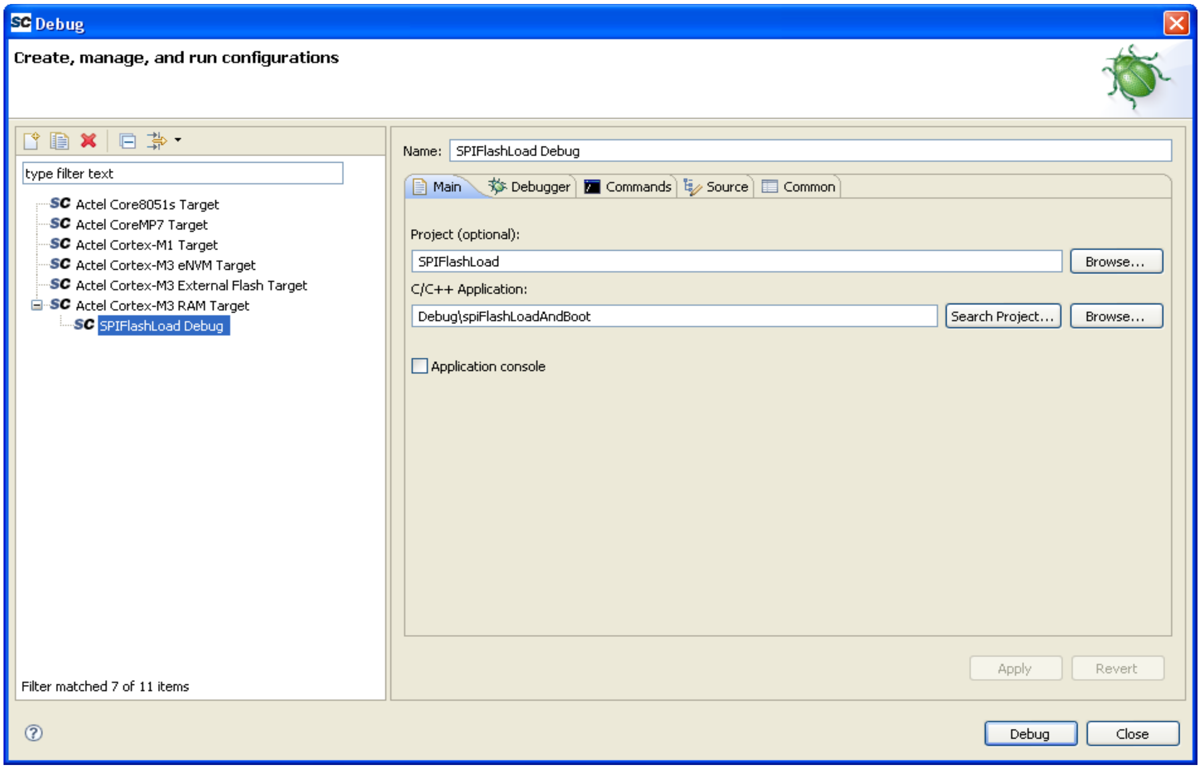


Figure 11. Project Window

9. How to rename the SoftConsole project and Debug target?

Projects can be renamed by selecting the project name in the SoftConsole Project Explorer, right-click the project file, and select **Rename**.

When projects are renamed, the debug target may still use the original project name for the debug application. For example, in Figure 12, a project that was named SPIFlashLoadAndBoot was renamed SPIFlashLoad. Note that the C/C++ Application field in the Debug dialog box still shows the original project name. Also, Hex and binary files that are generated use the original project name.



Name: SPIFlashLoad Debug

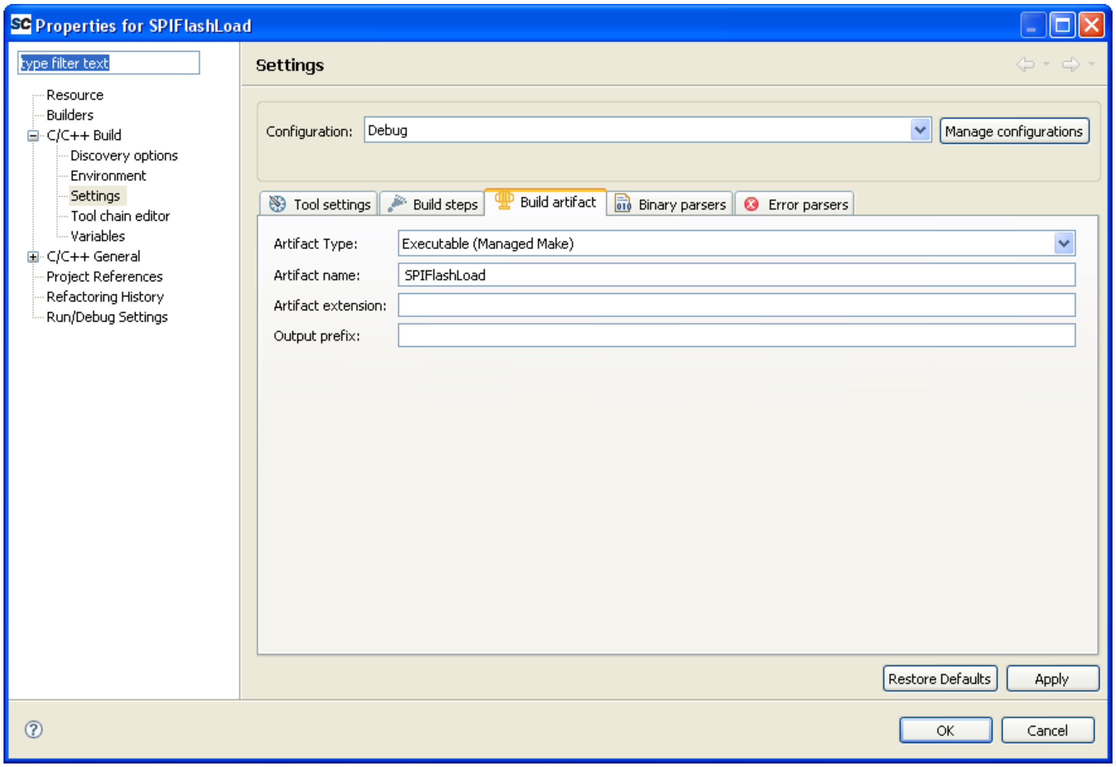
Project (optional): SPIFlashLoad

C/C++ Application: Debug\spiFlashLoadAndBoot

□ Application console

Figure 12. Debug Window

To correct this, open the project properties dialog box and select the **Build artifact** tab. Manually change the **Artifact name** to the new project name then, rebuild the project.



