

IAR 安装与使用

IAR Embedded Workbench (简称 EW) 的 C/C++ 交叉编译器和调试器是今天世界最完整的和最容易使用专业嵌入式应用开发工具。EW 对不同的微处理器提供一样直观用户界面。EW 今天已经支持 35 种以上的 8 位/16 位 32 位 ARM 的微处理器结构。

EW 包括: 嵌入式 C/C++ 优化编译器, 汇编器, 连接定位器, 库管理员, 编辑器, 项目管理器和 C-SPY 调试器中。使用 IAR 的编译器最优化最紧凑的代码, 节省硬件资源, 最大限度地降低产品成本, 提高产品竞争力。

EWARM 是 IAR 目前发展很快的产品, EWARM 已经支持 ARM7/9/10/11XSCALE, 并且在同类产品中具有明显价格优势。其编译器可以对一些 SOC 芯片进行专门的优化。如 Atmel, TI, ST, Philips。除了 EWARM 标准版外, IAR 公司还提供 EWARM BL (256K) 的版本, 方便了不同层次客户的需求。

IAR System 是嵌入式领域唯一能够提供这种解决方案的公司。EW 支持 35 种以上的 8 位/16 位/32 位的微处理器结构。

IAR Embedded Workbench 集成的编译器主要产品特征:

- 高效 PROMable 代码
- 完全标准 C 兼容
- 内建对应芯片的程序速度和大小优化器
- 目标特性扩充
- 版本控制和扩展工具支持良好
- 便捷的中断处理和模拟
- 瓶颈性能分析
- 高效浮点支持
- 内存模式选择
- 工程中相对路径支持

我们为什么要放弃使其他各种用免费的开发工具, 而选择需要支付费用来购买 IAR Systems 的开发工具? 主要包括以下几点原因:

由于 IAR 公司在微处理器 C/C++ 编译器设计方面的丰富经验, 目前没有任何一家公司的产品可以接近 IAR 公司针对 8 位、16 位、32 位处理器生产的 30 多种不同 C/C++ 编译器的水平。

经过反复实验证明, IAR Systems 的 C/C++ 编译器可以生成高效可靠的可执行代码, 并且应用程序规模越大, 效果明显。与其他的工具开发厂商相比, 系统同时使用全局和针对具体芯片的优化技术。连接器提供的全局类型检测和范围检测对于生成目标的代码的质量是至关重要。

IAR Systems 一贯使用精简的优化技术--基于我们最新技术架构的, 针对 AVR 的 IAR Embedded Workbench 4.10B 版, 生成的代码的尺寸比 3.20A 版缩小了 10%, 远远小于其他同类编译器生成的代码尺寸。IAR Embedded Workbench 生成的可以执行代码可以运行于更小尺寸、更低成本的为处理器之上, 从而降低产品的开发成本。

为什么小就意味着完美? 因为紧缩的代码, 就说明它可以很好的运行在更小、更便宜的芯片上! 假设公司要生产 10,000 设备, 而每一台因为使用了更小尺寸处理器的设备可以节

省2美元，这对公司来说将是一笔很客观的收入。产品的成本对于设计部门来说不是最先考虑的因素也不是开发工具的任务，但是它确实产品或销售经理最感兴趣的内容。

尺寸小不仅仅意味着廉价，它也为各种附加的功能留下的充足的扩展空间。假设你的客户中途需要为他们的产品设计增加一些新的功能特性，而在这个阶段再去选择另一款芯片是不可行的。这时，IAR Systems提供的高效的编译器加上代码检测服务为公司在最终期限之前完成任务提供了可能。我们应该清楚这种情况在我们以前的工作中会经常遇到。

忽略项目的最终期限，开发者需要依靠一些可靠的开发工具来完成任务。未能按时完成进度会给项目带来不便，而恶性循环将会导致所有进度安排的拖延，后果变得十分严重。IAR Embedded Workbench 被认为是一款稳定可靠的开发工具，它提供连续的工作流，使开发者可以专心于项目的开发，提高开发效率。

IAR Embedded Workbench 是一套完整的集成开发工具集合：包括从代码编辑器、工程建立到C/C++编译器、连接器和调试器的各类开发工具。它和各种仿真器、调试器紧密结合，使用户在开发和调试过程中，仅仅使用一种开发环境界面，就可以完成多种微控制器的开发工作。

除上述的几点之外，在IAR Embedded Workbench, IAR Systems 还提供了visualSTATE和IAR MakeApp两套图形开发工具帮助开发者完成应用程序的开发，它可以根据设计自动生成应用程序代码和自动生成驱动程序，使开发者摆脱这些耗时的任务同时保证了代码的质量。详细信息请参阅<http://www.iar.com>网站的相关内容。

