

# 龙芯 1x 嵌入式集成开发环境 1.0

## 使用说明书

苏州市天晟软件科技有限公司

[www.loongide.com](http://www.loongide.com)

2021 年 4 月

# 目 录

1、简介.....	5
1.1 主要特点.....	6
1.2 目录结构.....	6
1.3 文档约定.....	7
1.3.1 文件扩展名.....	7
1.3.2 指定文件名.....	7
1.3.3 头文件.....	7
1.4 项目开发过程.....	7
2、初次使用.....	8
2.1 语言设置.....	8
2.2 工作区目录.....	9
2.3 GNU 工具链.....	9
3、用户界面.....	12
3.1 菜单栏.....	12
3.2 工具栏.....	14
3.3 编辑面板.....	14
3.3.1 项目视图.....	15
3.3.2 代码解析.....	17
3.3.3 文本编辑器.....	18
3.3.4 消息窗口.....	19
3.4 调试面板.....	20
3.4.1 断点列表.....	21
3.4.2 CPU 寄存器.....	21
3.4.3 观察值.....	22
3.4.4 汇编代码.....	22
3.4.5 函数调用回溯.....	22
3.4.6 GDB 交互命令.....	22
3.5 状态栏.....	23
4、项目管理.....	24
4.1 新建项目向导.....	24
4.1.1 第一步 项目基本信息.....	24
4.1.2 第二步 设置Mcu、工具链和操作系统.....	25
4.1.3 第三步 实时操作系统选项.....	26
4.1.4 第四步 确认并完成向导.....	29
4.1.5 新建项目示例.....	30
4.2 基本操作.....	30
4.2.1 打开项目.....	30
4.2.2 保存项目.....	31
4.2.3 关闭项目.....	31

4.2.4	项目另存为 .....	31
4.2.5	成批添加文件 .....	32
4.2.6	成批移除文件 .....	33
4.3	项目属性 .....	34
5、	文档管理 .....	37
5.1	文件操作 .....	37
5.1.1	新建源代码文件 .....	37
5.1.2	新建头文件 .....	37
5.1.3	文件重命名 .....	38
5.1.4	文件移动 .....	38
5.1.5	文件删除 .....	39
5.2	文件夹操作 .....	39
5.2.1	新建文件夹 .....	39
5.2.2	重命名文件夹 .....	39
5.2.3	移动文件夹 .....	40
5.2.4	删除文件夹 .....	40
5.3	Drag & Drop .....	40
6、	文本编辑器 .....	41
6.1	编辑器选项 .....	41
6.1.1	常用 .....	41
6.1.2	字体 .....	42
6.1.3	颜色 .....	42
6.1.4	代码解析 .....	43
6.1.5	符号补全 .....	43
6.1.6	自动保存 .....	44
6.2	基本操作 .....	44
6.2.1	编辑 .....	44
6.2.2	查找 .....	44
6.2.3	替换 .....	45
6.2.4	在文件中查找 .....	46
6.3	其它操作 .....	47
6.3.1	打开头文件/文件夹 .....	47
6.3.2	定位语句定义原型 .....	47
6.3.3	代码解析项跳转 .....	48
6.4	插入代码向导 .....	49
6.4.1	插入RTOS 任务代码 .....	49
6.4.2	插入SPI/I2C驱动代码 .....	50
6.5	信息提示 .....	52
7、	项目编译 .....	53
7.1	编译选项 .....	53
7.1.1	MIPS & BSP Options .....	54
7.1.2	GNU C Compiler - C 编译器 .....	54
7.1.3	GNU Assembler - 汇编语言编译器 .....	58
7.1.4	GNU C++ Compiler - C++ 编译器 .....	58

7.1.5 GNU C Linker - C 链接器 .....	58
7.1.6 软浮点算术库 .....	60
7.2 开始编译 .....	61
7.2.1 编译成功 .....	61
7.2.2 编译失败 .....	62
7.3 项目清理 .....	62
8、项目调试 .....	63
8.1 调试选项 .....	63
8.1.1 主要项 .....	63
8.1.2 调试器 .....	64
8.1.3 启动项 .....	65
8.1.4 源代码 .....	66
8.2 调试断点 .....	66
8.2.1 在编辑器中设置断点 .....	66
8.2.2 断点列表 .....	67
8.3 开始调试 .....	67
8.3.1 代码下载 .....	67
8.3.2 单步运行 .....	68
8.3.3 连续运行 .....	69
8.3.4 停止调试 .....	69
8.3.5 观察值 .....	69
8.3.6 函数调用回溯 .....	70
9、实用工具 .....	71
9.1 NOR Flash 编程 .....	71
9.2 NAND Flash 编程 .....	72
9.3 硬件设计助手 .....	73
9.3.1 龙芯 1B 芯片 .....	74
9.3.2 龙芯 1C 芯片 .....	75
10、系统安装 .....	76
10.1 运行环境 .....	76
10.1.1 安装MSYS 1.0 .....	76
10.1.2 安装MSYS2 .....	76
10.2 安装LoongIDE .....	76
10.2.1 运行安装向导 .....	77
10.2.2 LxLink驱动 .....	77
10.3 GNU 工具链 .....	78
10.3.1 SDE Lite for MIPS工具链 .....	78
10.3.2 RTEMS GCC for MIPS工具链 .....	78
10.4 注意事项 .....	78

## 1、简介

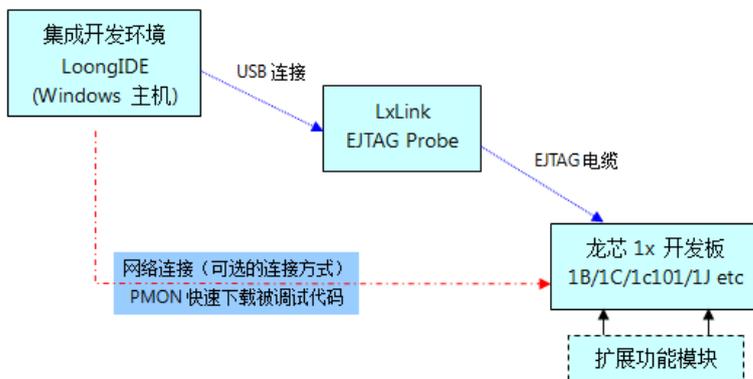
### 缩写

LS1x DTK: Embedded DTK for LS1x 龙芯 1x 嵌入式开发工具

LoongIDE: Embedded IDE for LS1x 龙芯 1x 嵌入式集成开发环境

### LS1x DTK

用于龙芯 1x 芯片的嵌入式开发解决方案，包括创建、构建以及调试用户应用项目，完美支持龙芯 1x 芯片的工业级应用的开发。



### LoongIDE

LoongIDE 是 LS1x DTK 的人机交互界面。

- 实现龙芯 1x 项目的 C/C++ 和 MIPS 汇编程序的编辑、编译和调试，软件功能强大、简单易用；
- 包含基于实时操作系统 RTEMS 的龙芯 1x BSP 包，可以让用户快速部署工业级应用的项目开发；
- 包含基于 RTThread/FreeRTOS/uCOS/裸机编程的龙芯 1x 应用程序框架，方便用户选择适用的 RTOS。

### LxLink

用于芯片级调试的 EJTAG 接口设备，通过 USB 连接主机。LxLink 由上位机软件驱动运行，实现两大功能：

- 标准 EJTAG 接口，实现龙芯 1x 的在线调试；
- 实现 SPI 硬件接口，支持 NOR Flash 芯片的编程。

### 开发板

基于龙芯 1x 芯片设计，方便用户模拟和实现各种自动化、工业控制、物联网等应用场景。

本文档介绍 LoongIDE 的使用。

## 1.1 主要特点

- Windows 环境下安装，无需配置，直接使用；
- 支持英、汉双语版本；
- 以项目为单位进行源代码管理，支持 C/C++ 的项目开发；
- 功能强大的 C/C++ 代码编辑器，支持代码折叠、高亮语法、宏注释灰色显示等功能；
- 实时代码解析引擎，实现光标处头文件、类、变量、函数等定义原型的快速定位；
- 集成 LS1x 的 RTEMS BSP 包，提供 LS1x 片上设备的驱动程序，支持 posix1003.1b 软件标准；
- 支持基于 RTThread/FreeRTOS/uCOS/裸机项目的开发，自动生成项目框架代码和提供部分 libc 库函数支持；
- 使用优化的 RTEMS GCC for MIPS 工具链，支持 -mips32/-mips32r2 和 -mhard-float 等编译选项；
- 基于 EJTAG 设备的 C、C++ 和 MIPS 汇编语言的源代码级的单步调试；
- 实现基于 PMON TCP/IP 快速下载待调试项目到目标板；
- 提供多种 Flash 的编程方式，方便用户部署项目应用；
- 提供基于 LGPL 协议的 yaffs2、lua、modbus、sqlite、agg 等第三方软件包；
- 支持 16 点阵宋体一级字库显示。

## 1.2 目录结构

系统安装目录下：

Help	帮助文档
Lang	语言
mips-2011.03	SDE Lite for MIPS 编译器
rtems-4.10	RTEMS for MIPS 编译器
bin	
include	
info	
lib	
libexec	
make	
man	
mips-rtems4.10	
bin	
include	
lib	
ls1b_core	龙芯1B RTEMS BSP包
ls1c_core	龙芯1C RTEMS BSP包
share	
Template	项目模板文件
gdbproxy.dll	动态链接库
LoongIDE.exe	LoongIDE 主程序
LoongIDE.ini	LoongIDE 配置文件

## 1.3 文档约定

### 1.3.1 文件扩展名

- .lxp 项目文件；
- .layout 项目配置文件；
- .S 大写，MIPS 汇编语言源文件；
- .c/.cpp 项目源文件；
- .h/.hpp 项目头文件；
- .inl 用于#include 的源文件。

### 1.3.2 指定文件名

用于裸机编程项目，包括 RTThread/FreeRTOS/uCOS 的项目：

- start.S 启动文件，内部包含应用程序 start 入口，不可改名；gcc 链接时链接的第一个文件；
- ld.script/ linkcmds 链接脚本文件；生成 makefile.mk 时先查找 ld.script，再查找 linkcmds。

### 1.3.3 头文件

LoongIDE 对 c/c++ 源文件进行预处理，约定 ""、<>的使用：

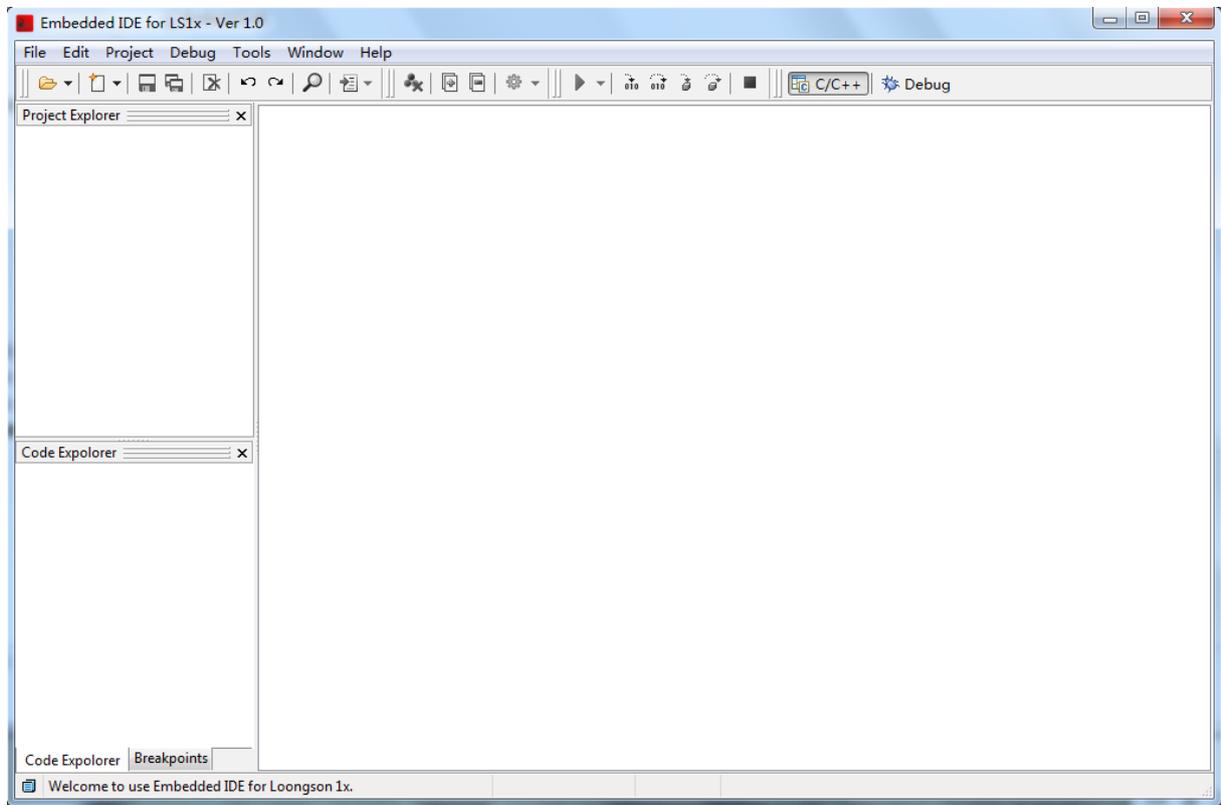
- #include "xxx"：查找顺序为当前目录、本地搜索路径、系统搜索路径；
- #include <xxx>：查找顺序为系统搜索路径、当前目录、本地搜索路径。

## 1.4 项目开发过程

1. 使用“新建项目向导”创建用户项目框架；
2. 在新建项目中添加源代码等文件，实现项目业务逻辑；
3. 编译、修正代码语法错误；
4. 调试、修正代码逻辑错误；
5. 编译、调试正确通过后，使用“Nand Flash 编程”工具部署应用。

