

# Configuring Compiler Options for Optimal Performance of ColdFire Devices

## 1 Introduction

This document provides two sets of options in the CodeWarrior tools to produce optimal performance of the ColdFire devices. One set optimizes speed; another set optimizes code size.

### Contents

1 Introduction .....	1
2 Optimizing Speed .....	1
3 Optimizing Code Size .....	5

## 2 Optimizing Speed

To optimize ColdFire devices for speed you can configure compiler settings from:

- MCU 10.x Eclipse IDE
- Command Line

---

**NOTE** The following procedure assumes that you have already created a project for a ColdFire device.

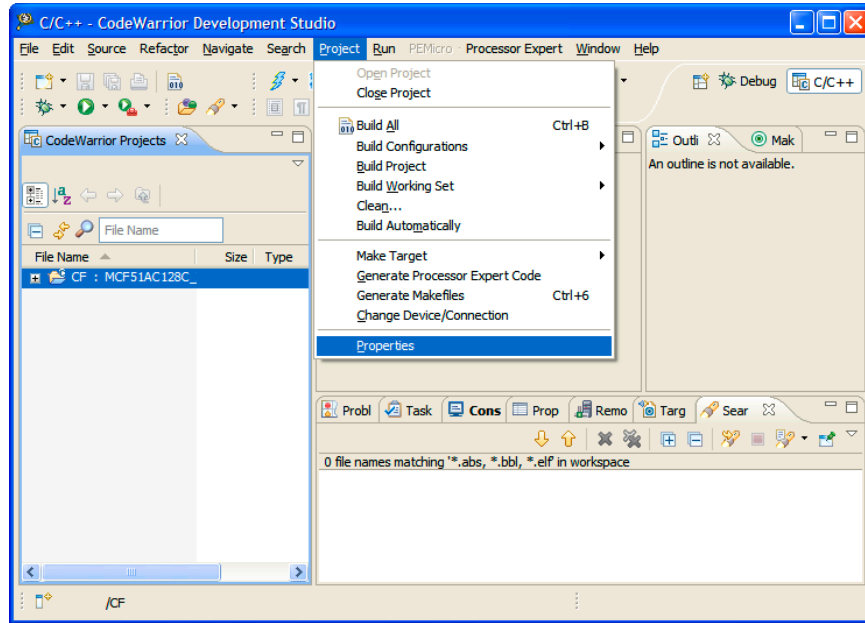
---

## 2.1 Optimizing Speed from MCU 10.x Eclipse IDE

To optimize ColdFire devices for speed:

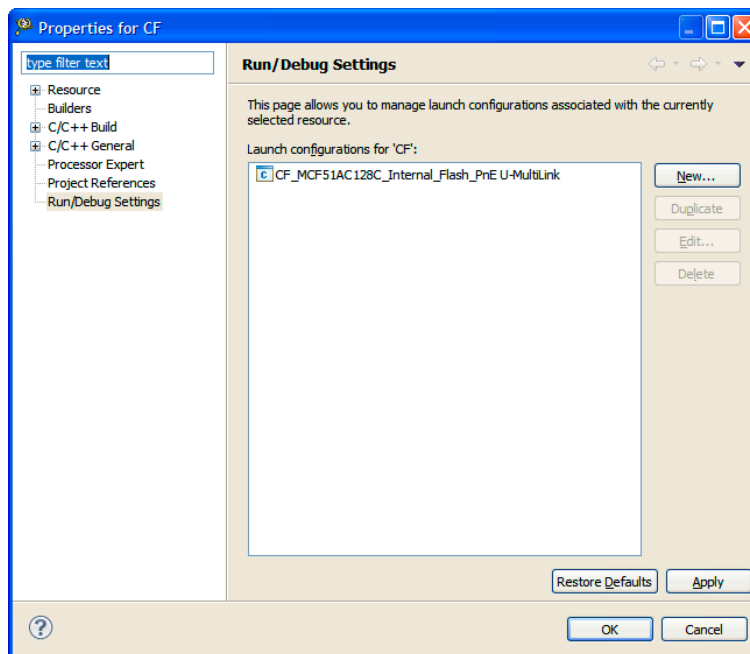
1. Select a ColdFire project in the **CodeWarrior Projects** view.
2. Select **Project > Properties**.

