第1章 IAR EWARM入门

1.1 IAR EWARM集成开发环境介绍

1.1.1 IAR EWARM简介

IAR Embedded Workbench for ARM（下面简称 IAR EWARM）是一个针对 ARM 处理器的集成开发环境，它包含项目管理器、编辑器、C/C++编译器和 ARM 汇编器、连接器 XLINK 和支持 RTOS 的调试工具 C-SPY。在 IAR EWARM 环境下可以使用 C/C++和汇编语言方便地开发嵌入式应用程序。比较其他的 ARM 开发环境，IAR EWARM 具有入门容易、使用方便和代码紧凑等特点。

目前IAR EWARM支持ARM Cortex-M3 内核的最新版本是 5.11，该版本支持Luminary全系列的MCU。为了方便用户学习评估，IAR 提供一个限制 32k 代码的免费试用版本。用户可以到IAR 公司的网站www.iar.com/ewarm下载。

IAR EWARM 中包含一个全软件的模拟程序（simulator）。使用它不需要任何硬件支持就可以模拟各种 ARM 内核、外部设备甚至中断的软件运行环境，从中可以了解和评估 IAR EWARM 的功能和使用方法。

1.1.2 LM LINK调试器简介

LM LINK 是由广州致远电子有限公司开发的低成本高性能 USB JTAG 调试器，它专门用于对 Luminary 系列单片机程序的调试与下载。该调试器结合 IAR EWARM 集成开发环境可支持所有 LM3S 系列 MCU 的程序的下载与调试。

LM LINK 采用 USB 接口与电脑连接，打破传统的用并口和串口下载程序的方式，无论是台式电脑还是笔记本电脑都应用自如。透明外壳封装、设计小巧、晶莹剔透、外形比手机还小、价格低廉、性价比极高、调试下载更快、使用更加方便。

图 1.1 LM LINK（USB2.0 JTAG）调试器

1.2 IAR EWARM的安装

1.2.1 IAR EWARM安装步骤

1、从 IAR 的官方网站上 www.iar.com/ewarm 下载 IAR 5.11，文件名为：EWARM-KS-WEB-511.exe;

2、运行 EWARM-KS-WEB-511.exe;

3、点击Install the IAR Embedded Workbench，开始安装。如图 1.2所示；

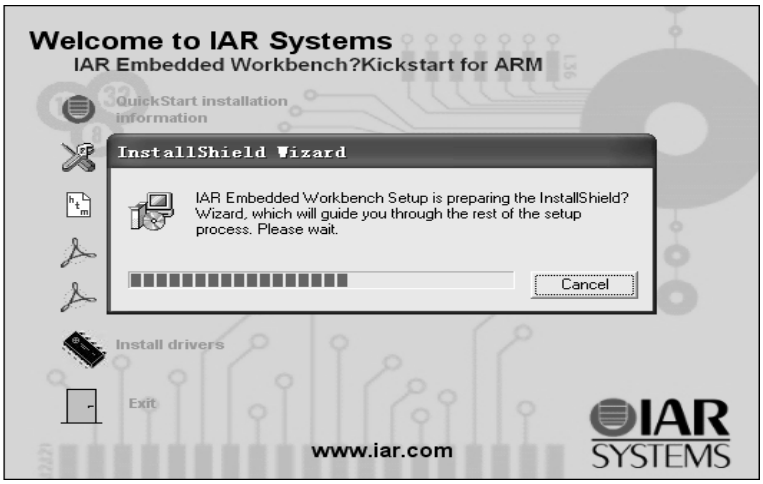

图 1.2　IAR EWARM 安装

4、输入许可证号（License）和密钥（License key）

用户从下载的软件包中的文本文件中提取许可证号（License）和密钥（License key），分别输入下面两个窗口如图 1.3和图 1.4所示。许可接受后建议按默认设置安装。

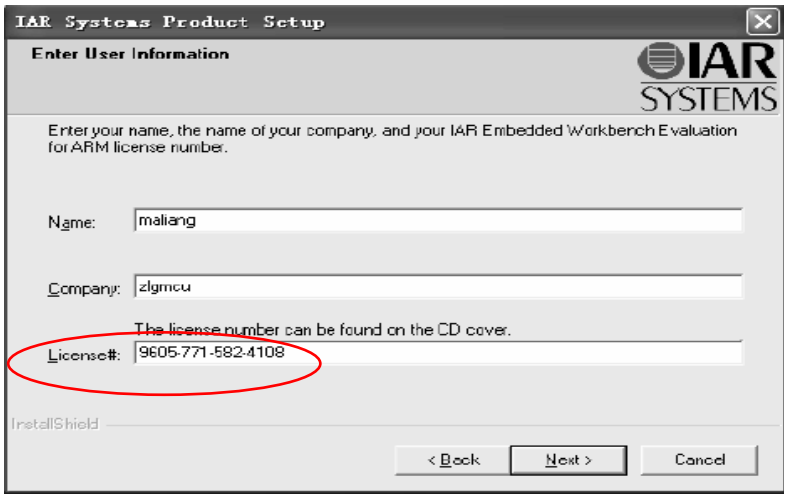

图 1.3　输入 License

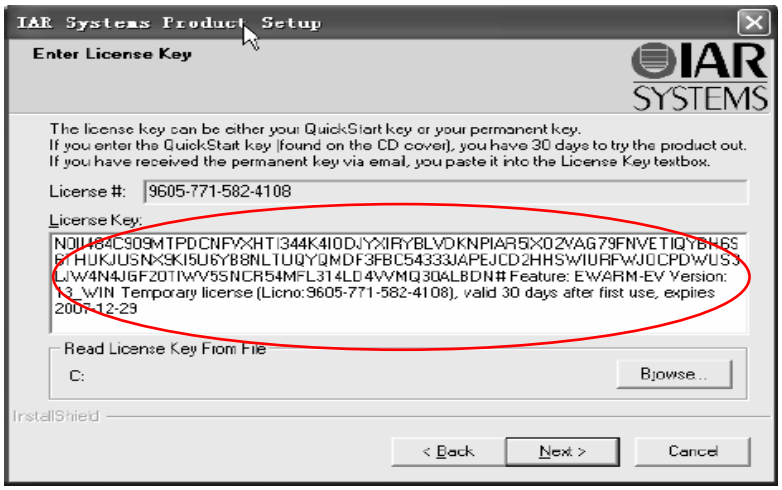

图 1.4 输入密钥

5、点击“下一步”直到软件安装完成。

注意：不可以通过网络安装。

1.2.2 安装LM LINK驱动

1、将光盘内的“LM LINK驱动”文件夹下的“FTDI”文件夹复制到C盘根目录下。如图1.5、图 1.6所示。

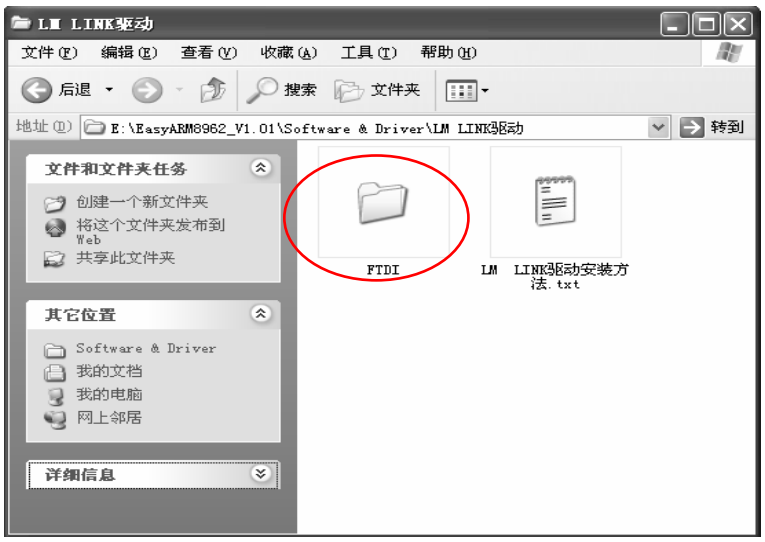

图 1.5 安装 LM LINK 驱动程序

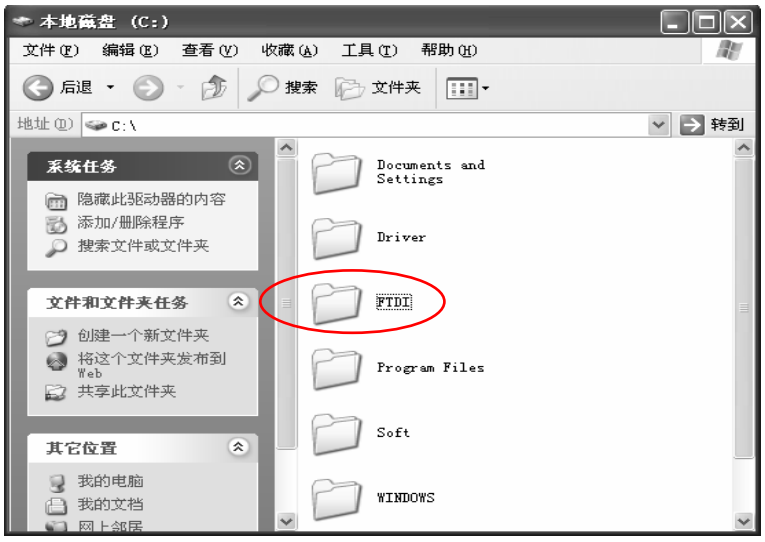

图 1.6 复制到 C 盘根目录下的 FTDI 文件

2、用USB电缆将LM LINK与PC相连，这时会弹出硬件安装向导窗口，如图 1.7所示，点击“下一步”继续安装就可以了。

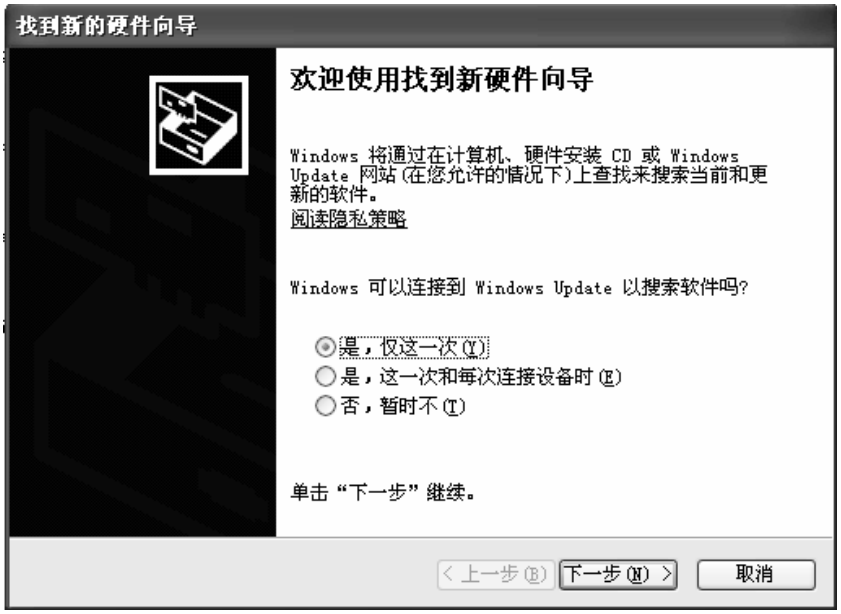

图 1.7 硬件安装向导

选择从指定位置安装，指向C盘的FTDI文件夹，如图 1.8所示。

图 1.8 指向 FTDI 目录

3、点击“下一步”进行安装，安装过程中将出现如图 1.9所示的窗口，选择“仍然继续”就可以了。

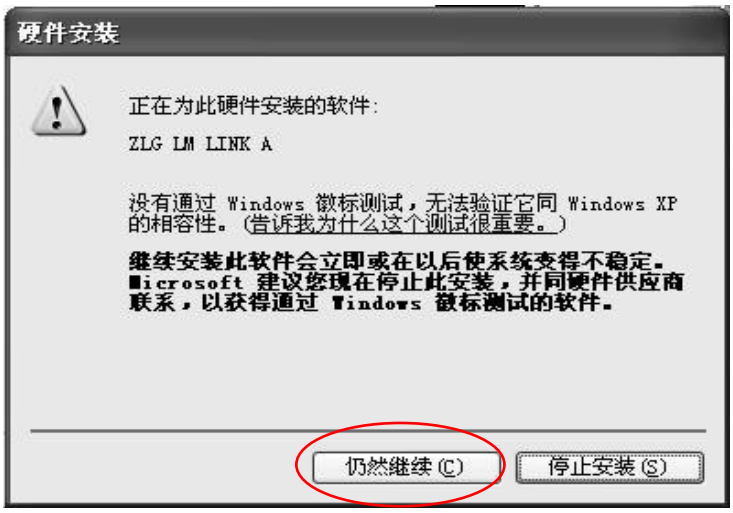

图 1.9 硬件安装过程中出现的窗口

这次安装完成后还会提示一次“硬件安装”，按照上面同样的过程再安装一次就可以了。安装完成后在“设备管理器”可以看到如图 1.10所示的信息，这表示驱动已经成功的安装好了。