## 2、初次使用

运行 LoongIDE.exe，显示界面如下：

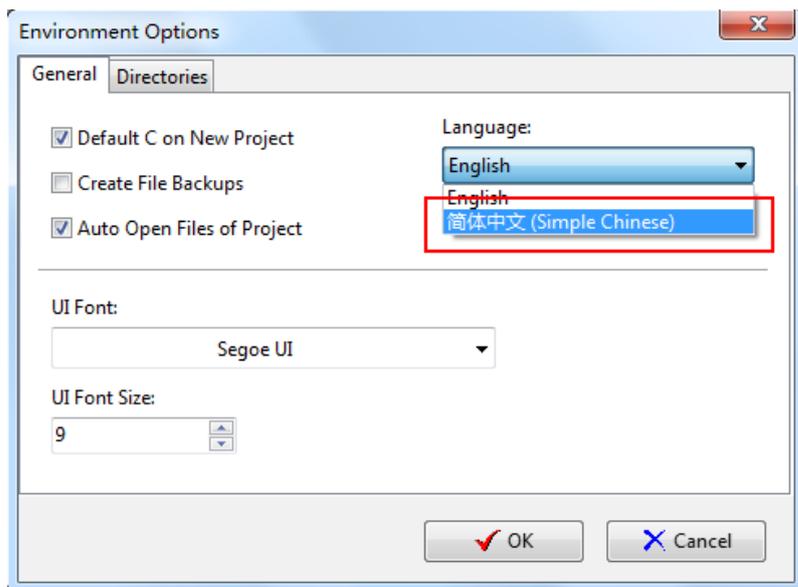


在使用 LoongIDE 前，进行初始化设置：

- 语言设置
- 工作区目录设置
- GNU 工具链设置

### 2.1 语言设置

打开菜单“Tools → Enviroment Options”，进入环境变量设置窗口如下，选择合适的语言环境。

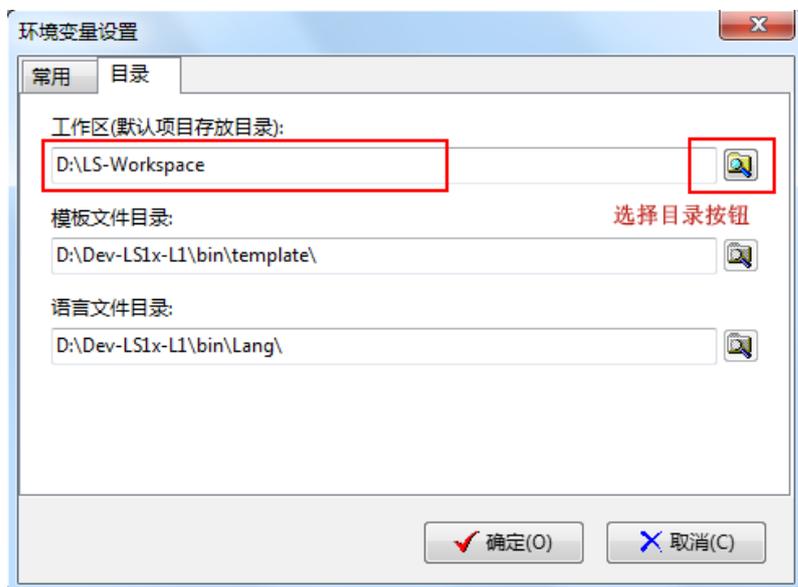


选择“简体中文”，按【OK】退出，这时主窗口的标题、菜单等文本以中文显示。

说明：本文档以“中文”环境进行介绍。

## 2.2 工作区目录

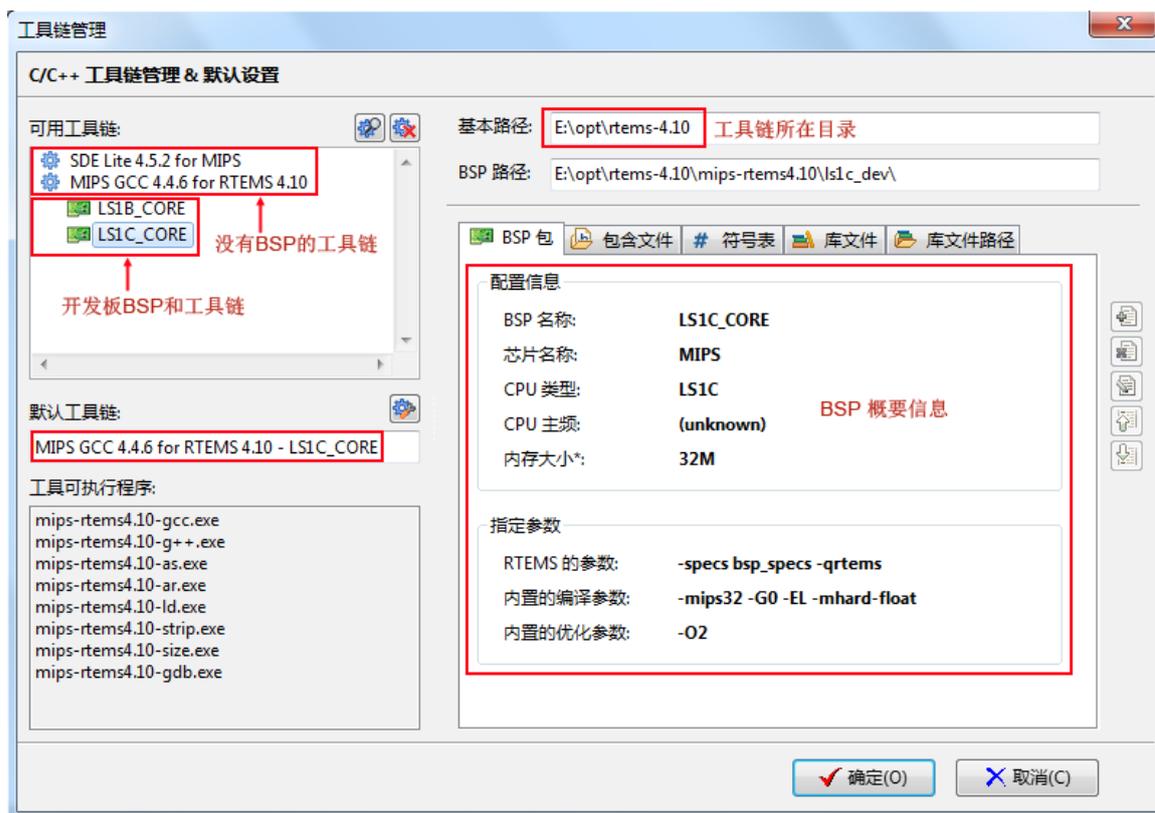
打开菜单“工具 → 环境参数设置”，选择“目录”页：



从文件系统选择工作区目录，作为新建项目的默认存放目录。

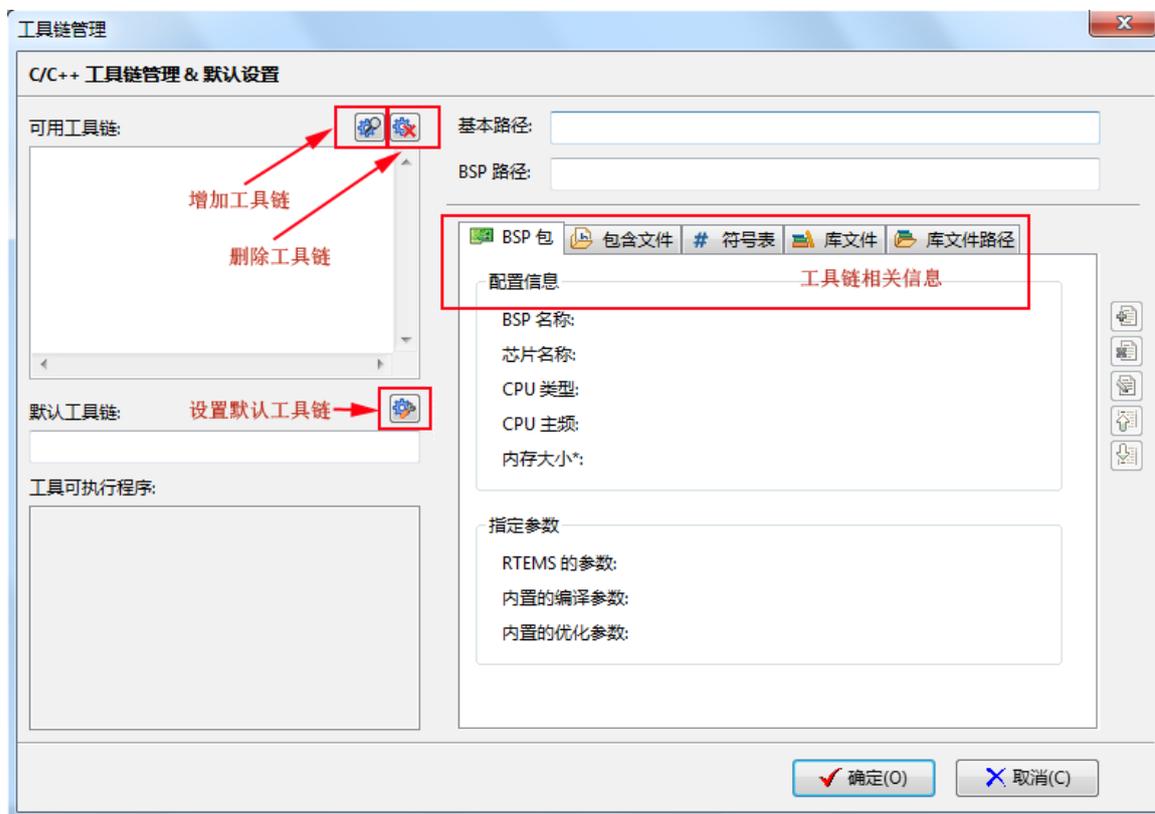
## 2.3 GNU 工具链

打开菜单“工具 → C/C++工具链管理”，显示窗口如下：



系统安装目录下的工具链，在 LoongIDE 初次运行时自动加载。

如果系统安装时没有选择安装工具链或者工具链已安装在其它目录，单击【增加工具链】按钮，选择工具链所在目录，系统将查找并加载工具链和 BSP 包。



从“可用工具链”中选中的一个，单击【设置默认工具链】，在“默认工具链”文本框显示。

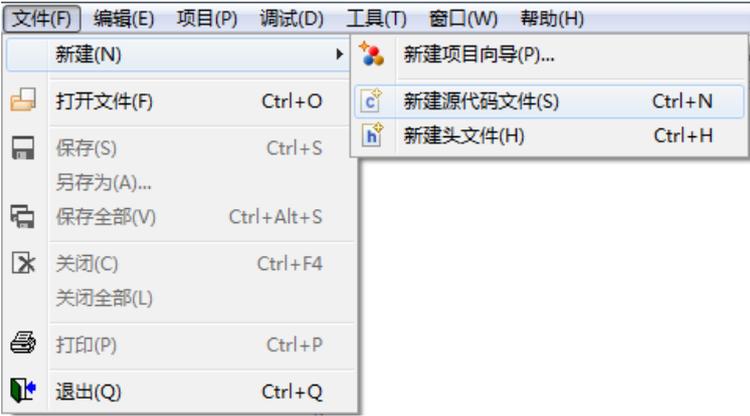
窗口右侧页框，可以查看“工具链”和“BSP 包”的基本信息，其中“符号表”在编程时可引用为 C/C++/ASM 的宏定义。

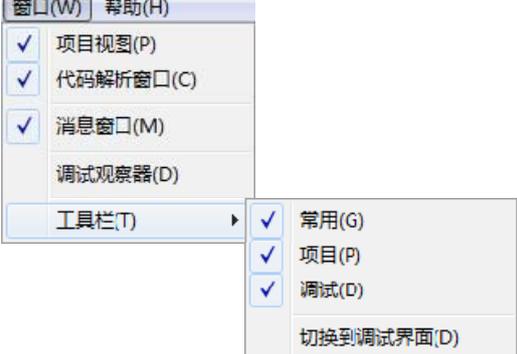
### 3、用户界面

LoongIDE 用户界面有菜单栏、快速工具栏、操作界面和状态栏四部分构成。其中操作界面分为“编辑”和“调试”两种显示形式，分别用于源代码编辑和源代码调试。在单击工具栏“开始调试”按钮  后自动切换到调试界面；在单击工具栏“停止调试”按钮  后自动回到编辑界面；或者使用工具栏按钮   实现“编辑”和“调试”界面的快速切换。

#### 3.1 菜单栏

菜单栏集中了用户操作的主要功能，各菜单项如下：

文件	 <p>文件(F) 编辑(E) 项目(P) 调试(D) 工具(T) 窗口(W) 帮助(H)</p> <ul style="list-style-type: none"> <li>新建(N) <ul style="list-style-type: none"> <li>新建项目向导(P)...</li> <li>新建源代码文件(S) Ctrl+N</li> <li>新建头文件(H) Ctrl+H</li> </ul> </li> <li>打开文件(F) Ctrl+O</li> <li>保存(S) Ctrl+S</li> <li>另存为(A)...</li> <li>保存全部(V) Ctrl+Alt+S</li> <li>关闭(C) Ctrl+F4</li> <li>关闭全部(L)</li> <li>打印(P) Ctrl+P</li> <li>退出(Q) Ctrl+Q</li> </ul>
编辑	 <p>编辑(E) 项目(P) 调试(D) 工</p> <ul style="list-style-type: none"> <li>撤销(U) Ctrl+Z</li> <li>重做(R) Ctrl+Y</li> <li>剪辑(T) Ctrl+X</li> <li>复制(C) Ctrl+C</li> <li>粘贴(P) Ctrl+V</li> <li>全选(A) Ctrl+A</li> <li>取消选择(D)</li> <li>查找(F) Ctrl+F</li> <li>替换(L) Ctrl+R</li> <li>继续查找(G) F3</li> <li>在文件中查找(I)</li> <li>字符编码转换(E)</li> <li>插入代码向导(I)...</li> </ul>

项目	 <p>项目(P) 调试(D) 工具(T) 窗口(W)</p> <ul style="list-style-type: none"> <li>打开项目(P)</li> <li>另存项目为(A)...</li> <li>关闭项目(C)</li> <li>添加文件到项目(F)...</li> <li>从项目删除文件(V)</li> <li>编译选项(O) F2</li> <li>编译(B)... Ctrl+F9</li> <li>清理(N)</li> <li>项目属性(J)</li> </ul>
调试	 <p>调试(D) 工具(T) 窗口(W) 帮</p> <ul style="list-style-type: none"> <li>启动(S) F9</li> <li>单步进入(R) F5</li> <li>跳过(T) F6</li> <li>单步进入指令(I) F7</li> <li>跳过指令(O) F8</li> <li>继续(C)</li> <li>停止(P) F10</li> <li>增加监视变量(W)</li> <li>调试选项(B) F4</li> </ul>
工具	 <p>工具(T) 窗口(W) 帮助(H)</p> <ul style="list-style-type: none"> <li>GNU C/C++ 工具链(T) F11</li> <li>编辑器选项(E)</li> <li>MCU 硬件设计助手...</li> <li>NOR Flash 编程</li> <li>NAND Flash 编程</li> <li>环境参数设置(P)</li> </ul>
窗口	 <p>窗口(W) 帮助(H)</p> <ul style="list-style-type: none"> <li><input checked="" type="checkbox"/> 项目视图(P)</li> <li><input checked="" type="checkbox"/> 代码解析窗口(C)</li> <li><input checked="" type="checkbox"/> 消息窗口(M)</li> <li>调试观察器(D)</li> <li>工具栏(T) <ul style="list-style-type: none"> <li><input checked="" type="checkbox"/> 常用(G)</li> <li><input checked="" type="checkbox"/> 项目(P)</li> <li><input checked="" type="checkbox"/> 调试(D)</li> <li>切换到调试界面(D)</li> </ul> </li> </ul>
帮助	 <p>帮助(H)</p> <ul style="list-style-type: none"> <li>内容(H) F1</li> <li>RTEMS C 用户手册 Shift+F1</li> <li>GNU C/C++ 语言参考</li> <li>软件更新(U)...</li> <li>关于(A)...</li> </ul>