Figure 1. Project Menu



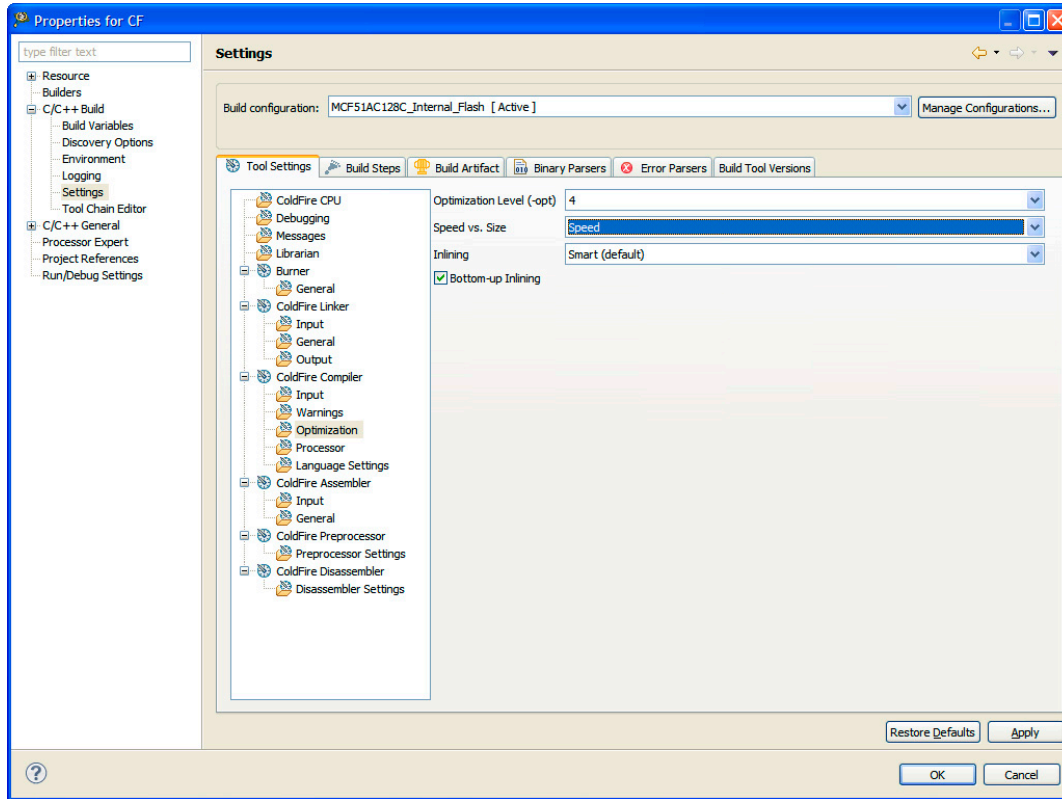
The **Properties for <project\_name>** window appears.

