► IAR EW430 总体介绍

瑞典 IAR System 公司推出的 IAR EW 软件是一种非常有效的嵌入式系统开发工具，它使用户能够充分有效地开发并管理嵌入式应用项目，其界面类似于 MS Visual C++,可以在 Windows 平台上运行，功能十分完善。包含有源程序文件编辑器，项目管理器，源程序调试器等，并且为 C/C++编译器，汇编器，连接定位器等提供了单一而灵活的开发环境。源级浏览器功能可以快速浏览源文件；还提供了对第三方工具软件的接口，允许启动用户指定的应用程序。

IAR EW 适用于开发基于 8 位，16 位以及 32 位的处理器的嵌入式系统，其具有同一界面，用户可以针对多种不同的目标处理器,在相同的集成开发环境中进行基于不同CPU 嵌入式系统应用程序的开发。另外IAR 的链接定位器(XLINK)可以输出多种格式的目标文件，使用户可以采用第三方软件进行仿真调试。

针对 TI MSP430，IAR 也有相应的 IAR EW430 软件。其具有上面所说的所有 IAR 软件共有功能，另外还有所有 MSP430 也包括 MSP430X 设备的配置文档，C-SPY 调试器支持 FET (TI's Flash Emulation Tool) 驱动，并支持实时操作系统相关信息的调试，还提供 MSP430 的项目例子以及相关的代码模板等。

► 现给以演示来具体说明其特性和使用：

♦ 1 在安装好的文件夹下面开打 IAR_KickStartCard 应用工程，右击鼠标选择 OPTIONS 会弹出如图 1 所示，在 CATEGORY 中选择 GENERAL OPTION 在右边点击 Target，从下面的 Device 右边的浏览器中可以看见 IAR EW430 所支持的所有常见的具体设备，在选择好具体的设备后 IAR 软件会自动的在后台调用相应的 I/O 头文件，以及设备描述文件(C-SPY 为了能对不同的器件的中断系统进行正确仿真，必须了解当前使用器件关于中断的详细信息，这类信息由设备描述文件.ddf 文件提供)。

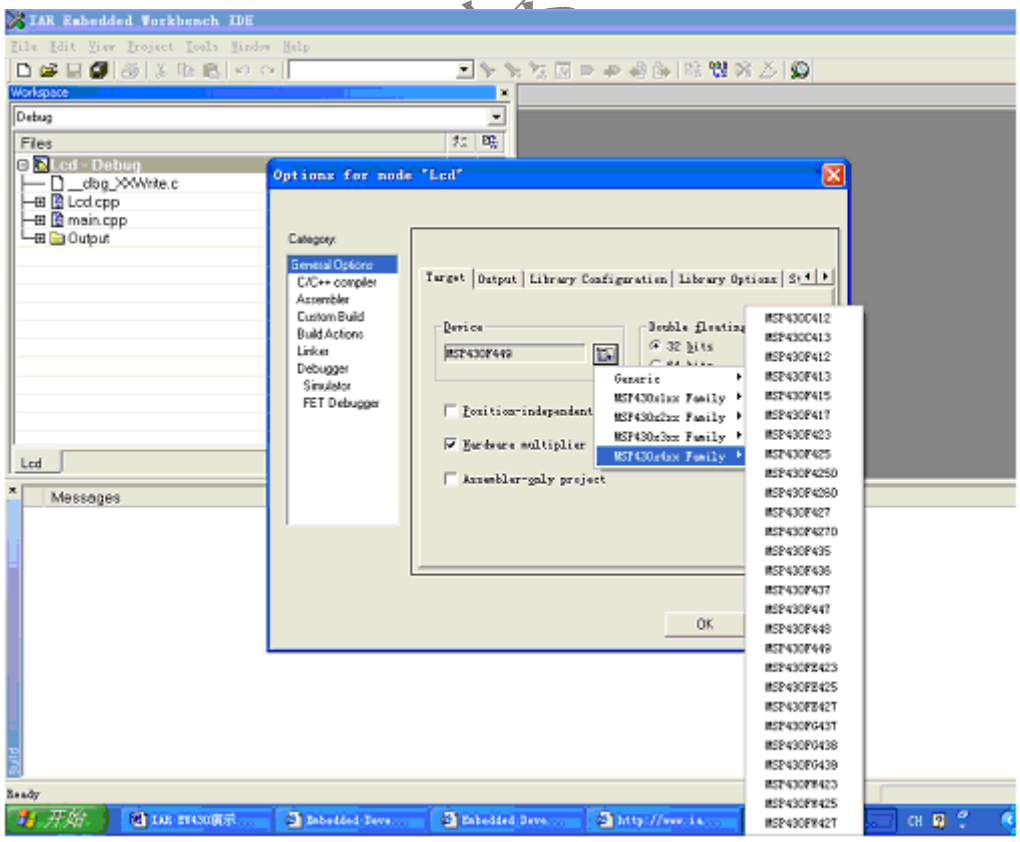


图 1

♦ 2 编译器：在 CATEGORY 中点击 C/C++ COMPILER，IAR 的编译器提供了 DLIB 库，支持符合 ANSI C 标准的 C/C++编程语言以及多字节参数和 MISRA 标准等。从图 2 中可见其的选择。(MISRA (The Motor Industry Software Reliability Association 汽车工业软件可靠性联会)所谓的 MISRA C Coding Standard，这一标准中包括了 127 条 C 语言编码标准，通常认为，如果能够完全遵守这些标准，则你的 C 代码是易读、可靠、可移植和易于维护的)