说明：菜单栏的使能状态由 Loong IDE 的当前操作状态决定。

### 3.2 工具栏

工具栏是用户操作的常用功能。

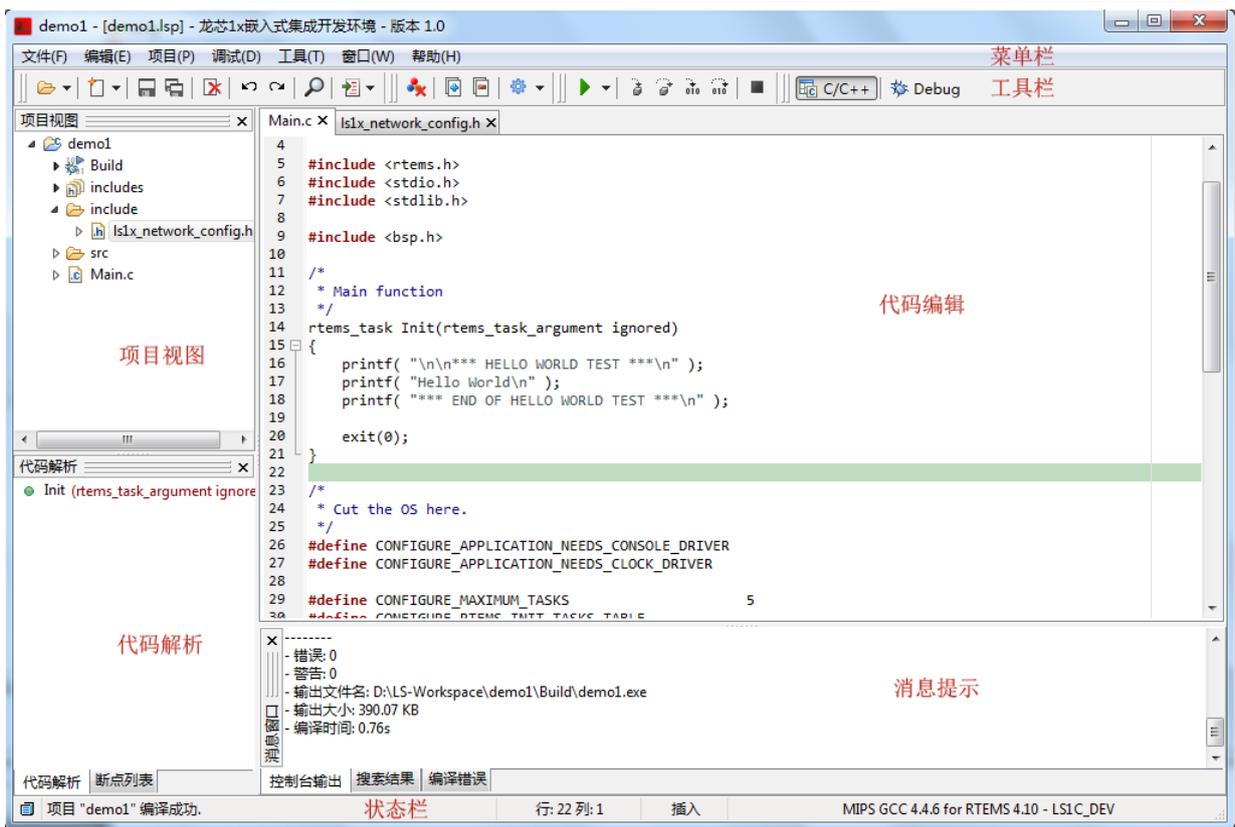
文件操作	 <ul style="list-style-type: none"> <li>→ 插入代码向导</li> <li>→ 查找</li> <li>→ 重做</li> <li>→ 撤销</li> <li>→ 关闭当前编辑的文件</li> <li>→ 保存全部</li> <li>→ 保存当前编辑的文件</li> <li>→ 新建文件</li> <li>→ 打开文件</li> </ul>
项目操作	 <ul style="list-style-type: none"> <li>→ 关闭项目</li> <li>→ 成批删除文件</li> <li>→ 成批添加文件</li> <li>→ 编译</li> </ul>
调试操作	 <ul style="list-style-type: none"> <li>→ 开始调试</li> <li>→ 单步进入指令(汇编)</li> <li>→ 单步进入</li> <li>→ 单步跳过指令(汇编)</li> <li>→ 单步跳过</li> <li>→ 停止调试</li> </ul>
界面切换	 <ul style="list-style-type: none"> <li>→ 切换到编辑界面</li> <li>→ 切换到调试界面</li> </ul>

说明：工具栏按钮的使能状态由 LoongIDE 的当前操作状态决定。

### 3.3 编辑面板

是用户常用的操作区，主要功能如下：

- 项目的创建、管理、编译等操作
- 文件夹的创建、移动、删除等操作
- 文件的创建、编辑、修改、删除等操作



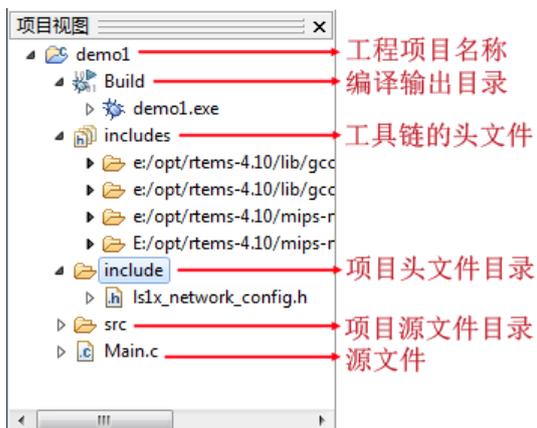
面板区域:

由上图所示的中间四个区构成;

其中用于代码编辑的“文本编辑器”不可关闭,“项目视图”、“代码解析”和“消息提示”可以根据操作需要进行打开或者关闭。

### 3.3.1 项目视图

以树形表展示当前工程项目的全部文件夹、文件, 如图所示:



文件夹:

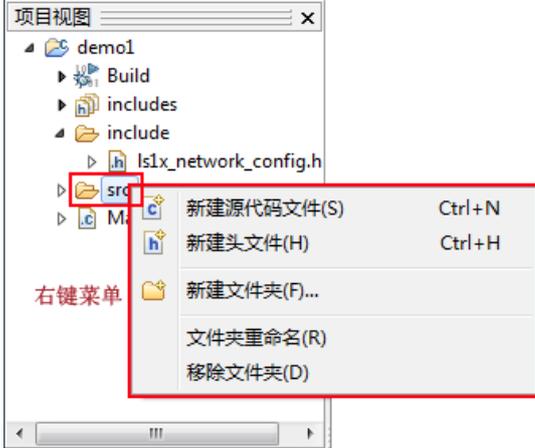
- “Build”: 系统目录, 存放当前项目最终编译输出的可执行文件
- “includes”: 虚拟目录, 列出工具链用到的库文件、头文件的目录, 便于用户查找

- “include”：用户目录，存放当前项目的头文件，编译时自动包含本目录为搜索路径之一，（用户可以不使用、改名或者删除）
- “src”：用户目录，存放当前项目的源文件，（用户可以不使用、改名或者删除）

**操作：**

- 右键菜单：用户根据操作需要，选用不同的弹出菜单项
- 鼠标拖放：用户定义的文件、文件夹可使用拖放操作，实现“移动”
- Double-Click/Ctrl+Click/Enter：文件名节点处，执行文件打开操作
- Delete：文件名节点处，执行文件删除操作
- Ctrl+Click：文件夹节点处，执行文件夹打开操作

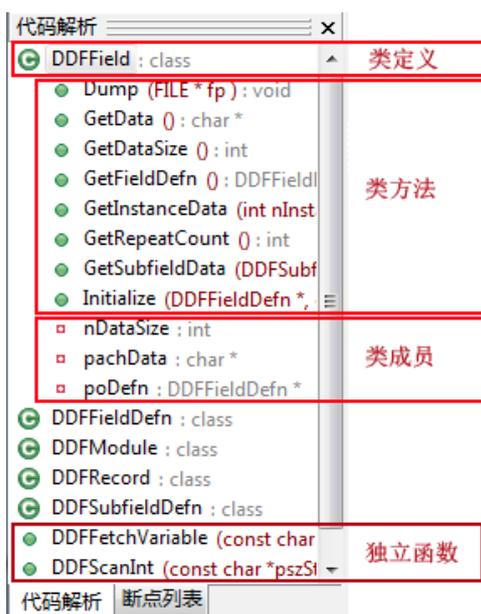
**右键菜单：**

<p>没有项目打开</p>	
<p>项目名称处</p>	
<p>项目文件夹处</p>	



### 3.3.2 代码解析

以树形表显示当前文件编辑器编辑的源代码、头文件的解析结果。(不解析工具链的头文件)



操作:

- Click: 单击节点, 在文本编辑器中快速定位到节点名称的定义原型的所在行
- Double-Click: 如果有子节点, 展开/折叠子节点

### 3.3.3 文本编辑器

实现文本文件, 包括 C/C++、MIPS 汇编等源文件和头文件的编辑, 支持 C/C++语法高亮显示, 支持剪贴板操作。

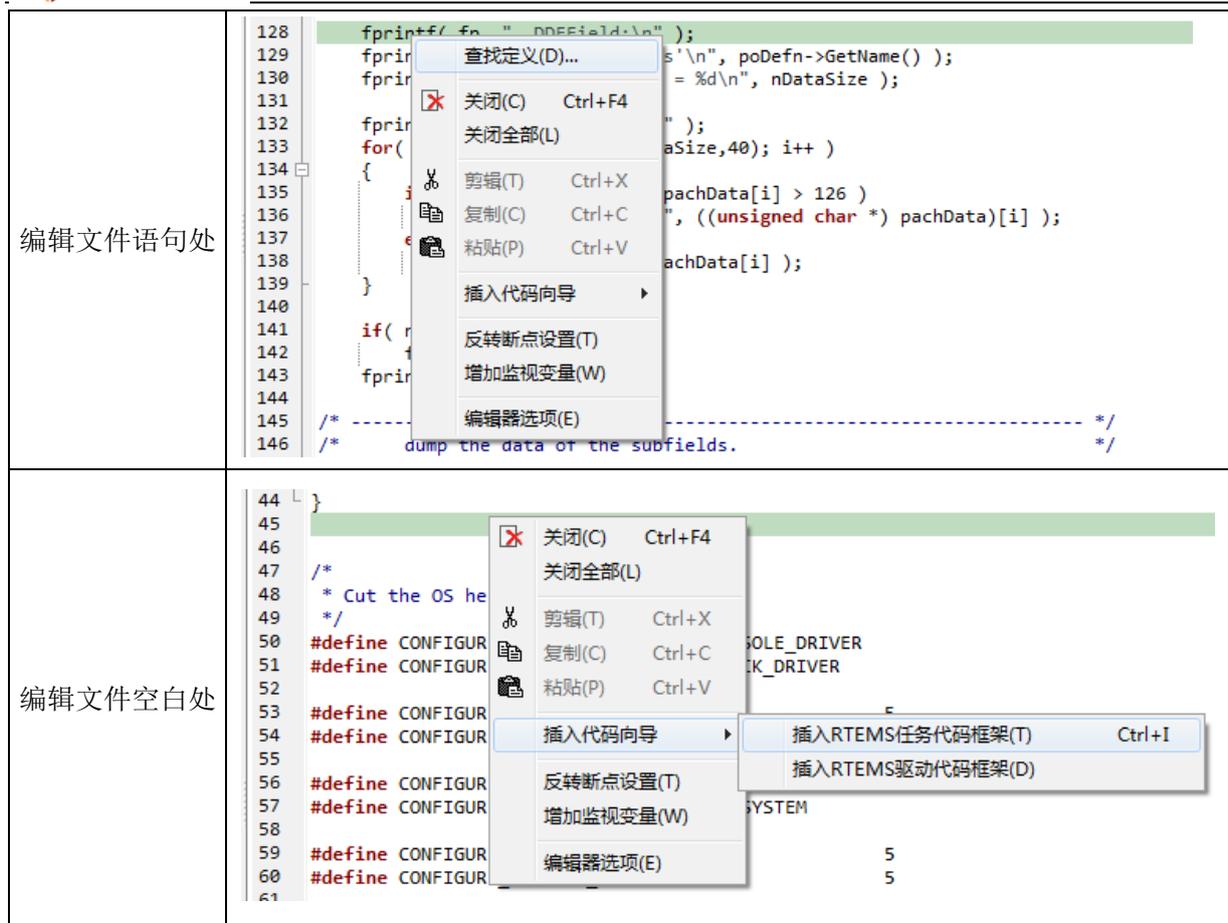


操作: 快速定位源代码中的语句定义原型, 包括宏、类型、变量、函数、类、类成员等。

- Ctrl+Click: 在文本处按住 Ctrl 键, 待光标变成  后单击鼠标左键, 实现定位操作
- 右键菜单: 在文本处打开弹出菜单, 使用“查找定义”菜单项, 实现定位操作

右键菜单:





### 3.3.4 消息窗口

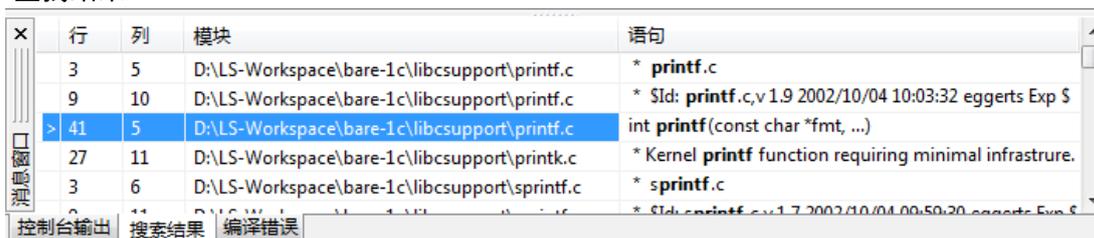
用于向用户反馈当前操作的状态信息，由“控制台输出”、“查找结果”、“编译错误”三个页框组成。

- “控制台输出”用于通用信息交互
- “查找结果”用于菜单“编辑→在文件中查找”的查找结果，并可快速定位
- “编译错误”用于显示项目编译的错误信息，并可快速定位

#### 控制台输出



#### 查找结果



Double-Click 行实现快速定位。

### 编译错误

行	列	模块	信息
87		..\iso8211\ddffield.cpp	In file included from ..\iso8211\ddffield.cpp
35	22	..\iso8211\iso8211.h	[Error] cpl_port.h: No such file or directory
88	22	..\iso8211\ddffield.cpp	[Error] cpl_conv.h: No such file or directory
87		..\iso8211\ddffield.cpp	In file included from ..\iso8211\ddffield.cpp
100		..\iso8211\iso8211.h	[Error] 'FILE' has not been declared

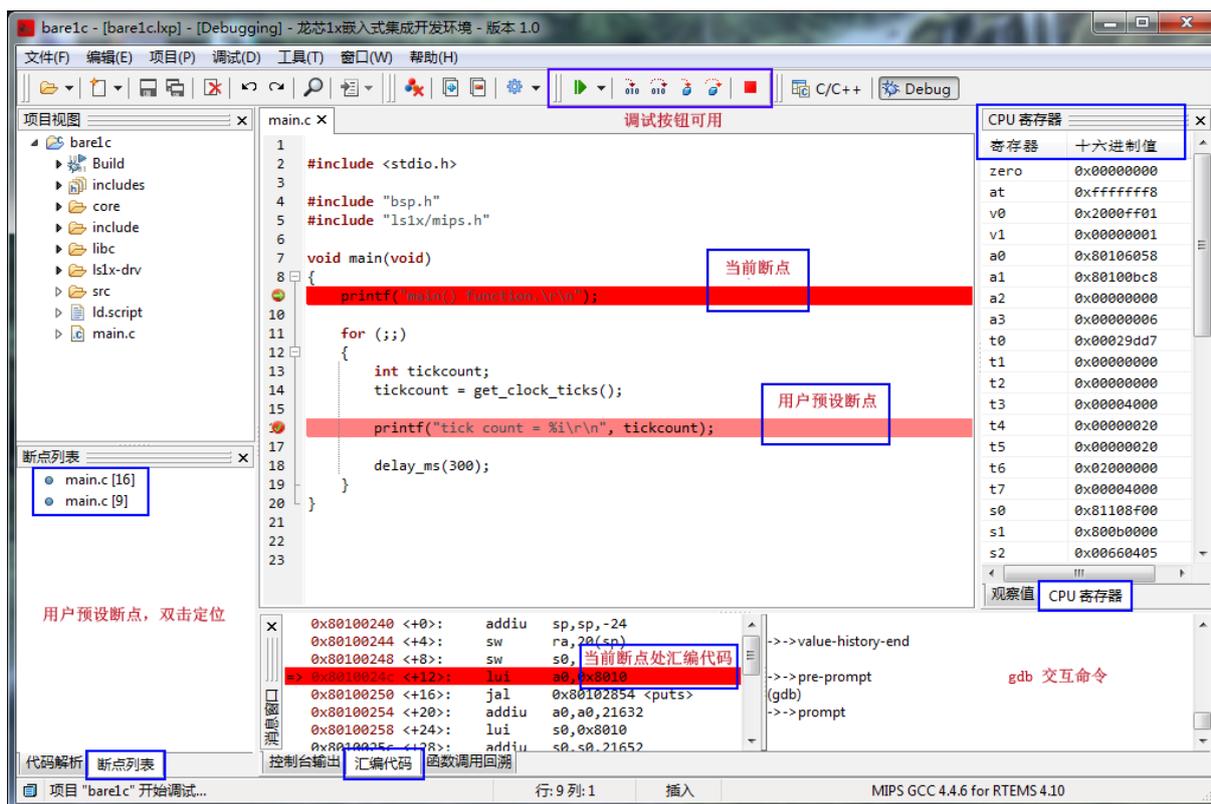
Double-Click 行实现快速定位。

## 3.4 调试面板

当用户项目编译成功，并在“Build”目录下生成可执行文件后：

- 使用菜单“调试→启动”
- 使用工具栏按钮  启动调试

LoongIDE 进入用户调试界面如下：

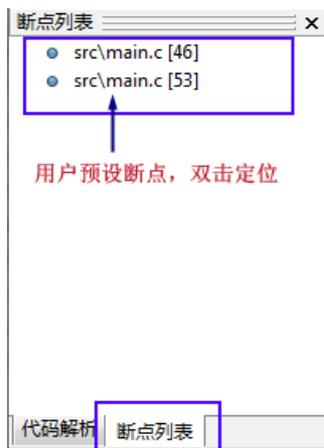


### 面板区域

- 文本编辑器
- 断点列表
- 消息窗口

- CPU 寄存器
- 观察值
- 消息窗口
  - 控制台输出
  - 汇编代码
  - 函数调用跟踪
  - GDB 交互命令

### 3.4.1 断点列表



#### 断点设置:

- Click: 单击文本编辑器源代码行的行号处，执行增加断点/移除断点操作
- Right-Click: 在文本编辑器的源代码行上，使用右键菜单的“反转断点设置”，执行增加断点/移除断点操作