Figure 2. Properties for <project\_name> Window



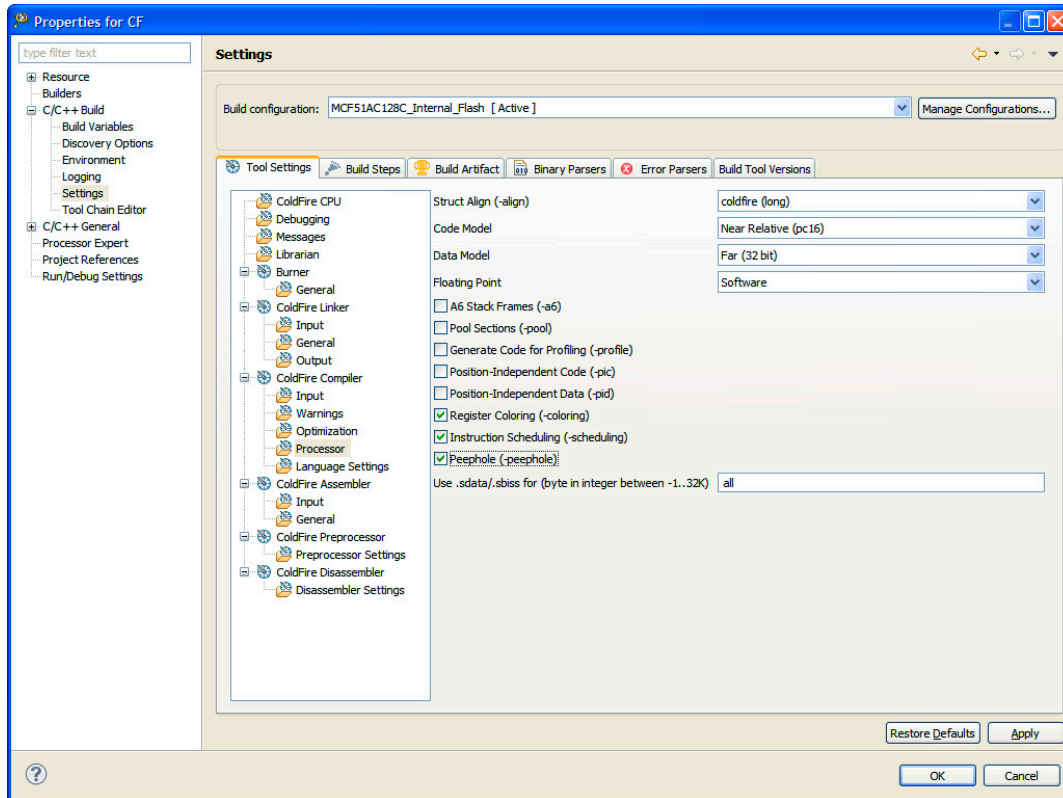
3. Select **C/C++ Build > Settings > ColdFire Compiler > Optimization**.
4. From the **Optimization Level (-opt)** drop-down list select **4**.
5. From the **Speed vs. Size** drop-down list select **Speed**.

**Figure 3. ColdFire Compiler > Optimization Panel**



6. Select **C/C++ Build > Settings > ColdFire Compiler > Processor**.
7. Check the following checkboxes.
  - **Register Coloring (-coloring)**
  - **Scheduling (-scheduling)**
  - **Peephole (-peephole)**