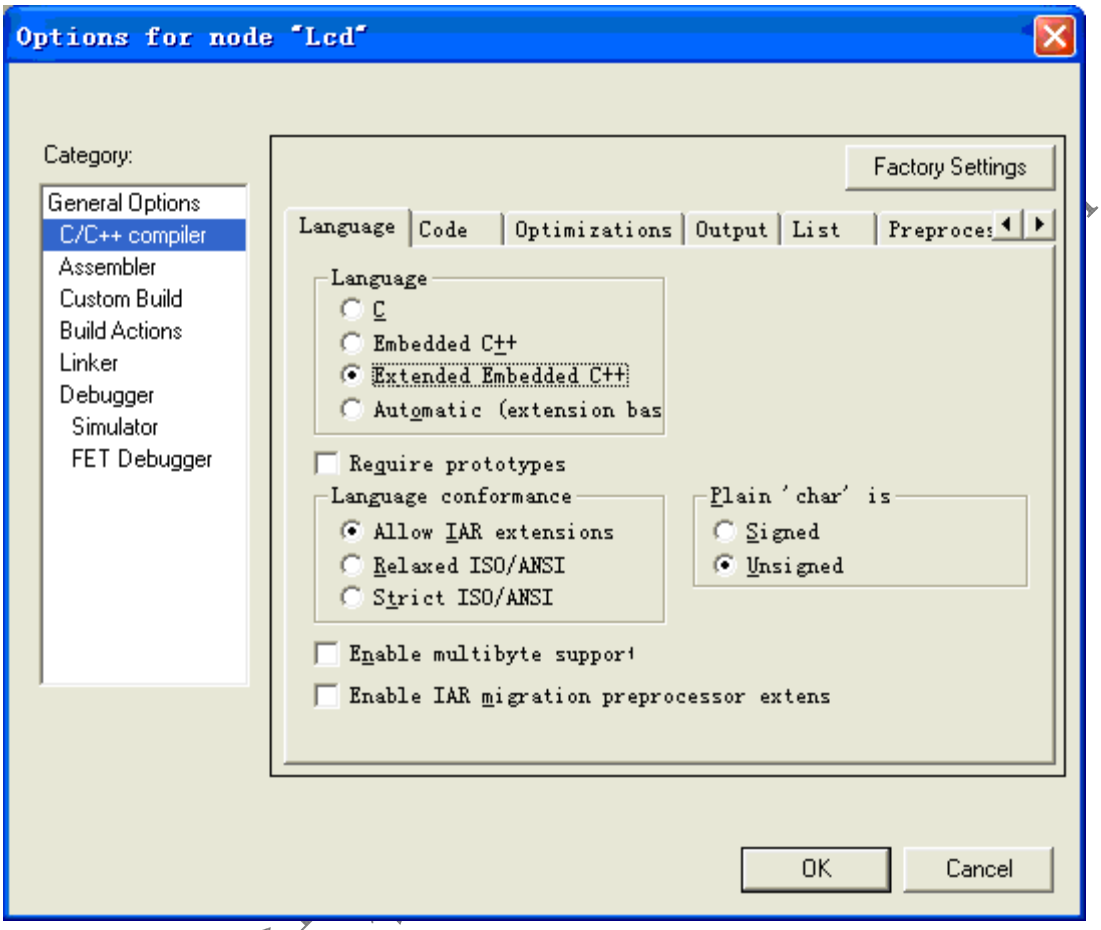


图 2

Language 选项区域用于设置希望采用的编程语言，如果您选择 Automatic 单选按钮，则根据源程序文件的扩展名自动选择。如扩展名为 .C 时作为 C 源程序进行编译。Enable multibyte support 允许在 C/C++源程序文件中使用多字节符号。图 3 中可以看见 MISRA C 选项卡，单击 ALL 后选择所有 MISRA C 规则校验模块当然也可以增加删除 MISAR C 规则校验模块，其作用就是按照 MISRA C 标准来检查校验您的代码。

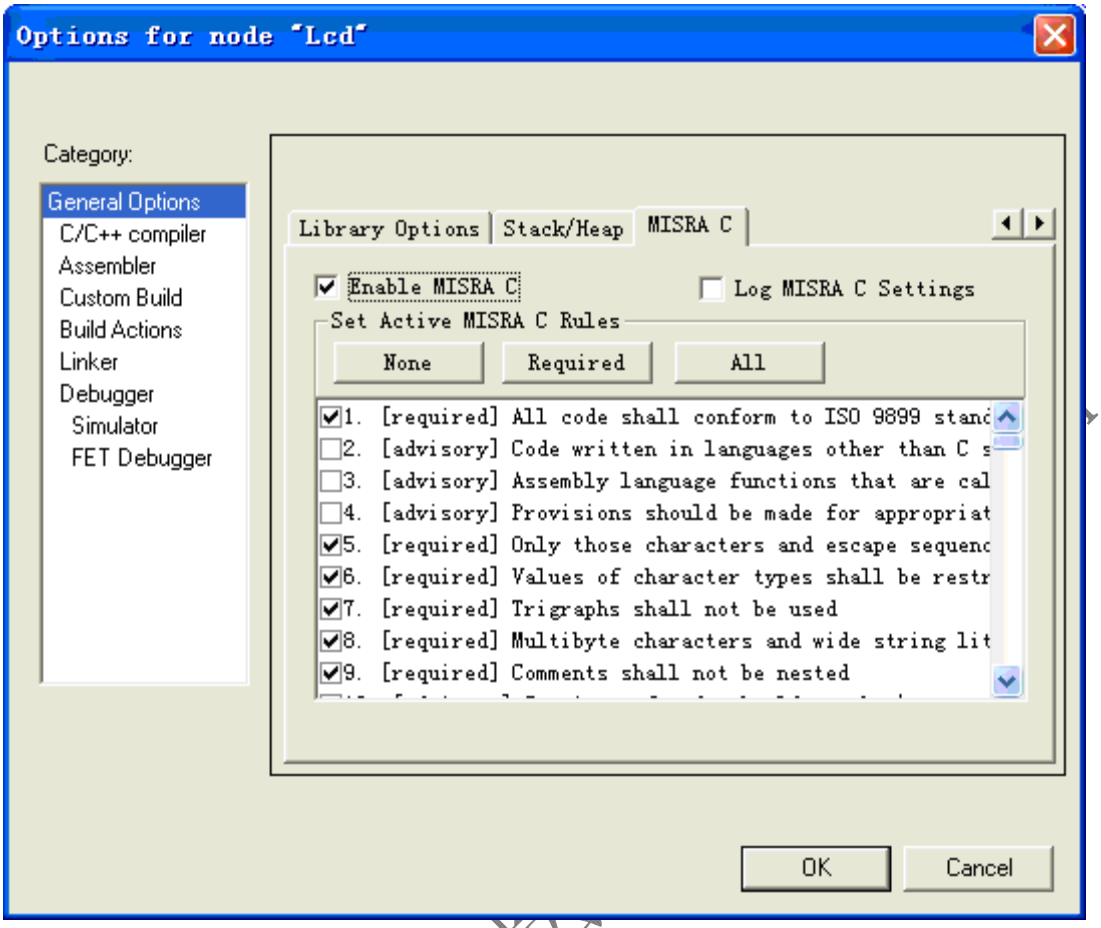
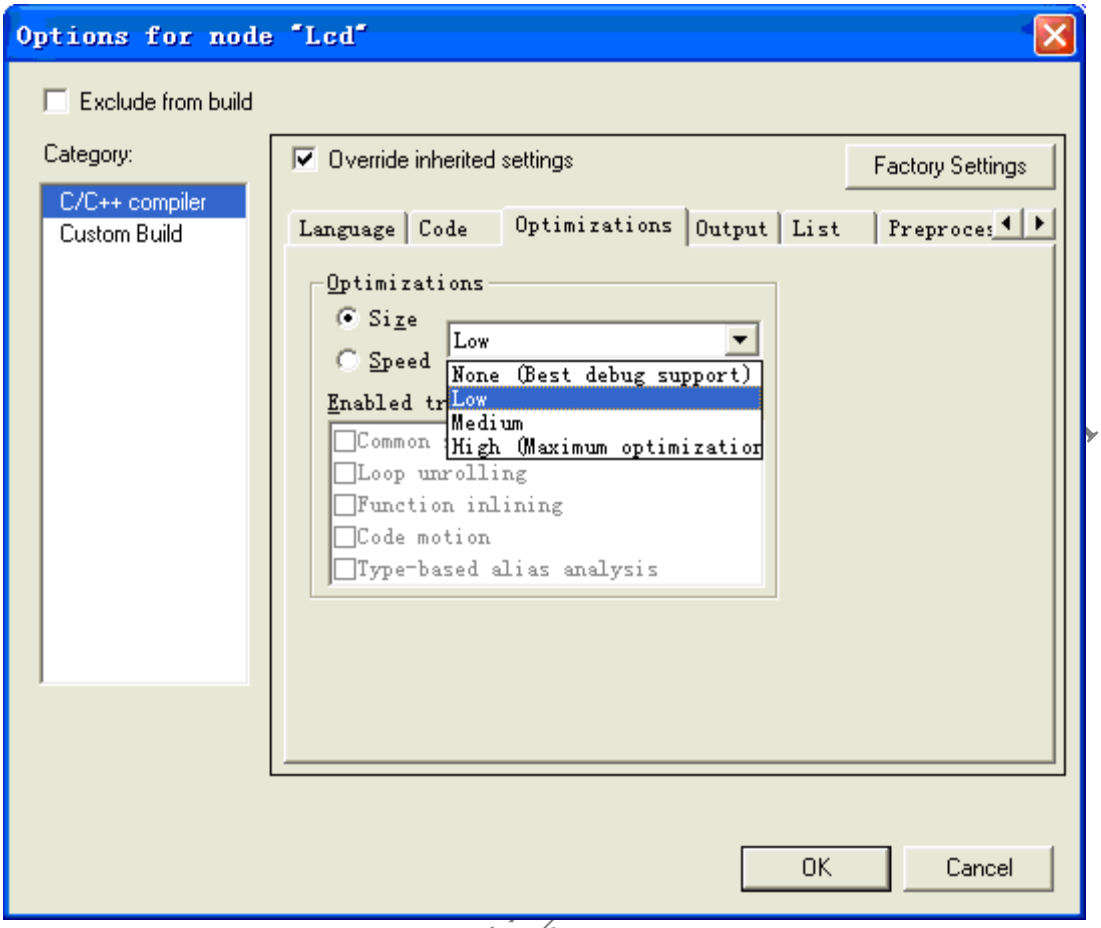