### 3.4.2 CPU 寄存器

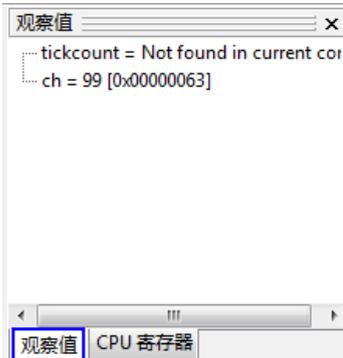
显示当前断点处的 CPU 寄存器值。当调试到达断点处时更新，仅用于查看。

寄存器	十六进制值	+	^
zero	0x00000000		
at	0x801007ec		
v0	0x01600f80		
v1	0x00000f80		
a0	0x00000003		
a1	0xbfe48003		
a2	0x801005b4		
a3	0xffffffff		
t0	0xffffffff		
t1	0x00000005		
t2	0x00000001		

观察值 CPU 寄存器

### 3.4.3 观察值

显示用户设置的 Watch Var，当调试到达断点处时更新。



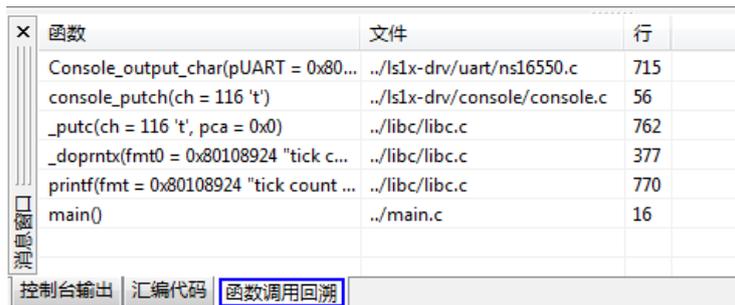
### 3.4.4 汇编代码

显示当前断点处的汇编代码。当调试到达断点处时更新，仅用于查看。



### 3.4.5 函数调用回溯

显示当前断点所在函数的调用层次关系，用于用户分析代码。



### 3.4.6 GDB 交互命令

用于显示调试过程中 GDB 使用的命令。

```
Source directories searched: D:/LS-Workspace/bare-1c;$cdir;$cwd
--> pre-prompt
(gdb)
--> prompt
--> post-prompt
Source directories searched: D:/LS-Workspace/bare-1c/include;D:/LS-Workspace/bare-1c;$cdir;$cwd
```

**GDB 交互命令**

### 3.5 状态栏

用户操作的状态、信息提示。



提示信息:

- 文本编辑器当前光标位置
- “进度条”标示长耗时操作的进度

## 4、项目管理

### 4.1 新建项目向导

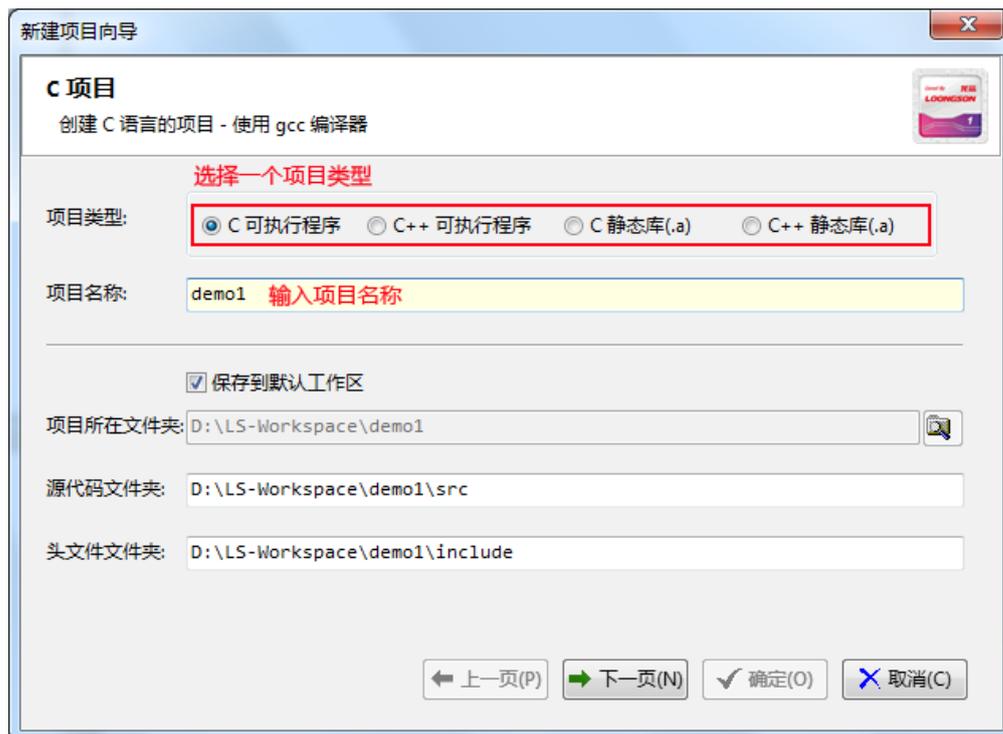
LoongIDE 新建项目，通过本向导实现。

- 使用菜单：“文件→新建→新建项目向导...”创建新项目
- 当没有项目被打开时，使用“项目视图”、“代码解析”面板的右键快捷菜单“新建项目向导...”创建新项目

执行菜单“新建项目向导...”，开始[新项目设置...](#)

#### 4.1.1 第一步 项目基本信息

输入新项目基本信息，用于在工作区创建一个以“项目名称.lxp”命名的新项目。



项目类型：

- C Executable: C 可执行程序（ELF 格式的.exe 文件），项目使用 gcc 编译
- C++ Executable: C++ 可执行程序（ELF 格式的.exe 文件），项目使用 g++ 编译
- C Static Library: C 静态库（.a 文件），项目使用 gcc 编译
- C++ Static Library: C++ 静态库（.a 文件），项目使用 g++ 编译

项目名称:

用户自定义的本项目名称

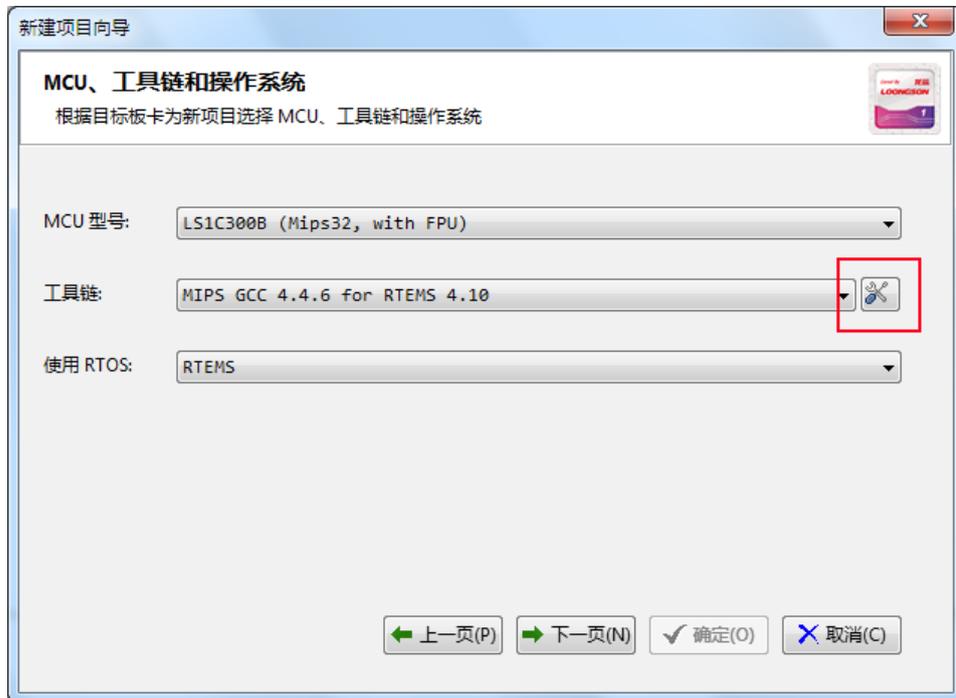
项目保存目录:

指定项目的文件夹位置，项目文件的扩展名为: .lxp

项目基本信息设置完成后，单击【[下一页](#)】...

#### 4.1.2 第二步 设置 Mcu、工具链和操作系统

根据项目的需求，选择 Mcu 等配置选项。



芯片型号:

根据新项目将运行的目标芯片选择芯片型号。

工具链:

为新项目选择编译使用的工具链:

- MIPS GCC 4.4.6 for RTEMS 4.10
- MIPS GCC 4.9.3 for RTEMS 4.11
- SDE Lite 4.5.2 for MIPS
- SDE Lite 4.9.2 for MIPS

单击“工具链管理”按钮，进入系统工具链管理窗口。

使用 RTOS:

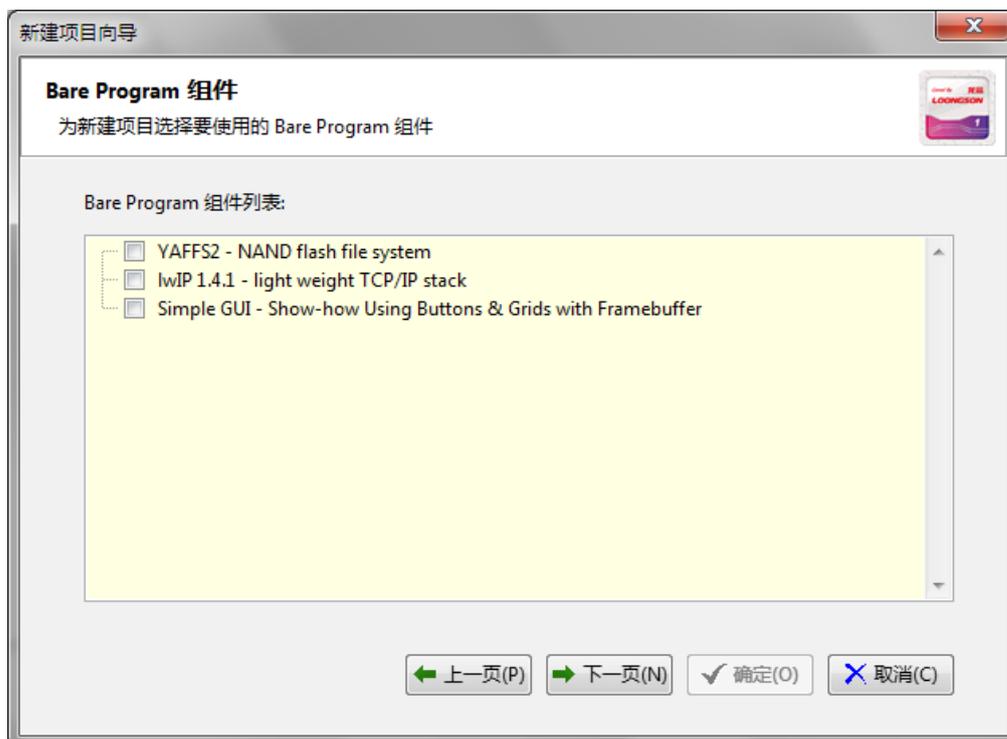
根据新项目需求，使用裸机编程或者选用实时操作系统进行项目开发。

- None (bare programming)
- RT-Thread
- FreeRTOS
- uCOS-II
- RTEMS

单击【[上一步](#)】进行更改或者【[下一步](#)】继续操作...