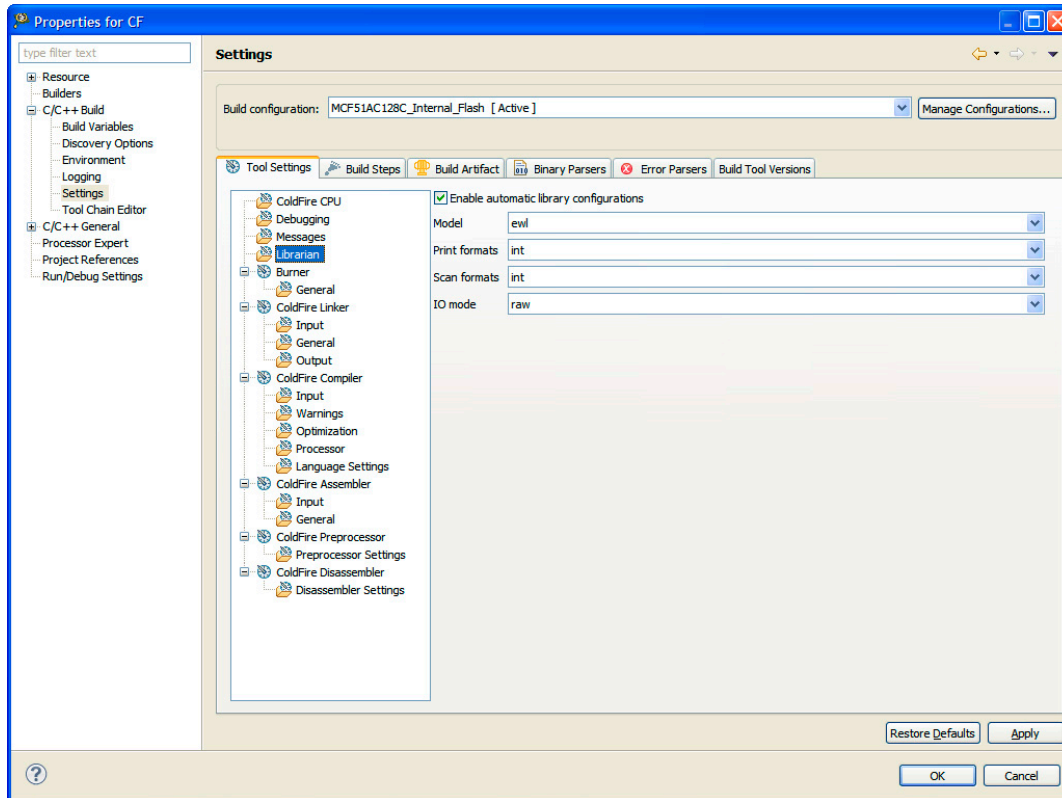
Figure 4. ColdFire Compiler > Processor



**NOTE** If the current project can be accommodated with a smaller library following is the best configuration but can be modified as per the requirement. However, the following steps are optional.

8. Select **C/C++ Build > Settings > Librarian**.
9. Check the **Enable library automatic configurations** checkbox.
10. From the **Model** drop-down list select **ewl**.
11. From the **Print formats** drop-down list select **int**.
12. From the **Scan formats** drop-down list select **int**.
13. From the **IO mode** drop-down list select **raw**.

Figure 5. Settings &gt; Librarian Panel



14. Click **Apply**.

15. Click **OK**.

---

**NOTE** Alternatively, select **Project > Properties > C/C++ Build > Settings > ColdFire Compiler > Language Settings** and select **File** from the **IPA** drop-down list.

---

## 2.2 Optimizing Speed from Command Line

From the command line, the compiler should get these options

```
-opt level=4 -opt speed -coloring -scheduling -peephole
```

(Optional)

```
-lavender model=ewl ,print=int ,scan=int ,io=raw
```

## 3 Optimizing Code Size

To optimize ColdFire devices for code size you can configure compiler settings from:

- MCU 10.x Eclipse IDE
- Command Line

---

**NOTE** The following procedure assumes that you have already created a project Optimizing Code Size from MCU 10.x Eclipse IDE.