不论客户在哪里，IAR Systems 都可以为其提供完善的技术支持和设计服务。

下面我们就从安装到设置一步一步地学习如何使用IAR集成开发环境。

IAR 集成开发环境

在本节将逐步介绍 IAR 安装、IAR 开发环境如何添加文件、新建程序文件、设置工程选项参数、编译和连接、程序下载、仿真调试。

IAR 安装

如同 Windows 操作系统其它一般的软件安装一样，单击 setup.exe 进行安装，你将会看到如图 1 的界面。



图 1 IAR 安装 1

单击“Next”至下一步，将分别需要填写你的名字、公司以及认证序列，如图 2 所示。

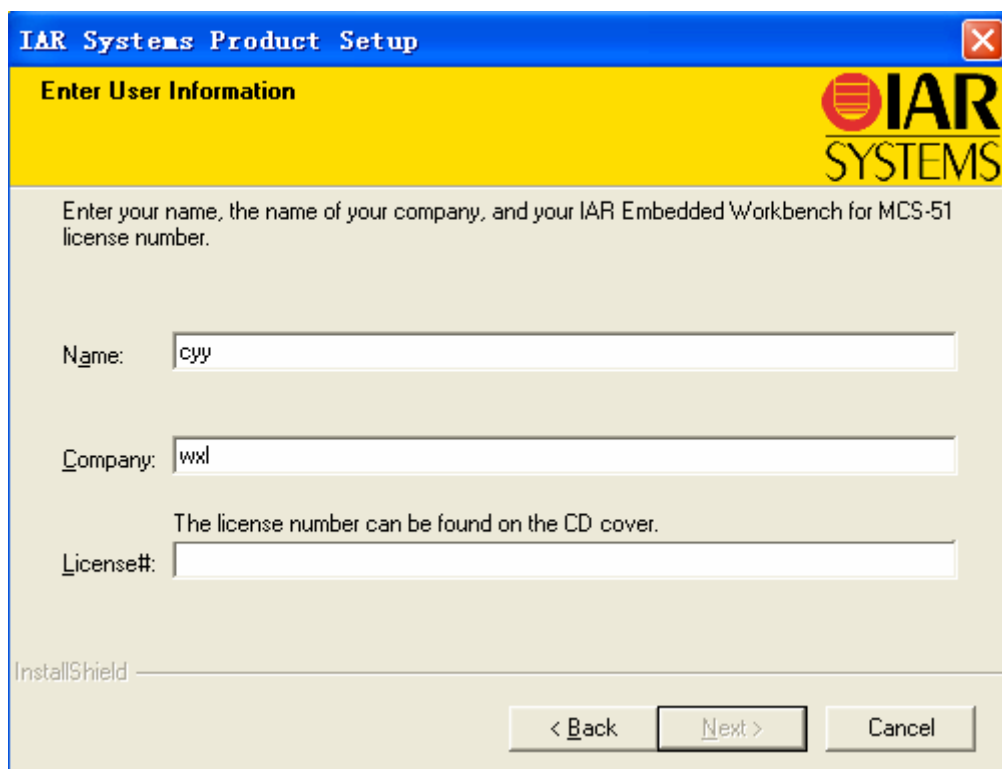


图 2 IAR 安装 2

正确填写后，单击“Next”至下一步，将分别需要由你计算机的机器码和认证序列生成的序列钥匙，如图 3 所示。

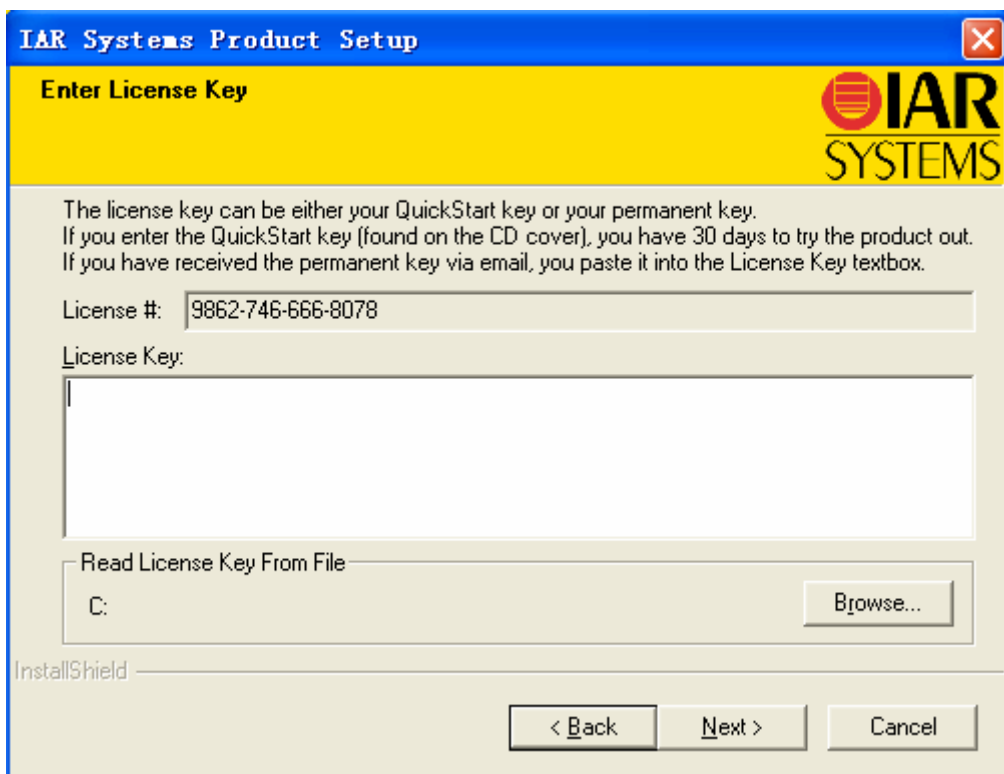


图 3 IAR 安装 3

输入的认证序列以及序列钥匙正确后，单击“Next”到下一步。如图 4 所示，在你将选择完全安装或是典型安装，在这里我们选择第 1 个也就是完全安装。

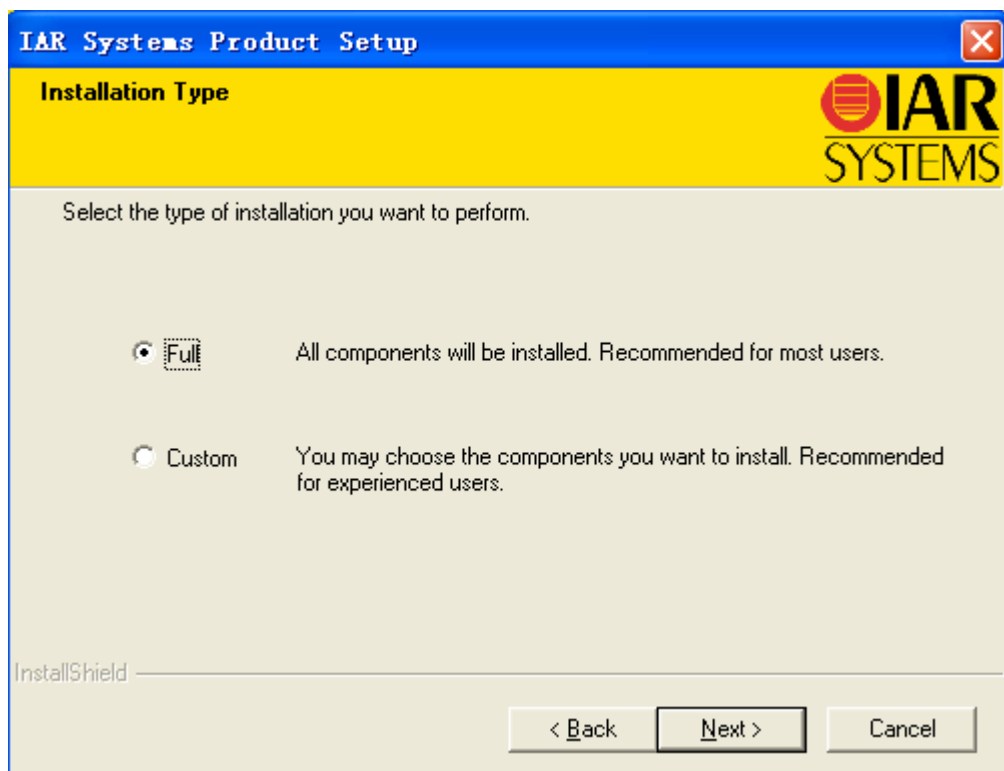


图 4 IAR 安装 4

单击“Next”到下一步，在这里你将查证看你输入的信息是否正确，如图 5 所示。如果

需要修改，单击“Back”返回修改。

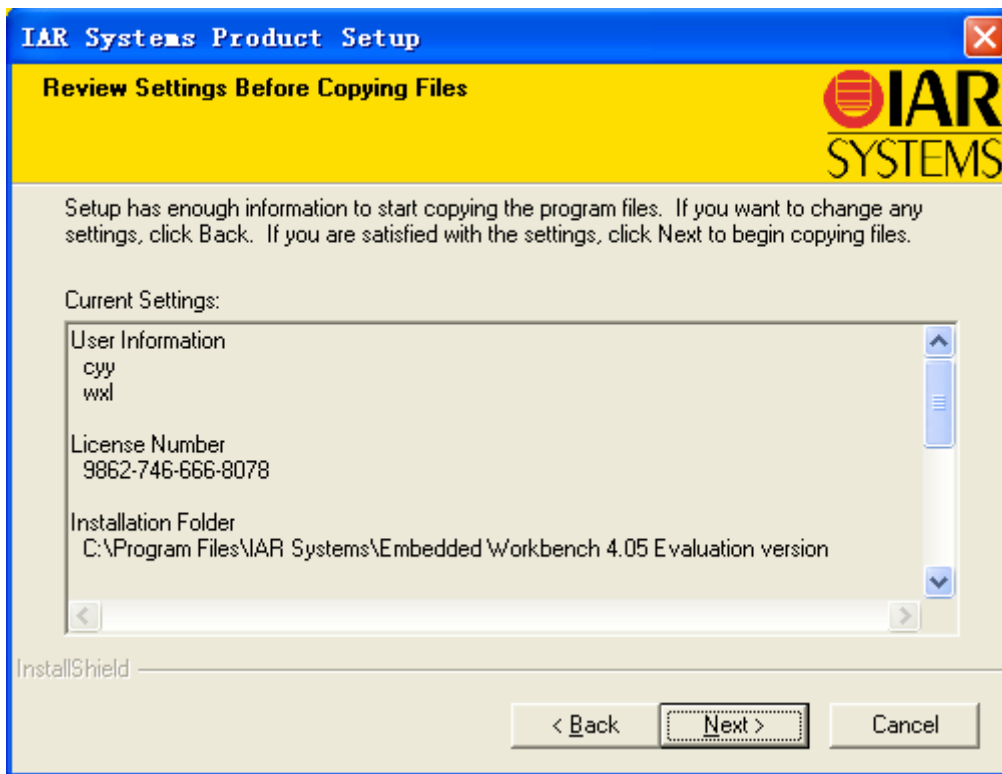


图 5 IAR 安装 5

单击“Next”正式开始安装，如图 6 所示。在这你将看到安装进度，这将需要几分钟时间的等待，现在你需要耐心等待。

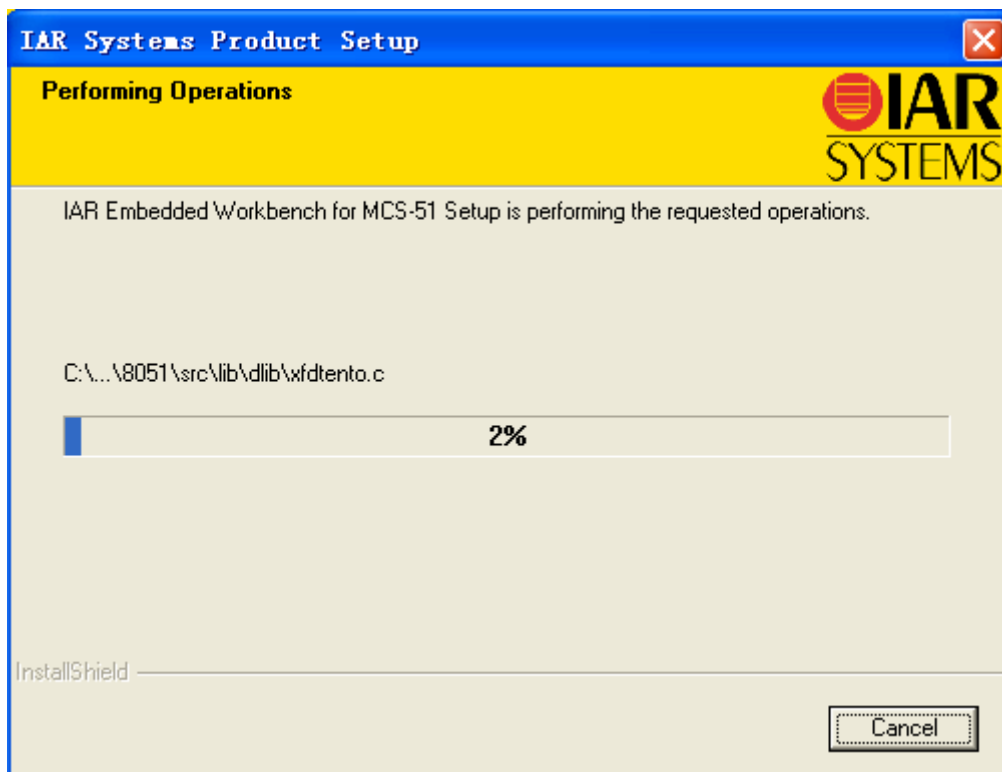


图 6 IAR 安装 6

当进度到 100%时，它将跳到下一个界面，如图 7 所示。在此你可选择查看 IAR 的介绍以及是否立即运行 IAR 开发集成环境。单击“Finish”来完成安装。

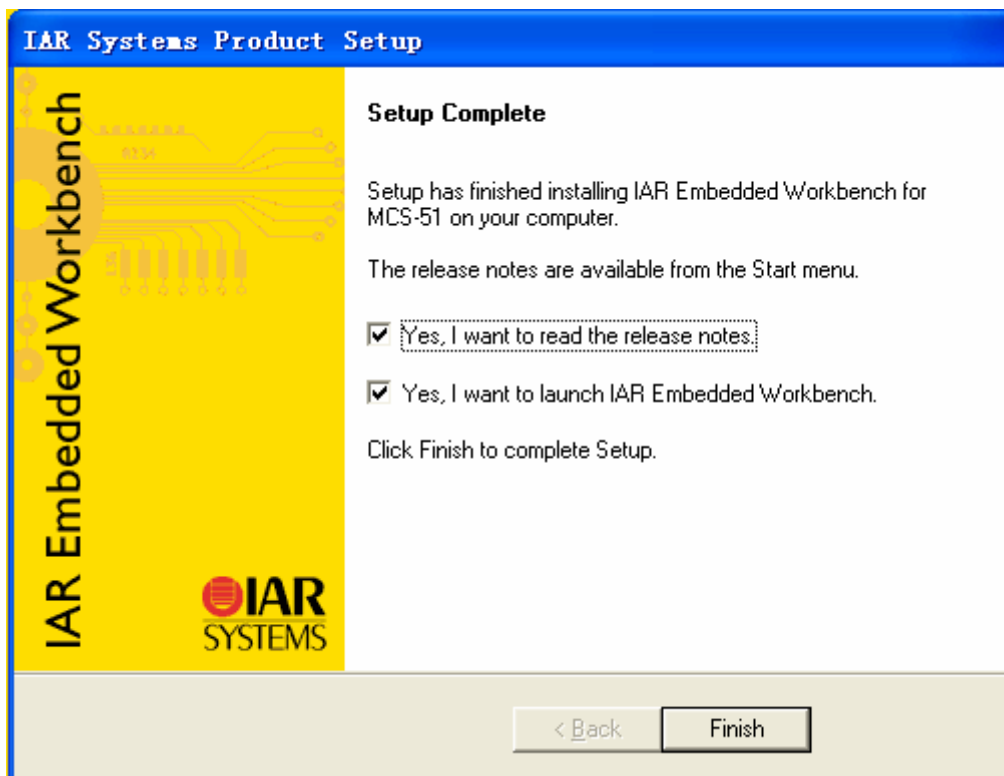


图 7 IAR 安装 7

完成安装后，你可以从“开始”那里找到刚刚安装的 IAR 软件，如图 8 所示。

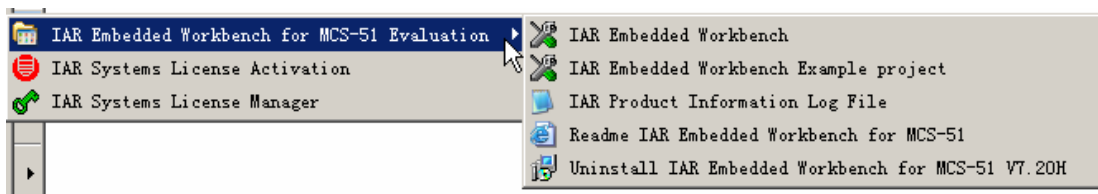


图 8 IAR 安装 8

现在你可以通过在桌面的快捷方式或在“开始”按键中选择程序来启动你的 IAR 软件开发环境。

现在你可以通过在桌面的快捷方式或在“开始”按键中选择程序来启动你的 IAR 软件开发环境。

使用 IAR 开发环境首先应建立一个新的工作区。在一个工作区中可创建一个或多个工程。用户打开 IAR Embedded Workbench 时，已经建好了一个工作区，一般会显示如下图 9 窗口，可选择打开最近使用的工作区或向当前工作区添加新的工程。