图 1.10 安装后显示的驱动信息

安装完成后即可启动 EWARM 环境。

将 LM LINK 与目标开发板的 JTAG 接口插座相连，接通目标开发板电源，然后按下面的步骤执行后面的操作。

1.3 相关准备工作

在安装好 EWARM 集成开发环境后，就可在该环境下新建工程了。但在新建工程之前，为了使以后工程中的设置更加的简单化，在这里就需要一些准备工作，将某些文件拷贝到默认路径下，具体的操作方式将在随后介绍。

至于为什么要这样做，在工程的设置时就会体会出其优越性。

注意：本文是以 IAR 5.11 版（32K 的试用版）为例，作讲解。如果用正式版可以参照本文进行设置。

1.3.1 下载最新库文件

从流明诺瑞官方网站http://www.luminarymicro.com下载最新的驱动库文件。假设保存于“C:\”，并对下载的压缩文件解压，解压后如图 1.11所示。

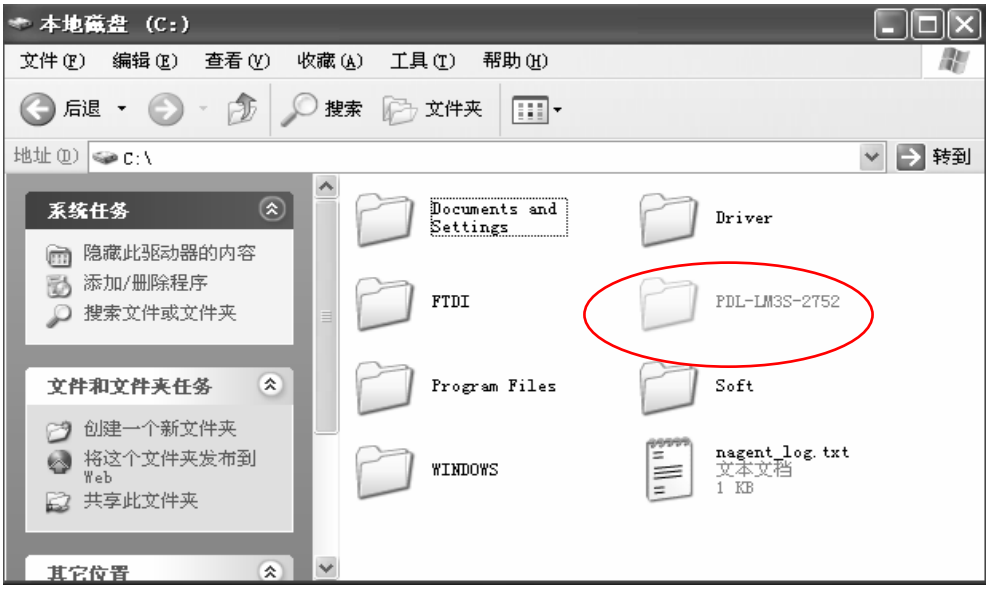

图 1.11 驱动库文件存放目录

1.3.2 生成“driverlib.a”文件

1、启动 IAR EWARM，在“File”菜单中选择“New>Workspace”,新建一个工作区；

2、在“Project”菜单中选择“Add Existing Project…”，将 C:\PDL-LM3S-2752\DriverLib\src 目录下的工程“driverlib.ewp”添加进来，如图 1.12所示；

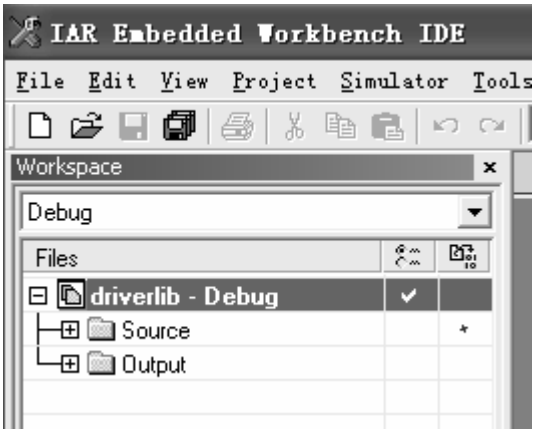

图 1.12 添加“driverlib.ewp”文件

3、单击保存按钮，弹出保存对话框，将新建的工作区取名为“LM3S_DriverLib.eww”，并保存在“C:PDL_LM3S_2752\DriverLib\src”文件夹下；

4、鼠标右击窗口左边的“driverlib - Debug”，选择 Options，打开选项，对某些选项进行必要的配置；

● 选择“Category:”里的“C/C++”，在“Optimizations”选项卡里，建议选中优化级别Level为“None”，这可以更好地支持将来的在线仿真调试，如图 1.13所示。

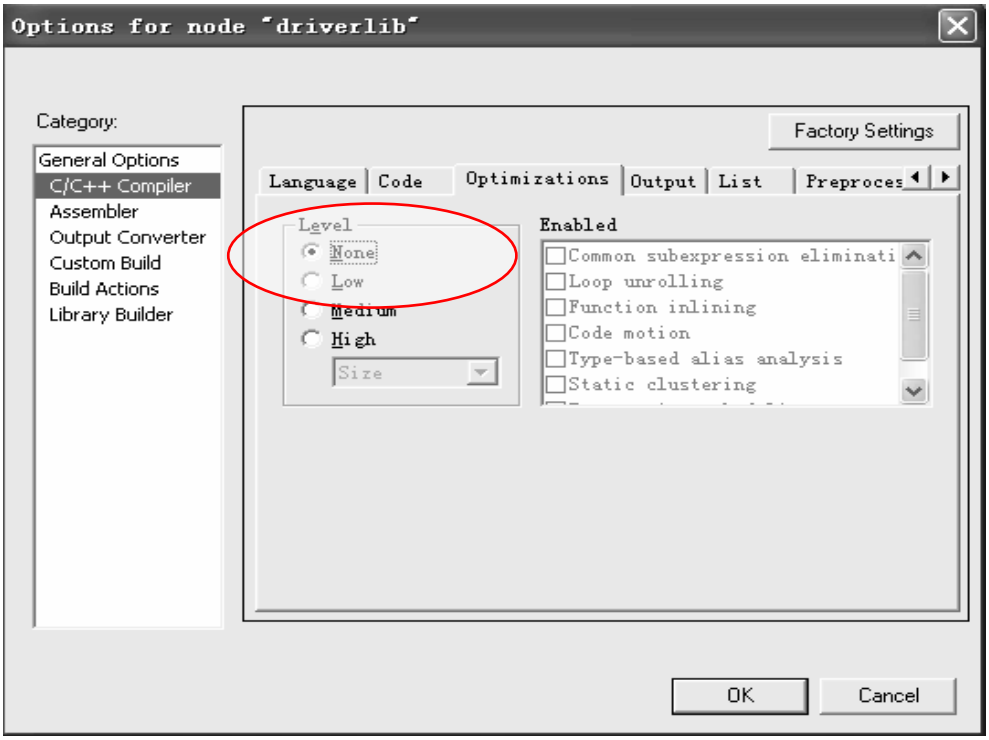

图 1.13 C/C++ Compiler 选项设置

● 选择“Category:”里的“Library Builder(Linker的Config选项)”，勾中“Override default”，如图 1.14所示。

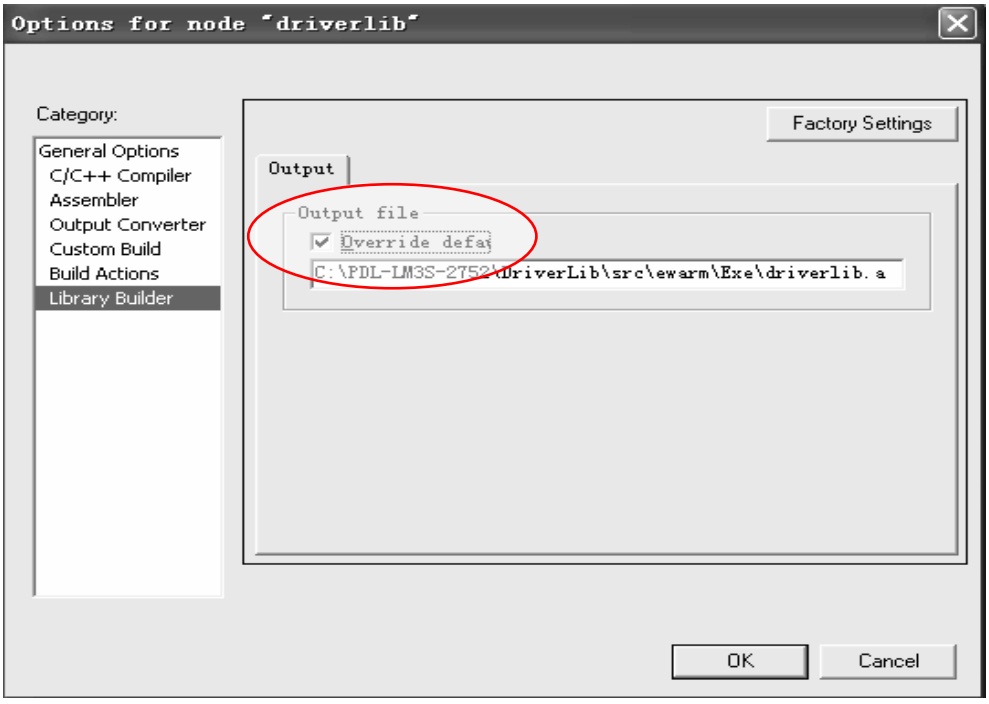

图 1.14 Library Builder 选项设置

4、在“Project”菜单中选择“Make”进行编译，将在 C:\PDL_LM3S_2752\Driver\src\Exe 下自动生成一个“driverlib.a”文件。

注：生成“driverlib.a”文件，对以后所建工程将带来很多方便，不需要重复上面的这些步骤。C:\PDL-LM3S-2752\DriverLib\src\ewarm\Exe”