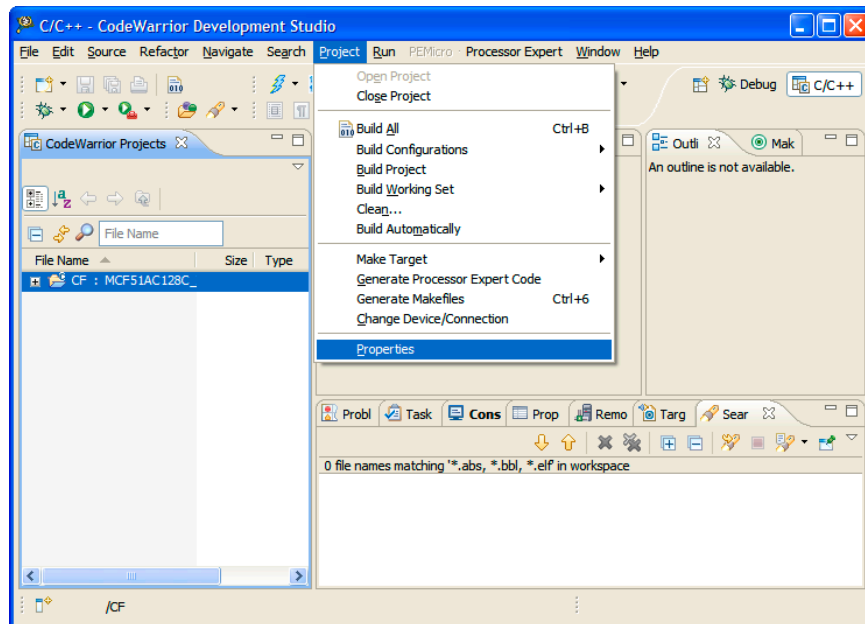
---

### 3.1 Optimizing Size from MCU 10.x Eclipse IDE

To optimize ColdFire devices for code size:

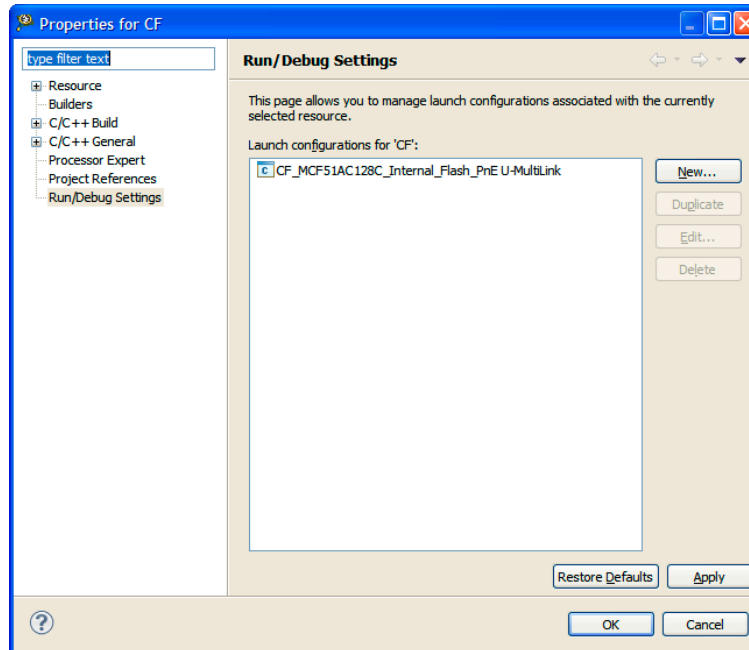
1. Select a ColdFire project in the **CodeWarrior Projects** view.
2. Select **Project > Properties**.