Configuration: Debug

Artifact Type: Executable (Managed Make)

Artifact name: SPIFlashLoad

Artifact extension:

Output prefix:

Figure 13. Properties for SPIFlashLoad

10. Does SoftConsole v2.3 and v3.1 support C++ projects?

No, SoftConsole versions 2.3 and 3.1 do not support C++ projects. However, SoftConsole v3.3 and later versions support C++ projects.

11. Does SoftConsole work with FlashPro Lite?

No. Only FlashPro3, FlashPro4, and the LC Programmer stick support the SoftConsole tool for debugging.

Note: Some Microsemi boards include the FlashPro3 circuitry in the board design.

12. Does the soft console compiler support thumb mode?

Yes. You can add the option -mthumb under the **Properties** ->**C/C++ Build** for your project. Insert it after the Command for both "GCC C Compiler" and "GCC C Linker". After that, click **Apply** or **OK**. Your project can be rebuilt using the '-mthumb' flag to make it compile your code for the thumb target and link in the thumb versions of crt0 and libraries like libc.

13. Can I use SoftConsole to write to external memory for CoreMP7?

No, SoftConsole cannot be used to write to an external memory for CoreMP7.

During debugging, SoftConsole communicates with the target processor through the hardware target board's FlashPro3 JTAG header through the FlashPro3 programmer/debugger. To enable debugging through the FlashPro3 header, appropriately configure certain aspects of your design:

For CoreMP7 debugging, the CoreMP7Bridge Debug configurable option must be set to RealView or FlashPro3. Your design must have RAM in the code memory space, as only debugging from RAM is supported.

- You also need to connect the UJTAG interface of CoreMP7Bridge to the top level of your design. Promote the UJTAG interface to the top level of the SmartDesign tool (or CoreConsole project), and route this interface to the top-level module of the overall design when the SmartDesign (or CoreConsole) subsystem is instantiated as a component in a higher level module.
- There is no need to apply pin assignments to the UJTAG signals as Microsemi's designer tool automatically recognizes that the JTAG pins of the device are used for these signals.

14. How can I obtain the .h and .c and HAL (Hardware Abstraction layer) files for my SoftConsole project?

You can use the firmware catalog which is installed along with Libero tool. Open Firmware catalog from **Start-> Programs > Libero > Firmware Catalog vx.x > Firmware Catalog**.

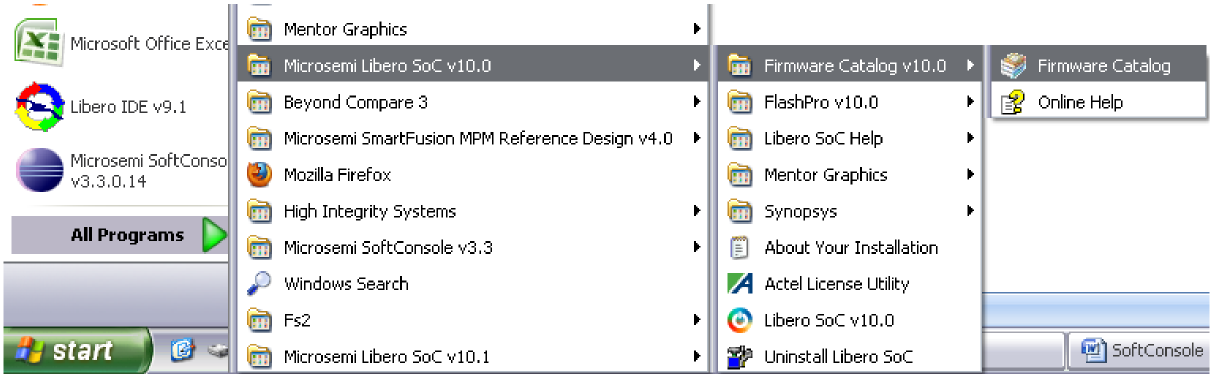
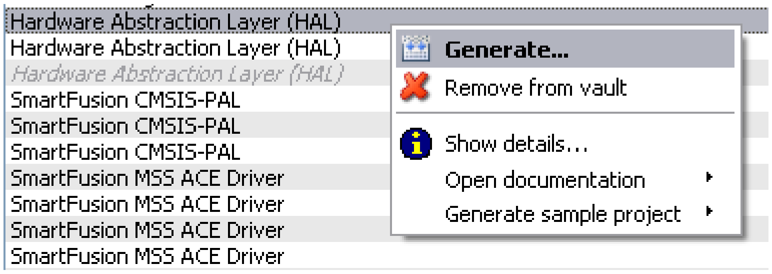