选择 File\New\Workspace。现在用户已经建好一个工作区，可创建新的工程并把它放入工作区。

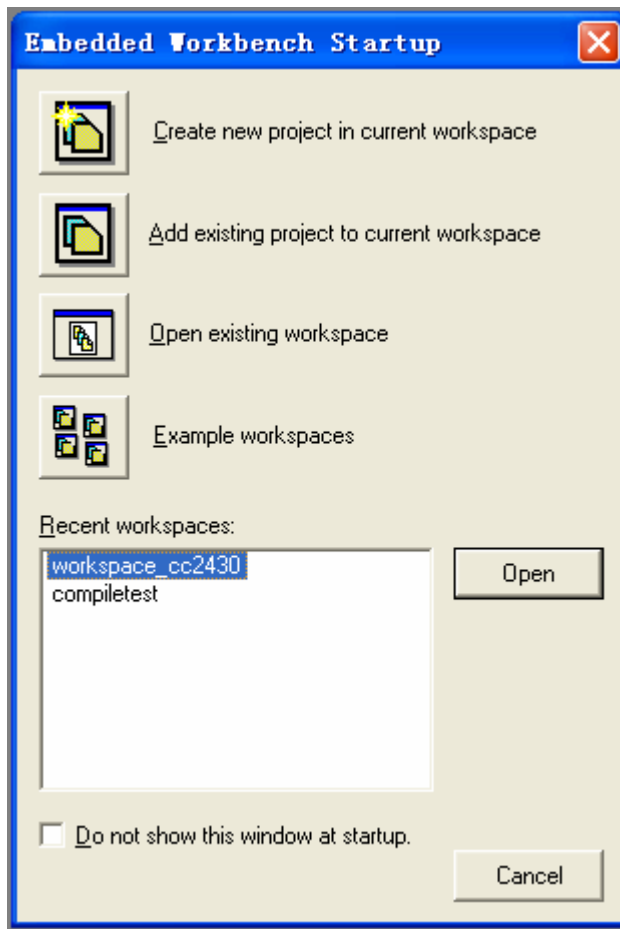


图 9 打开一个工作区

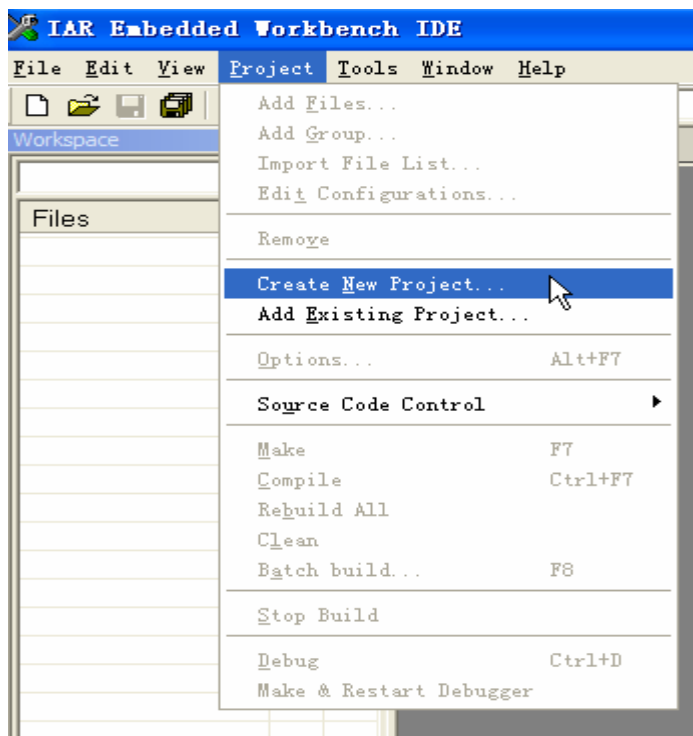


图 10 建立一个新工程

单击 Project 菜单，选择 Create New Project，如图 10 所示。

通往无线的桥梁 无线世界的先锋

弹出图 11 建立新工程对话框，确认 Tool chain 栏已经选择 8051，在 Project templates: 栏选择 Empty project 单击下方 OK 按钮。

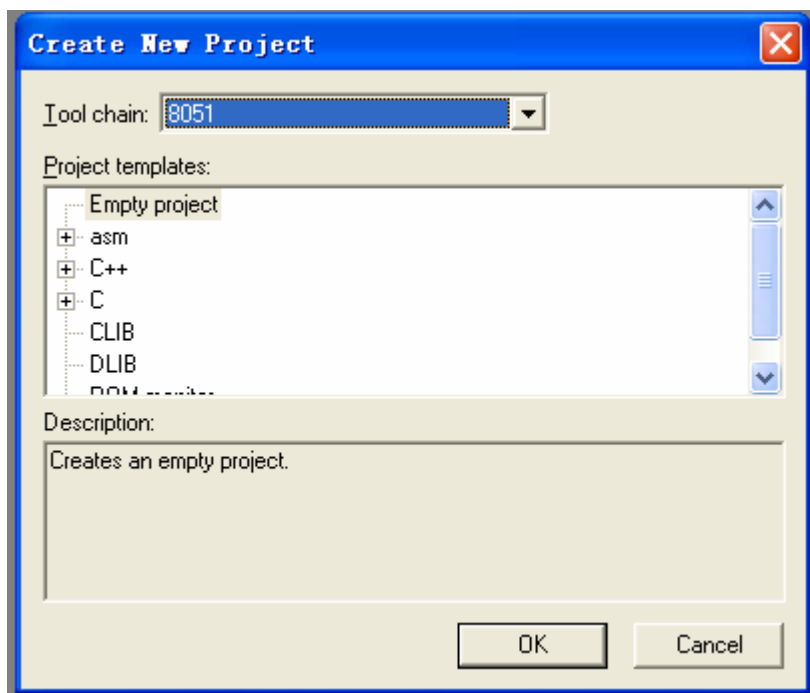


图 11 选择工程类型

根据需要选择工程保存的位置，更改工程名，如 ledtest 单击 Save 来保存，如图 12 所示。这样便建立了一个空的工程。

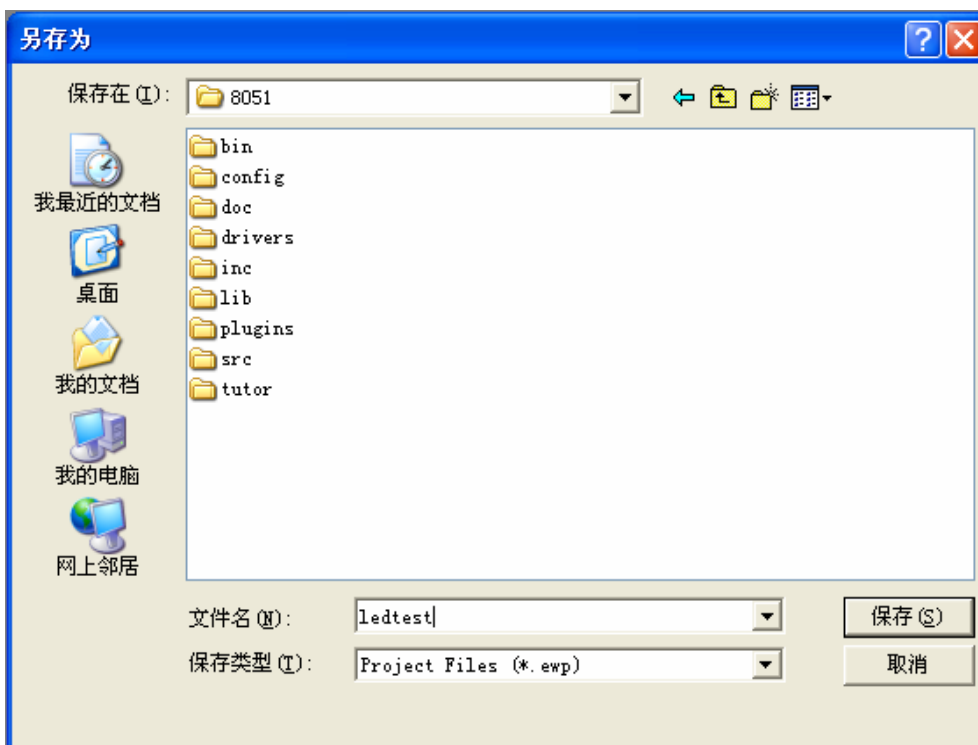


图 12 保存工程

这样工程就出现在工作区窗口中了，如图 13 所示。

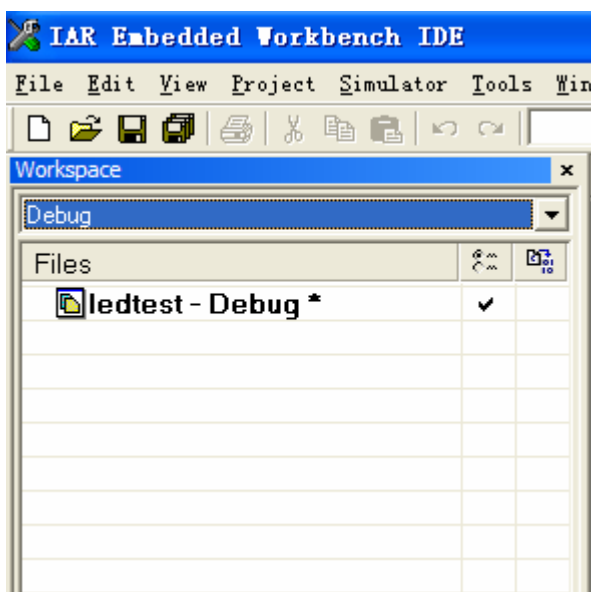


图 13 工作区窗口中的工程

系统产生两个创建配置：调试和发布。在这里我们只使用 Debug 即调试。项目名称后的星号(*)指示修改还没有保存。选择菜单 File\Save\Workspace ,保存工作区文件，并指明存放路径，这里把它放到新建的工程目录下。单击 Save 保存工作区，如图 14 所示。