**Figure 6. Project Menu**



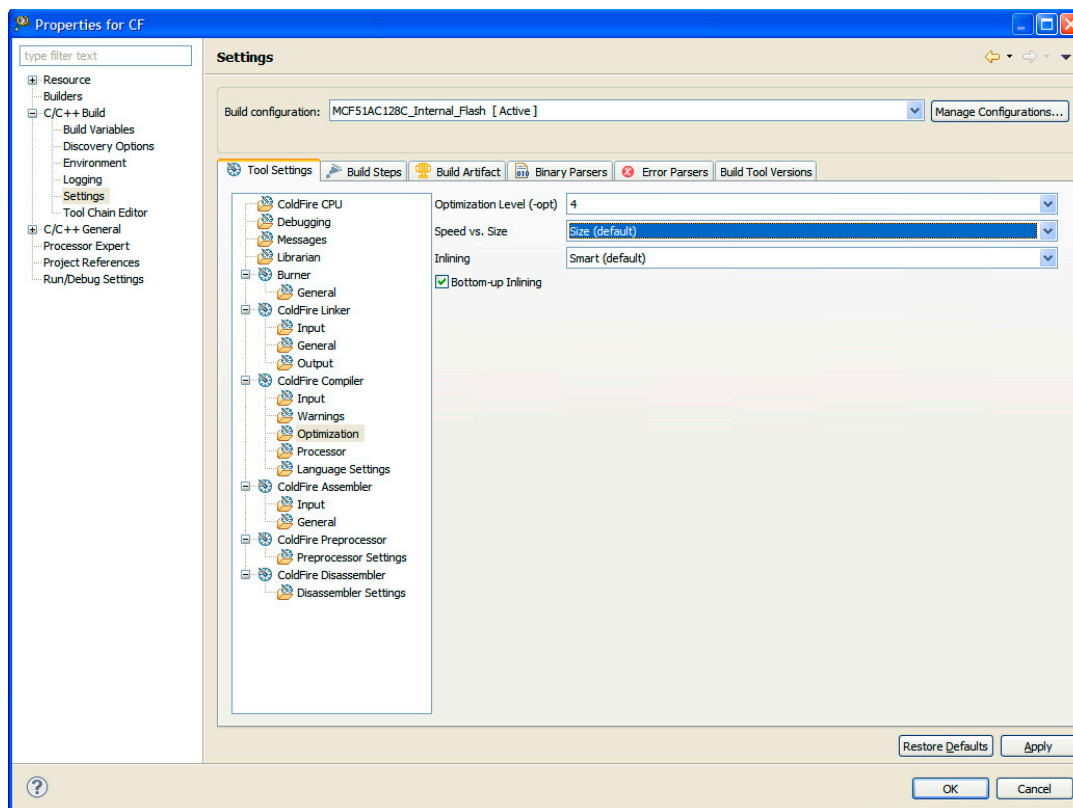
The **Properties for <project\_name>** window appears.

Figure 7. Properties for &lt;project\_name&gt; Window



3. Select **C/C++ Build > Settings > ColdFire Compiler > Optimization**.
4. From the **Optimization Level (-opt)** drop-down list select **4**.
5. From the **Speed vs. Size** drop-down list select **Size (default)**.
6. Select **C/C++ Build > Settings > ColdFire Compiler > Processor**.
7. Check the following checkboxes.
  - **Register Coloring (-coloring)**
  - **Scheduling (-scheduling)**
  - **Peephole (-peephole)**