Figure 14. Programs List Window



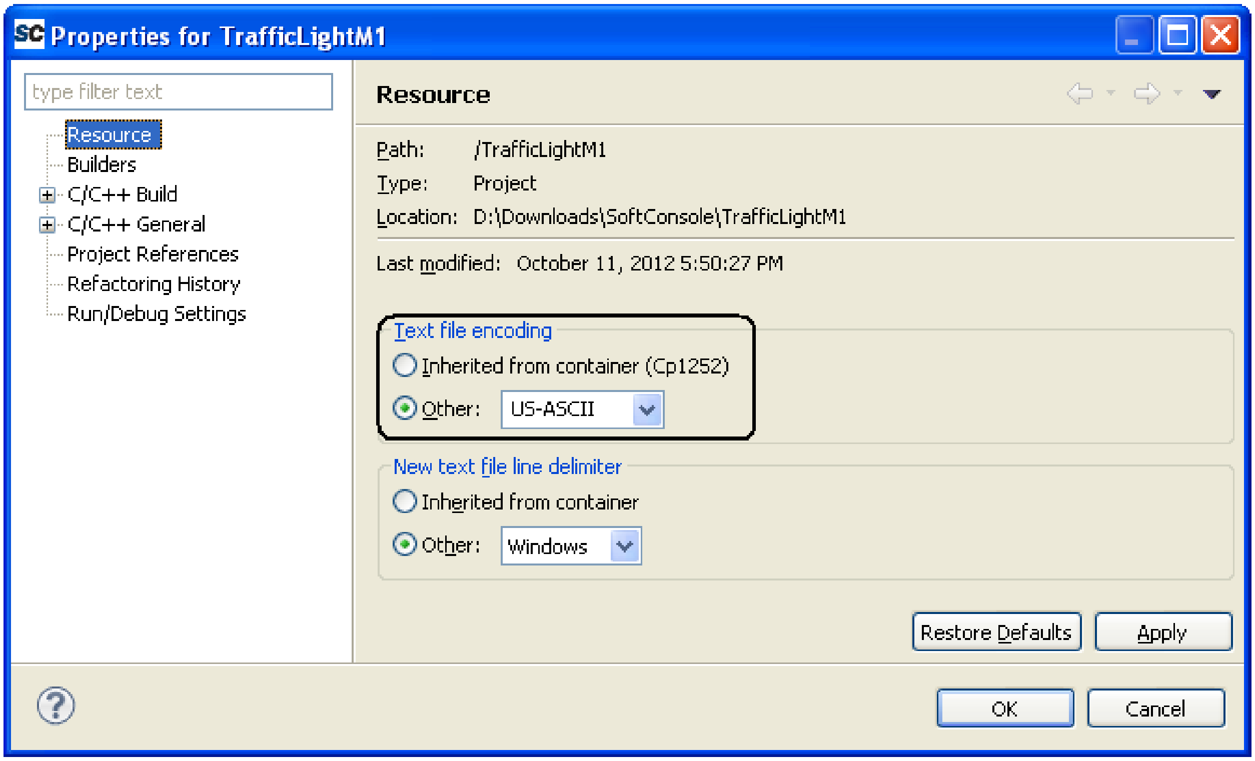
Hardware Abstraction Layer (HAL)
Hardware Abstraction Layer (HAL)
Hardware Abstraction Layer (HAL)
SmartFusion CMSIS-PAL
SmartFusion CMSIS-PAL
SmartFusion CMSIS-PAL
SmartFusion MSS ACE Driver
SmartFusion MSS ACE Driver
SmartFusion MSS ACE Driver
SmartFusion MSS ACE Driver

Figure 15. Selection Window

Click on “**Generate**” to generate HAL drivers. You can make platform.h and other header files to match your hardware. Use the memory map from your Libero SoC project as a guide.

15. Why do we get error message “No builder for clean” while cleaning a SoftConsole project?

This error occurs when the font used for text file encoding is not “US ASCII”. To fix the problem,right-click on the project and click on “**Properties**”. Then select **Resource**. In the Text file encoding field, check “**Other**”. Then select **US ASCII** from the pull-down menu.



Properties for TrafficLightM1

Resource

Path: /TrafficLightM1
Type: Project
Location: D:\Downloads\SoftConsole\TrafficLightM1

Last modified: October 11, 2012 5:50:27 PM

Text file encoding
○ Inherited from container (Cp1252)
● Other: US-ASCII

New text file line delimiter
○ Inherited from container
● Other: Windows

Figure 16. Properties for TrafficLightM1

16. Does SoftConsole with the Cortex-M1 support the C-runtime library?

Yes, it supports the C-runtime library. For any functions that you want to add, you can always add a compiler switch. For example, the standard C library is automatically included when using the linker scripts that we provide with SoftConsole. SoftConsole uses GNU GCC. Using linker scripts other than those supplied by Microsemi is not supported.

17. Is there any document which explains how to build a software project using SoftConsole so that it can run from Non-Volatile Memory (NVM)?

Yes, we have a document. Please refer to the following tutorial:
www.microsemi.com/soc/documents/CortexM1_Proc_SW_Tutorial_UG.pdf

18. How can I find the size of my executable code (eNVM space)? (ARM)

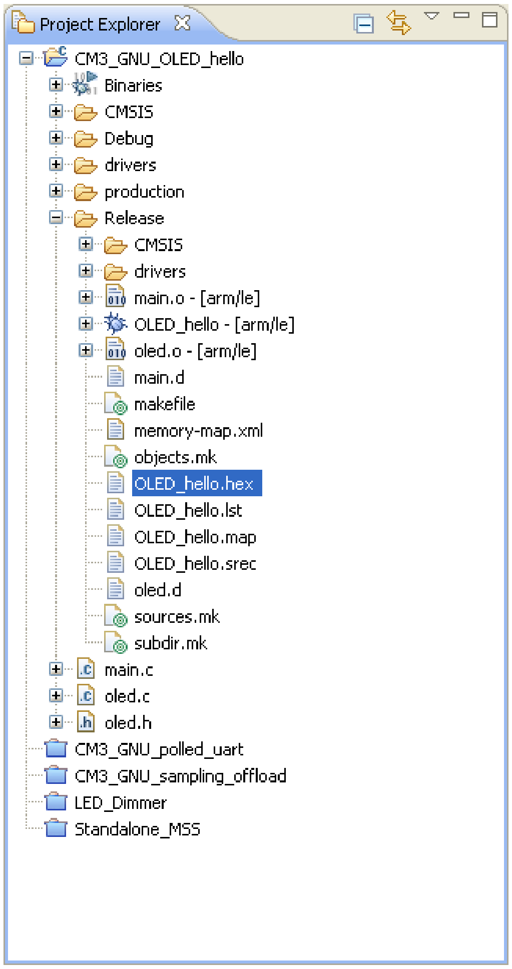
The easiest way is to the use the GNU arm-none-eabi-size command line utility provided with SoftConsole. Open a Windows command line box, move to the Debug directory of your project and enter "arm-none-eabi-size <project-name>", replacing <project name> with the name of the project. This returns two lines of text providing the size of the text, data, and bss sections. The executable size is the text size in bytes.