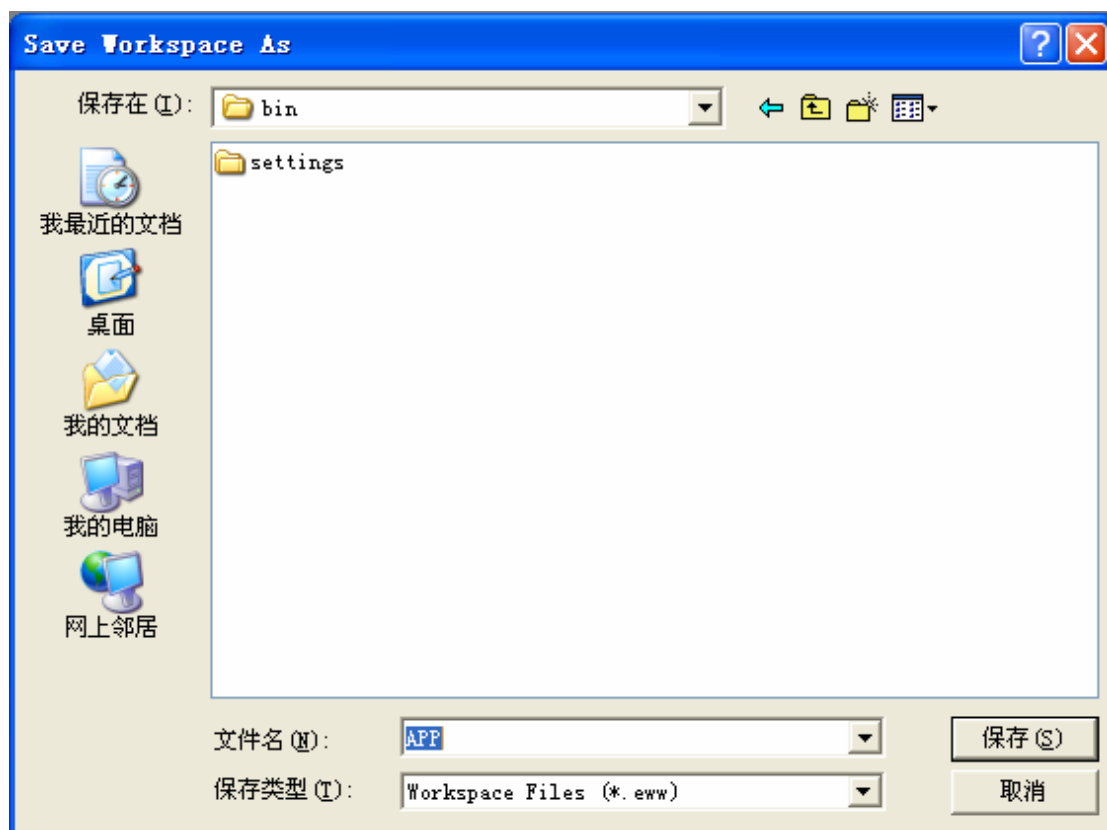



图 14 保存工作区

添加文件或新建程序文件

选择菜单 Project\Add File 或在工作区窗口中，在工程名上点右键，在弹出的快捷菜单

中选择 Add File，弹出文件打开对话框，选择需要的文件单击“打开”退出。

如没有建好的程序文件也可单击工具栏上的  或选择菜单 File\New\File 新建一个空文本文件，向文件里添加程序清单 1 代码。

程序清单 1 如下：

```
#include "ioCC2430.h"
void Delay(unsigned char n)
{
    unsigned char i;
    unsigned int j;
    for(i = 0; i < n; i++)
        for(j = 1; j <= 255; j++);
}
void main(void)
{
    // CC2430 中，I/O 口做普通 I/O 使用时和每个 I/O 端口相关的寄存器有 3 个，分别是//PxSEL
    //功能选择寄存器，PxDIR 方向寄存器，PxINP 输入模式寄存器，其中 x 为 0, 1, 2。
    //这里选择 P1.0 上的 色 LED 作为 I/O 测试。
    SLEEP &= ~0x04;
    while(!(SLEEP & 0x40)); //晶体振荡器开启且稳定
    CLKCON &= ~0x47; //选择 1-32MHz 晶体振荡器
    SLEEP |= 0x04;
    P1SEL = 0x00; //P1.0 为普通 I/O 口
    P1DIR = 0x01; //P1.0 输出
    while(1)
    {
        P1_0 = 1;
        Delay(10);
        P1_0 = 0;
        Delay(10);
    }
}
```

选择菜单 File\Save 弹出保存对话框，如图 15 所示。

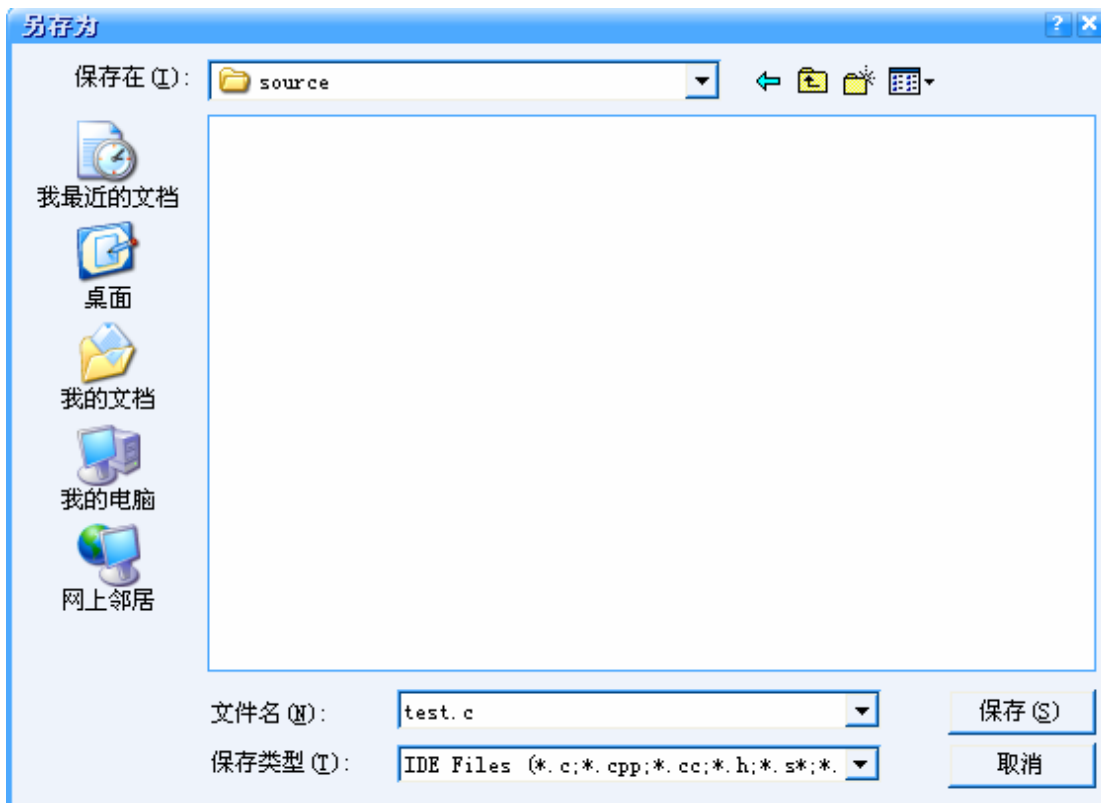


图 15 保存程序文件

新建一个 source 文件夹，将文件名改为 test.c 后保存到 source 文件夹下。按照前面添加文件的方法将 test.c 添加到当前工程里，完成的结果如下图 15 所示。

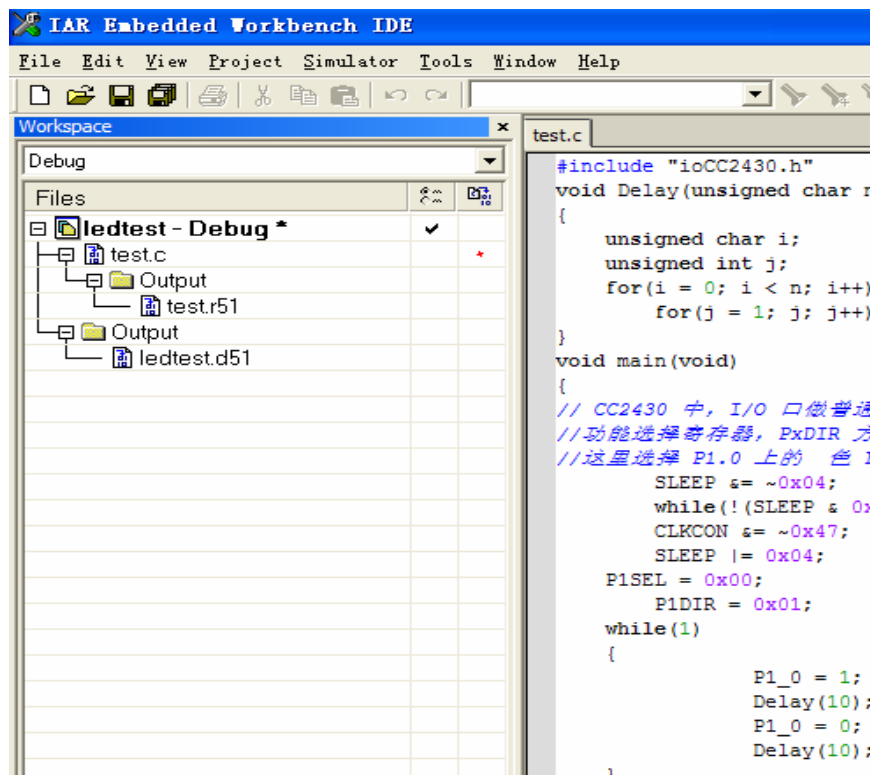


图 16 添加程序文件后的工程
通往无线的桥梁 无线世界的先锋

设置工程选项参数

选择 Project 菜单下的 Options 配置与 CC2430 相关的选项。

Target 标签

按下图 17 配置 Target, 选择 Code model 和 Data model, 以及其它参数。

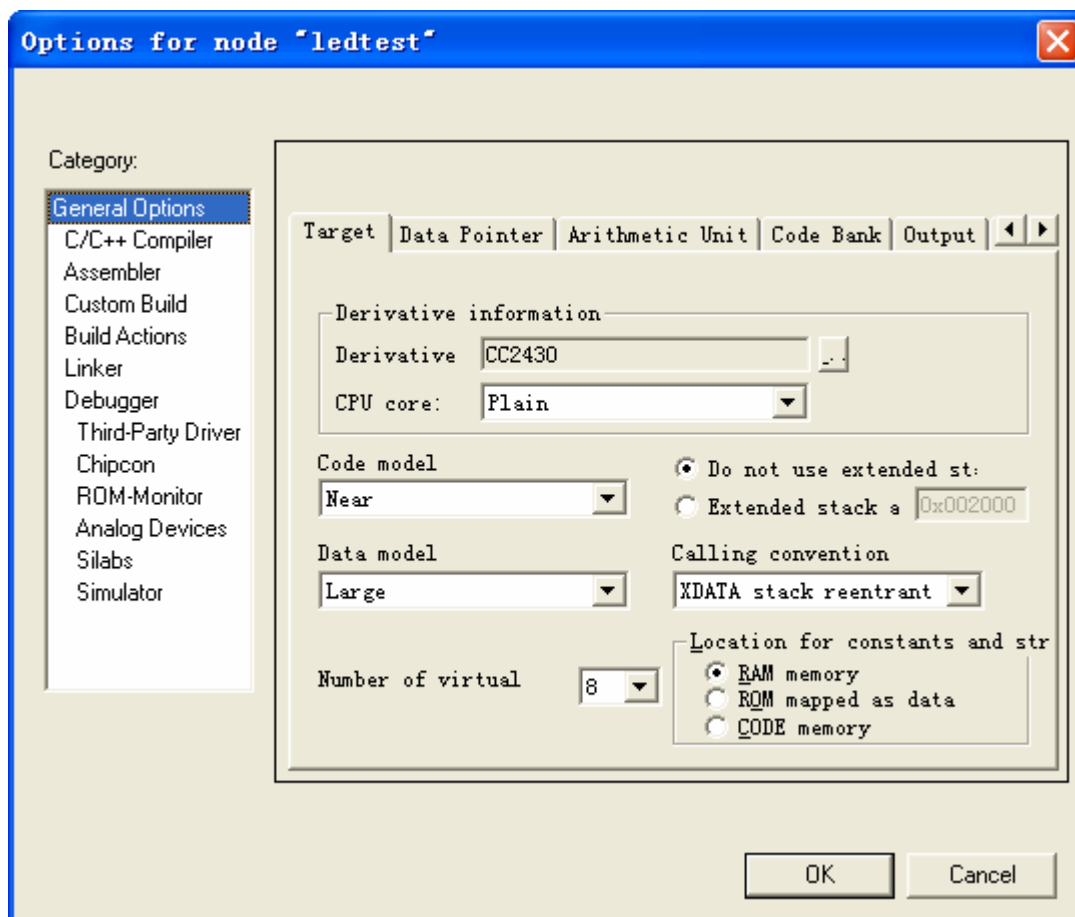


图 17 配置 Target

单击 Derivative information 栏右边的按钮, 选择程序安装位置如这里是 IAR Systems\Embedded Workbench4.05 Evaluation version\8051\config\derivatives\chip-con 下的文件 CC2430.i51。