图 3

IAR 的编译器支持对代码大小和运行速度的多层次优化，在 Optimizations 选择区域可选择优化方法有 SIZE 和 SPEED 两种，前者以代码大小进行优化，后者以运行速度进行优化。另外还有 NONE 不优化对调试支持最好，LOW 低级优化，MEDIUM 中级优化和 HIGH 高级优化 4 种不同的优化级别。根据您所选择的优化方法和优化级别，Enabled transformations 框将自动选择不同的优化项目。另外，针对一个项目中不同的源文件也可以选择不同的具体优化代码方式。具体见图 4



IAR 提供了特殊性质的扩展关键字，可以直接在源程序中使用这些关键字，而不用用汇编语言写任何的函数而达到操作硬件设备命令。如 monitor 用于定义监视函数，其在执行期间禁止中断，从而允许完成操作等。除以上功能外 IAR 编译器还支持内嵌汇编语言，符合 IEEE 标准的 32 位和 64 位浮点运算等。

♦ 3 调试器：C-SPY 调试器完全集成在 IAR EW 软件中，通过不同驱动 DRIVER 实现与目标系统通信和仿真控制。MSP430 软件的 C-SPY 调试器提供了 2 种类型的驱动：纯软件仿真驱动和硬件仿真器驱动。纯软件仿真驱动可以实现在没有实际硬件的条件下，采用软件模拟方式进行用户程序的仿真也就是 C-SPY 提供的 Simulator 方式；硬件仿真器驱动为 C-SPY 调试器和专用硬件仿真器（例如 J-LINK）提供接口，实现对目标系统的实时在线仿真调试。在调试过程中可以编辑代码不用退出调试环境，并且可以在调试之前在代码中直接设置断点。选择 Project—Debug 后进入调试状态见下图 5，可以选择您感兴趣的窗口。例如打开 Register 窗口，还可以选择不同的寄存器见下图 6