The arm-none-eabi-size executable is contained in the <drive>:\<SoftConsole v3.3 installation>\Sourcery-G++\bin folder.

Example:

To determine the size of the executable for the OLED_hello sample project, perform the following steps:

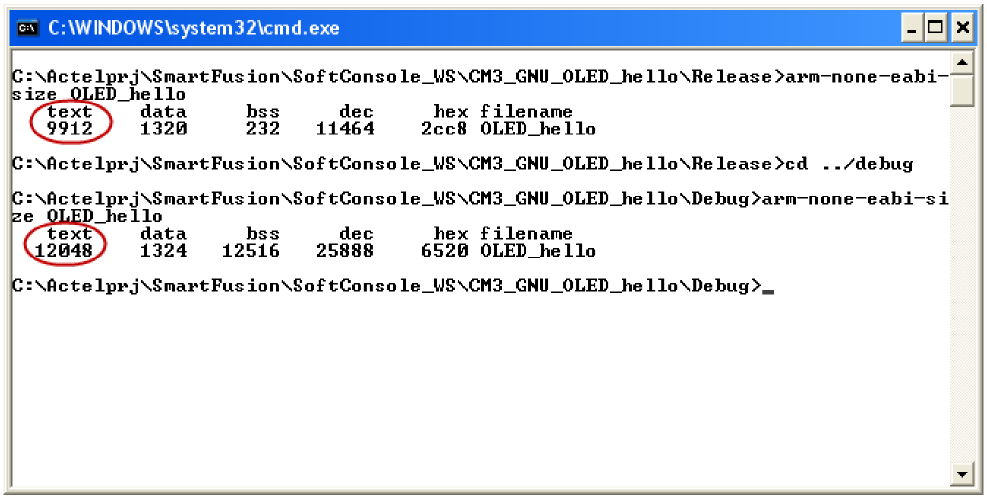
1. Perform a build for the debug or release configuration to generate the hex file OLED_hello.hex (Figure 17).



```
Project Explorer
CM3_GNU_OLED_hello
    Binaries
    CMSIS
    Debug
    drivers
    production
    Release
        CMSIS
        drivers
        main.o - [arm/le]
        OLED_hello - [arm/le]
        oled.o - [arm/le]
        main.d
        makefile
        memory-map.xml
        objects.mk
        OLED_hello.hex
        OLED_hello.lst
        OLED_hello.map
        OLED_hello.srec
        oled.d
        sources.mk
        subdir.mk
    main.c
    oled.c
    oled.h
CM3_GNU_polled_uart
CM3_GNU_sampling_offload
LED_Dimmer
Standalone_MSS
```

Figure 17. Project Explorer List

2. Open a DOS command prompt and move to the debug (or release) folder under the SoftConsole project. Then enter “arm-none-eabi-size OLED_hello”:



```
C:\Actelprj\SmartFusion\SoftConsole_WS\CM3_GNU_OLED_hello\Release>arm-none-eabi-
size OLED_hello
   text    data     bss     dec     hex filename
   9912    1320     232   11464    2cc8 OLED_hello

C:\Actelprj\SmartFusion\SoftConsole_WS\CM3_GNU_OLED_hello\Release>cd ../debug

C:\Actelprj\SmartFusion\SoftConsole_WS\CM3_GNU_OLED_hello\Debug>arm-none-eabi-si
ze OLED_hello
   text    data     bss     dec     hex filename
  12048    1324   12516   25888    6520 OLED_hello

C:\Actelprj\SmartFusion\SoftConsole_WS\CM3_GNU_OLED_hello\Debug>_
```

Figure 18. Command Window

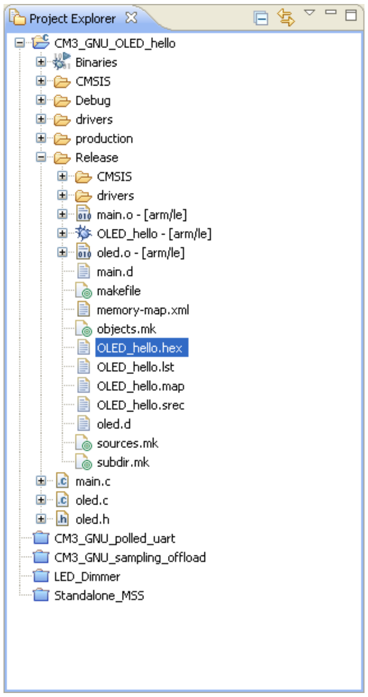
The executable size is the text size (circled in Red color in Figure 18).

Note: Ensure that the SoftConsole installation path (Ex: C:\Actel\Libero_v9.1\SoftConsole\Sourcery-G++\bin) is set in the Environment variables in your PC.

In SoftConsole 3.3, a Make target can be created to allow the arm-none-eabi-size OLED_hello function to run without opening a DOS command window.