1.3.3 在IAR安装路径下创建Stellaris外设驱动库

这一步是将库文件手动复制到 IAR EWARM 的默认路径下面，减轻了每次在选择库文件时的添加库文件步骤。

1、进入“C:\Program Files\IAR Systems\Embedded Workbench 5.0 Kickstart\ARM\lib”，创建一个新文件夹，取名“Luminary”，将“C:\PDL-LM3S-2752\DriverLib\src\ewarm\Exe”目录下的“driverlib.a”文件复制过来，如图 1.15所示。

图 1.15 复制到相应目录下的"driverlib.a"文件

2、将“C:\PDL-LM3S-2752\DriverLib”下所有的“*.h”文件复制到“C:\Program Files\IAR Systems\Embedded Workbench 5.0 Kickstart\ARM\inc\Luminary”下；

3、将“C:\PDL-LM3S-2752\DriverLib\src”下所有的“*.c”、“*.h”文件以及“CPU.S”复制到“C:\Program Files\IAR Systems\Embedded Workbench 5.0 Kickstart\ARM\inc\Luminary”下;复制后如图 1.16所示。

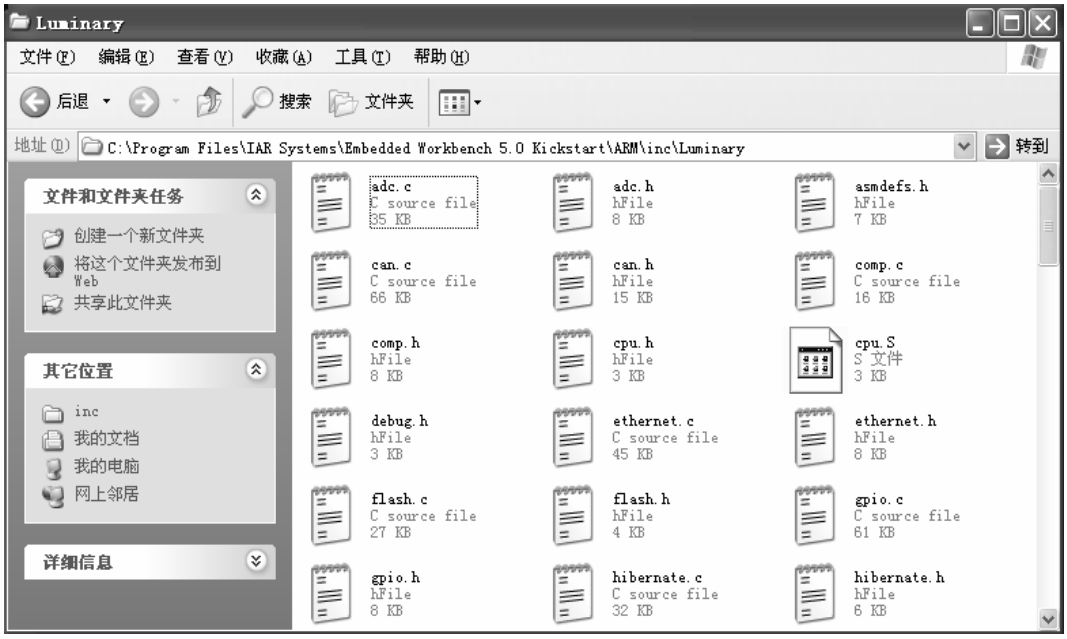

图 1.16 驱动库头文件存放目录

4、在“C:\Program Files\IAR Systems\Embedded Workbench 5.0 Kickstart\ARM\config”下，创建一个新文件夹，取名“Luminary”，在“Luminary”文件夹下创建一个新文件，取名“LM3S.icf”，这是个配置文件，内容如下：

```
//****************************************************************************
//
// LM3S.icf - Linker configuration file for timers.
//
// Copyright (c) 2005-2008 Luminary Micro, Inc.  All rights reserved.
//
// Software License Agreement
//
// Luminary Micro, Inc. (LMI) is supplying this software for use solely and
// exclusively on LMI's microcontroller products.
//
// The software is owned by LMI and/or its suppliers, and is protected under
// applicable copyright laws.  All rights are reserved.  You may not combine
// this software with "viral" open-source software in order to form a larger
// program.  Any use in violation of the foregoing restrictions may subject
// the user to criminal sanctions under applicable laws, as well as to civil
// liability for the breach of the terms and conditions of this license.
//
// THIS SOFTWARE IS PROVIDED "AS IS".  NO WARRANTIES, WHETHER EXPRESS, IMPLIED
// OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF
// MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE.
// LMI SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL, OR
// CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.
//
// This is part of revision 2752 of the Stellaris Peripheral Driver Library.
//
//****************************************************************************


//
// Define a memory region that covers the entire 4 GB addressible space of the
// processor.
//
define memory mem with size = 4G;
//
// Define a region for the on-chip flash.
//
define region FLASH = mem:[from 0x00000000 to 0x0003ffff];
//
// Define a region for the on-chip SRAM.
//
```

```
define region SRAM = mem:[from 0x20000000 to 0x2000ffff];
//
// Define a block for the heap.    The size should be set to something other
// than zero if things in the C library that require the heap are used.
//
define block HEAP with alignment = 8, size = 0x00000000 { };
//
// Indicate that the read/write values should be initialized by copying from
// flash.
//
initialize by copy { readwrite };
//
// Indicate that the noinit values should be left alone.    This includes the
// stack, which if initialized will destroy the return address from the
// initialization code, causing the processor to branch to zero and fault.
//
do not initialize { section .noinit };
//
// Place the interrupt vectors at the start of flash.
//
place at start of FLASH { readonly section .intvec };
//
// Place the remainder of the read-only items into flash.
//
place in FLASH { readonly };
//
// Place all read/write items into SRAM.
//
place in SRAM { readwrite, block HEAP };
```

注：//后的内容可以省略不用。

5、把 C:\PDL-LM3S-2752\DriverLib\boards\ek-lm3s8962\hello 目录下的“startup_ewarm.c”文件复制到 C:\PDL-LM3S-2752\DriverLib\ewarm 目录下。

注：在“startup_ewarm.c”文件里 STACK_SIZE 定义的值是 64，建议将其改得大一些，如 256、2048 等等

1.4 在IAR EWARM中新建项目

要为某个目标系统开发一个新应用程序，必须先新建一个新项目。新建项目具体步骤下面将一一呈现。

1.4.1 建立一个项目文件目录

首先应该为新项目创建一个目录，用来存放与项目有关的各种文件。项目开发过程中生成的一系列文件，如：工作区文件，开发环境的配置，编译、连接和调试选项配置，各种列

表文件和输出文件等都将被存放在这个目录下。用户也可以选择把各种源文件也放在这个目录下。在下面的例子中我们生成一个 E:\DEMO 目录。

1.4.2 新建工作区

IAR EWARM 虽然是按项目进行管理，但是要求把所有的项目都放在工作区内（Workspace）。用户如果是第一次使用 IAR EWARM 开发一个新项目，必须先创建一个新工作区，然后才能在工作区中创建新项目。一个工作区中允许存放一个或多个项目。如果用户过去已经建立了一个工作区并且希望把目前要建的新项目放在老工作区内，则可以直接打开老工作区并执行第三步生成新项目。创建新工作区方法如下：

启动IAR EWARM开发环境，如图 1.17所示。

图 1.17 启动 IAR EWARM 开发环境

选择主菜单的File >New>Workspace命令，然后开启一个空白工作区窗口，如图 1.18所示。

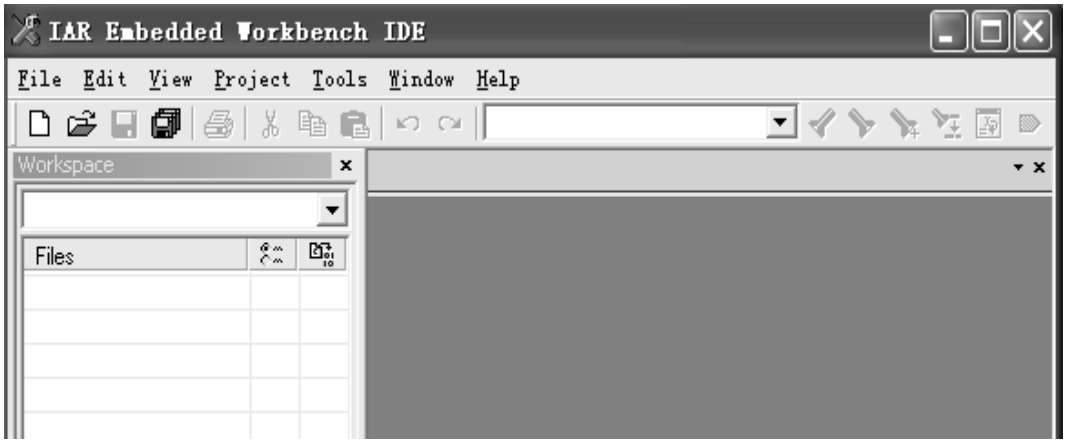

图 1.18 空白工作区窗口

1.4.3 生成新项目

创建新项目方法如下：

1、选择主菜单Project>Create New Project，弹出生成新项目窗口中。IAR EWARM提供

几种应用程序和库程序的项目模板。如果选择Empty project，表示采用默认的项目选项设置，为一个空工程。在本例中我们选择Empty project，如图 1.19所示。

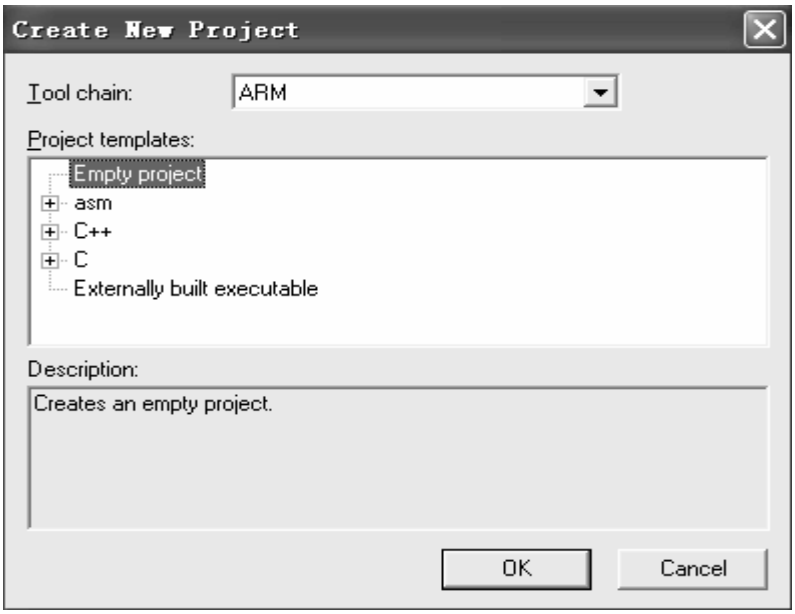

图 1.19 生成新项目窗口

2、在Tool chain栏中选择ARM，点击OK按钮，弹出“另存为”窗口。如图 1.20所示。

图 1.20 “另存为”窗口

3、在“另存为”窗口中浏览和选择新建的E:\DEMO目录，输入新项目的文件名为demo，然后保存。这时在屏幕左边的Workspace窗口中将显示新建的项目名和输出代码模式，如图1.21所示。