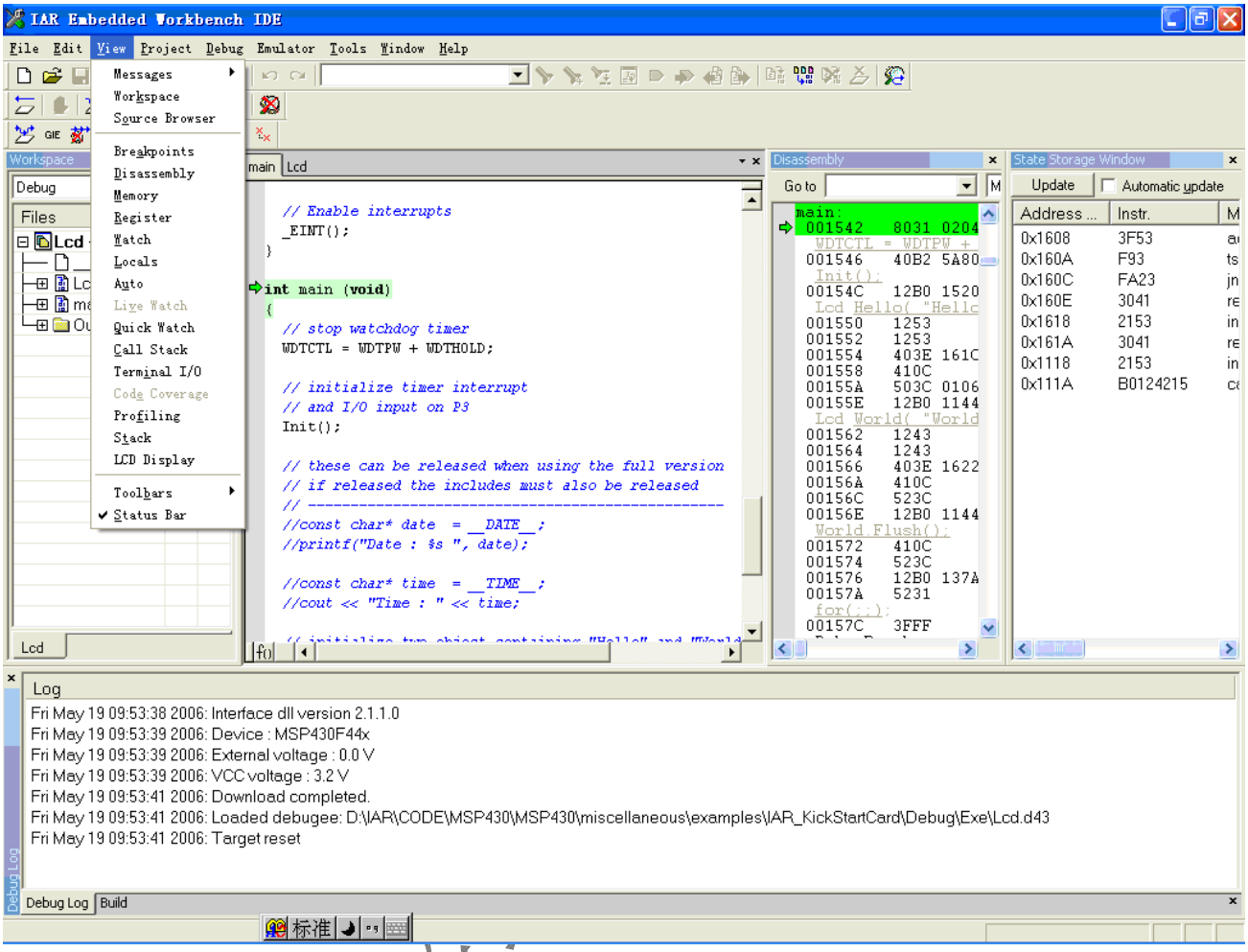

IAR Embedded Workbench IDE
File Edit View Project Debug Emulator Tools Window Help
Messages
Workspace
Source Browser
Breakpoints
Disassembly
Memory
Register
Watch
Locals
Auto
Live Watch
Quick Watch
Call Stack
Terminal I/O
Code Coverage
Profiling
Stack
LCD Display
Toolbars
Status Bar
Workspace
Debug
Files
Lcd
main Lcd
// Enable interrupts
_EINT();
}
int main (void)
{
// stop watchdog timer
WDTCTL = WDTPW + WDTHOLD;
// initialize timer interrupt
// and I/O input on P3
Init();
// these can be released when using the full version
// if released the includes must also be released
//const char* date = __DATE__;
//printf("Date : %s ", date);
//const char* time = __TIME__;
//cout << "Time : " << time;
Disassembly
Go to
main:
001542 8031 0204
WDTCTL = WDTPW +
001546 40B2 5A80
Init();
00154C 12B0 1520
Lcd_Hello("Hello
001550 1253
001552 1253
001554 403E 161C
001558 410C
00155A 503C 0106
00155E 12B0 1144
Lcd_World("World
001562 1243
001564 1243
001566 403E 1622
00156A 410C
00156C 523C
00156E 12B0 1144
World.Flush();
001572 410C
001574 523C
001576 12B0 137A
00157A 5231
for(;;);
00157C 3FFF
State Storage Window
Update
Automatic update
Address ... Instr.
0x1608 3F53
0x160A F93
0x160C FA23
0x160E 3041
0x1618 2153
0x161A 3041
0x1118 2153
0x111A B0124215
Log
Fri May 19 09:53:38 2006: Interface dll version 2.1.1.0
Fri May 19 09:53:39 2006: Device : MSP430F44x
Fri May 19 09:53:39 2006: External voltage : 0.0 V
Fri May 19 09:53:39 2006: VCC voltage : 3.2 V
Fri May 19 09:53:41 2006: Download completed.
Fri May 19 09:53:41 2006: Loaded debugee: D:\IAR\CODE\MSP430\MSP430\miscellaneous\examples\IAR_KickStartCard\Debug\Exe\Lcd.d43
Fri May 19 09:53:41 2006: Target reset
Debug Log
Build
标准