#### 4.1.3 第三步 实时操作系统选项

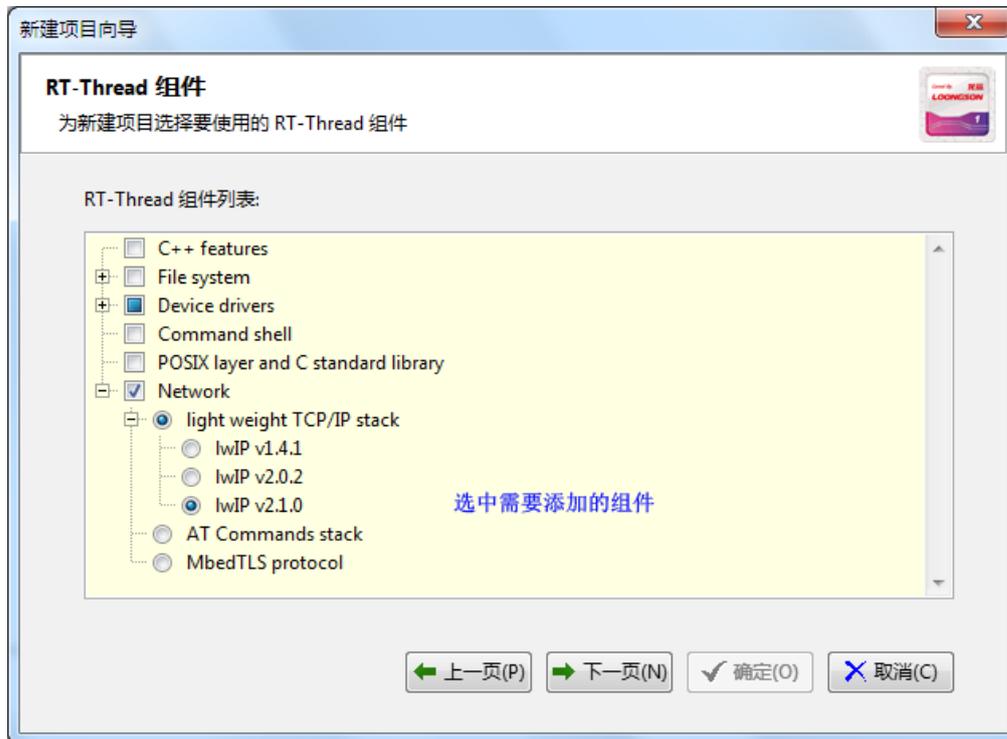
1、当项目选用“裸机/uCOSII/FreeRTOS”进行编程时，进入如下界面：



#### 组件列表

- 用户根据项目的实际需求，选择需要添加到新项目的组件；
- 如果需要生成 DEMO 项目，则需要添加组件。

2、当项目选用“RT-Thread”进行编程时，进入如下界面：

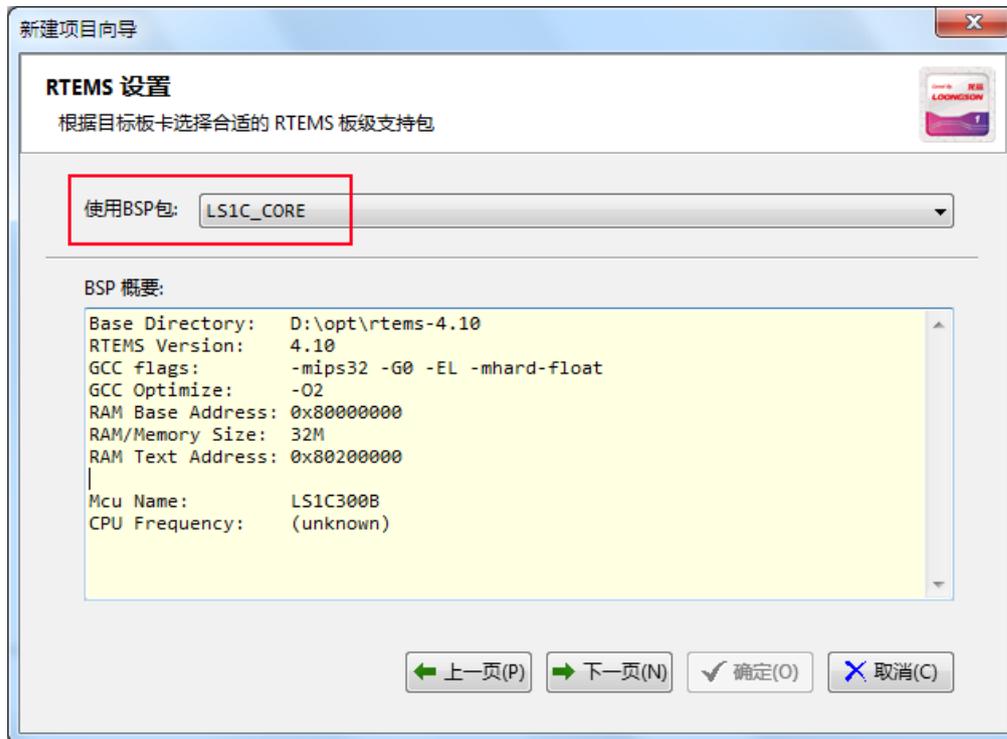


### RT-Thread 组件列表

- 用户根据项目的实际需求，选择需要添加到新项目的组件；
- 在项目创建完成后，可以通过添加相关文件以实现 RT-Thread 组件的添加；

注：本组件选择使用第 1 项的组件替代，其控制参数为 Template.ini 的 RTTComponents 段的 choiceme 的值。

3、当项目选用“RTEMS”进行编程时，进入如下界面：



### 使用 BSP 包

项目将使用已移植好的龙芯 1x RTEMS 板级支持包进行开发。

### BSP 概要

选择的 BSP 包的主要信息。

单击【[上一步](#)】进行更改或者【[下一步](#)】继续操作...

#### 4.1.4 第四步 确认并完成向导

显示新建项目汇总信息如下：



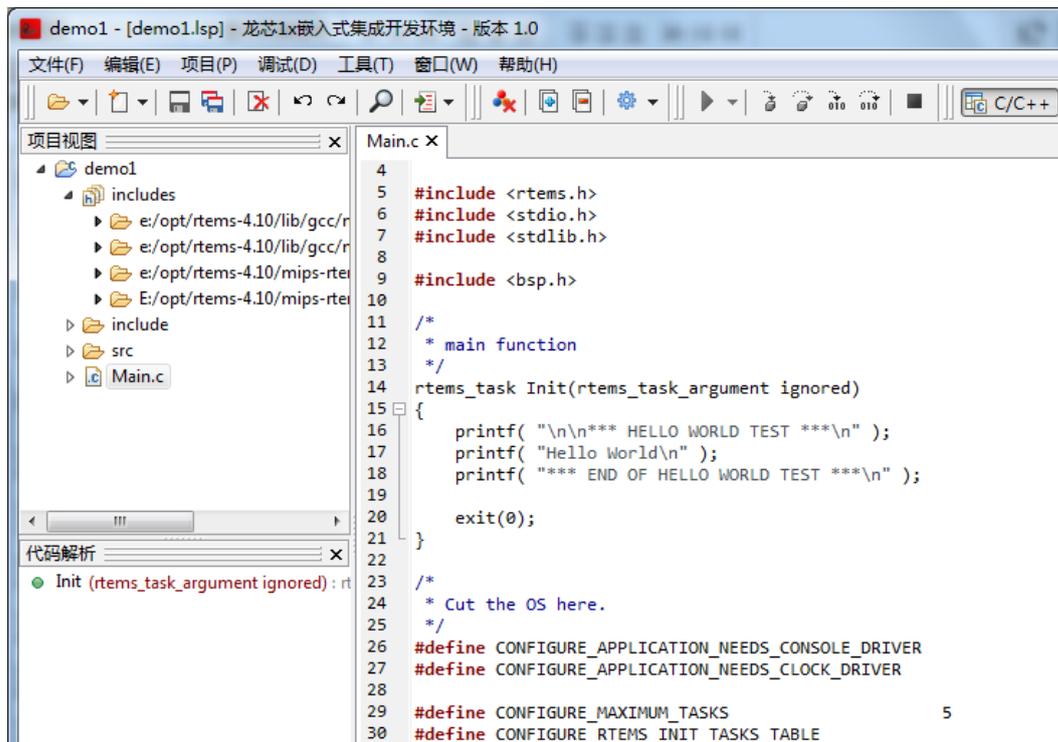
从模板加入源代码文件：

根据向导选择，在创建新项目的时候，添加相应的文件：

- 裸机编程项目，向导将为新建项目创建：
  - ◆ 启动文件 start.S；
  - ◆ 链接脚本文件 ld.script 或者 linkcmds；
  - ◆ libc 库文件(部分)；
  - ◆ 项目基本代码文件。
- RTEMS 项目：
  - ◆ 加入 main.c 文件或 main.cpp 文件；
  - ◆ ls1x\_network\_config.h 头文件；
  - ◆ 项目基本代码文件。
- RT-Thread、FreeRTOS、uCOS-II 项目：
  - ◆ 加入 RT-Thread、FreeRTOS 内核文件；
  - ◆ 没有提供 uCOS-II 内核文件；
  - ◆ 加入相应 RTOS 的移植文件；
  - ◆ 项目框架代码文件。
- 如果选择添加“组件”，项目添加相应的组件源文件；
- 如果选择“为新建项目加入演示代码”，添加演示项目的源文件。

单击【[上一步](#)】进行更改或者【[确定](#)】完成新项目创建。

## 4.1.5 新建项目示例



项目新建完成后，可以执行以下操作：

- 在项目中新建、添加、编辑、保存源代码文件
- 执行编译，并对编译错误进行修改处理
- 在项目编译成功后，启动调试

## 4.2 基本操作

LoongIDE 基于项目对用户应用开发进行管理，编译项目中的源代码文件，链接生成可执行的应用程序。

- 打开、保存和关闭项目
- 将项目另存为一个新项目
- 成批添加文件到项目
- 从项目中成批删除文件

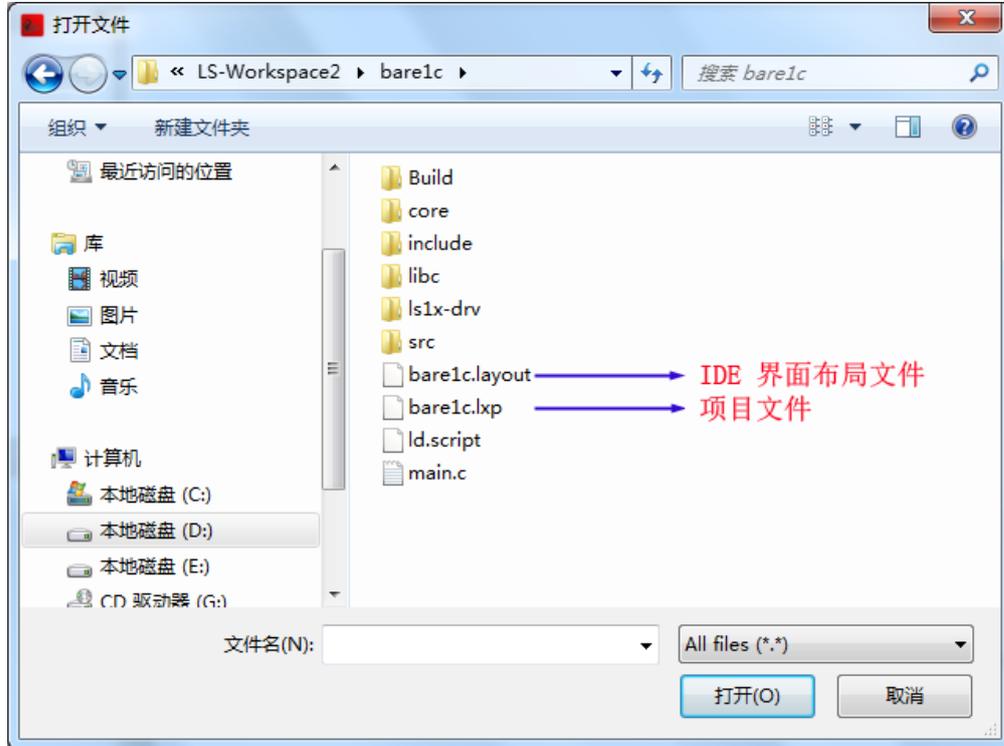
### 4.2.1 打开项目

通过执行以下操作之一打开项目（注：项目文件扩展名为 .lxp）：

- 使用主菜单：“项目→打开项目...”
- 使用工具栏按钮  打开项目

- 使用工具栏按钮  的下拉菜单“打开项目...”
- 使用“项目视图”面板的右键快捷菜单“打开项目...”

显示窗口如下：



选择要打开的 **.lxp** 项目文件，单击【打开】。

#### 4.2.2 保存项目

- 使用主菜单：“文件→保存全部”
- 使用工具栏按钮 

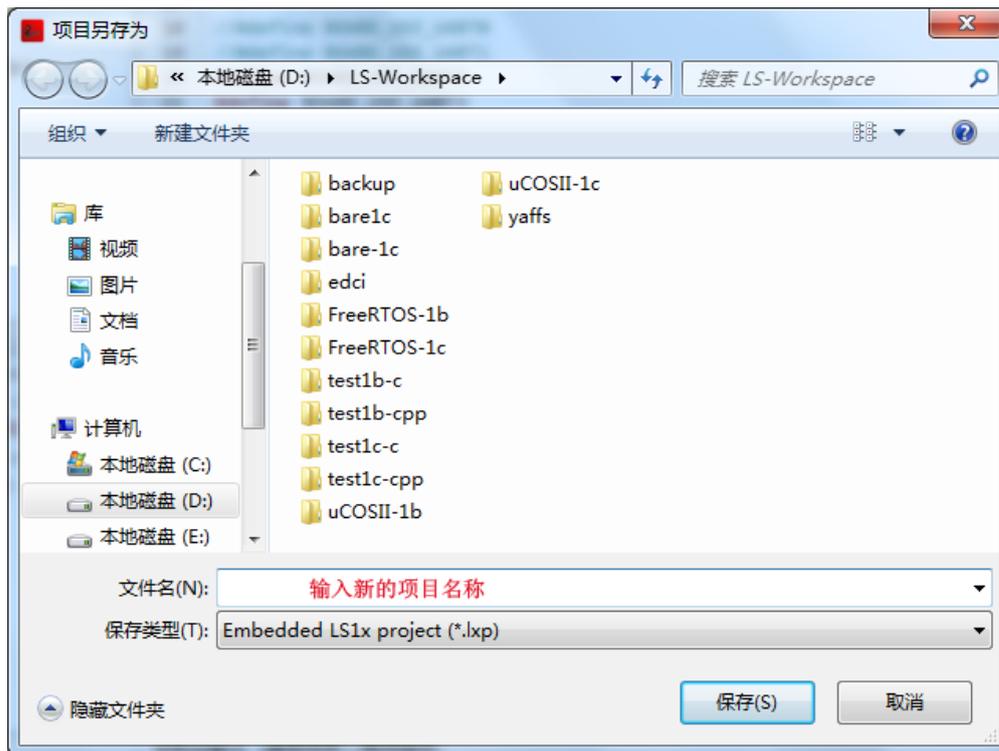
如果项目被修改，关闭项目前将提示是否保存。

#### 4.2.3 关闭项目

- 使用主菜单：“项目→关闭项目”
- 使用工具栏按钮 

#### 4.2.4 项目另存为

打开主菜单“项目→另存项目为...”：

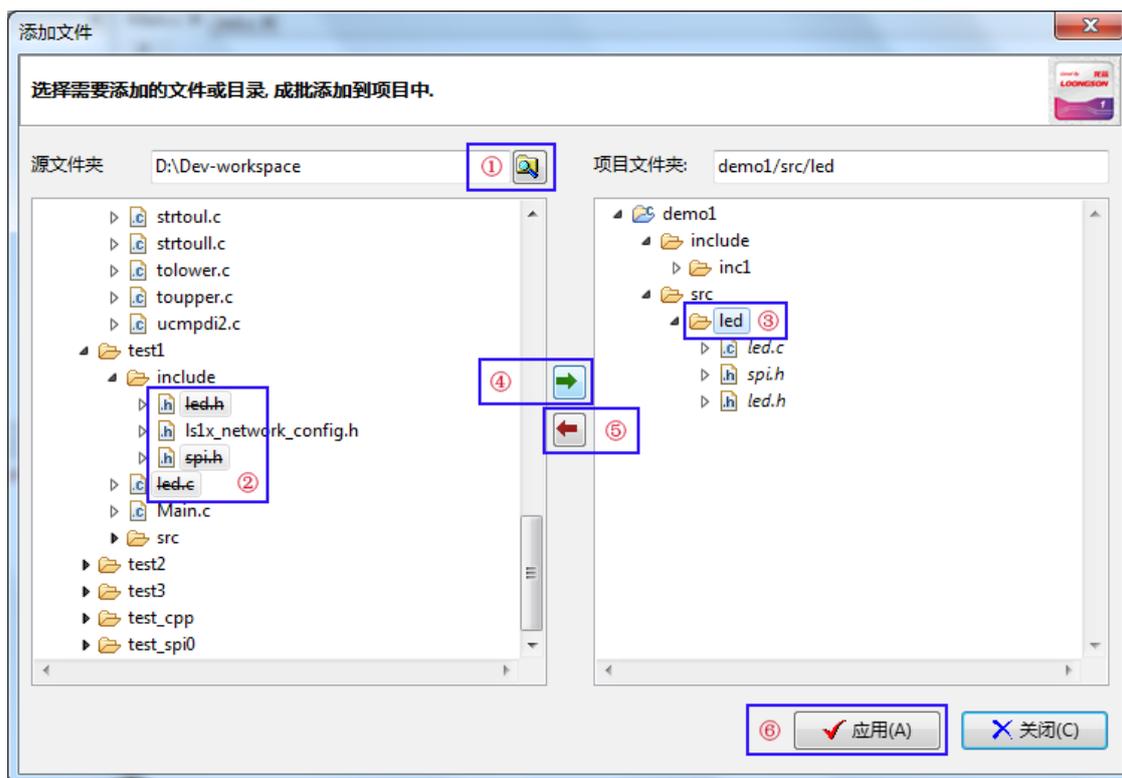


输入一个新项目名称，单击按钮【保存】。

- 如果另存项目与原项目在同一个目录，本操作仅创建一个新的项目文件
- 如果不在同一个目录，本操作除将在目标目录创建项目文件之外，将复制全部项目源文件到新项目
- 另存项目操作执行完成后，当前打开项目切换到另存的新项目