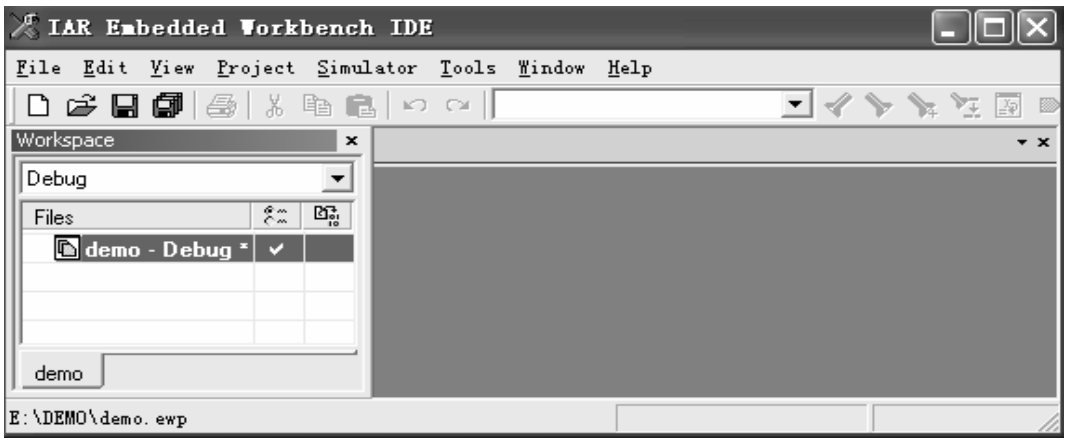

图 1.21 新建项目名

项目名后面的 Debug 表示输出含调试信息的代码文件。IAR EWARM 能为项目提供两种输出代码模式：Debug 和 Release。Debug 模式生成含调试信息的程序代码，用户利用它可以在 IAR EWARM 环境下调试应用程序。而 Release 模式生成不含调试信息的发行版本的程序代码，其代码比较紧凑。用户可以从 Workspace 窗口顶部的下拉菜单中选择两种项目配置之一，本例我们选择 Debug。

现在 DEMO 目录下已生成一个 demo.ewp 文件，该文件中将包含与 demo 项目设置有关的信息，如编译、连接（build）的选项等。

注意：demo-Debug 后的*号表示当前的工作区和项目经修改后还没有被保存。

4、保存工作区

新生成的工作区需要保存才有效，所以在添加项目后IAR EWARM要求执行保存工作区操作。保存工作区选择主菜单File>Save Workspace，浏览并选择E:\DEMO目录。然后将工作区取名为demo输进File name输入框，按保存按钮退出，如图 1.22所示。

这时在 E:\DEMO 目录下又生成一个 demo.eww 文件。同时在 E:\DEMO 目录下还生成一个 settings 子目录，这个目录下存放保存窗口设置和断点设置等与当前操作有关信息的其他文件。

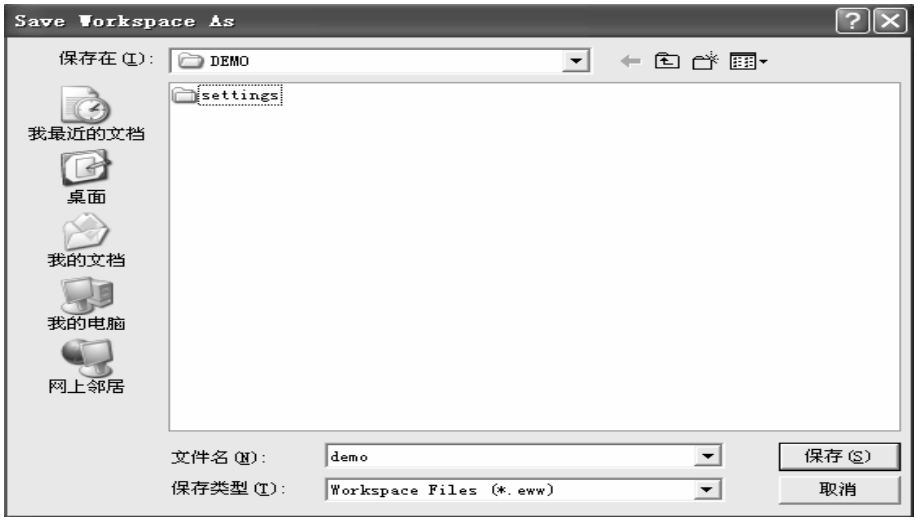

图 1.22 保存工作区

注意：保存操作完成后项目名后的*号已经消失。

1.4.4 添加/新建文件

保存工作区后，下一步就是在项目中新建文件或添加已有文件。项目中的文件允许分组，用户可以根据项目的需要和自己的习惯来组织源文件。为举例说明，这里新建以下几个文件组：一个 startup 文件组，一个 src 文件组，一个 lib 文件组。

注意：往项目中添加文件时只需添加汇编语言和 C 语言的源程序，不需要添加头文件(即.h 头文件)。但是用户必须在配置项目的编译器、连接器选项时指明包含头文件的路径和目录。关于项目配置选项的设定我们会在后面详细介绍。

1、建立文件组

右击“demo-Debug”然后选择ADD>ADD Group…，如图 1.23所示。

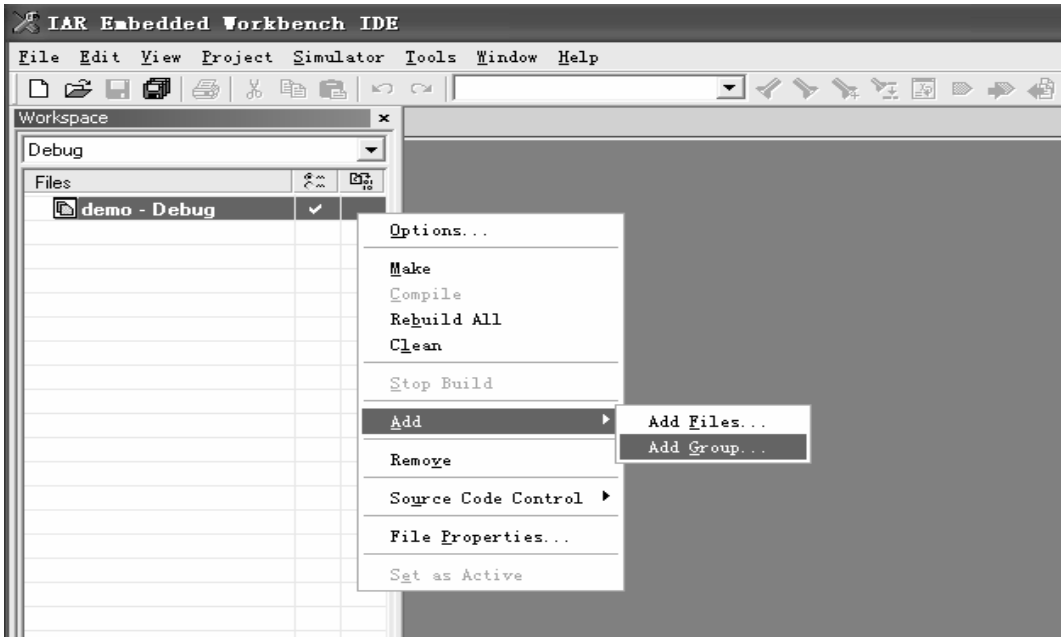

图 1.23 建立文件组

新建 3 个文件组：startup文件组，src文件组，lib文件组，如图 1.24所示

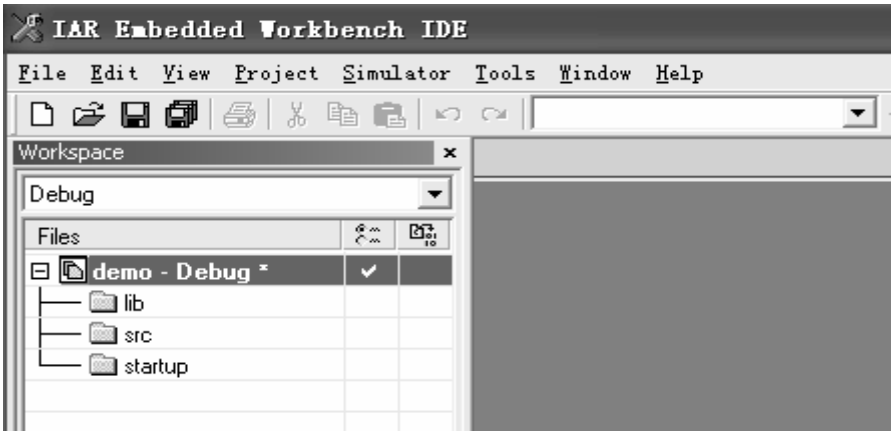

图 1.24 新建 3 个文件组

2、添加对应文件

向文件组添加对应文件，如图 1.25所示

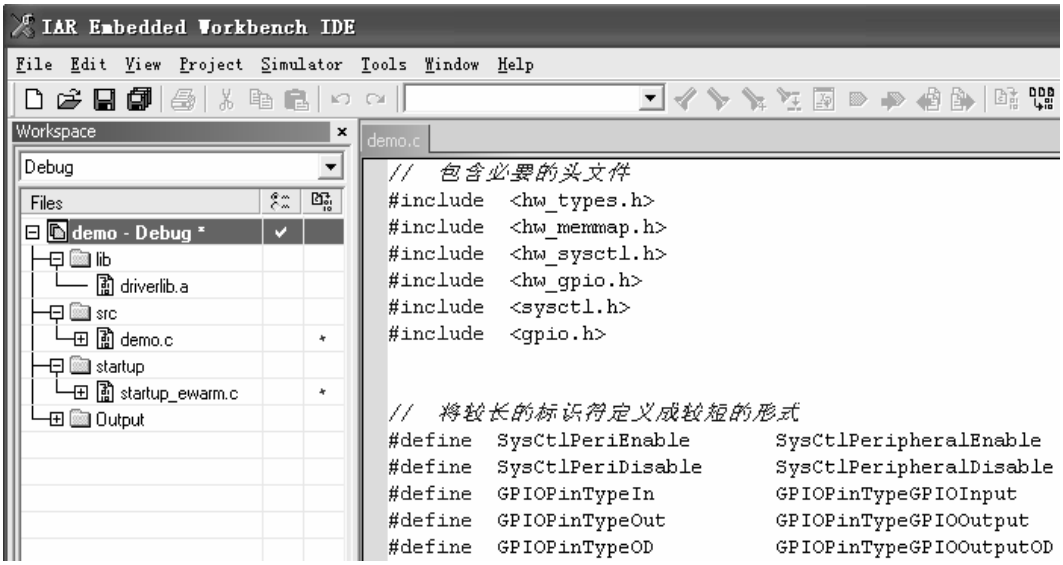

图 1.25 向文件组添加对应文件

● 在 lib 组添加 driverlib.a 文件

添加方法：右击lib，选择ADD>ADD Files…，在弹出的对话框中选择：“C:\Program Files\IAR Systems\Embedded Workbench 5.0 Kickstart\arm\lib\Luminary”，选择需要添加的库文件driverlib.a，如图 1.26所示。