图 5

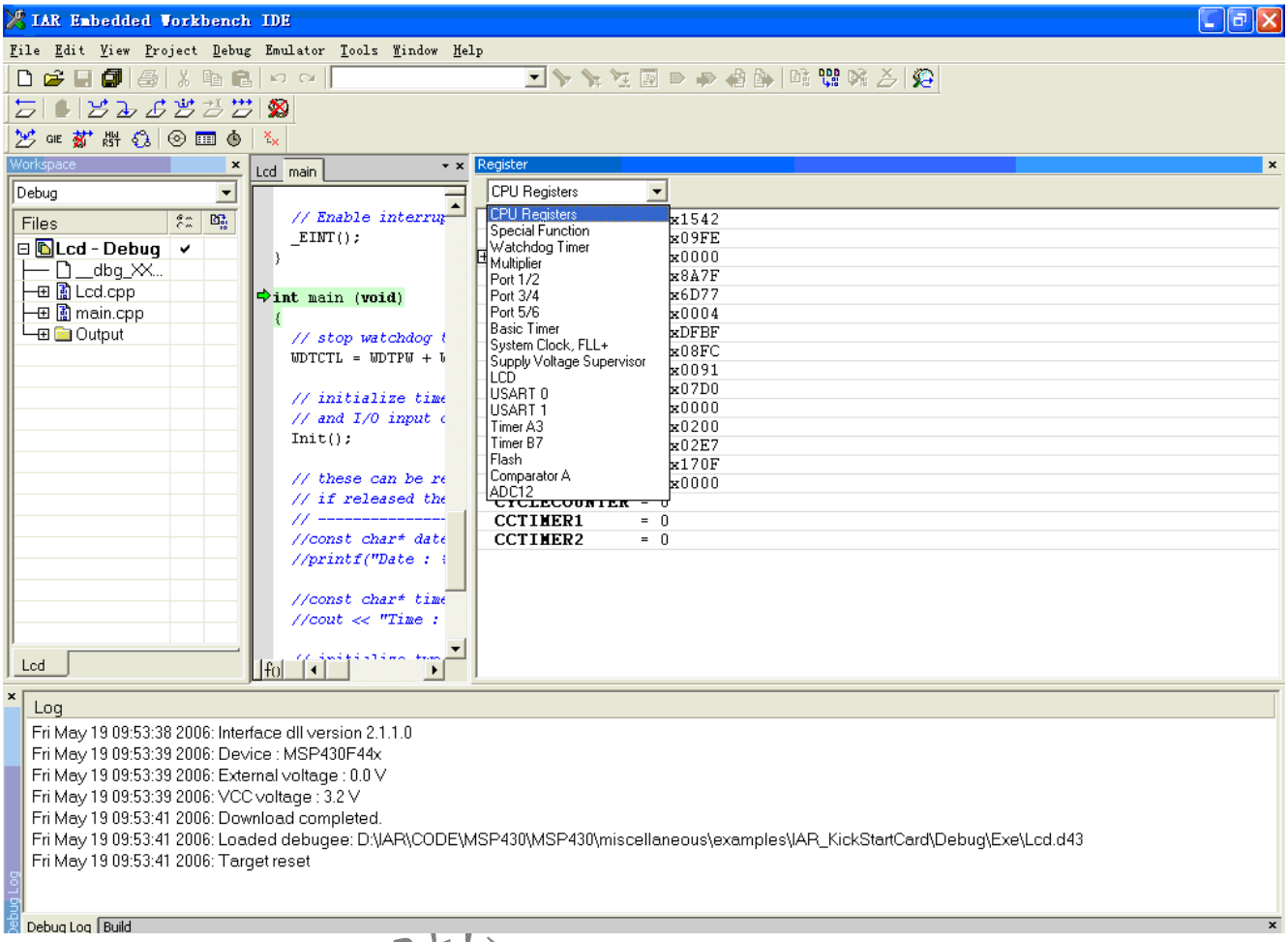


图 6

还可以看不同的存储器，通过 AUTO 看当前语句相关变量和表达式的值，LOCAL 查看当前运行函数的局部变量和函数参数值。如图 7 C-SPY 还提供了函数刨析功能和代码覆盖功能，用于对应用程序进行分析以确定其运行的瓶颈问题（注意 FET 仿真驱动没有代码覆盖功能），函数刨析功能可以找出程序运行过程中对一个给定激发信号耗时最长的函数，从而使用户能够集中精力研究如何更好地对这些函数进行优化。例如在编译时选择速度优化模式，或者将函数移到能更高效寻址的存储器中运行等。在 VIEW 中选择 PROFILING 开打刨析记录。启动程序全速运行，当运行到一个断点或程序结束时，窗口中将显示对当年程序所有函数运行的刨析记录结果，如图 8：