Figure 8. ColdFire Compiler > Optimization Panel

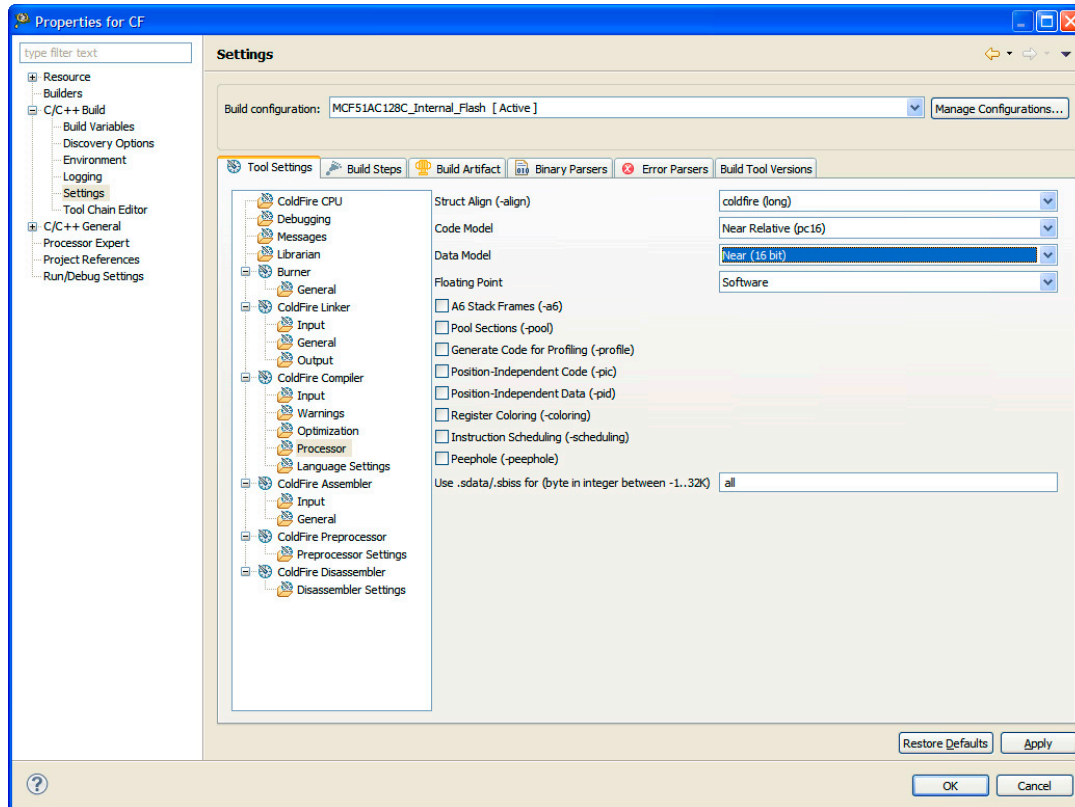


(Optional (1)), if the current project can fit its code / data in 16-bit following is the best configuration but can be modified as per the requirement.

8. From the **Code Model** drop-down list select **Near Relative (pc16)**.
9. From the **Data Model** drop-down list select **Near (16 bit)**.



Figure 9. ColdFire Compiler &gt; Processor Panel



(Optional (2)), depending on the complexity of the compiled code, one can use A6 Stack Frames, but with the cost of reserving A6 register. If the code is too complex and requires more data registers, reserving A6 might come with register allocation penalties, resulting in worse size as before. Using this option is not predictable and should be used only if better size is achieved.

10. Check the **A6 Stack Frames (-a6)** checkbox.

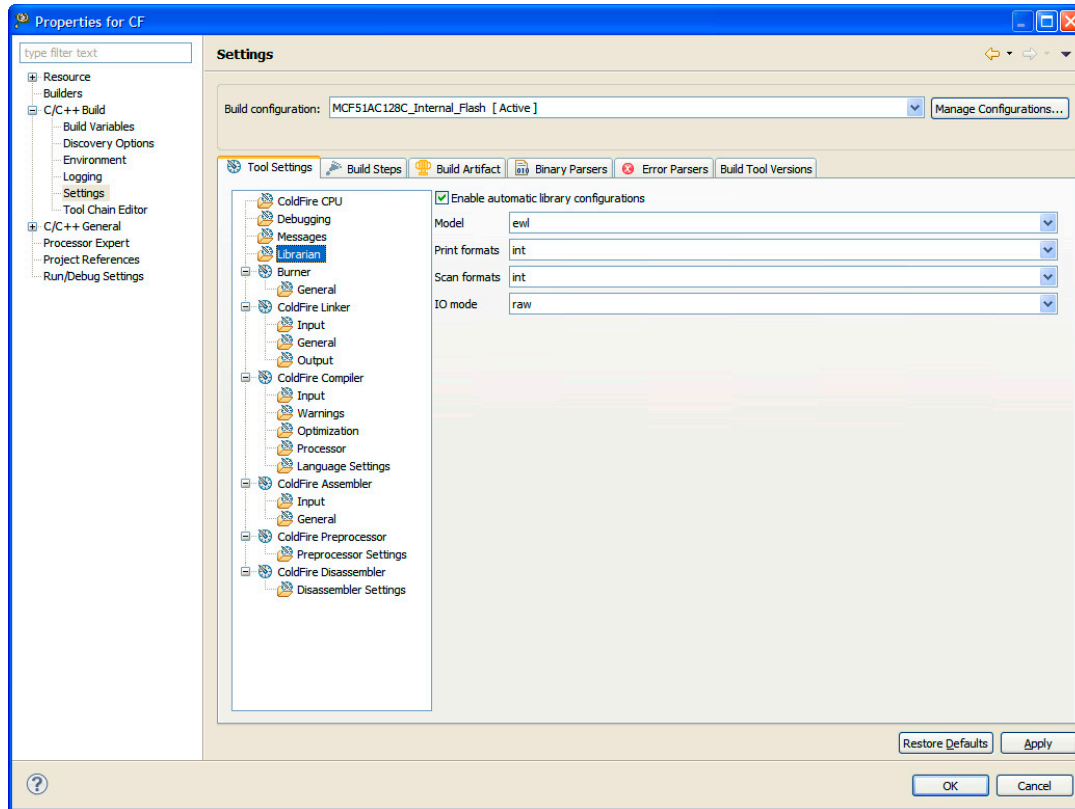
(Optional (3)), depending on the complexity of the compiled code, one can use Small Data Areas `.sdata/.sbss`, but with the cost of reserving A5 register. If the code is too complex and requires more data registers, using SDA might come with register allocation penalties, resulting in worse size as before.