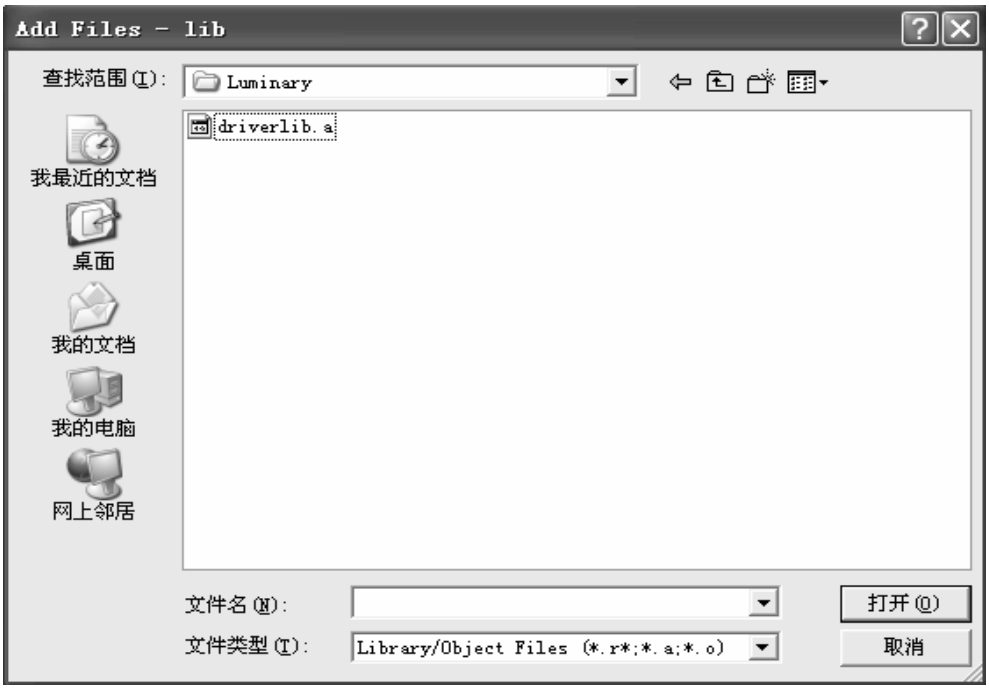

图 1.26 选择需要添加的库文件

● 在 startup 组添加 startup_ewarm.c 文件

将“C:\PDL-LM3S-2752\DriverLib\ewarm”下的startup_ewarm.c文件复制到工程目录E:\DEMO下面。然后右击startup，选择ADD>ADD Files…，在弹出的对话框中选择目录E:\DEMO，添加startup_ewarm.c文件，如图 1.27所示。

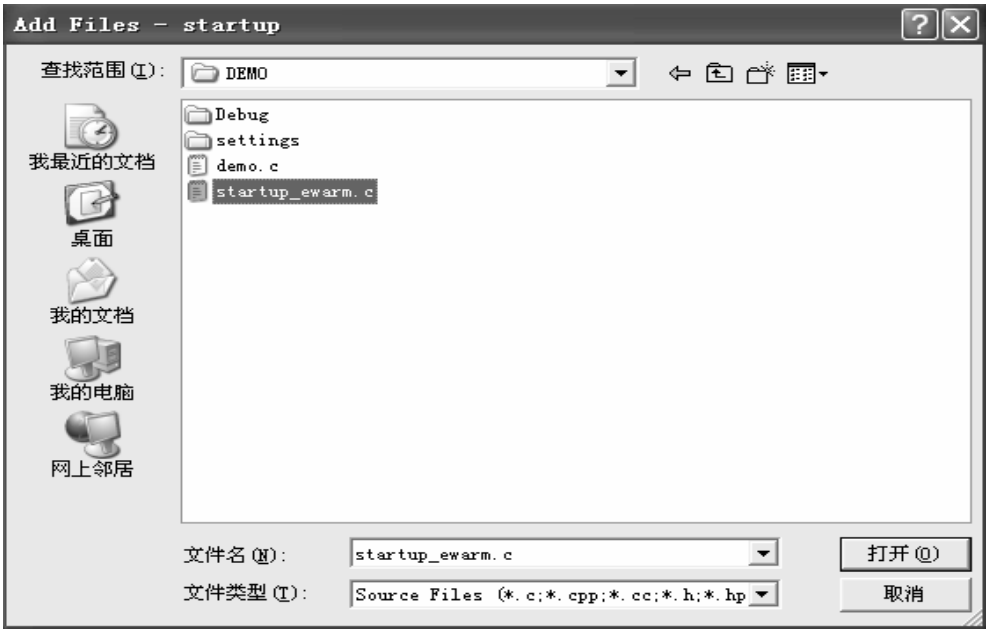

图 1.27 添加 startup_ewarm.c 文件

● 在 src 组中新建需要的 demo.c 文件或添加已有的 demo.c 文件，即主程序在这里编辑。

这里新建一个demo.c，首先单击src 组，选择File > New > File（也可以选择New document），将在窗口中出现一个空白页，再选择File > Save，弹出另存为对话框，保存在E:\DEMO，保存为demo.c。

然后右击src 组，选择ADD > ADD Files…，在弹出的对话框中选择目录E:\DEMO，添加demo.c 文件。如图 1.28所示。

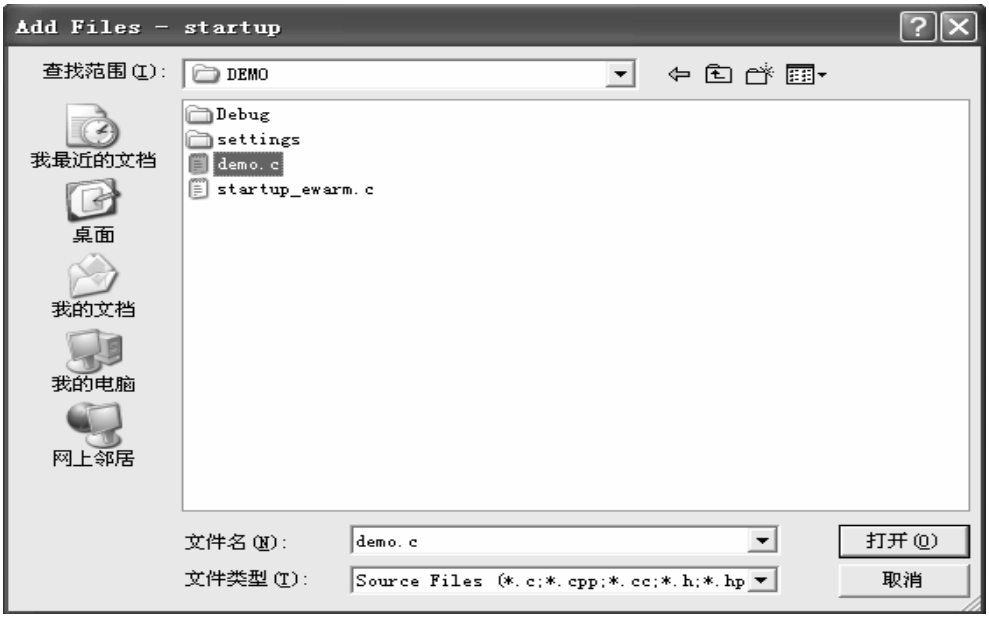

图 1.28 添加 demo.c 文件

此时，便可以在该main.c 文件中编辑需要的程序，这里编写了一个LED 灯闪烁的示例程序。

1.4.5 项目选项设置

生成新项目和添加文件后的下一步是为项目设置选项。设置项目选项是非常重要的一步，如果设置不当，编译、连接就会出错，就无法生成正确的代码文件。大家记得，在创建新项目时我们选择了Empty project 模板，表示采用默认的项目选项设置。但是这些默认的设置还要根据具体项目的需要进行修改。IAR EWARM 提供的项目选项内容繁多，初学者可能会感觉到摸不着头脑、无从下手。实际上关键的选项并不多，只要把它们设置正确了，其它的采用默认设置就不会出错。下面我们把这些关键选项设置逐条介绍。

注意：文中没有提及的选项均采用默认设置。

1.4.6 通用选项设置

IAR EWARM 允许为工作区中的任何一级目录和文件单独设置选项，但是用户必须首先为整个项目设置通用的选项General Option。

设置方法是：选中工作区中的项目名demo – Debug，按鼠标右键在弹出菜单中选择Options…或选择主菜单 Project > Options…。在弹出的Options 窗口左边的目录（Category）中选择第一项General Options。然后分别在：

●Target 设置

●在Processor Variant 框中选择Device。并点击右边的器件选择按钮，选择芯片型号Luminary LM3Sx9xx（针对不同型号芯片，可对其进行灵活选择）。同时Endian mode 选择Little。如图 1.29所示；

●其它选项采用默认值。

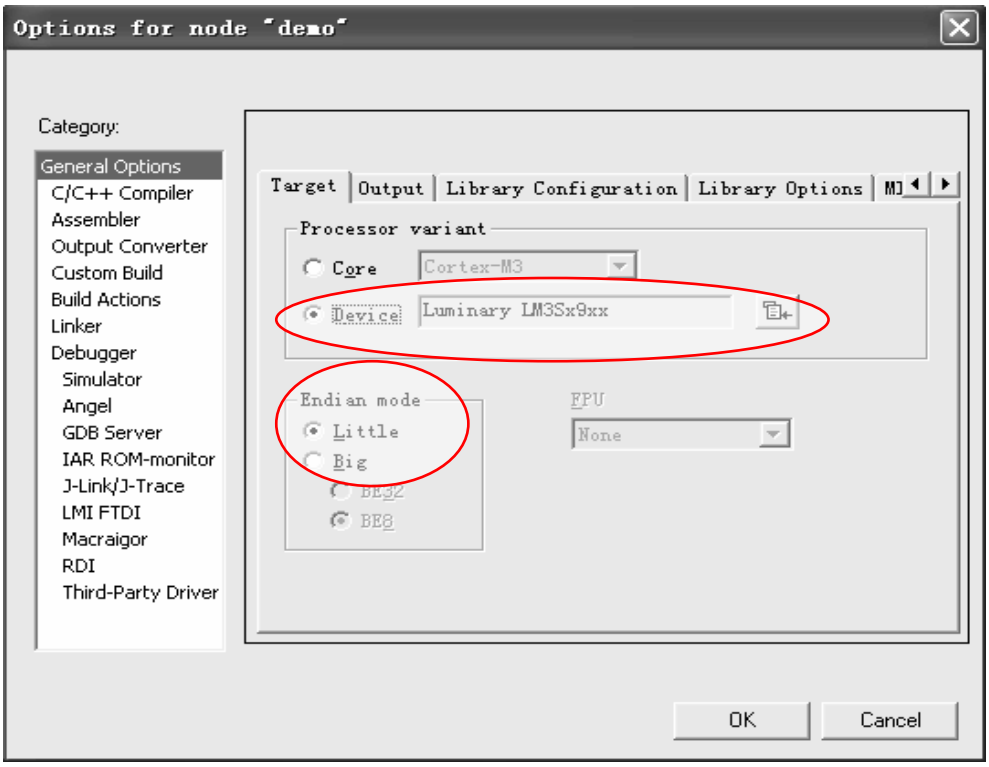

图 1.29 General Option 选项设置

1.4.7 C/C++编译器选项设置

在Options 窗口的目录Category 中选择第二项C/C++ Compiler。C/C++编译器的选项设置如下：

●Preprocessor 设置

Preprocessor 页面中，列有标准的include 文件的目录。如果用户的include 文件不在标准目录下时，必须在Additional include directories 输入包含该项目include 文件的目录。一个目录用一行描述，有多个目录时允许用多行。

在Preprocessor 框中的Additional include directories(one per line) 项目中输入“$TOOLKIT_DIR$\INC\Luminary”，前面的拷贝库文件目的就在此。如图 1.30 所示。