Data Pointer 标签

如图 18 所示, 选择数据指针数 1 个, 16 位。

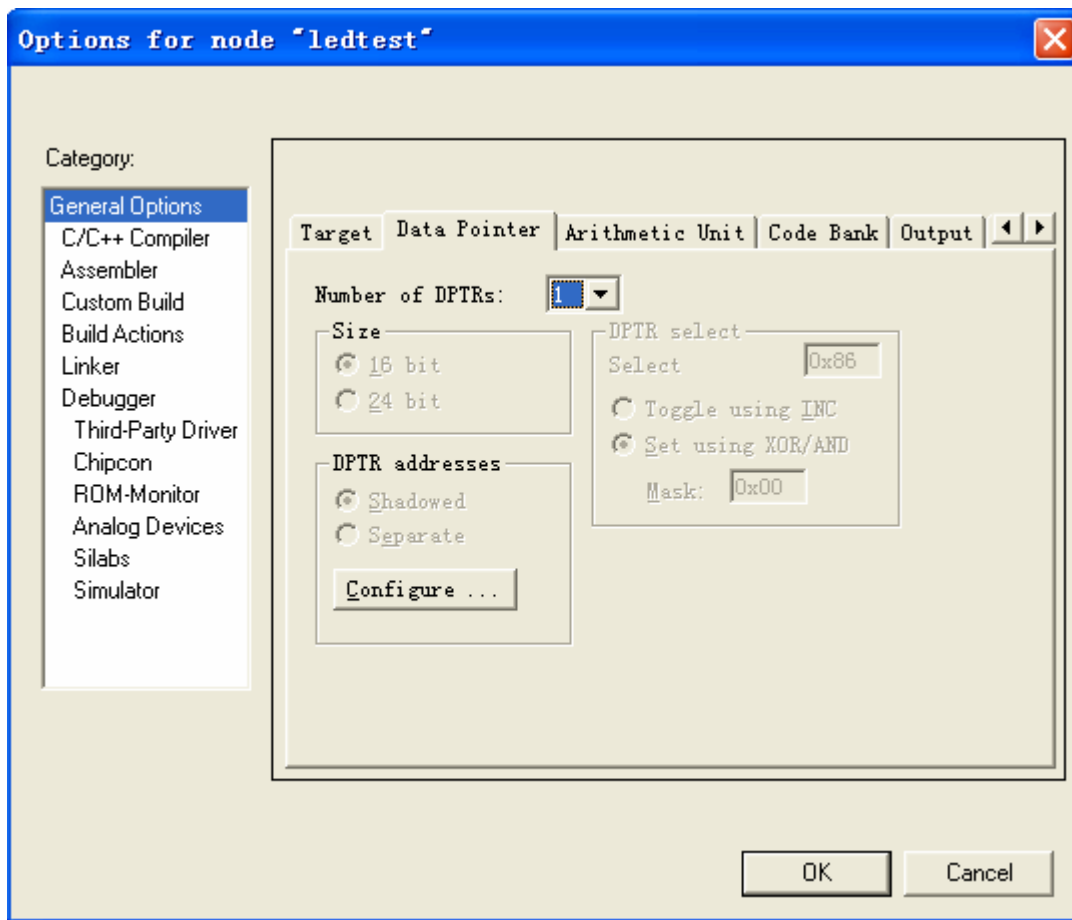


图 18 数据指针选择

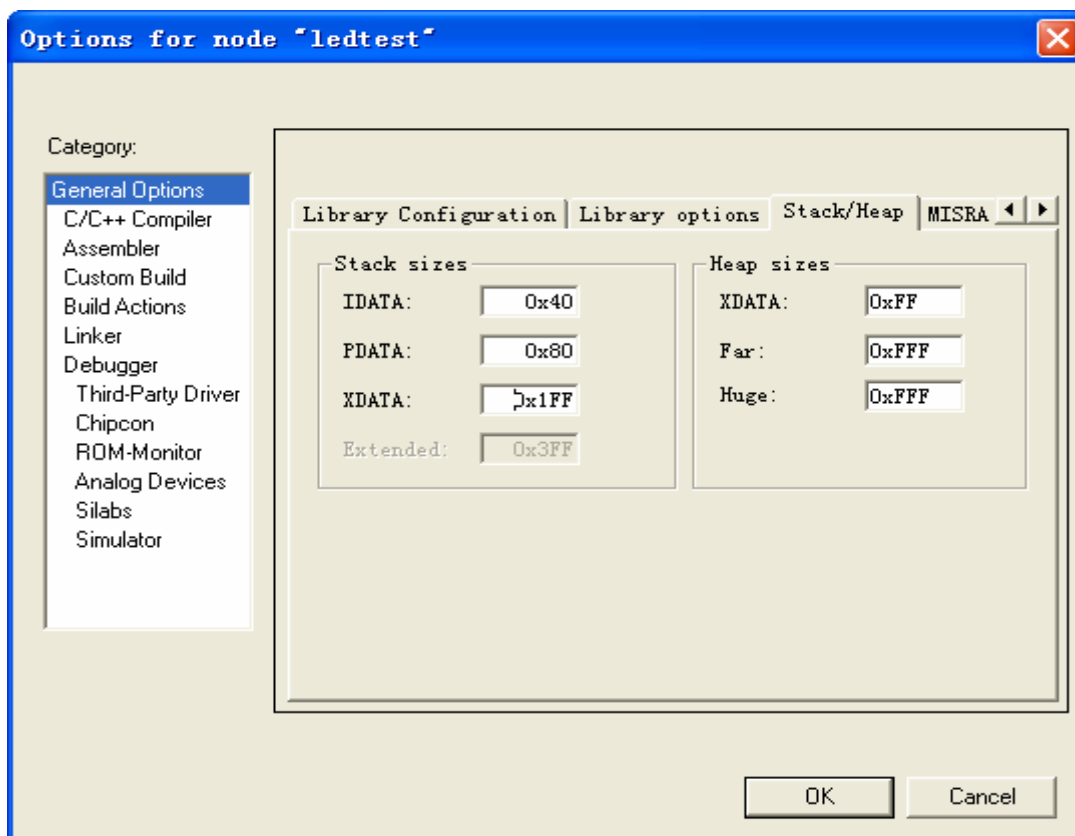


图 19 Stack/Heap 设置

Stack/Heap 标签

如图 19 所示，改变 XDATA 栈大小到 0x1FF。

单击 Options 中右边框架内的 Linker 选项，配置相关的选项。

Output 标签

选中 Override default 可以在下面的文本框中更改输出文件名。如果要用 C-SPY 进行调试，选中 format 下面的 Debug information for C-SPY，如图 20 所示。