#### 4.2.5 成批添加文件

使用菜单“项目→添加文件到项目”或者工具栏按钮 ，实现向项目批量添加文件。



#### 操作步骤：

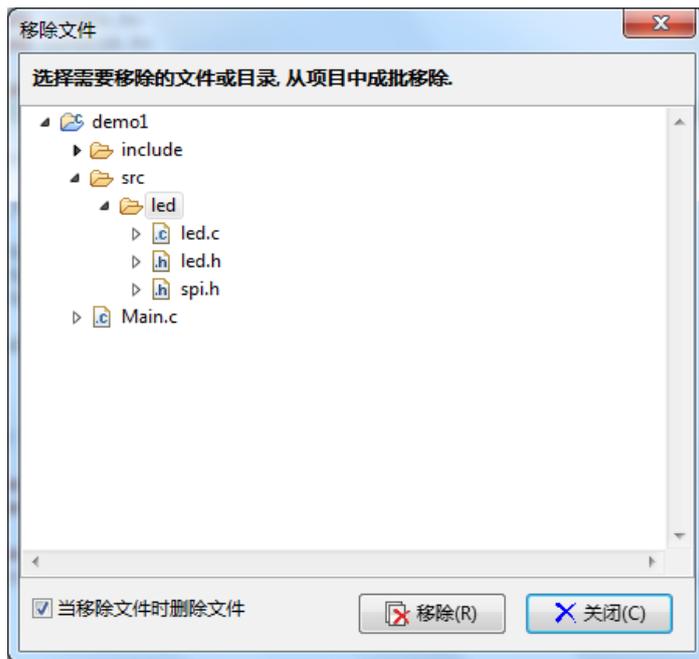
- ◆ ①、选择源文件夹，文件夹下的所有源文件显示在窗口左侧树形表中；
- ◆ ②、从源文件树形表选择需要添加的文件、文件夹，使用 **Ctrl+Click** 进行多选；
- ◆ ③、窗口右侧树形表是当前项目的目录结构，选择一个文件夹用于存放将添加的文件；
- ◆ ④、点击窗口中部的  按钮，选中的文件被添加到当前项目的文件夹下；
- ◆ ⑤、从目标树形表选择文件，点击窗口中部的  按钮，将移除已添加的文件；
- ◆ ⑥、添加选择完成后，单击【应用】按钮，确认本次操作。

注：源文件树形表在文件或文件夹被添加到当前项目后，以删除线字体显示；  
当前项目树形表以斜体字体显示待添加的文件或文件夹。

#### 4.2.6 成批移除文件

使用菜单“项目→从项目删除文件”或者工具栏按钮 ，实现从项目批量移除文件。

当前项目的所有源文件以树形表显示如下：



#### 操作步骤：

- ◆ ①、从树形表选择需要移除的文件、文件夹，使用 **Ctrl+Click** 进行多选；
- ◆ ②、如果要从磁盘删除文件，选中“当移除文件时删除文件”单选框；
- ◆ ③、最后，单击【移除】按钮，确认本次操作。

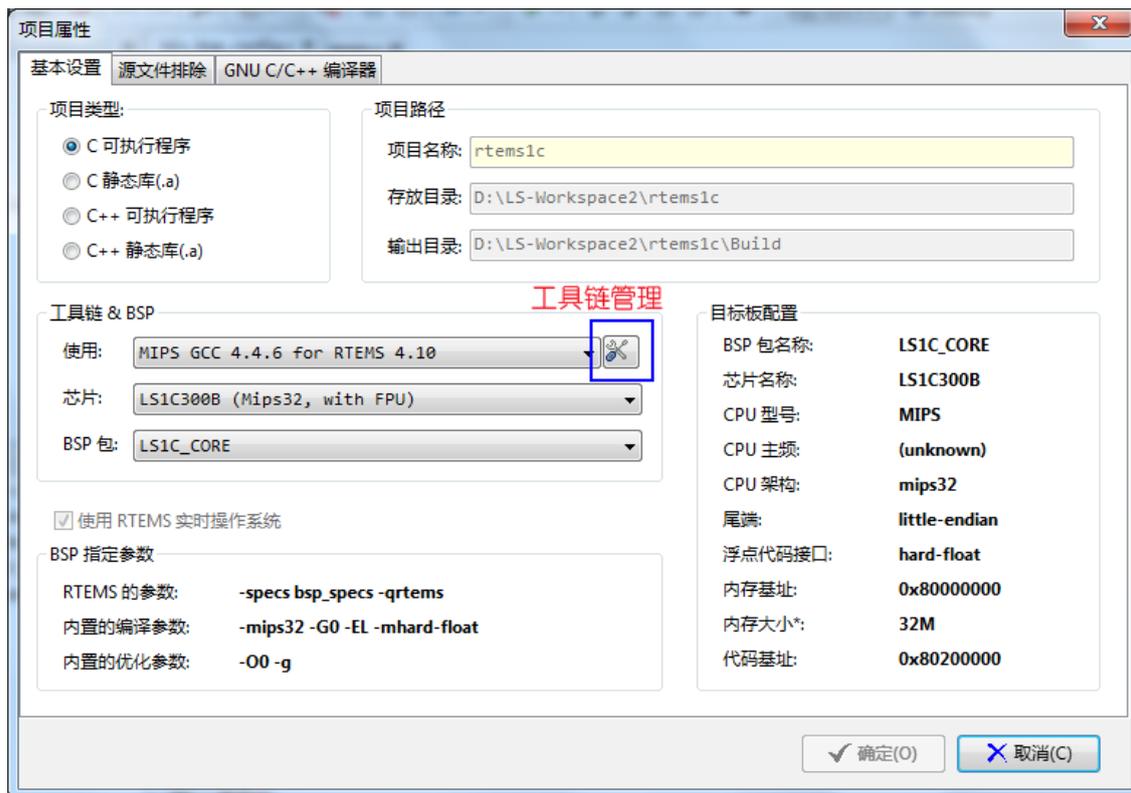
注：推荐不要选中“当移除文件时删除文件”单选框，这样执行本操作后，文件仍保留在磁盘上，仅从当前项目执行移除。

### 4.3 项目属性

对当前打开的项目进行配置修改，包括工具链、芯片、BSP 等。

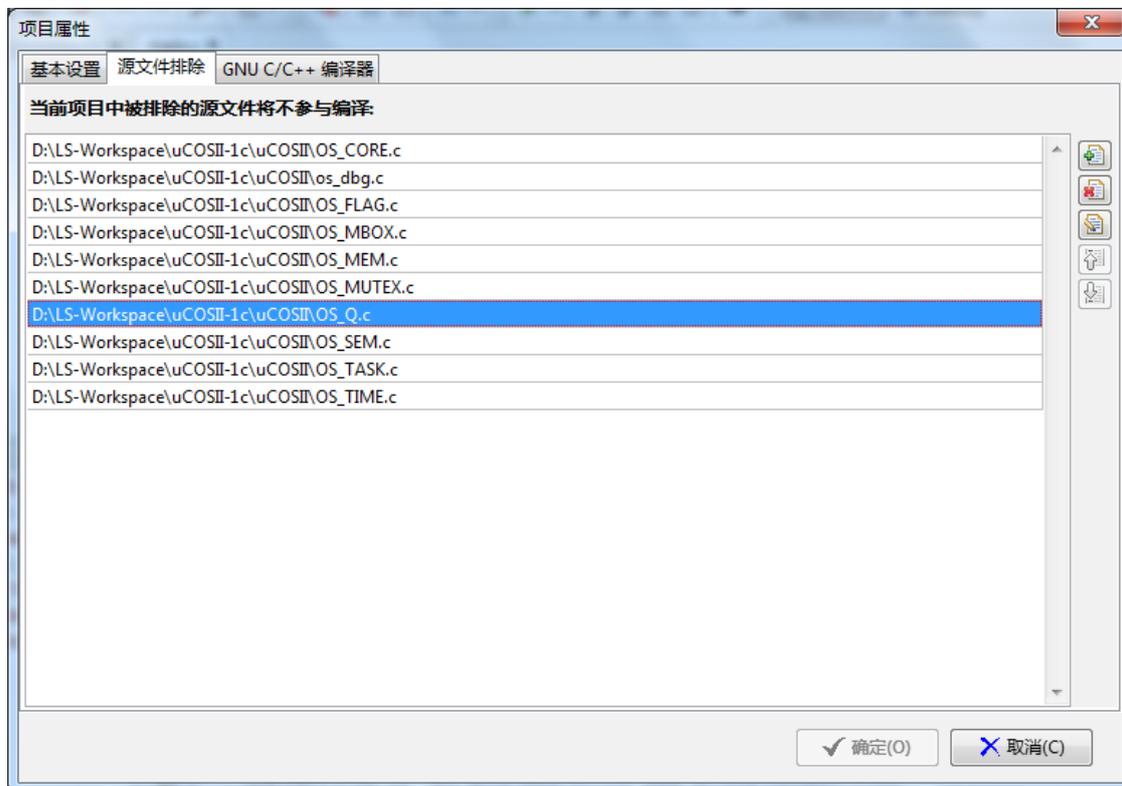
- 使用主菜单“项目→项目属性”
- 使用项目视图面板的右键快捷菜单，选择“项目属性”

基本设置：



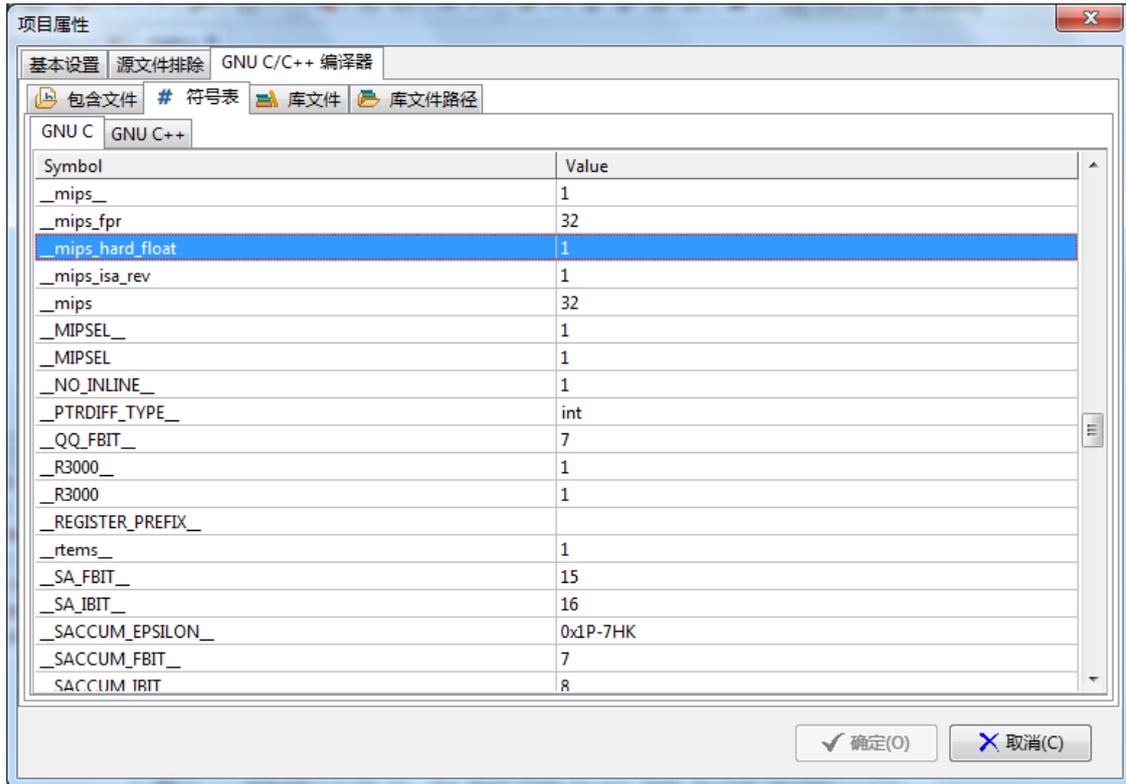
警告：一般情况下不要修改项目类型

源文件排除：



列表中的文件不参与直接编译，一般被其它源文件#include。

GNU C/C++ 编译器:



各页框内容如下:

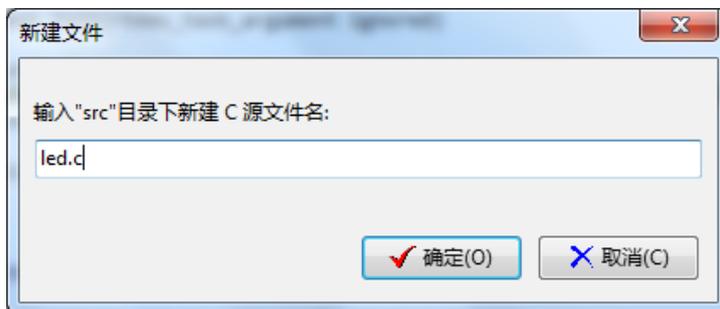
- 包含文件: 编译器内置头文件搜索路径;
- 符号表: 编译器内置宏定义;
- 库文件: 工具链使用的用户定义的库文件 (.a);
- 库文件路径: 工具链使用的用户定义的库文件搜索路径。

## 5、文档管理

### 5.1 文件操作

#### 5.1.1 新建源代码文件

- 使用主菜单“文件→新建→新建源代码文件”
- 使用工具栏按钮 ，从下拉菜单选择“新建源代码文件”
- 当项目打开时，在项目视图面板的文件夹处，使用右键快捷菜单“新建源代码文件”



输入新建的源文件名，按【确定】按钮。

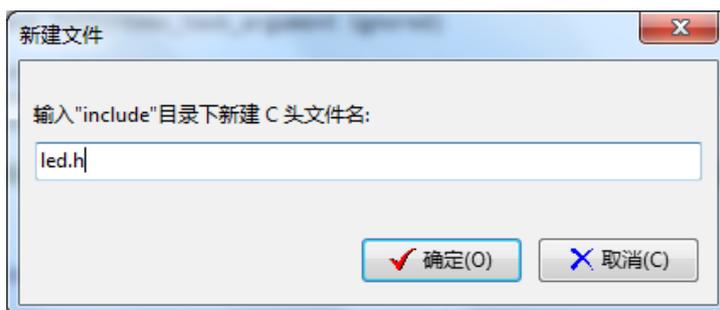
- 如果有项目被打开，新建源文件将保存到“项目视图”面板选中的文件夹下
  - 如果新建文件名在磁盘上已存在，系统提示是否将文件加入到当前项目
  - 如果没有项目被打开，新建源文件是独立文件，保存时需要选择保存路径
- 新建文件成功后，编辑器打开新建文件。

源文件扩展名：

- C 项目下合法的源文件扩展名：.c 和 .S
- C++ 项目下合法的源文件扩展名：.c、.cpp 和 .S

#### 5.1.2 新建头文件

- 使用主菜单“文件→新建→新建头文件”
- 使用工具栏按钮 ，从下拉菜单选择“新建头文件”
- 当项目打开时，在项目视图面板的文件夹处，使用右键快捷菜单“新建头文件”