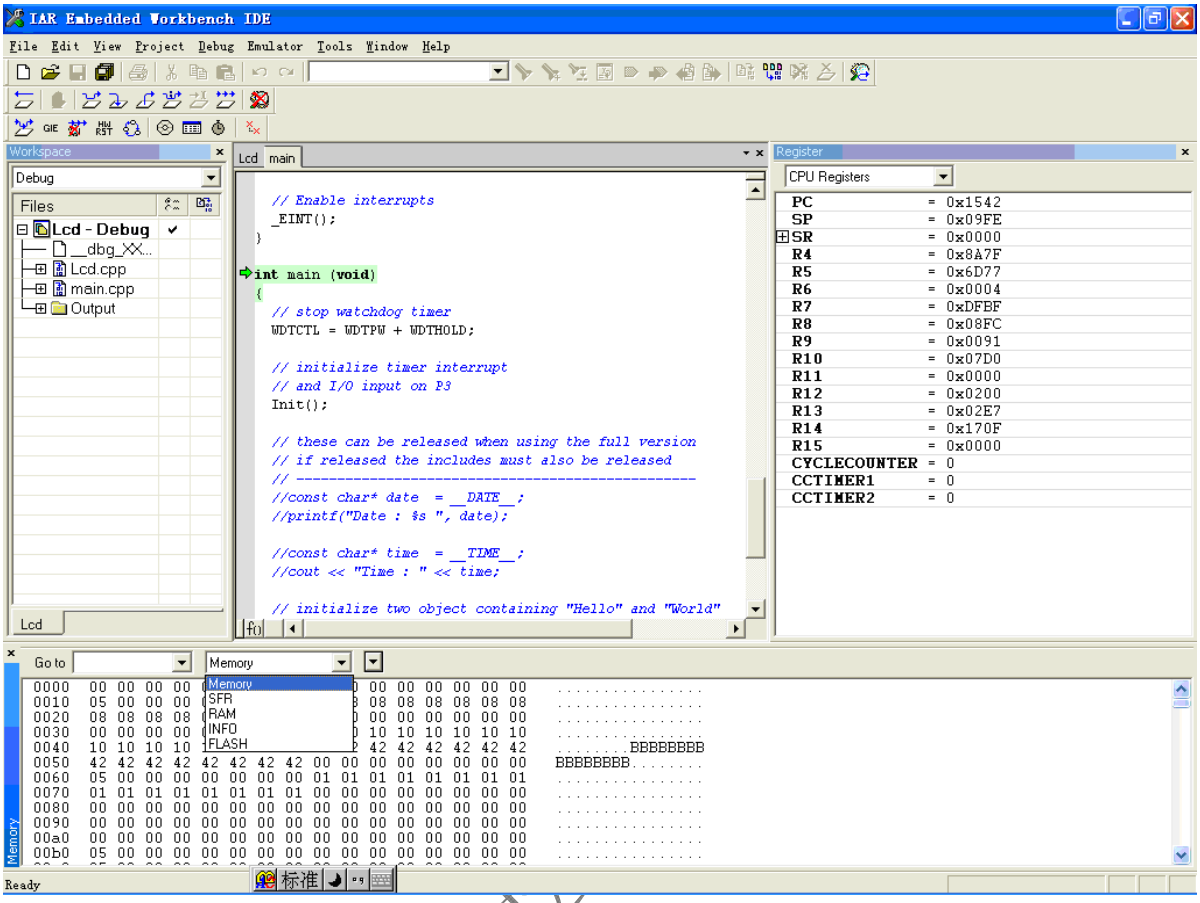

IAR Embedded Workbench IDE
File Edit View Project Debug Emulator Tools Window Help
Workspace
Debug
Files
Lcd - Debug
__dbg_XX...
Lcd.cpp
main.cpp
Output
Lcd
Lcd main
// Enable interrupts
_EINT();
}
int main (void)
{
// stop watchdog timer
WDTCTL = WDTPW + WDTHOLD;
// initialize timer interrupt
// and I/O input on P3
Init();
// these can be released when using the full version
// if released the includes must also be released
//const char* date = __DATE__;
//printf("Date : %s ", date);
//const char* time = __TIME__;
//cout << "Time : " << time;
// initialize two object containing "Hello" and "World"
Register
CPU Registers
PC = 0x1542
SP = 0x09FE
SR = 0x0000
R4 = 0x8A7F
R5 = 0x6D77
R6 = 0x0004
R7 = 0xDFBF
R8 = 0x08FC
R9 = 0x0091
R10 = 0x07D0
R11 = 0x0000
R12 = 0x0200
R13 = 0x02E7
R14 = 0x170F
R15 = 0x0000
CYCLECOUNTER = 0
CCTIMER1 = 0
CCTIMER2 = 0
Go to
Memory
Memory
SFR
RAM
INFO
FLASH
Ready

图 7

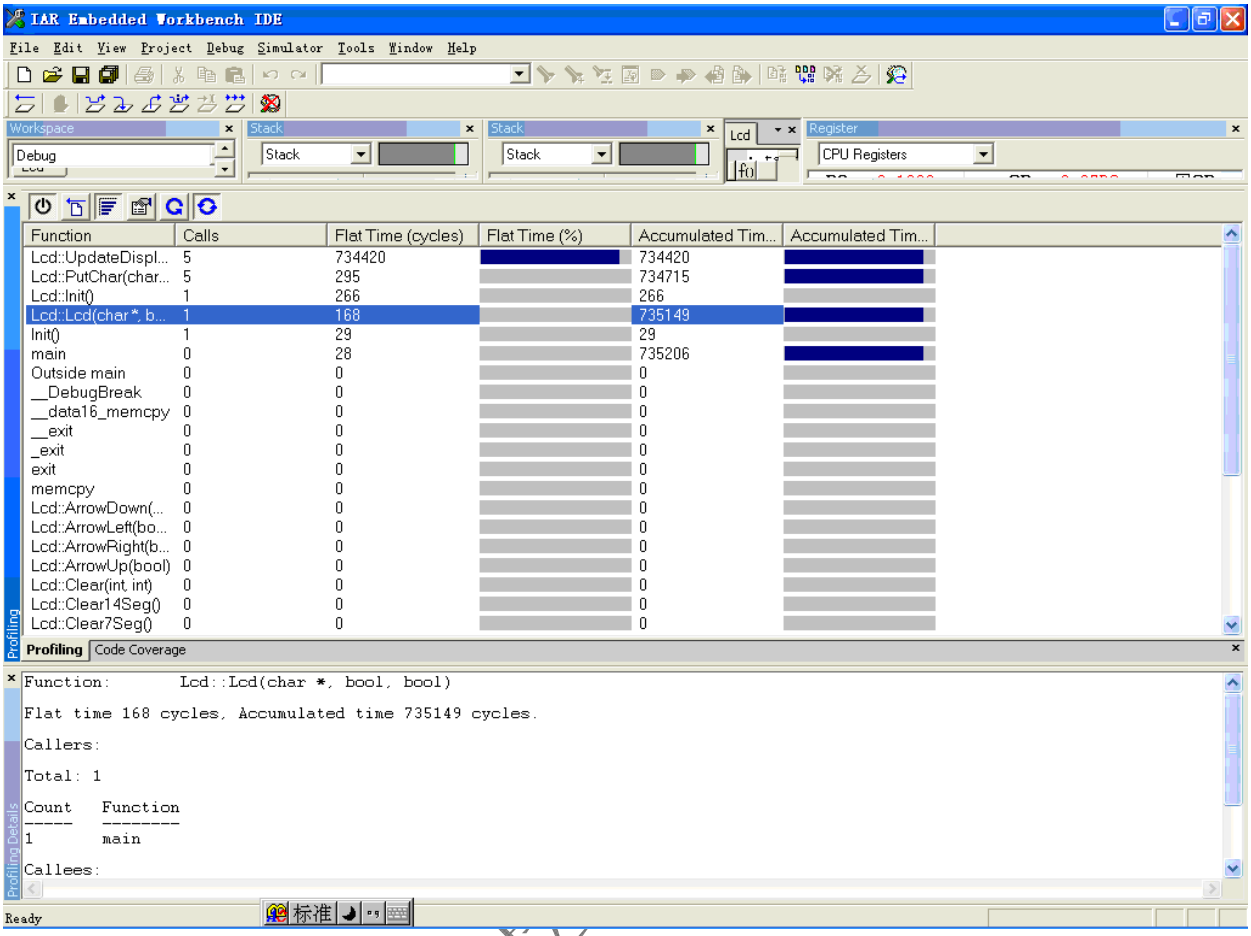


图 8

当一个函数由不含自身源代码的函数调用时，例如库函数，则没有耗时测试。当要查看与该函数有关的调用和被调用详细信息时，可以看创析细节窗口。

代码覆盖分析用于帮助用户确认是否所有程序代码都得到执行，这对于鉴别程序代码中是否有不能被执行的部分特别有用。打开 VIEW-CODE COVERAGE 窗口，如图 9 显示的是当前代码覆盖分析状态报告，即哪部分代码在分析开始后至少执行了一次。IAR C/C++编译器在应用程序每条语句以及每个函数调用处以 STEP POINT 步点形式生成详细的步进信息，以百分比统计了所有已经执行的布点数量，并列出了所有还没有执行的布点等。