Follow the steps given below:

1. Perform a build for the debug or release configuration to generate the hex file OLED_hello.hex (as shown in Figure 19).



```
CM3_GNU_OLED_hello
  Binaries
  CMSIS
  Debug
  drivers
  production
  Release
    CMSIS
    drivers
    main.o - [arm/le]
    OLED_hello - [arm/le]
    oled.o - [arm/le]
    main.d
    makefile
    memory-map.xml
    objects.mk
    OLED_hello.hex
    OLED_hello.lst
    OLED_hello.map
    OLED_hello.srec
    oled.d
    sources.mk
    subdir.mk
  main.c
  oled.c
  oled.h
CM3_GNU_polled_uart
CM3_GNU_sampling_offload
LED_Dimmer
Standalone_MSS
```

Figure 19. Projects List

2. Select the project name in the Make view (right side of the C/C++ perspective) then right-click and select **New.**

Note: called Make Targets in SoftConsole v3.3

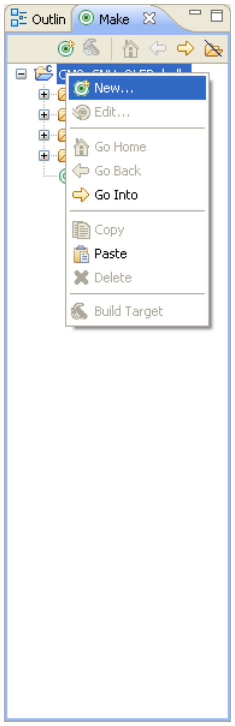
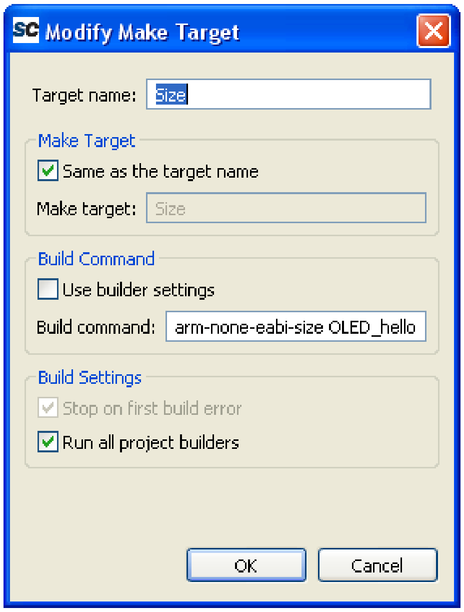


Figure 20. Project Creation

3. Enter the following in the **Modify Make Target** dialog box:

- Target Name: Enter a name (As shown in Figure 21. Here the target is named as **Size**)
- Build Command: Un-check Use Builder Settings; enter arm-none-eabi-size <name of executable>; in my example the executable is named OLED_hello
- Click **OK**



Modify Make Target

Target name: Size

Make Target

☑ Same as the target name

Make target: Size

Build Command

□ Use builder settings

Build command: arm-none-eabi-size OLED_hello

Build Settings

☑ Stop on first build error

☑ Run all project builders

Figure 21. Modify Target

To use this, select the build configuration and build the project. Click on the **Make Target** in the Make View.

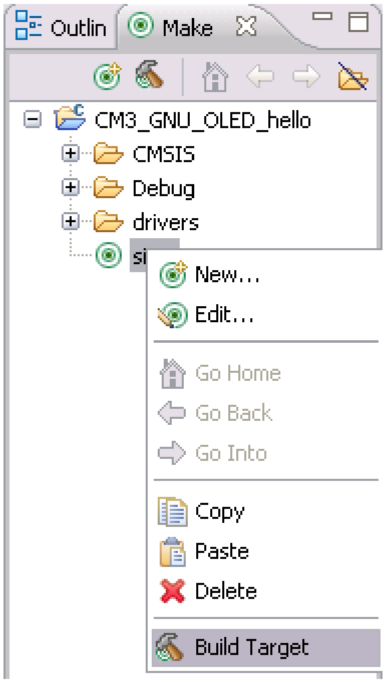
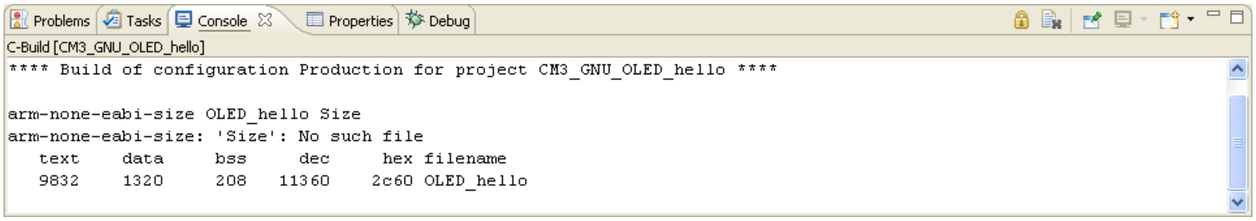


Figure 22. Build Target

The size is displayed in the Console view:



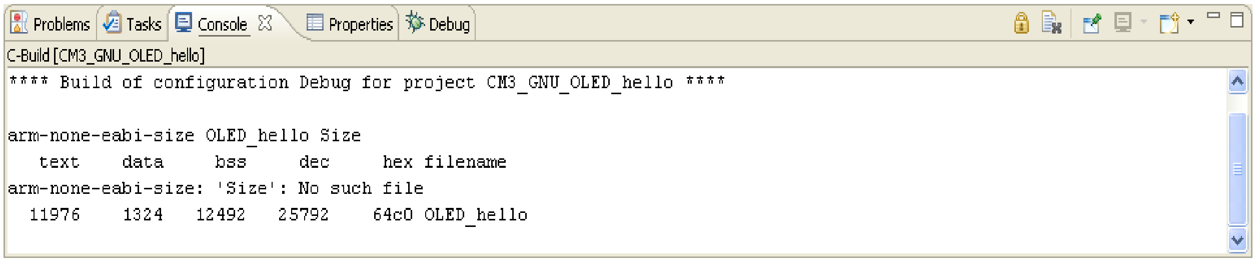
C-Build [CM3_GNU_OLED_hello]

```
**** Build of configuration Production for project CM3_GNU_OLED_hello ****

arm-none-eabi-size OLED_hello Size
arm-none-eabi-size: 'Size': No such file
   text    data     bss     dec     hex filename
   9832    1320     208   11360    2c60 OLED_hello
```

Figure 23. Console View 1

Note: The size shown in Figure 23 is in bytes.