输入新建的头文件名，按【确定】按钮。

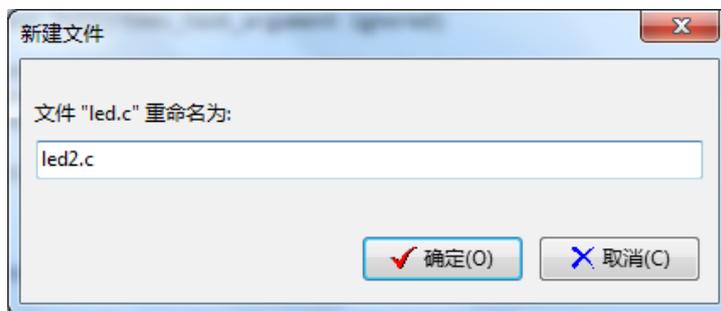
- 如果有项目被打开，新建头文件将保存到“项目视图”面板选中的文件夹下
  - 如果新建文件名在磁盘上已存在，系统提示是否将文件加入到当前项目
  - 如果没有项目被打开，新建头文件是独立文件，保存时需要选择保存路径
- 新建文件成功后，编辑器打开新建文件。

头文件扩展名：

- C 项目下合法的头文件扩展名：.h
- C++ 项目下合法的头文件扩展名：.h 和 .hpp

### 5.1.3 文件重命名

在项目视图面板的项目文件处，使用右键快捷菜单“文件重命名”。

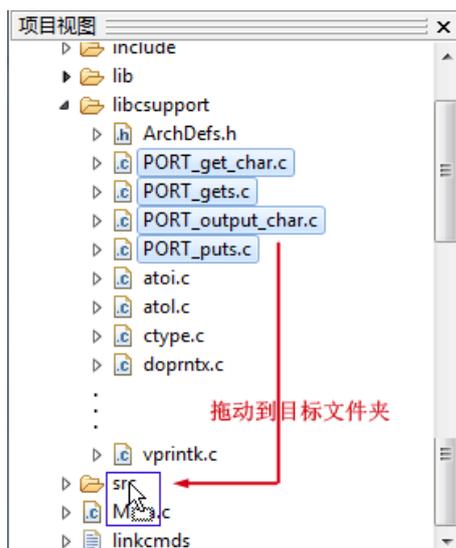


输入新的文件名，单击按钮【确定】。

- 如果新文件名在当前项目中已经存在，文件重命名失败
- 如果新文件名在磁盘存在、但不在当前项目中，系统提示用户选择覆盖操作

### 5.1.4 文件移动

在“项目视图”面板内，使用鼠标拖放操作，实现项目文件的移动。

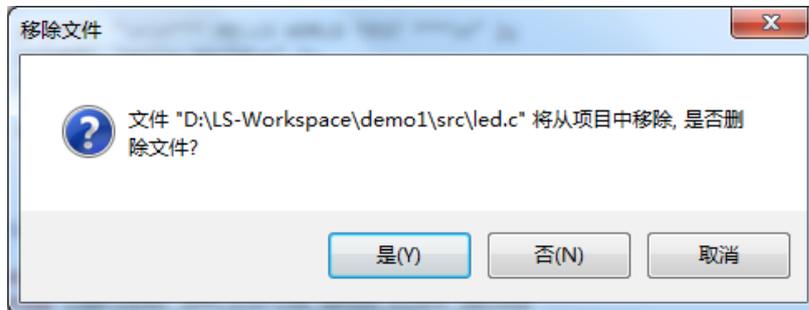


按住 Ctrl 键，实现多选操作。

LoongIDE 创建的“includes”和“Build” 文件夹及其下文件，禁止本操作。

### 5.1.5 文件删除

在项目视图面板的项目文件处，使用右键快捷菜单“移除文件”。



操作选择：

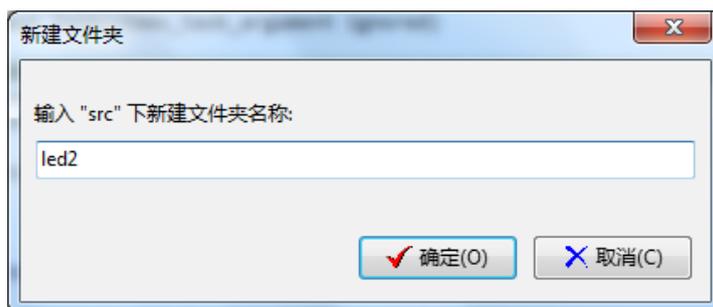
- 选择【是】：从项目移除文件，并且从磁盘删除改文件
- 选择【否】：从项目移除文件，但文件仍保留在磁盘上
- 选择【取消】：放弃本次操作

## 5.2 文件夹操作

### 5.2.1 新建文件夹

当有项目打开时：

- 使用工具栏按钮  下拉菜单“新建文件夹”
- 使用项目视图文件夹处右键快捷菜单“新建文件夹”

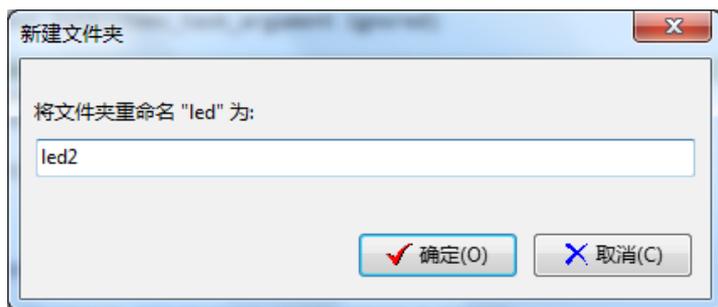


输入新建文件夹名称，单击【确定】按钮。

- 如果新文件夹名称在当前项目中已经存在，新建文件夹失败
- 如果新文件夹名称在磁盘存在、但不在当前项目中，系统提示用户是否加入到当前项目

### 5.2.2 重命名文件夹

当有项目打开时，使用项目视图文件夹处右键快捷菜单“文件夹重命名”

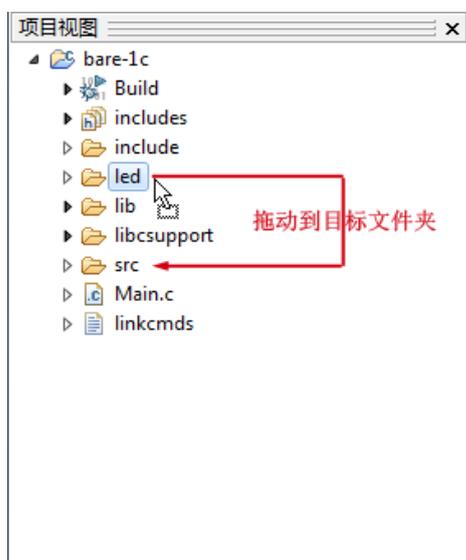


输入新的文件夹名称，单击按钮【确定】。

- 如果新文件夹名称在当前项目中已经存在，文件夹重命名失败
- 如果新文件夹名称在磁盘存在、但不在当前项目中，文件夹重命名失败

### 5.2.3 移动文件夹

在“项目视图”面板内，使用鼠标拖放操作，实现项目文件夹的移动。



如果移动的文件夹在目标文件夹下已经存在，移动失败。

### 5.2.4 删除文件夹

在“项目视图”文件夹处，使用右键快捷菜单“移除文件夹”。

注：被删除文件夹下没有文件时，才可以执行删除文件夹操作。

## 5.3 Drag & Drop

“项目视图”面板支持操作系统的 Drag 和 Drop 操作，将文件、文件夹放入当前项目。

能够用于拖放操作的文件格式有：

- 文件扩展名：.c、.cpp、.S、.inl、.txt
- 链接脚本：ld.script、lindcmds、link.ld

## 6、文本编辑器

LoongIDE 的文本编辑器使用著名开源组件 SynEdit 来实现。

SynEdit 是一个高级的多行文本编辑控件，支持语法高亮、word-wrap、代码自动完成、模版组件、文本导出等功能。

### 6.1 编辑器选项

编辑器选项设置，影响文本编辑器的外观和操作特性，用户可以根据个人需求进行设置。

打开操作：

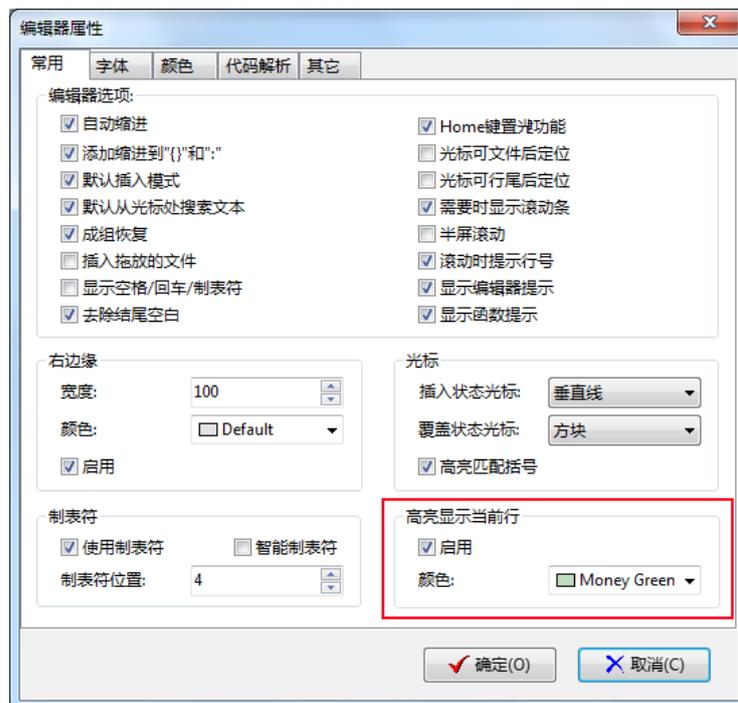
- 使用主菜单“工具→编辑器选项”
- 使用文本编辑器的右键弹出菜单，选择“编辑器选项”菜单项