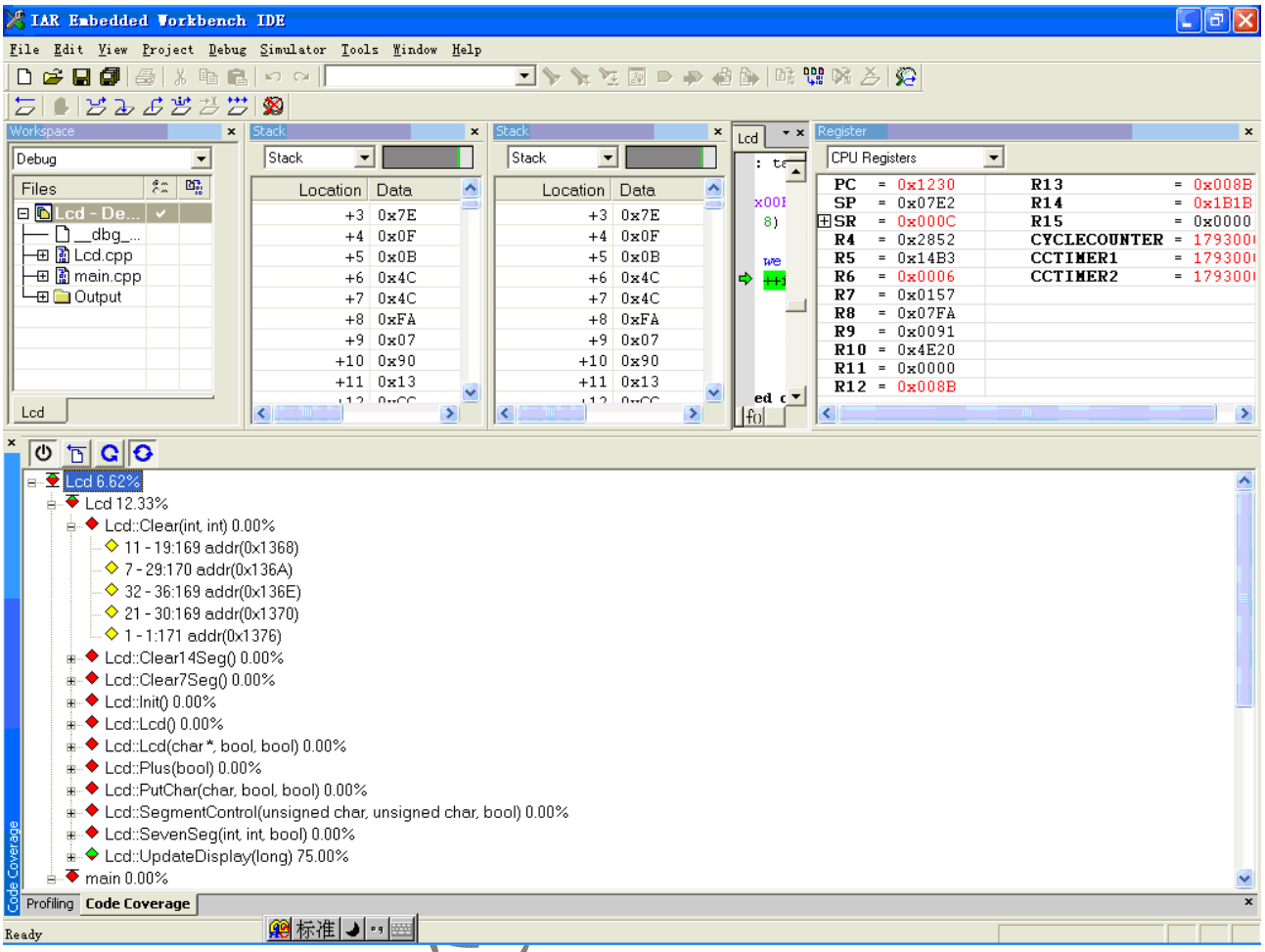
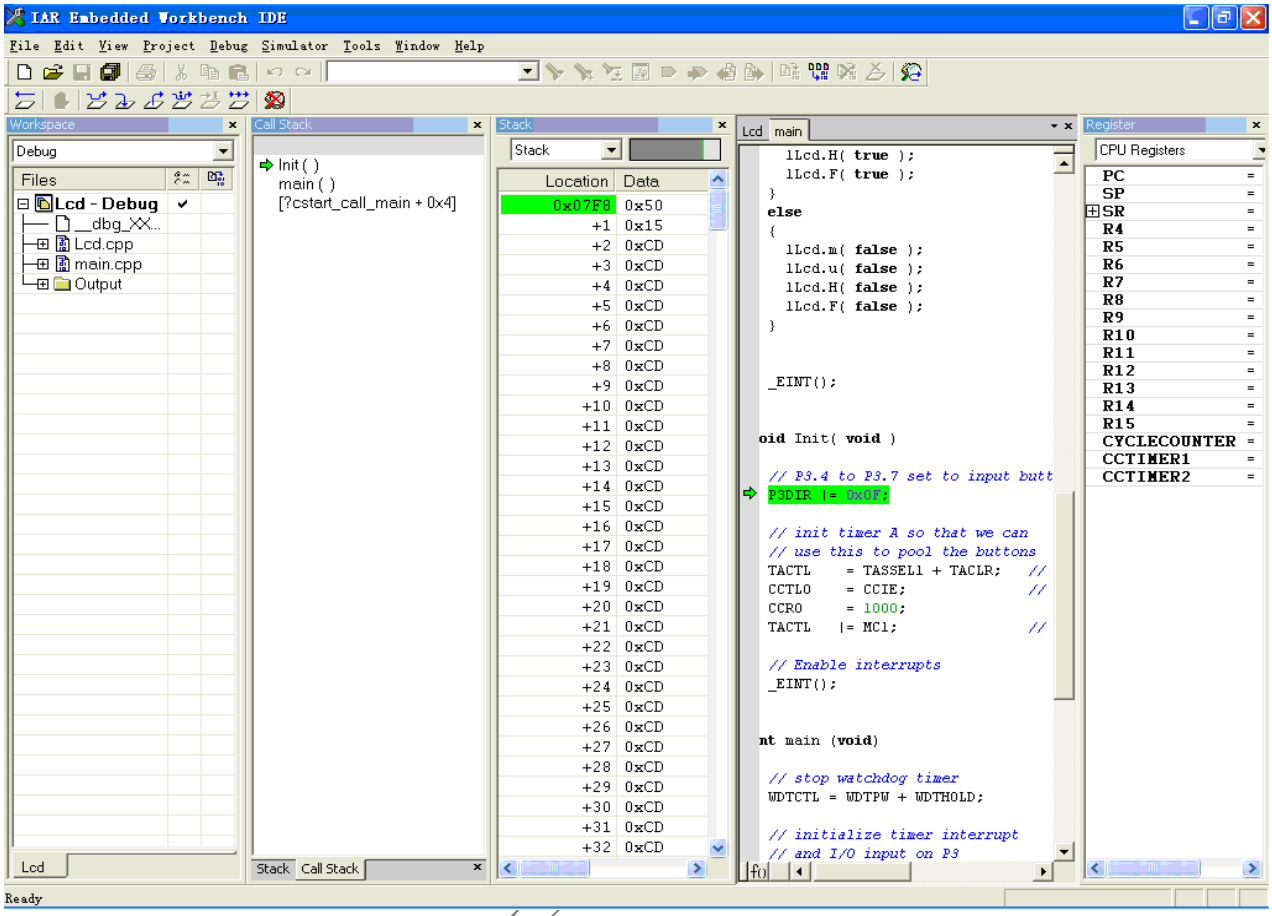