If multiple build configurations are used (Debug, Release, etc.) set the active build configuration (**Build Configurations** > **Set Active** > <configuration name> then run the **Make Target**.



C-Build [CM3_GNU_OLED_hello]

```
**** Build of configuration Debug for project CM3_GNU_OLED_hello ****

arm-none-eabi-size OLED_hello Size
   text    data     bss     dec     hex filename
arm-none-eabi-size: 'Size': No such file
  11976    1324   12492   25792    64c0 OLED_hello
```

Figure 24. Console View 2

19. How can I find the size of my executable code (eNVM space) (8051s)?

The GNU binutils size utility should be run on the ELF file for example.

```
C:\dev\Sandbox\SoftConsole\test33\test8051\Debug>c8051-elf-size
test8051.elf

   text    data     bss     dec     hex filename
   9238       0     757    9995    270b test8051.elf
```

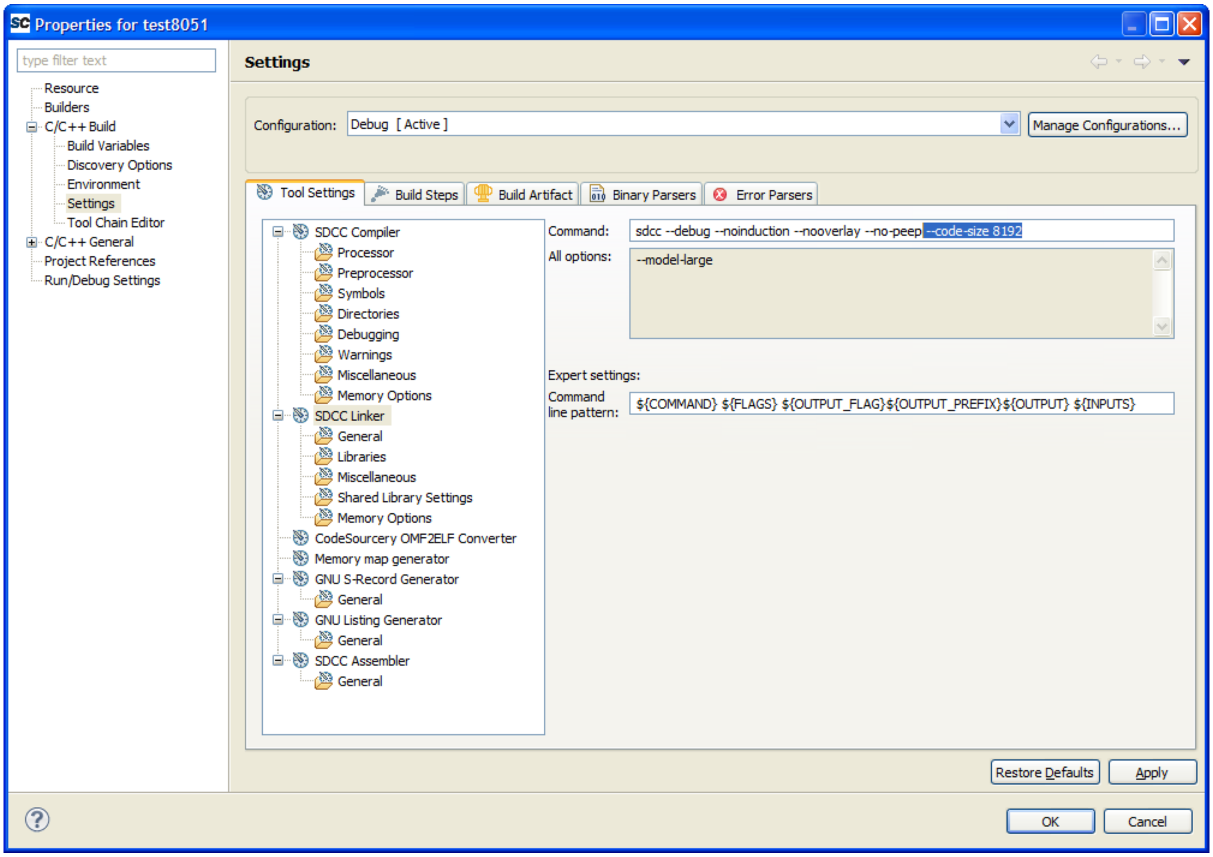
An easier and more reliable way to check the size of the program may be to look at the `*.mem` file generated by SDCC in the *Debug* or *Release* folder – e.g. my test8051s.mem has:

```
Internal RAM layout:
      0 1 2 3 4 5 6 7 8 9 A B C D E F
0x00:|0|0|0|0|0|0|0|0| | | | | | | | |
0x10:| | | | | | | | |a|a|a|a|a|a|a|a|
0x20:|b|b|b|b|b|b|b|b|b|b|b|b|b|b|b|b|
0x30:|b|b|b|b|S|S|S|S|S|S|S|S|S|S|S|S|
0x40:|S|S|S|S|S|S|S|S|S|S|S|S|S|S|S|S|
0x50:|S|S|S|S|S|S|S|S|S|S|S|S|S|S|S|S|
0x60:|S|S|S|S|S|S|S|S|S|S|S|S|S|S|S|S|
0x70:|S|S|S|S|S|S|S|S|S|S|S|S|S|S|S|S|
0x80:|S|S|S|S|S|S|S|S|S|S|S|S|S|S|S|S|
0x90:|S|S|S|S|S|S|S|S|S|S|S|S|S|S|S|S|
0xa0:|S|S|S|S|S|S|S|S|S|S|S|S|S|S|S|S|
0xb0:|S|S|S|S|S|S|S|S|S|S|S|S|S|S|S|S|
0xc0:|S|S|S|S|S|S|S|S|S|S|S|S|S|S|S|S|
0xd0:|S|S|S|S|S|S|S|S|S|S|S|S|S|S|S|S|
0xe0:|S|S|S|S|S|S|S|S|S|S|S|S|S|S|S|S|
0xf0:|S|S|S|S|S|S|S|S|S|S|S|S|S|S|S|S|

0-3:Reg Banks, T:Bit regs, a-z:Data, B:Bits, Q:Overlay, I:iData,
S:Stack, A:Absolute

Stack starts at: 0x34 (sp set to 0x33) with 204 bytes available.

Other memory:
   Name             Start    End      Size     Max
   ---------------- -------- -------- -------- --------
   PAGED EXT. RAM                         0      256
   EXTERNAL RAM     0x0000   0x02c1     706    65536
   ROM/EPROM/FLASH  0x0000   0x2415    9238    65536
```