11. Set the **Use `.sdata/.sbss` for (byte in integer between -1..32K)** option as per the requirement.

(Optional (4)), if the current project can be accommodated with a smaller library following is the best configuration but can be modified as per the requirement.

12. Select **C/C++ Build > Settings > ColdFire Compiler > Librarian**.

Figure 10. ColdFire Compiler > Librarian Panel



13. Check the **Enable library automatic configurations** checkbox.
14. From the **Model** drop-down list select **ewl**.
15. From the **Print formats** drop-down list select **int**.
16. From the **Scan formats** drop-down list select **int**.
17. From the **IO Mode** drop-down list select **raw**.
18. Click **Apply**.
19. Click **OK**.

---

**NOTE** Alternatively, select **Project > Properties > C/C++ Build > Settings > ColdFire Compiler > Language Settings** and select **File** from the **IPA** drop-down list.

---

### 3.2 Optimizing Speed from Command Line

From the command line, the compiler should get these options

```
-opt level=4 -opt size -coloring -scheduling -peephole
```

(optional (1))

```
-model nearRelCode -model nearData
```

(optional (2))

```
-a6
```

(optional (3))

```
-sdata all
```

(optional (4))

```
-lavender model=ewl ,print=int ,scan=int ,io=raw
```

*How to Reach Us:*

Home Page:  
[www.freescale.com](http://www.freescale.com)

E-mail:  
[support@freescale.com](mailto:support@freescale.com)

USA/Europe or Locations Not Listed:  
Freescale Semiconductor  
Technical Information Center, CH370  
1300 N. Alma School Road  
Chandler, Arizona 85224  
+1-800-521-6274 or +1-480-768-2130  
[support@freescale.com](mailto:support@freescale.com)

Europe, Middle East, and Africa:  
Freescale Halbleiter Deutschland GmbH  
Technical Information Center  
Schatzbogen 7  
81829 Muenchen, Germany  
+44 1296 380 456 (English)  
+46 8 52200080 (English)  
+49 89 92103 559 (German)  
+33 1 69 35 48 48 (French)  
[support@freescale.com](mailto:support@freescale.com)

Japan:  
Freescale Semiconductor Japan Ltd.  
Headquarters  
ARCO Tower 15F  
1-8-1, Shimo-Meguro, Meguro-ku,  
Tokyo 153-0064, Japan  
0120 191014 or +81 3 5437 9125  
[support.japan@freescale.com](mailto:support.japan@freescale.com)

Asia/Pacific:  
Freescale Semiconductor Hong Kong Ltd.  
Technical Information Center  
2 Dai King Street  
Tai Po Industrial Estate  
Tai Po, N.T., Hong Kong  
+800 2666 8080  
[support.asia@freescale.com](mailto:support.asia@freescale.com)

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale, the Freescale logo, CodeWarrior and ColdFire are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. ColdFire+, Kinetis, Processor Expert, and Qorivva are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. ARM is the registered trademark of ARM Limited. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2011 Freescale Semiconductor, Inc. All rights reserved.