图 9

在程序调试时往往希望能对函数的调用过程进行跟踪，C-SPY 可以在调试过程中随时显示整个调用链，跟踪显示当前调用函数内容，以方便调试和修改源代码中的错误。在 VIEW-CALL STACK 打开调用堆栈窗口其中显示的是程序调用过程中函数调用的列表。并将当前函数置顶。见图 10



C-SPY 共有 4 种单步运行命令：Step Over,step Into,Step out 和 Next Statement 等。Step Into 是单步运行无论是否在同一个函数内，对于 Step Over 是在同一函数中将运行到下一布点。Step out 命令立即执行完整个函数调用并返回到调用话句的下一条语句。

另外 IAR 软件的输出格式也支持其他的调试器如下图您可以选择您用的调试器可读格式供 30 多种工业标准格式供选择如 UBROF，ELE/DWARF，COFF，Inter-extended,Motorola 等见下图：

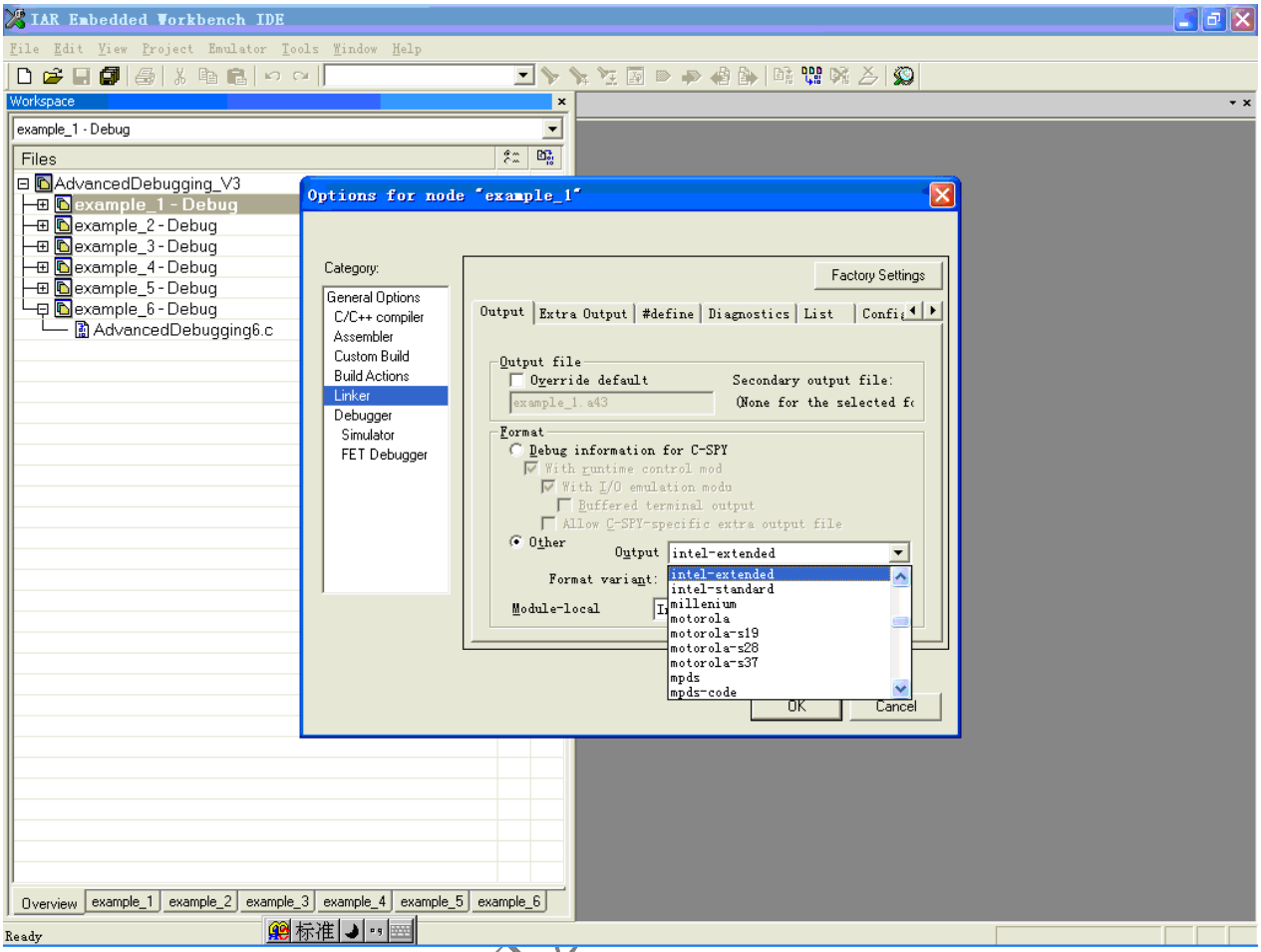

IAR Embedded Workbench IDE
File Edit View Project Emulator Tools Window Help
Workspace
example_1 - Debug
Files
AdvancedDebugging_V3
example_1 - Debug
example_2 - Debug
example_3 - Debug
example_4 - Debug
example_5 - Debug
example_6 - Debug
AdvancedDebugging6.c
Options for node "example_1"
Category:
General Options
C/C++ compiler
Assembler
Custom Build
Build Actions
Linker
Debugger
Simulator
FET Debugger
Factory Settings
Output
Extra Output
#define
Diagnostics
List
Output file
Override default
Secondary output file:
example_1.a43
(None for the selected f
Format
Debug information for C-SPY
With runtime control mod
With I/O emulation modu
Buffered terminal output
Allow C-SPY-specific extra output file
Other
Output
intel-extended
Format variant:
Module-local
intel-extended
intel-standard
millenium
motorola
motorola-s19
motorola-s28
motorola-s37
mpds
mpds-code
OK
Cancel
Overview
example_1
example_2
example_3
example_4
example_5
example_6
Ready

图 11