Note: The code size reported by both of these methods tally in this case.

20. How do I tell the SDCC compiler how much CODE space is implemented?

Use the SDCC linker `--code-size` flag (Refer to the chapter “Using SDCC” in http://sdcc.sourceforge.net/doc/sdccman.pdf).



Configuration: Debug [Active]

Command: sdcc --debug --noinduction --nooverlay --no-peep --code-size 8192

All options: --model-large

Expert settings:

Command line pattern: ${COMMAND} ${FLAGS} ${OUTPUT_FLAG}${OUTPUT_PREFIX}${OUTPUT} ${INPUTS}

Figure 25. Properties for Test8051

21. How do I tell the SDCC compiler how much DATA space is implemented?

Some additional useful SDCC linker options are:

[**--iram-size** <Value>]

- It causes the linker to check if the internal RAM usage is within limits of the given value.

[**--xram-size** <Value>]

- It causes the linker to check if the external RAM usage is within limits of the given value.

[**--stack-size** <Value>]

- It causes the linker to check if minimum <Value> bytes for stack are available or not.

22. Is it possible to use SoftConsole debug with the core8051s via ‘user I/O’ debug setting instead of ‘UJTAG’?

No, SoftConsole only uses JTAG interface for debugging. However, Keil works with FS2 debugger as shown in the application note provided in the following: www.microsemi.com/soc/documents/CORE8051s_AC354_AN.pdf (for AX devices).

23. What is the significance of various styles followed while declaring the "#include" statements in the code?

Usually the #include statement is declared in the following 3 different styles each having its own significance:

1. #include <stddef.h>

The <> brackets indicates that this is a library file and that the compiler should look for this in the standard location for library files.

2. #include "cpu_types.h"

The quote mark says this is a user file and compiler should look in user's project for this using whatever 'include paths' have been defined. This is probably the most common way of using #include.

3. #include "../../CMSIS/a2fxxxm3.h"

This is a user file and the '../../' indicates a specific path in user's project.

24. How is the printf function implemented and how does it perform the operation of printing characters through UART?

Printf( ) function has been implemented for SmartFusion devices. printf() function uses Newlib C library. It then calls a board support package (BSP) that can write output on the serial port.
The SmartFusion CMSIS-PAL supports this redirection of Newlib printf() output to the SmartFusion MSS UART0 using the ACTEL_STDIO_THRU_UART and ACTEL_STDIO_BAUD_RATE pre-processor symbols in the CMSIS/startup_gcc/newlib_stubs.c source file.
So, basically during linking process, printf() function is linked to the MSS_UART_polled_tx() function defined under newlib_stubs.c file.

25. Is there a provision in SoftConsole which helps in optimization of code?

Yes. Here are the different optimization techniques:

1. "Fine grained" linking
It is possible to enable more "fine grained" linking by modifying the project build properties under C/C++ Build > Settings …

a) GNU C Compiler > Miscellaneous > Other flags: add -ffunction-sections -fdata-sections

This causes the compiler to omit every function and data symbol in its own individual section rather than omitting sections potentially containing multiple symbols with the whole section linked in if ANYTHING in the that section is referenced.

b) GNU C Linker > Miscellaneous > Linker flags: add -gc-sections

This allows the linker to omit ("garbage collect") sections that are not used/ referenced. Used in combination with the compilation flags above it normally results in a significantly smaller linked executable due to allowing for more fine grained compilation and linking.
For example view the following website for discussion of these options:
http://www.avrfreaks.net/index.php?name=PNphpBB2&file=viewtopic&p=676388

Some care must be taken so that otherwise unreferenced but still required sections are not omitted causing problems.

2. Optimization

Please use optimization levels given in SoftConsole C/C++ Build settings as explained below:
C/C++ Build > **Settings** > **GNU C Compiler** > **Optimization** > **Optimization Level**: Optimize more (-O2)
or
C/C++ Build > Settings > GNU C Compiler > Miscellaneous > Other flags: add -Os

3. Newlib usage
a) Use the integer only *iprintf() APIs rather than the "full blown" *printf() ones as the former are significantly smaller than the latter – e.g. siprintf() etc.
Some useful links:
http://sourceware.org/newlib/libc.html#Stdio
http://sourceware.org/newlib/libc.html#siprintf
http://www.eetimes.com/discussion/other/4024637/Embedding-with-GNU-newlib

b. Do not use floats/doubles unnecessarily. Using floats/doubles causes additional library support to be pulled in which could be avoided. Wherever possible, use only integer types and use the integer only *iprintf() support in newlib.

26. When I debug my project; I get an error message saying "Memory map not found". How do I resolve it?

If you forget to build a project or the <build-folder>/memory-map.xml file is missing for some other reason then you will get the error saying "Memory map not found". To resolve it, you will need to build the project before debugging works. Note that you will also need to manually terminate the failed debug session because it remains running even though it is no use.

27. Does SoftConsole allow you to debug via RealView ICE debug Hardware?

No; SoftConsole only allows you to debug with the FlashPro3.