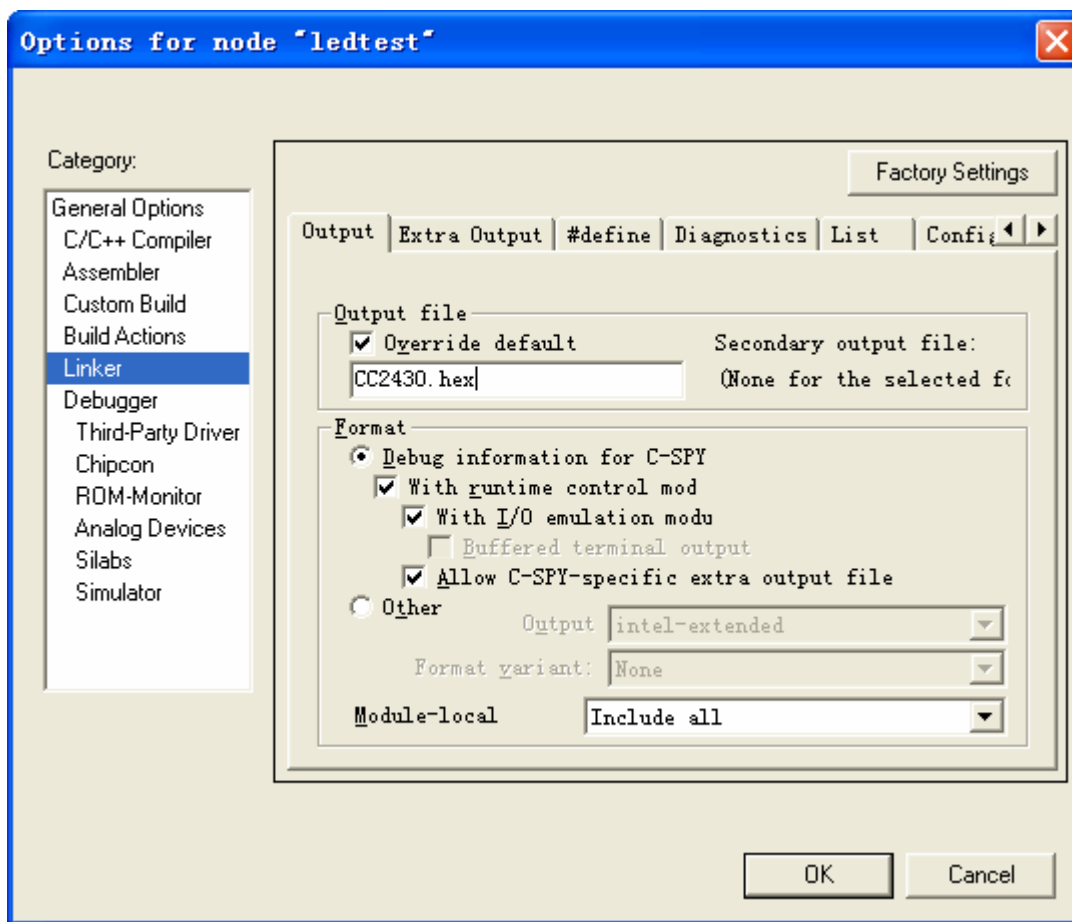


图 20 输出文件设置

Config 标签

如图 21 所示，单击 Linker command file 栏文本框右边的按钮，选择正确的连接命令文件，如表 5.1 所示。

表 5.1 Code Model 关系表

Code Model	File
Near	lnk51ew_cc2430.xcl
Banked	lnk51ew_cc2430b.xcl

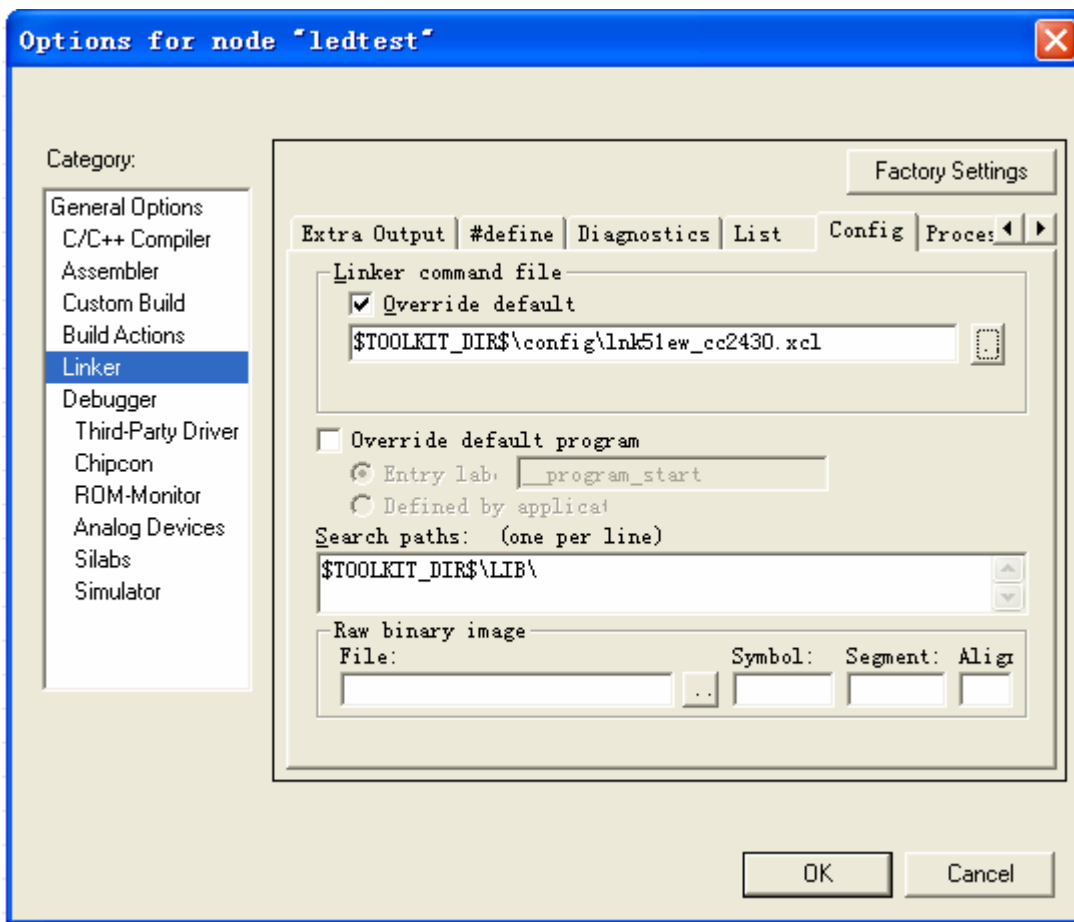


图 21 选择连接命令文件

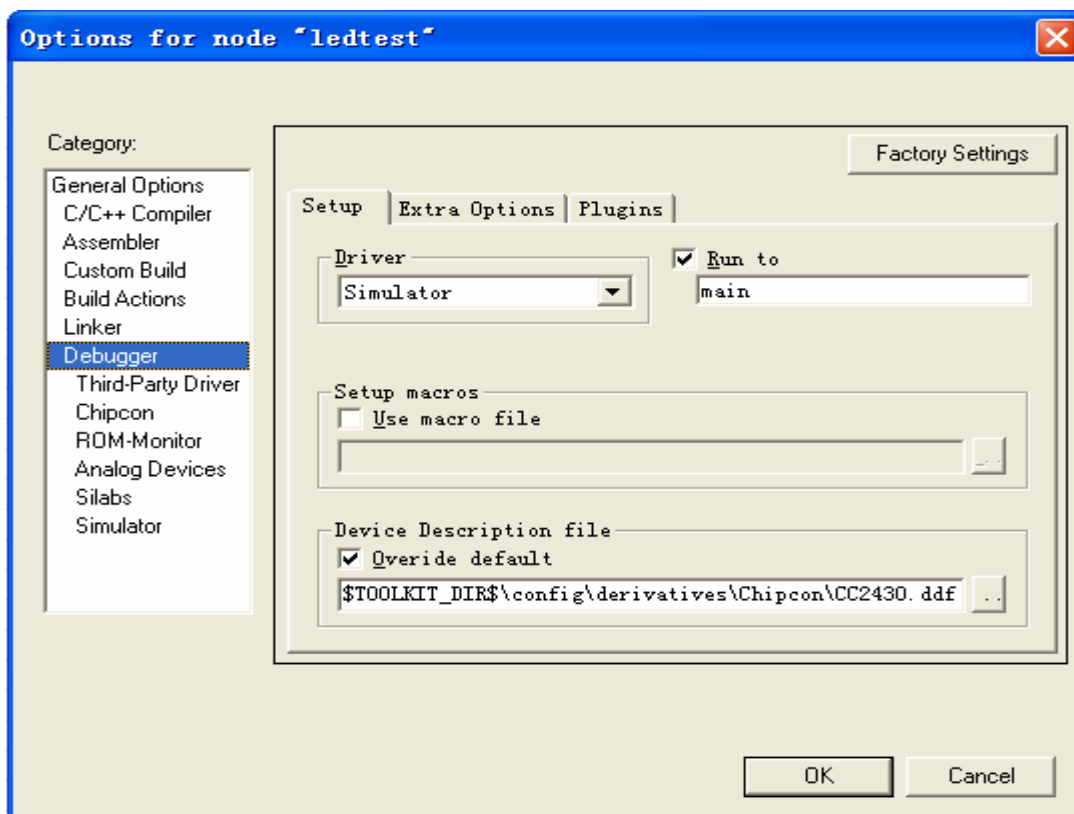


图 22 配置调试器

Debugger

单击 Options 中右边框架内的 Debugger 选项，配置相关的选项。在 Setup 标签按下图 22 所示设置。

在 Device Description file 选择 CC2430.ddf 文件，其位置在程序安装文件夹下如 C:\Program Files\IAR Systems\Embedded Workbench 4.05 Evaluation version\8051\Config\derivatives\chipcon。

最后按下“ok”保存设置。

编译、连接、下载

选择 Project\Make 或按 F7 键编译和连接工程，如图 23 所示。

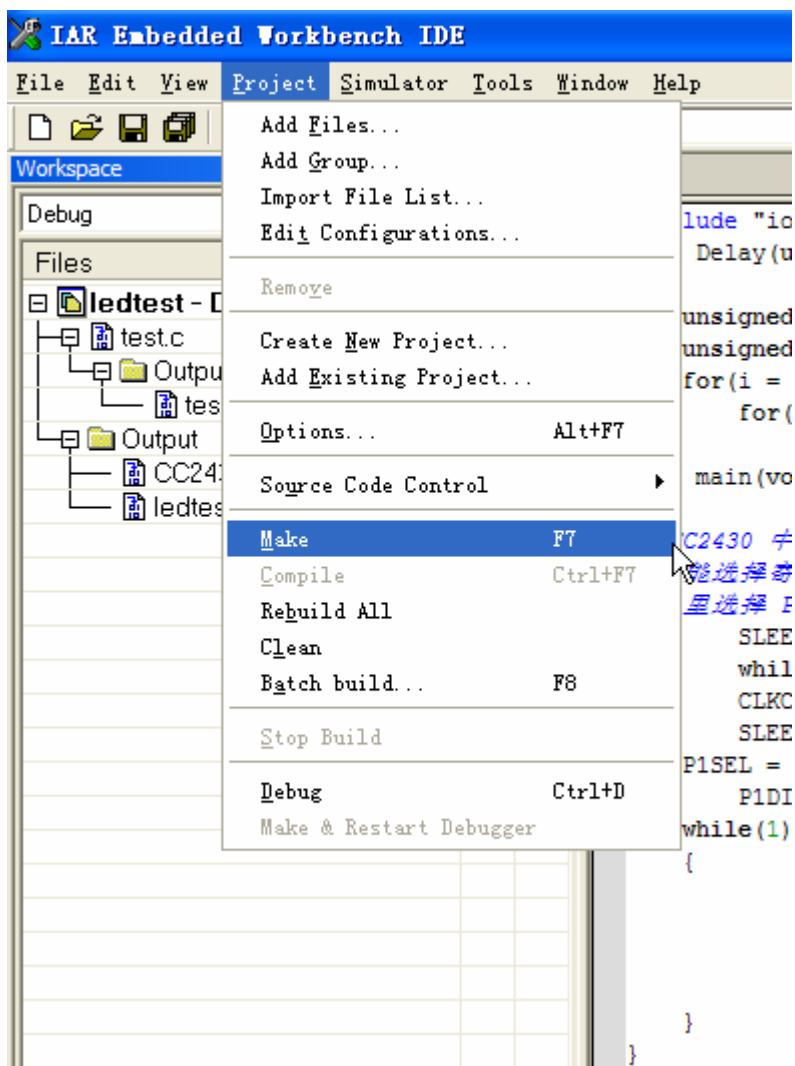
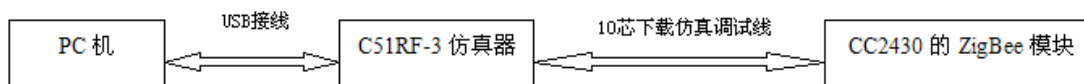


图 23 编译和连接工程

成功编译工程，并且没有错误信息提示后，按照下图连接硬件系统(注：“CC2430 的 ZigBee 模块”部分可以换成网关、扩展板等与仿真器引脚相配器件)。



选择 IAR 集成开发环境中菜单 Project\Debug(如图 30 所示) 或按快捷键 CTRL+D 进入调试状态, 也可按工具栏上按钮进入程序下载, 程序下载完成后, IAR 将自动跳转至仿真状态。

仿真调试

编译好后接下来就是调试程序了。首先你需要连接你的硬件平台才能进行调试。在计算机与 ZigBee 硬件系统连接前, 你需要确保你已在你的计算机上安装了必要的仿真器驱动。

安装仿真器驱动---手动

安装仿真器前确认 IAR Embedded Workbench 已经安装。手动安装适用于系统以前没有安装过仿真器驱动的情况。

将仿真器通过开发系统附带的 USB 电缆连接到 PC 机, 在 Windows XP 系统下, 系统找到新硬件后提示如下对话框, 选择“从列表或指定位置安装”, 点下一步如图 24 所示。