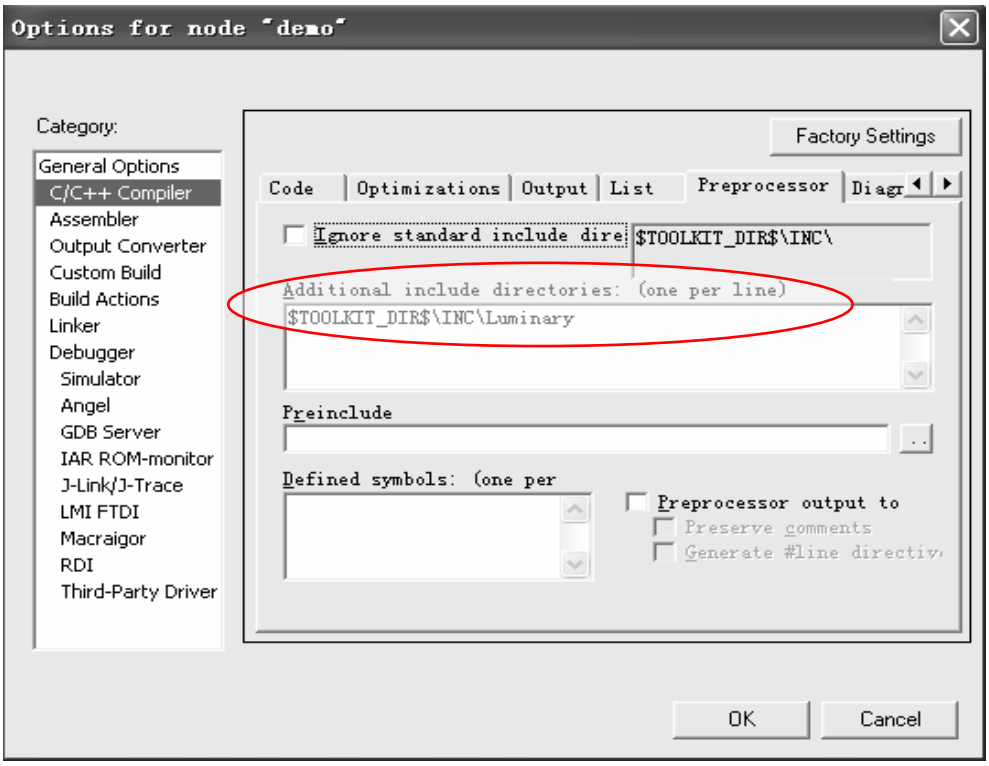

图 1.30 C/C++ Compiler 编译选项设置

其它的选项采用默认值。

1.4.8 Assembler选项设置

在Options 窗口的目录Category 中选择第三项Assembler。汇编器的选项设置采用默认设置。

1.4.9 Output Converter选项设置

在“Output”选项卡中，勾中“Generate additional output”，在“Output format”里选中“binary”，再勾中“Override default”，这样，将来编译时会在“E:\Demo\Debug\Exe”自动生成二进制文件“Demo.bin”，方便用LMFlashProgrammer软件下载，如图 1.31所示。

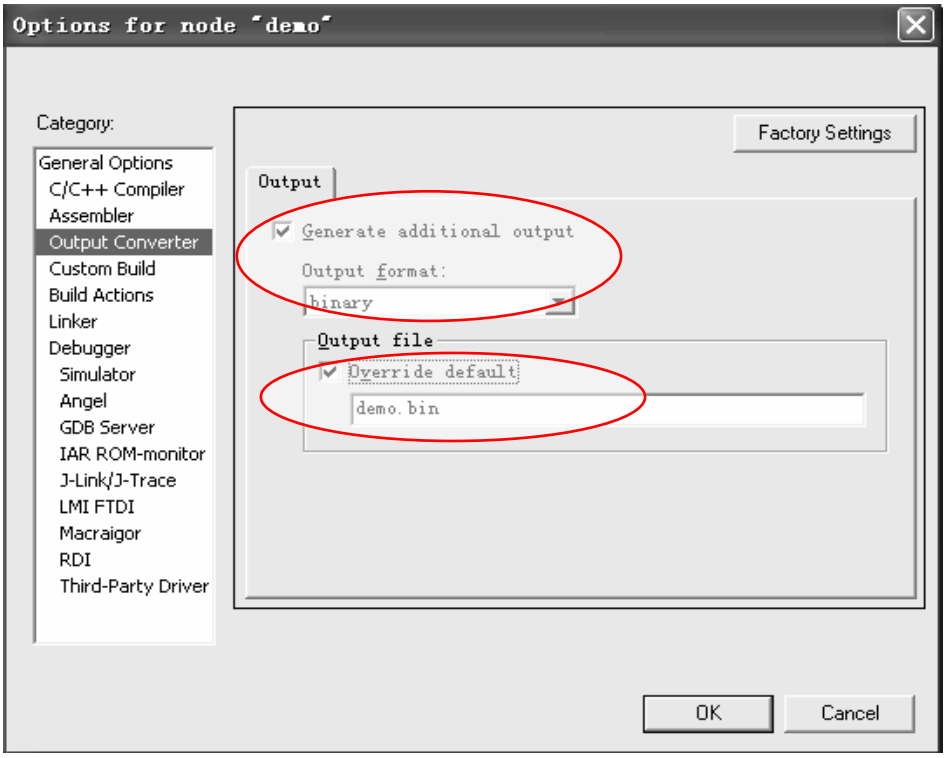

图 1.31 Output Converter 选项设置

1.4.10 Linker 选项设置

在Options 窗口的目录Category 中选择第七项Linker。连接器的选项中的设置主要有以下几个：

●Config 设置

主要是定义连接器命令文件（Linker Command File）。这是连接器选项中最重要同时也是最复杂的设置。连接器命令文件中包含连接器的各项命令行参数，主要用于控制程序各种代码段和数据段在存储器中如何分布。用户一定要吃透和掌握如何生成正确的连接器命令文件。为了帮助初学者理解，我们增加了下面一段介绍。

用户会采用不同半导体厂家的产品，每种芯片内部SRAM 和FLASH 的大小和地址分布都不同，另外用户目标系统配置的外部存储器也不同，用户应用软件要求的存储器分配也不相同。以上所有的不同最后落实到在运行时不同的代码段和数据段的存储器地址分配方案。而这种运行时存储器分配必需在连接器命令文件中说明，并由连接器IAR XLINK 生成。经XLINK 连接生成的代码文件下载到目标板时的地址，由FlashLoader 执行，后面将介绍。IAR EWAR 提供默认的连接器命令文件，它在IAR EWAR 安装目录的ARM\config 目录下，名字叫generic.icf。但是默认的连接器命令文件generic.icf 不能完全适用特定的目标系统，必须加以修改。LM3S.icf为LM3S 系列MCU 在EWARM 集成开发环境下的连接器命令文件。之前我们在ARM\config\Luminary目录下新建的LM3S.icf，就是为了这一步很方便的选择LM3S.icf。

在Link Comamnd file 中，选中Override default，点击右边选择按钮，打开选项选择LM3S.icf。如图 1.32 所示。

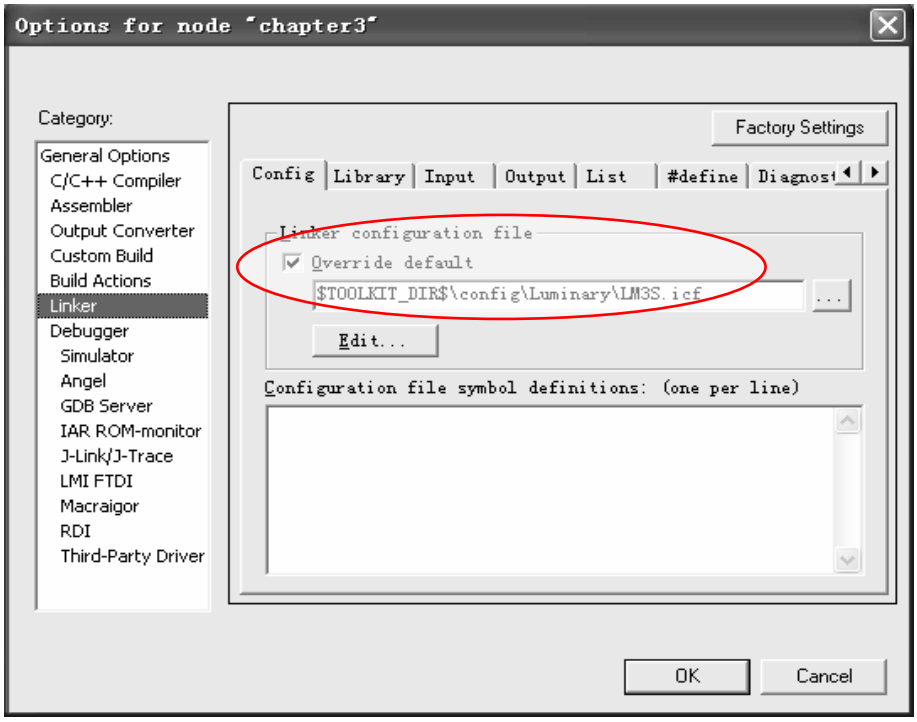

图 1.32 Linker 选项的 Config 设置

注：LM3S.icf文件在“C:\Program Files\IAR Systems\Embedded Workbench 5.0 Kickstart\ARM\config\Luminary”目录下。