In general, the ARM processors we offer (Cortex-M1 or CoreMP7) allow debug via one of the two options below:

- FlashPro3 and SoftConsole

www.microsemi.com/soc/products/software/softconsole/default.aspx

- RealView ICE and RealView Software (debugger + development suite)

http://www.arm.com/products/tools/debug-adapters/rvi-rvt2.php

So, in order to use the RealView ICE debug hardware, you would need to switch to the RealView

Software tools instead of SoftConsole.

28. Which tool has to be used for zipping SoftConsole Projects

When zipping SoftConsole projects, use Windows Explorer's *Send To* > *Compressed (zipped) folder* or 7Zip. Do not use WinZip or any other tool that omits files/folders whose names start with a dot (e.g. .metadata).

29. Where can I find more information on how to use SoftConsole?

The SoftConsole user guide contains detailed information on how to use the SoftConsole:

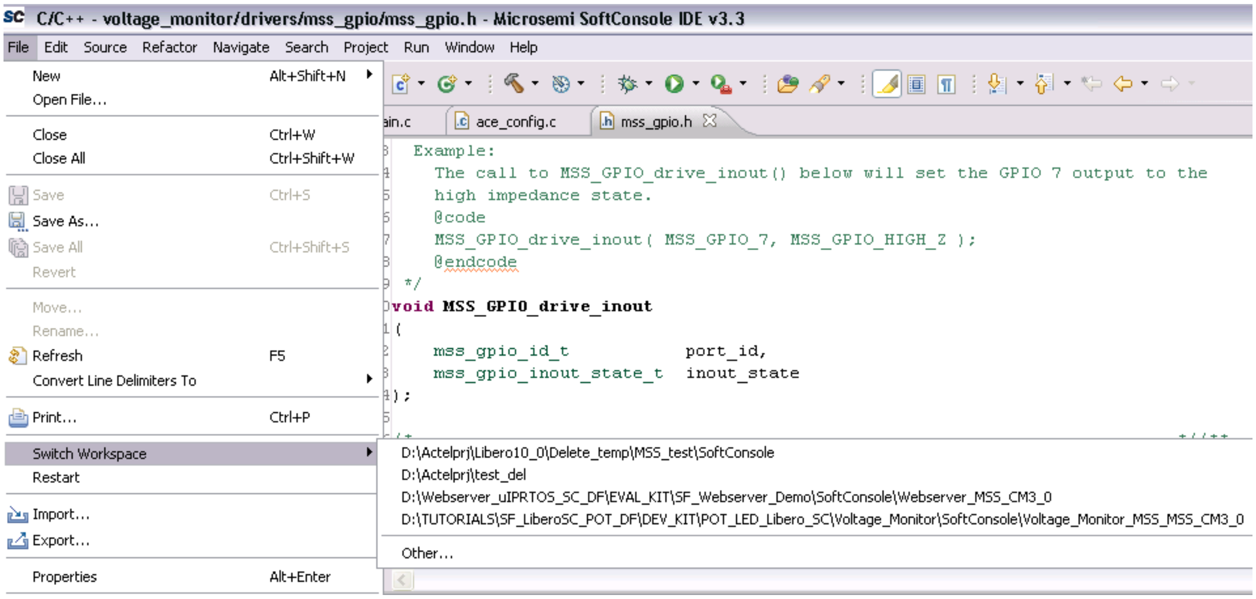
www.microsemi.com/soc/documents/SoftConsole_UG.pdf

30. Does SoftConsole work on Windows x64 system?

SoftConsole is not tested on windows x64 systems. However, no repeatable issues have been reported.

31. How to switch between two workspaces in SoftConsole environment?

To switch from an existing workspace to another, click on **File** and then select **Switch workspace** option as shown below:


SC C/C++ - voltage_monitor/drivers/mss_gpio/mss_gpio.h - Microsemi SoftConsole IDE v3.3
File Edit Source Refactor Navigate Search Project Run Window Help
New Alt+Shift+N
Open File...
Close Ctrl+W
Close All Ctrl+Shift+W
Save Ctrl+S
Save As...
Save All Ctrl+Shift+S
Revert
Move...
Rename...
Refresh F5
Convert Line Delimiters To
Print... Ctrl+P
Switch Workspace
Restart
Import...
Export...
Properties Alt+Enter
ain.c
ace_config.c
mss_gpio.h
D:\Actelprj\Libero10_0\Delete_temp\MSS_test\SoftConsole
D:\Actelprj\test_del
D:\Webserver_uIPRTOS_SC_DF\EVAL_KIT\SF_Webserver_Demo\SoftConsole\Webserver_MSS_CM3_0
D:\TUTORIALS\SF_LiberoSC_POT_DF\DEV_KIT\POT_LED_Libero_SC\Voltage_Monitor\SoftConsole\Voltage_Monitor_MSS_MSS_CM3_0
Other...

```
 Example:
   The call to MSS_GPIO_drive_inout() below will set the GPIO 7 output to the
   high impedance state.
   @code
   MSS_GPIO_drive_inout( MSS_GPIO_7, MSS_GPIO_HIGH_Z );
   @endcode
 */
void MSS_GPIO_drive_inout
(
   mss_gpio_id_t           port_id,
   mss_gpio_inout_state_t  inout_state
);
```

Figure 26. Switch Workspace

Select the required workspace. The existing workspace will be closed and a new SoftConsole workspace will be opened.

Microsemi Corporation (NASDAQ: MSCC) offers a comprehensive portfolio of semiconductor solutions for: aerospace, defense and security; enterprise and communications; and industrial and alternative energy markets. Products include high-performance, high-reliability analog and RF devices, mixed signal and RF integrated circuits, customizable SoCs, FPGAs, and complete subsystems. Microsemi is headquartered in Aliso Viejo, Calif. Learn more at **www.microsemi.com**.

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo CA 92656 USA
Within the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996