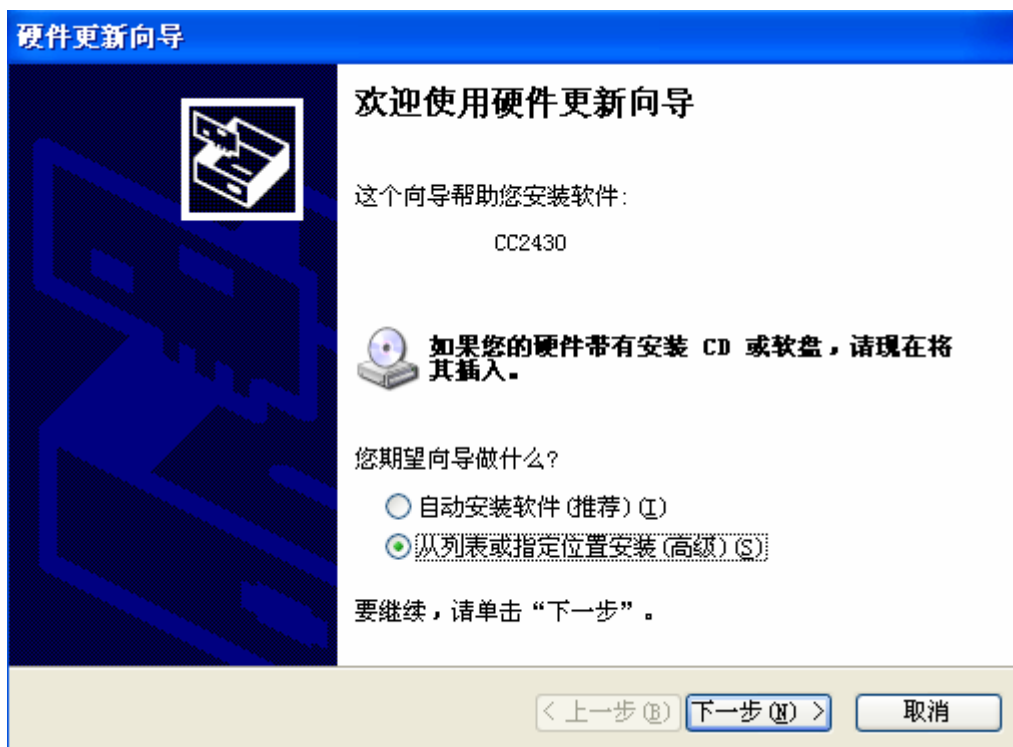


图 24 系统找到仿真器

如下图 25 设好驱动安装选项, 单击右边的“浏览”按钮选择驱动所在路径。

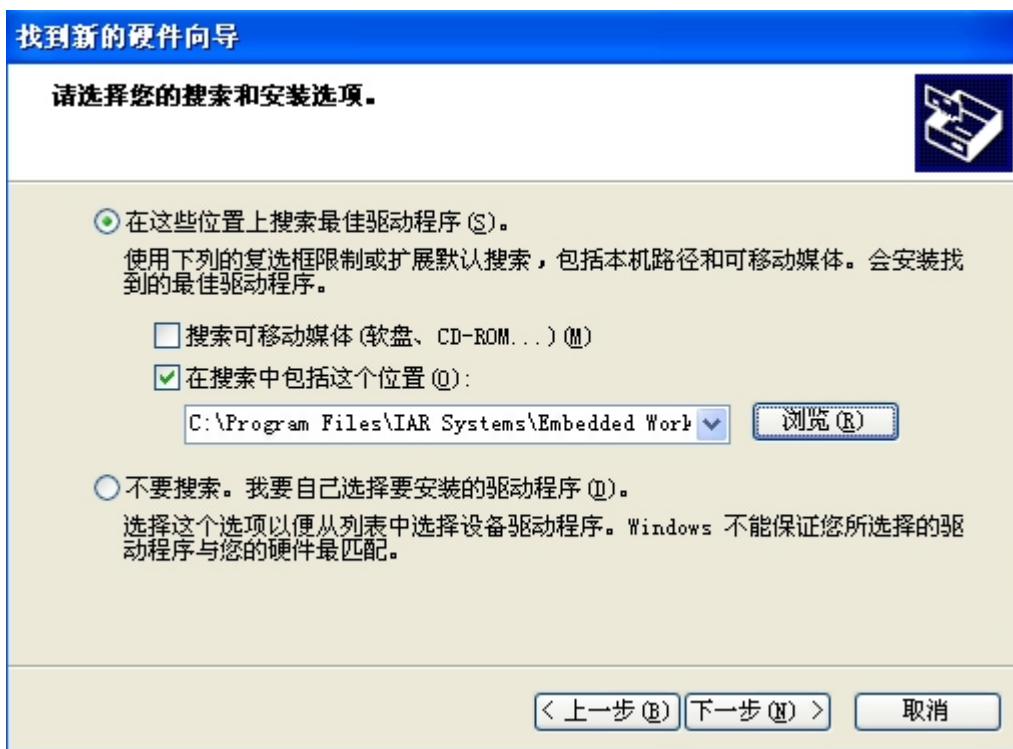


图 25 驱动安装选项

驱动文件在 IAR 程序安装目录下，在 C:\ Program Files\IAR Systems\Embedded Workbench 4.05 Evaluation version\8051\drivers\chipcon，如下图 26 所示。

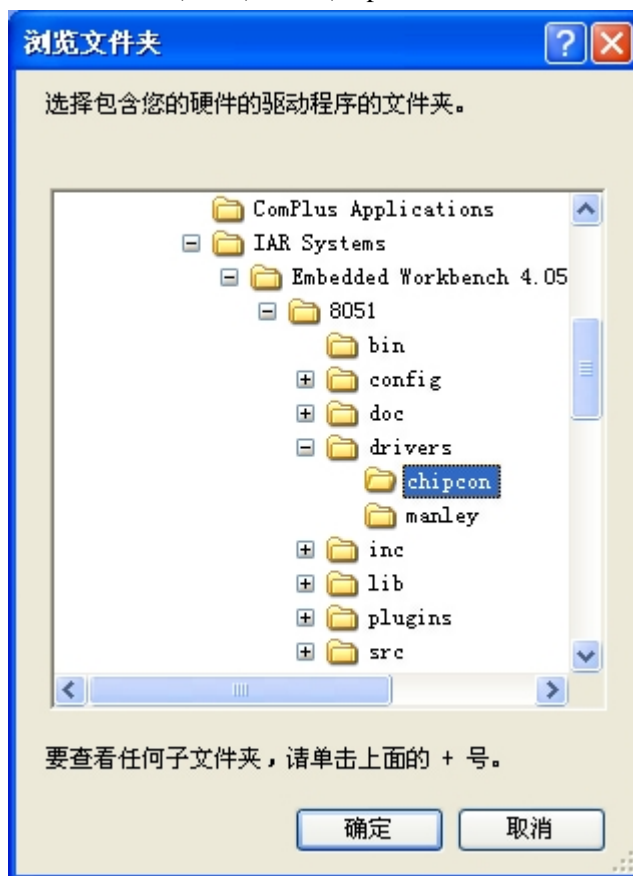


图 26 选择驱动路径

通往无线的桥梁 无线世界的先锋

选中 chipcon 文件夹，点“确定”退出，回到安装选项界面，点“下一步”，系统安装完驱动后提示完成对话框，单击“完成”退出安装。



图 27 完成驱动安装

安装仿真器驱动---自动

将仿真器通过开发系统附带的 USB 电缆连接到 PC 机，在 Windows XP 系统下，系统找到新硬件后提示如下对话框，选择“自动安装软件”，点“下一步”如图 28 所示。