- List 设置

选择Generate linker map file 或Generate log file，允许生成存储器分配MAP 文件或LOG文件。如图 1.33所示。

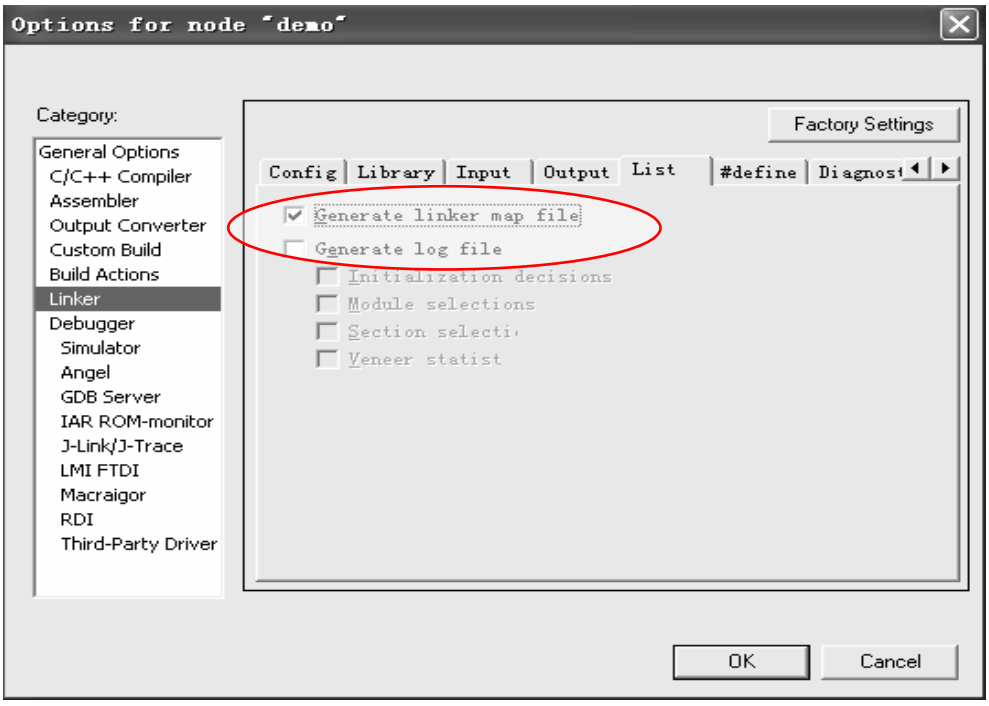

图 1.33 Linker 选项的 List 设置

1.4.11 Debugger 选项设置

在Options 窗口的目录Category 中选择第八项Debugger。调试器的选项设置如下：

● Setup 页面设置

本项选择所用的调试工具，我们选择的是Luminary 的LM FTDI，如图 1.34所示。

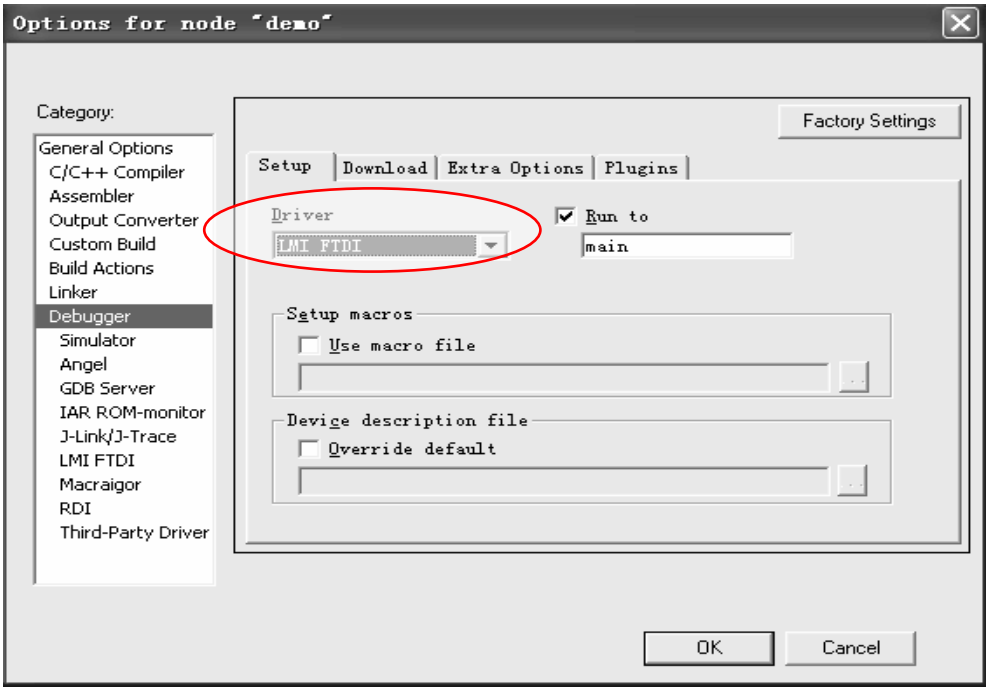

图 1.34 调试工具选择

● Download 页面设置

选择 Verify download 和 use flash load。如图 1.35 所示。

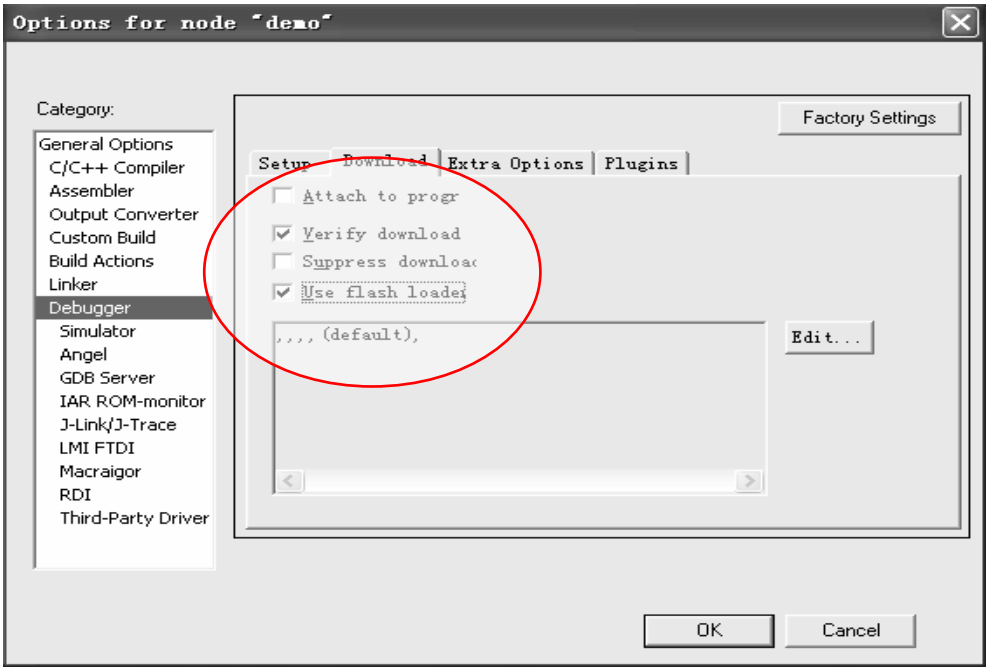

图 1.35 下载程序选项设置

1.4.12 LMI FTDI 选项设置

在LMI FTDI选项中Setup → JTAG speed，填写连接速度，一般不要超过 500kHz，推荐用 100kHz，如图 1.36所示。

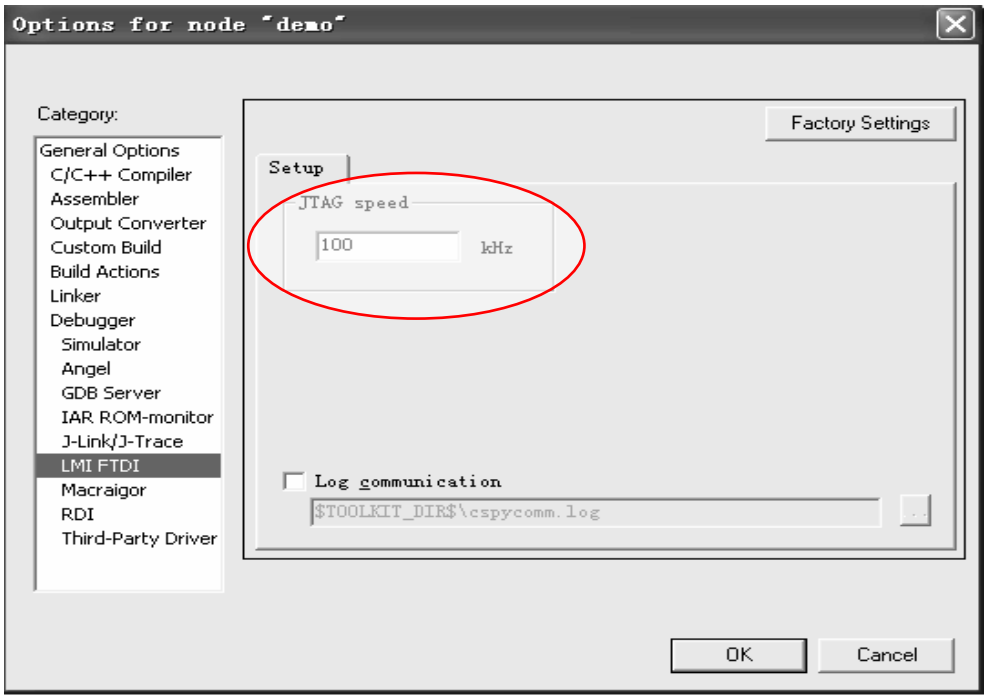

图 1.36 LMI FTDI 选项设置

1.5 编译和运行应用程序

按上述步骤完成所有的工程设置以后就可以开始编译程序了。

1.5.1 编译连接处理

选择主菜单Project > Make，或选中工作区中的项目名demo– Debug，按鼠标右键在弹出菜单中选择Make。如果你想重新编译所有的文件，选择主菜单Project > Rebuild All，或选中工作区中的项目名demo – Debug，按鼠标右键在弹出菜单中选择Rebuild All。

IAR EWARM 将执行编译连接处理，生成可调试代码文件。Build 消息窗口中将显示连接处理的消息。连接的结果将生成一个带调试信息的代码文件demo.o 和一个存储器分配（MAP）文件demo.map。

从编译连接后的工作区窗口中树结构中，我们可以看到每个源文件访问关联了哪些头文件，同时生成了哪些输出文件。因为我们在建立新项目时选择Debug 配置，所以在DEMO目录下自动生成一个Debug 子目录。Debug 子目录下又包含另3 个子目录，名字分别为List、Obj、Exe。

在Obj 目录下后缀为.o 的文件，用作IAR XLINK 连接器的输入文件。

在Exe 目录下后缀为.out 的文件，用作IAR C-SPY 调试器的输入文件，注意在执行连接处理之前这个目录是空的。

1.5.2 查看MAP 文件

双击Workspace 中的demo.map 文件名，编辑器窗口中将显示该MAP 文件。从MAP 文件中我们可以了解以下内容：

- 文件头中显示连接器版本，输出文件名以及连接命令使用的选项。
- MODULE SUMMARY 部分显示所有被连接的文件。每个文件中，作为应用程序一部分加载的有关模块的信息，包括各段和每个段中声明的全局符号都列出来。
- ENTRY LIST部分显示程序入口地址。

如果编译连接没有任何错误，则生成demo.out 应用程序代码，并可以用于在IAR C-SPY 中调试。

1.5.3 加载应用程序

选择主菜单Project > Debug 或工具条上的Debugger 按钮或者按键CTL+D，C-SPY 将开始装载demo.out。屏幕上将显示PC 机通过 LM LINK 加载的过程。

屏幕上除了原先已经打开的窗口外，将显示一组C-SPY 专用窗口。如Debug Log和Disassembly 窗口。如图 1.37 所示。

注意：如果在下载程序时，有提示信息出现，直接选择“否”就可以了。

图 1.37 IAR EWARM 的 Debug 窗口

注：到此，程序已经下载到Flash，也可以进行程序的调试了。

1.5.4 应用程序的相关调试

用IAR EWARM调试应用程序时，允许用户采用源代码调试模式和反汇编调试模式。在源代码调试模式下，编辑器窗口中显示C语言源代码程序，用户可以单步运行程序，同时监控变量和数据的值，这是应用程序开发最快捷的方式。在反汇编调试模式下，可以打开一个反汇编窗口，显示应用程序的助记符和汇编指令，每次准确地执行一条汇编指令，从而可以关注程序的关键部分，并对硬件进行精确控制。也可以采用混合调试模式，即在调试过程中，同时打开源代码窗口和反汇编窗口，以便于观察C语言语句与汇编语言指令代码之间的关系。不管采用哪种模式，调试过程中都可以随时显示或修改寄存器和存储器的内容，下面我们将一一呈显调试给你带来的“快感”。

● 程序的执行方式

IAR调试器在Debug菜单中提供了8种程序运行命令：Step Over、Step Into、Step Out、Next Statement、Run to Cursor、Go、Stop Debugging和Break。其对应的快捷键、工具按钮、功能说明见表1.1所示。

表 1.1 程序运行命令表

命令选项	快捷键	工具按钮	功能说明
Step Over	F10		在同一函数中将运行至下一步点，而不会跟踪进入调用函数内部
Step Into	F11		控制程序从当前位置运行至正常控制流中的下一个步点，无论它是否在同一函数内
Step Out	Shift+F11		使用Step Into单步运行踪跟进入一个函数体内之后，如不想一直跟踪到该函数末尾，运用此命令可执行完整个函数调用并返回到调用语句的下一条语句
Next Statement			直接运行到下一条语句
Run to Cursor			使程序运行至用户光标所在的源代码处，也可在反汇编窗口以及堆栈调用窗口中使用
Go	F5		从当前位置开始，一直运行到一个断点或是程序末尾
Break			中止程序运行
Stop Debugging			退出调试器，返回IAR EWARM环境

● 设置断点

用户可以设置不同类型的断点，以便使程序在某些关键位置暂停运行。用户可以设置Code(代码)断点，以检查程序逻辑结构是否正确；也可以设置Data（数据）断点，以观察数据是如何变化的；在使用纯软件模拟仿真时，还可以设置Immediate(立即)断点和特殊条件断点。这些断点的设置方法很多，这里只介绍最简单的方法。

使用Toggle Breakpoint命令。先在源代码窗口中选取要设置断点的位置；然后选择Edit>Toggle Breakpoint菜单项，或单击快捷工具按钮  ，即可在指定位置设置一个断点。同时源代码窗口相应语句位置将以高红色显示，左边空白处还将显示一个红色的标记，如图1.38所示。

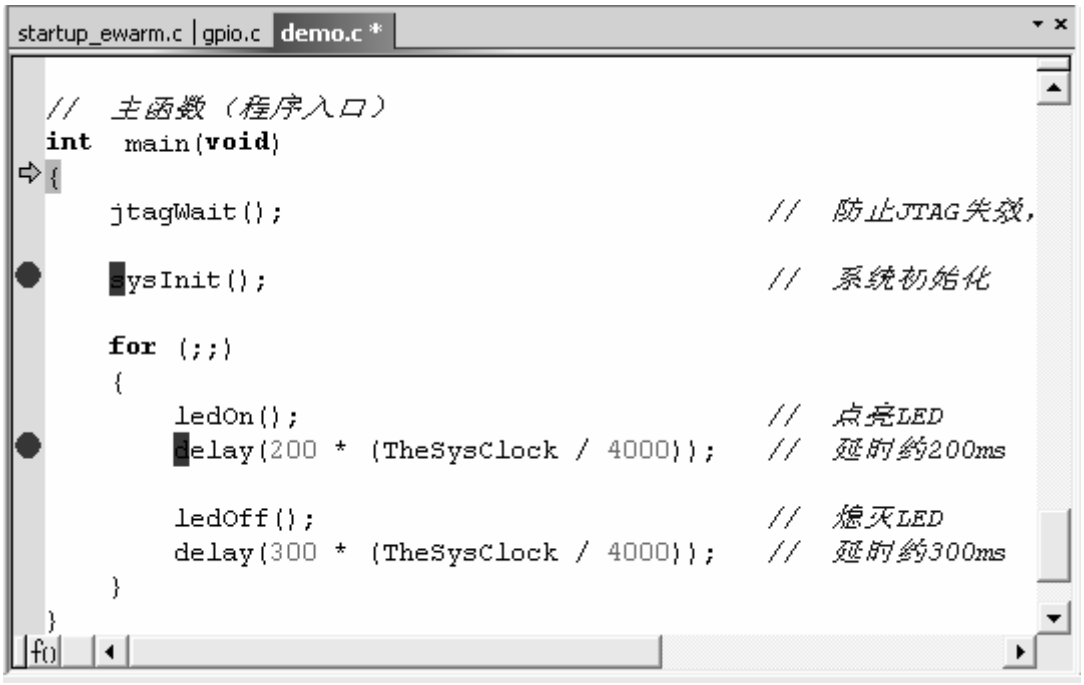

图 1.38 断点设置效果图

● 察看存储器和寄存器

在调试状态下选择View>Memory菜单项，开启存储器窗口，观察存储器单元的内容，如图1.39所示。在Go to下拉文本框中输入地址后按回车键，立即跳到指定的地址。在Memory右侧的下拉列表框中可以选择不同的存储器区域。可以同时打开多个窗口，指定显示存储器区域并允许进行编辑，从而可方便地监控不同存储器区域。窗口显示内容分为3列，最左边一列显示目前察看的地址，中间一列以用户待定的格式显示存储器内容，最右边一列以ASCII码显示存储器内容。将一个指定变量拖到存储器窗口，将立即显示其对应的存储器内容。

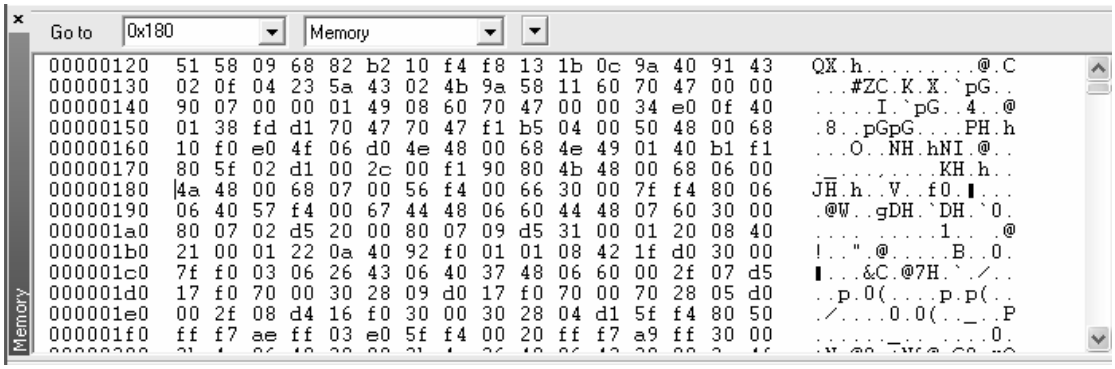

图 1.39 存储器口窗

在调试状态下选择View>Register菜单项，开启寄存器窗口，可观察相关寄存器的内容，如图1.40所示。每当程序暂停执行时，寄存器窗口将以高亮方式显示当前被改变的寄存器内容。双击某个寄存器可以进行编辑修改。

默认状态下，显示的是CPU Registers寄存器组。用户可以在寄存器下拉文本框中选择其它相应寄存器组。

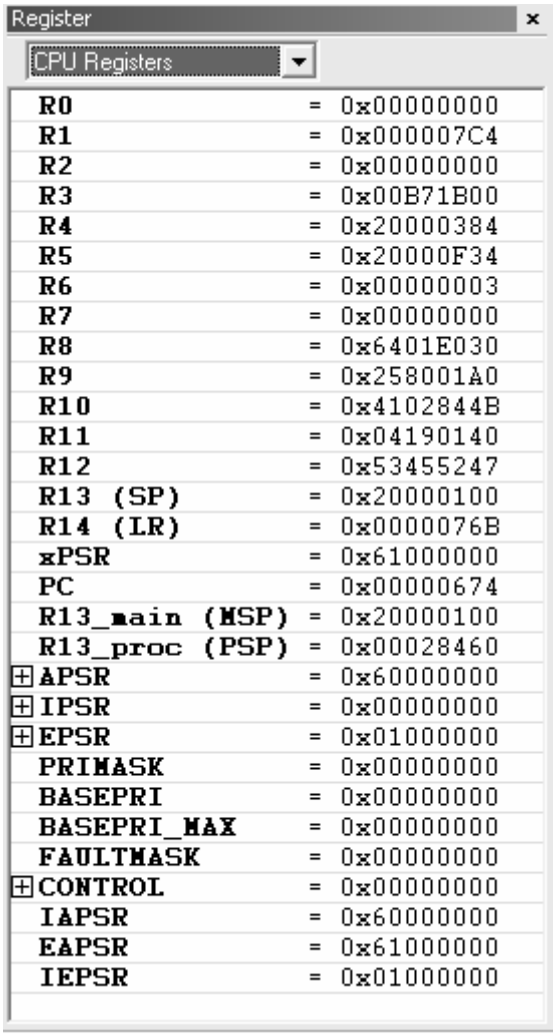

图 1.40 寄存器窗口

● 察看变量和表达式

可以通过以下几种方式来察看变量和表达式的值。

1、在源代码窗口中将鼠标指向希望察看的变量，对应的值将显示在该变量旁边。图1.41

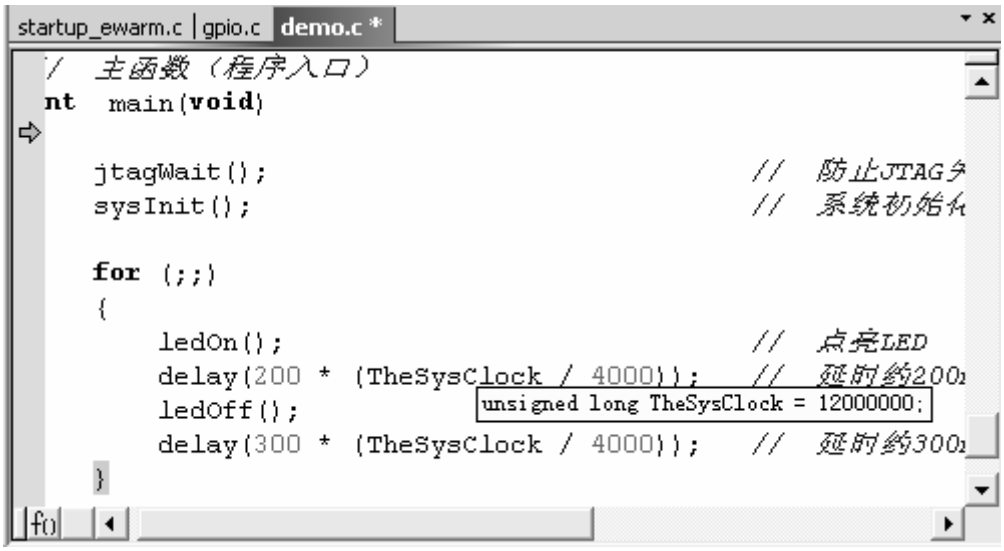

图 1.41 从源代码窗口中直接察看变量

中小矩形框所示内容就是对应变量TheSysClock的值。

2、选择View>Auto菜单项，从开启的Auto窗口中将自动显示与当前语句关相变量和表达式的值。

3、选择View>Locals菜单项，从开启的Locals窗口中将显示当前运行函数的局部变量和函数参数值，如图1.42所示。

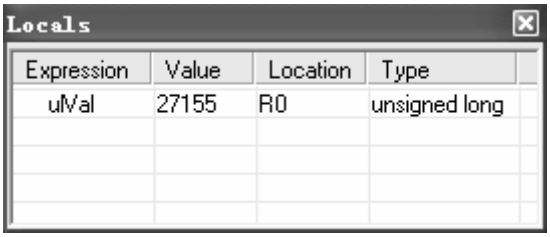

图 1.42 利用 Locals 窗口察看变量

4、选择View>Watch菜单项，开启Watch窗口。在窗口的Expression栏中定义用户希望查看的变量和表达式，也可以直接从源代码窗口将变量和表达式拖到Expression栏中，对应变量和表达式的值将随程序执行而不断更新显示，如图1.43所示。

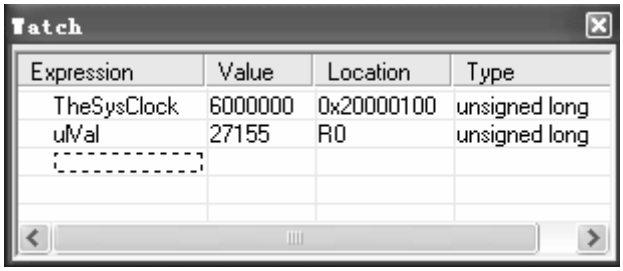

图 1.43 利用 Watch 窗口察看变量

到此为至，我们对一个工程的操作就基本结束了，有些细节用户还可以根据自己的需要进行设置和操作。不足之处还请指教。