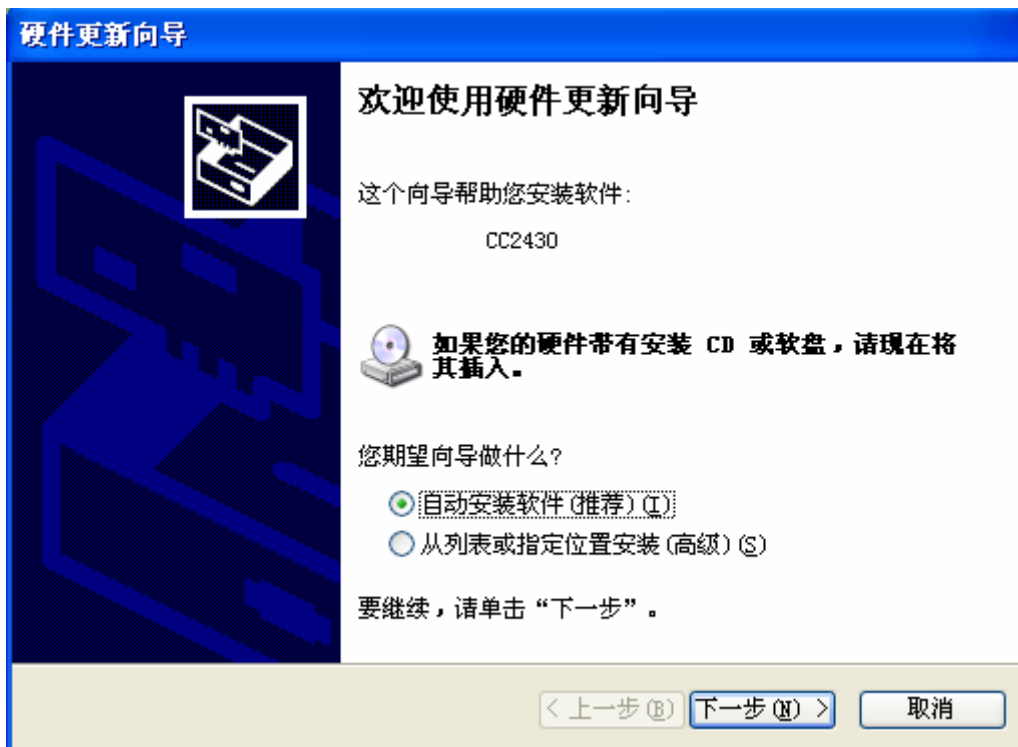


图 28 系统找到仿真器

向导会自动搜索并复制驱动文件到系统, 如图 29 所示。



图 29 安装驱动文件

系统安装完驱动后提示完成对话框, 单击“完成”退出安装, 如图 27 所示。

进入调试

安装完成仿真器驱动后, 通过 USB 接口把 ZigBee 开发系统与计算机连接(详细介绍请
通往无线的桥梁 无线世界的先锋

看第四章的介绍)后, 进入 IAR 编译环境进行仿真调试。

选择菜单 Project\Debug 或按快捷键 CTRL+D 进入调试状态, 也可按工具栏上按钮进入调试, 如图 30 所示。

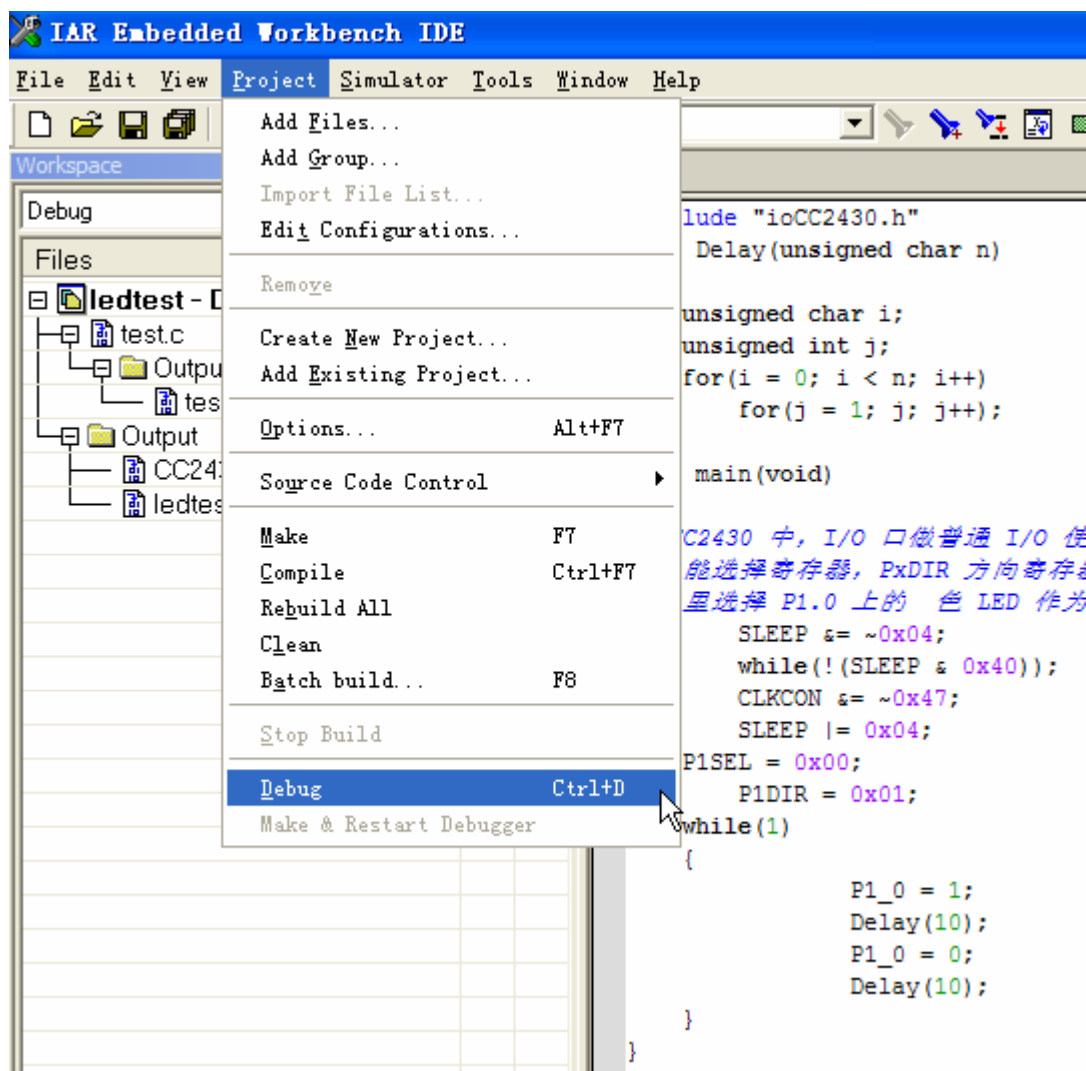


图 30 进入调试

进入调试后, 整体窗口如图 31 所示。

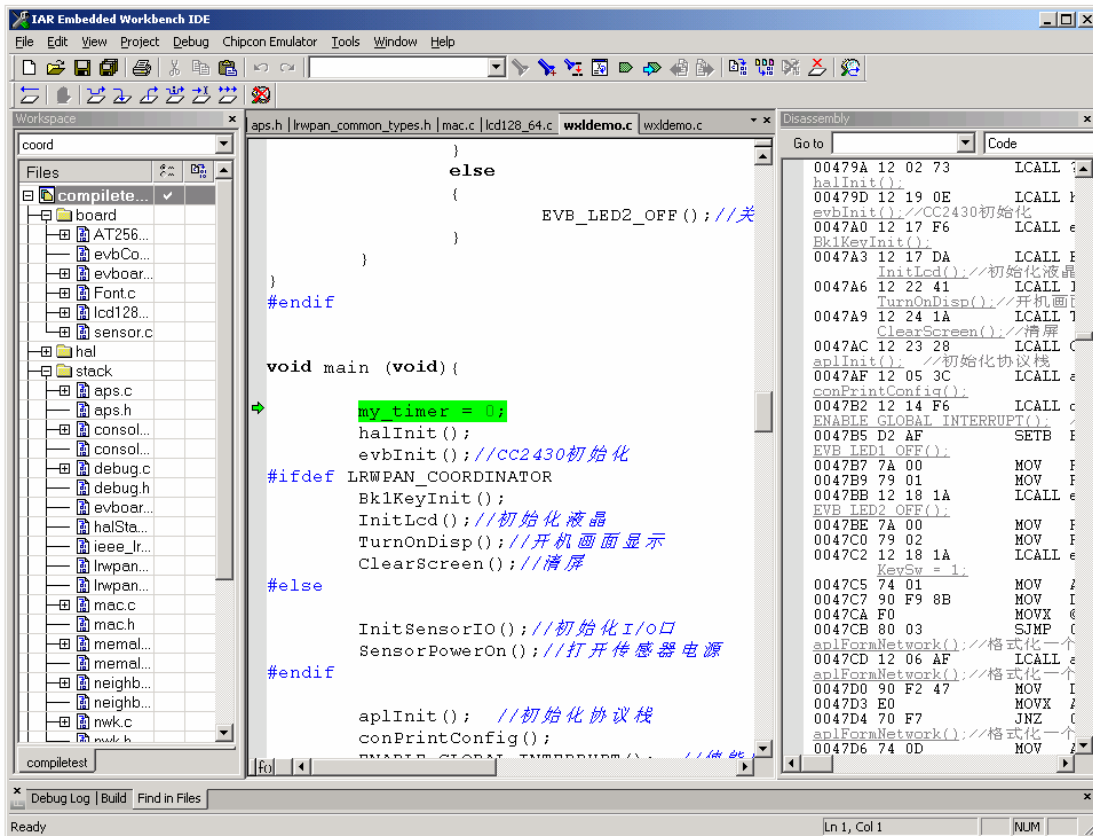


图 31 程序调试界面

调试窗口管理

在 IAR Embedded Workbench 中用户可以在特定的位置停靠窗口，并利用标签组来管理它们。用于也可以使某个窗口处于悬浮状态，即让它始终停靠在窗口的上层。状态栏位于主窗口底部，包含了如何管理窗口的帮助信息。更详细信息参见帮助文件中的 EW8051_UserGuide。

查看源文件语句:

Step Into 执行内部函数或子进程的调用

Step Over 每步执行一个函数调用

Next statement 每次执行一个语句

这些命令在工具栏上都有对应的快捷键。

调试管理

C-SPY 允许用户在源代码中查看变量或表达式，可在程序运行时跟踪其值的变化。选择菜单 View\Auto 开启窗口，如图 32 所示。自动窗口会显示当前被修改过的表达式。

Expression	Value	Location	Type
my_timer	0	XData:0xF97C	UINT32

图 32 自动窗口

连续观察 my_timer 的值的变化的情况。选择菜单 **View\Watch**，打开 Watch 窗口。单击 Watch 窗口中的虚线框，出现输入区域时键入 my_timer 并回车。也可以先选中一个变量将其从编辑窗口拖到 Watch 窗口，如图 33 所示。

Expression	Value	Location	Type
my_timer	0	XData:0xF97C	UINT32

图 33 Watch 窗口

单步执行，观察 my_timer 的变化。如果要在 Watch 窗口中去掉一个变量，先选中然后单击键盘上的 Delete 键或点右键删除。

设置并监控断点。使用断点最便捷的方式是将其设置为交互式的，即将插入点的位置指到一个语句里或靠近一个语句，然后选择 Toggle Breakpoint 命令。在 BK1KeyInit 语句出插入断点。在编辑窗口选择要插入断点的语句，选择菜单 Edit\Toggle Breakpoint。或者在工具栏上单击按钮，如图 34 所示。

```

void main (void) {
    my_timer = 0;
    halInit ();
    evbInit (); //CC2430初始化
#ifdef LRWPAN_COORDINATOR
    BklKeyInit ();
    InitLcd (); //初始化液晶
    TurnOnDisp (); //开机画面显示
    ClearScreen (); //清屏
#else
    InitSensorIO (); //初始化I/O口
    SensorPowerOn (); //打开传感器电源
#endif

    aplInit (); //初始化协议栈
    conPrintConfig ();
    ENABLE_GLOBAL_INTERRUPT (); //使能中断
    }
    
```

图 34 设置一个断点

这样在这个语句设置好一个断点，用高亮表示并且在左边标注一个红色的“X”显示有一个断点存在。可选择菜单 View\Breakpoint 打开断点窗口，观察工程所设置的断点。在主窗口下方的调试日志 Debug Log 窗口中可以查看断点的执行情况。如要取消断点，在原来断点的设置处再执行一次 Toggle Breakpoint 命令。

在反汇编模式中调试。在反汇编模式，每一步都对应一条汇编指令，用户可对底层进行完全控制。选择菜单 View\Disassembly ,打开反汇编调试窗口，用户可看到当前 C 语言语句对应的汇编语言指令，如图 35 所示。

```

Disassembly
Goto [ ] Code [ ]
main:
→ 00478D 90 F9 EA    MOV    DPTR,#0xF9EA
004790 78 08          MOV    R0,#0x08
004792 12 02 57      LCALL ?L_MOV_X
004795 90 F9 7C      MOV    DPTR,#0xF97C
004798 78 08          MOV    R0,#0x08
00479A 12 02 73      LCALL ?L_MOV_TO_X
halInit():
00479D 12 19 0E      LCALL halInit
evbInit();//CC2430初始化
0047A0 12 17 F6      LCALL evbInit
Bk1KeyInit():
× 0047A3 12 17 DA      LCALL Bk1KeyInit
  InitLcd();//初始化液晶
0047A6 12 22 41      LCALL InitLcd
  TurnOnDisp();//开机画面显示
0047A9 12 24 1A      LCALL TurnOnDisp
  ClearScreen();//清屏
0047AC 12 23 28      LCALL ClearScreen
aplInit(): //初始化协议栈
0047AF 12 05 3C      LCALL apsInit
conPrintConfig():
0047B2 12 14 F6      LCALL conPrintConfig
ENABLE GLOBAL INTERRUPT();//使能中断
0047B5 D2 AF      SETB  EA
EVB_LED1_OFF():
0047B7 7A 00          MOV    R2,#0x00
0047B9 79 01          MOV    R1,#0x01
0047BB 12 18 1A      LCALL evbLedSet
EVB_LED2_OFF():
0047BE 7A 00          MOV    R2,#0x00
0047C0 79 02          MOV    R1,#0x02
0047C2 12 18 1A      LCALL evbLedSet
    
```

图 35 汇编模式中调试程序

监控寄存器。寄存器窗口允许用户监控并修改寄存器的内容。选择菜单 View\Register，打开寄存器窗口，如图 36 所示。

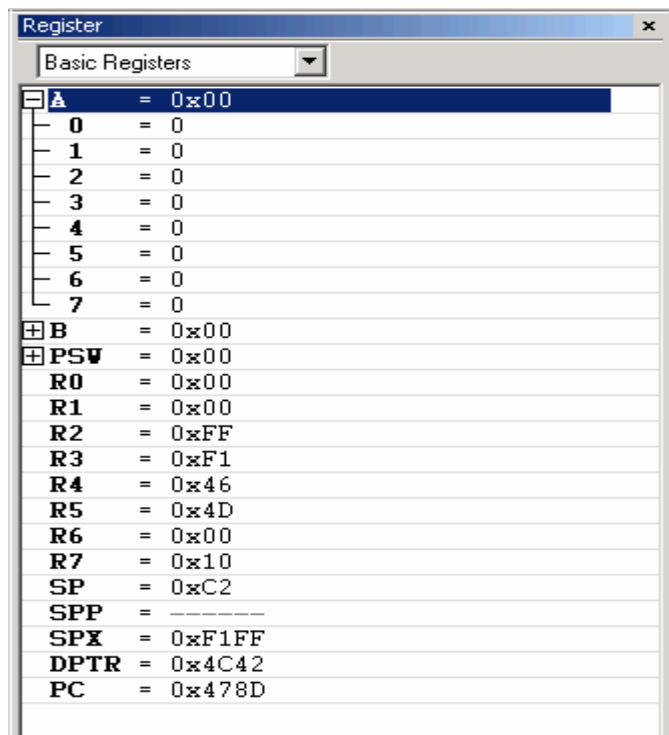


图 36 寄存器窗口

选择窗口上部的下拉列表，选择不同的寄存器分组。单步运行程序观察寄存器值的变化情况。

监控存储器。存储器窗口允许用户监控寄存器的指定区域。选择菜单 **View\Memory**，打开存储器窗口。打开 `my_timer` 所在文件，单击 `my_timer`，将它从源代码窗口拖到存储器窗口中。此时存储器窗口中对应的值也被选中，如图 37 所示。

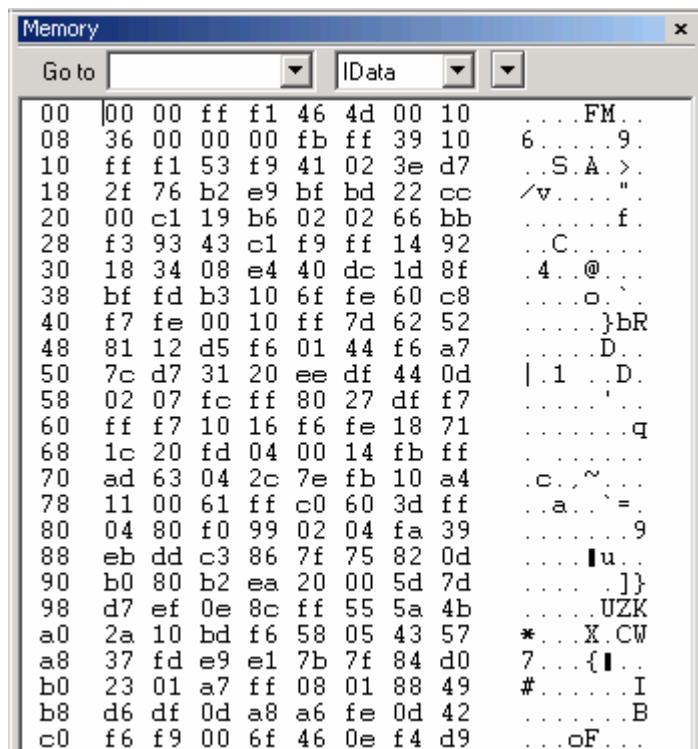




图 37 存储器窗口

单步执行程序，观察存储器中值的变化。用户可以在存储器窗口中对数据进行编辑，修改。在想进行编辑的存储器数值处放置插入点，键入期望值即可。

完整运行程序。选择菜单 **Debug\Go** 或点调试工具栏上  按钮，如果没有断点，程序将一直运行下去。可以看到在 ZigBee 的开发系统中相关的硬件反应。如果要停止，选择菜单 **Debug\Break** 或点调试工具栏上  按钮，停止程序运行。

退出调试。选择菜单 **Debug\Stop Debugging** 或单击调试工具栏上的按钮退出调试模式。

这里不是 IAR 开发环境的详细使用手册，关于 IAR 的详细说明文档请浏览 IAR 网站或安装文件夹下 `8051\doc` 里的支持文档。