用户设置分组为：

- 常用
- 字体
- 颜色
- 代码解析
- 其它

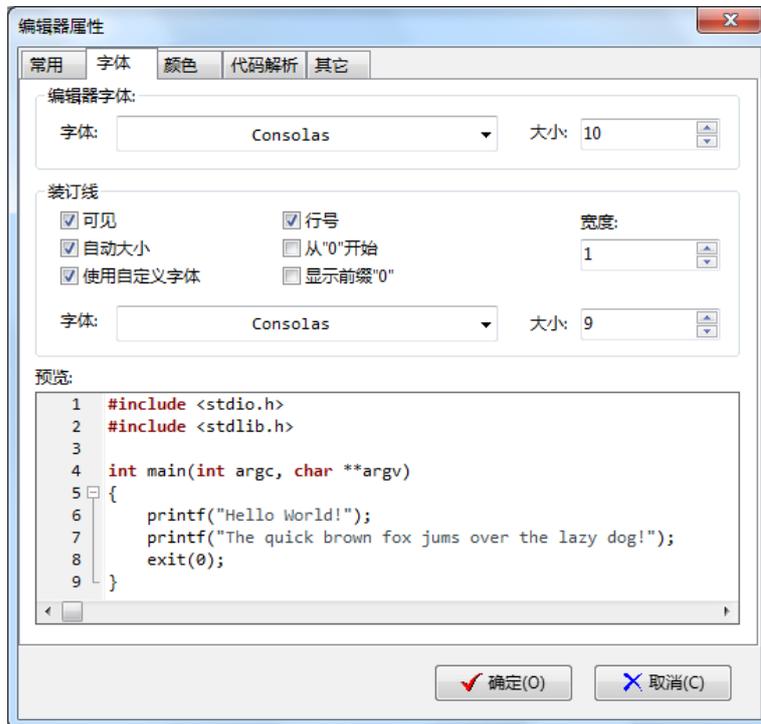
#### 6.1.1 常用

设置文本编辑器外观基本属性。



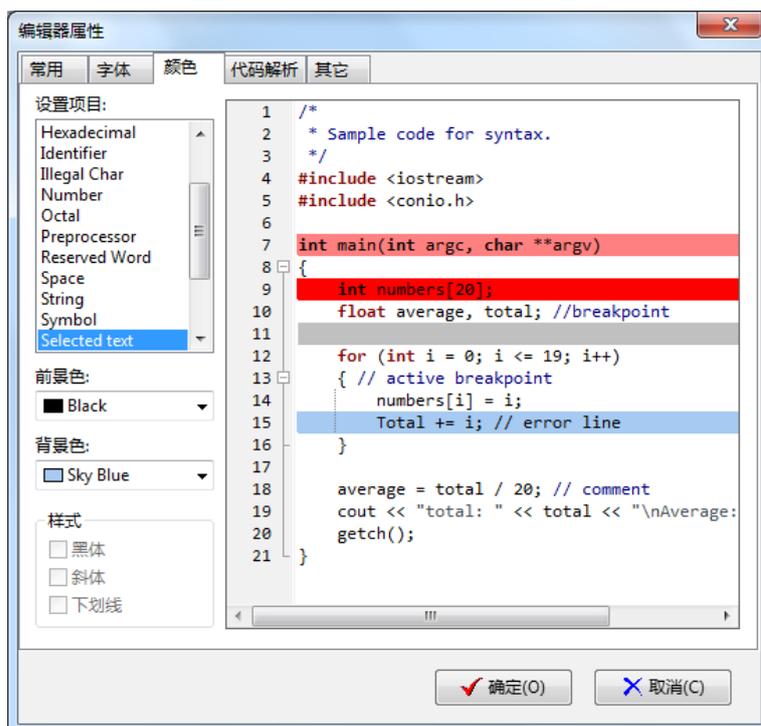
## 6.1.2 字体

设置文本编辑器的显示字体。



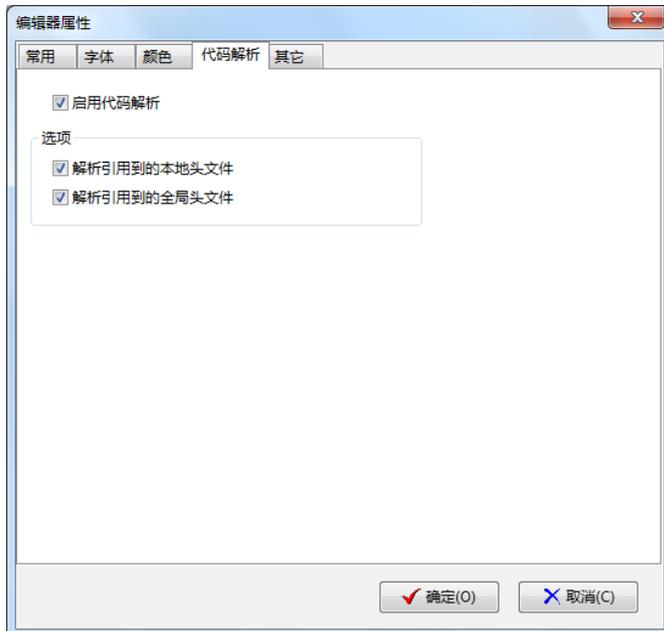
## 6.1.3 颜色

设置文本编辑器的字体颜色，语法高亮特性支持。



## 6.1.4 代码解析

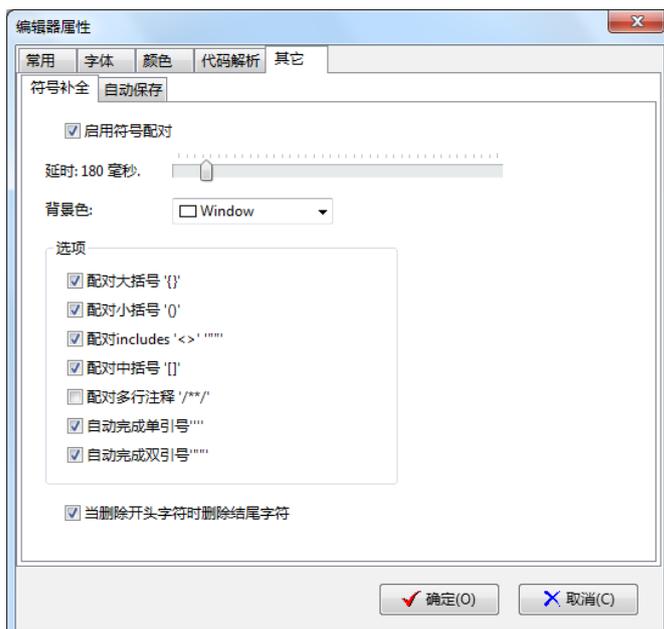
解析当前项目的 C/C++源文件和头文件，为代码的编辑操作提供更多支持。



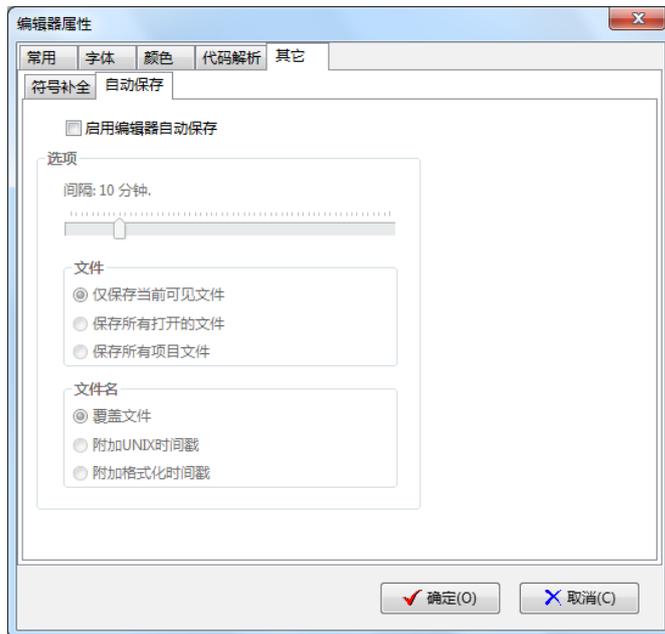
以下操作基于代码解析结果：

- 文本编辑器引用的头文件快速定位
- 文本编辑器引用的函数、变量、类、类成员等的快速定位
- 代码解析面板列出的函数、变量、类、类成员等的快速定位
- 编辑状态时，提示光标处函数定义原型
- 调试状态时，提示光标处变量当前值

## 6.1.5 符号补全



## 6.1.6 自动保存



## 6.2 基本操作

项目文件的编辑处理等操作。

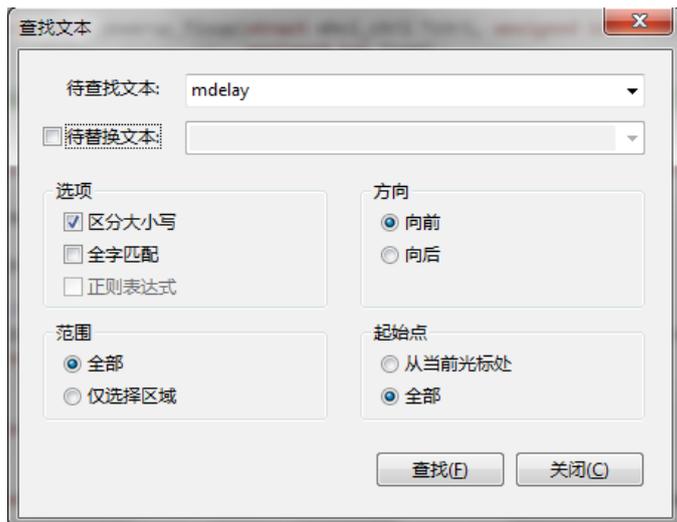
### 6.2.1 编辑

实现单个文件的文本编辑，同时支持剪贴板的 cut、copy、paste，支持 undo、redo 操作。

### 6.2.2 查找

当需要在当前编辑的文本中使用查找功能，打开查找窗口：

- 使用主菜单“编辑→查找”
- 使用快捷键“Ctrl+F”



当打开查找窗口时，编辑器有选择的文本或者光标位置的语句自动复制到“待查找文本”文本框。输入“待查找文本”和选项等，单击【查找】按钮，编辑器光标定位到第一个找到的文本处，同时按钮文本变为“下一个”。

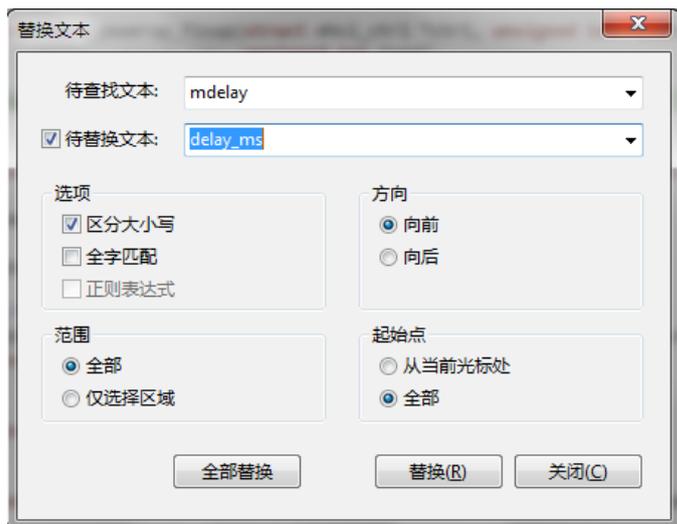
#### 查找下一个：

- 保持查找窗口打开，单击【下一个】按钮或者使用快捷键 <F3>，编辑器光标定位到下一个找到的文本处，或者提示未找到
- 关闭查找窗口，使用快捷键 <F3>，编辑器光标定位到下一个找到的文本处，或者提示未找到

### 6.2.3 替换

当需要在当前编辑的文本中使用替换功能，打开替换窗口：

- 使用主菜单“编辑→替换”
- 使用快捷键“Ctrl+R”



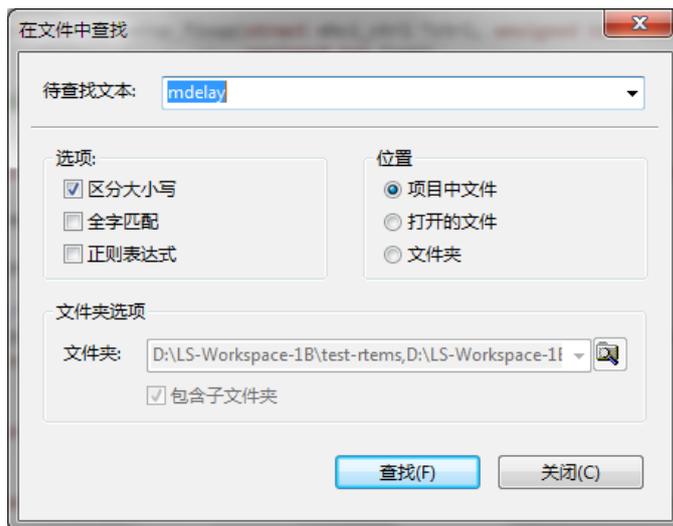
当打开替换窗口时，编辑器有选择的文本或者光标位置的语句自动复制到“待查找文本”文本框。输入“待查找文本”、“待替换文本”和选项等，单击【替换】按钮，编辑器光标定位到找到的第一个文本处执行并替换操作，同时按钮文本变为“下一个”。

替换下一个：

- 保持替换窗口打开，单击【下一个】按钮或者使用快捷键 <F3> ，编辑器光标定位到下一个找到的文本处并执行替换，或者提示未找到
- 关闭替换窗口，使用快捷键 <F3> ，编辑器光标定位到下一个找到的文本处并执行替换，或者提示未找到

## 6.2.4 在文件中查找

本功能实现在“项目中文件”、“打开的文件”、“文件夹”中查找指定文本。  
 使用主菜单“编辑→在文件中查找”，打开窗口如下：



输入“待查找文本”、选项等，单击【查找】按钮，查找结果显示在“消息窗口”中。

查找结果：

行	列	模块	语句
133	13	D:\LS-Workspace\bare-1c\lib\ls1c_pwm.c	switch (gpio)
140	44	D:\LS-Workspace\bare-1c\lib\ls1c_pwm.c	case LS1C_PWM0_GPIO04: // gpio04的第三复用
144	44	D:\LS-Workspace\bare-1c\lib\ls1c_pwm.c	case LS1C_PWM1_GPIO05: // gpio05的第三复用
148	44	D:\LS-Workspace\bare-1c\lib\ls1c_pwm.c	case LS1C_PWM2_GPIO52: // gpio52的第四复用
152	44	D:\LS-Workspace\bare-1c\lib\ls1c_pwm.c	case LS1C_PWM2_GPIO46: // gpio46的第四复用
156	44	D:\LS-Workspace\bare-1c\lib\ls1c_pwm.c	case LS1C_PWM3_GPIO47: // gpio47的第四复用
160	44	D:\LS-Workspace\bare-1c\lib\ls1c_pwm.c	case LS1C_PWM3_GPIO53: // gpio53的第四复用

操作：双击“查找结果”行，文本编辑器打开文件并将光标设置在指定的行和列。

警告：选择“包含子文件夹”的“文件夹”查找，限制查找结果 1000 条。使用本功能时，尽量缩小查找范围。



```

Main.c x led.c x tasks.h x
39 int LedShrink_start(void)
40 {
41     rtems_status_code rc;
42
43     /* create the task */
44     rc = rtems_task_create(rtems_build_name('L', 'E', 'D', '1'),
45                          100, // PRIORITY
46                          (8 * 1024), // Stack Size
47                          RTEMS_DEFAULT_MODES,
48                          RTEMS_DEFAULT_ATTRIBUTES,
49                          &LedShrink_taskid);

```

右键弹出菜单操作：

将鼠标光标移动到待查找语句处，打开右键菜单“查找定义”。

```

43     /* create the task */
44     rc = rtems_task_create(rtems_build_name('L', 'E', 'D', '1'),
45                          100, // PRIORITY
46                          (8 * 1024), // Stack Size
47                          RTEMS_DEFAULT_MODES,
48                          RTEMS_DEFAULT_ATTRIBUTES,
49                          &LedShrink_taskid);

```



执行结果：

在文本编辑器打开“语句”所在的文件，并将光标定位到该语句声明或者实现。

```

Main.c x led.c x tasks.h x
256 * id of the created task in ID.
257 */
258 rtems_status_code rtems_task_create(
259     rtems_name      name,
260     rtems_task_priority initial_priority,
261     size_t          stack_size,
262     rtems_mode      initial_modes,
263     rtems_attribute attribute_set,
264     rtems_id        *id
265 );
266

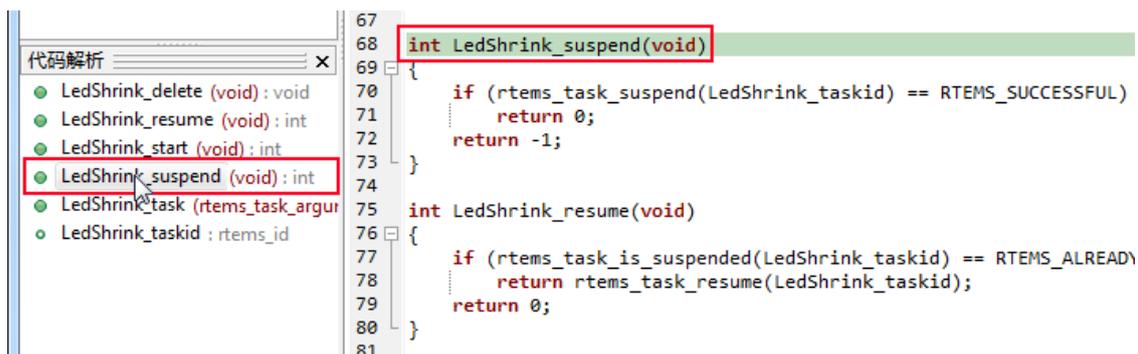
```

### 6.3.3 代码解析项跳转

文本编辑器中当前代码的实时解析结果显示在“代码解析”面板：

- 便于用户浏览当前代码的实现功能和架构
- 便于用户快速定位本代码中实现的某一功能模块

通过单击“代码解析”面板的显示项，快速定位到代码中语句的实现处：



```

67
68 int LedShrink_suspend(void)
69 {
70     if (rtems_task_suspend(LedShrink_taskid) == RTEMS_SUCCESSFUL)
71         return 0;
72     return -1;
73 }
74
75 int LedShrink_resume(void)
76 {
77     if (rtems_task_is_suspended(LedShrink_taskid) == RTEMS_ALREADY)
78         return rtems_task_resume(LedShrink_taskid);
79     return 0;
80 }
81

```

## 6.4 插入代码向导

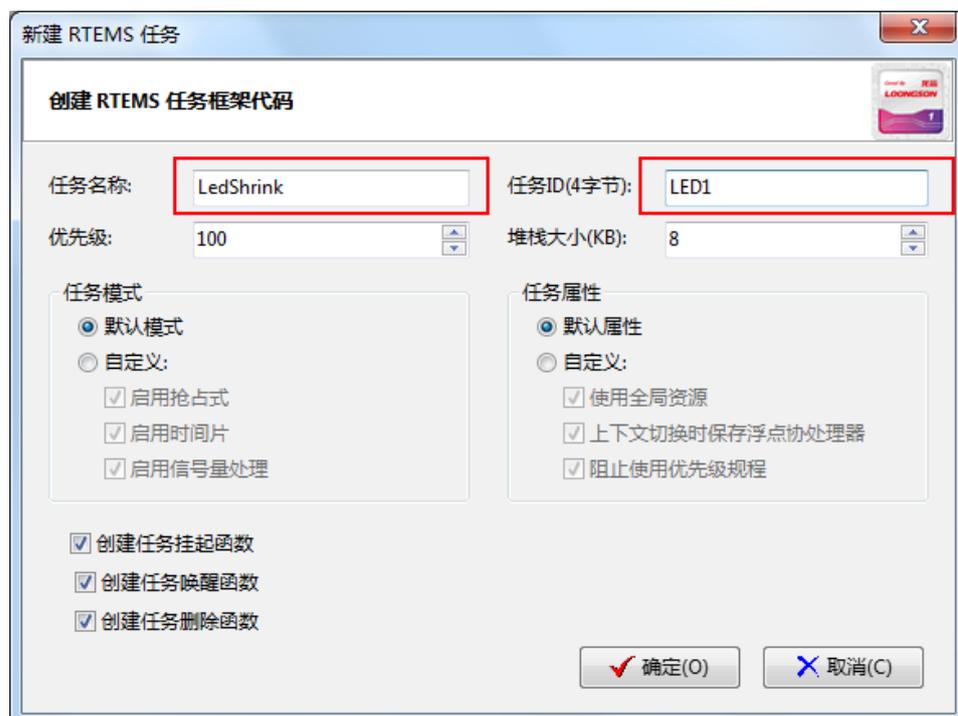
在当前源文件中插入基于 RTOS 的任务代码框架或者插入 SPI/I2C 设备驱动程序框架。生成的代码与当前项目选用的 RTOS 和 LS1x 芯片是一致的。

### 6.4.1 插入 RTOS 任务代码

仅当前项目基于 RTOS 时，本菜单可见。

- 使用快速工具栏按钮  的下拉菜单“插入 RTOS 任务代码框架”
- 使用文本编辑器右键弹出菜单“插入代码向导→插入 RTOS 任务代码框架”

打开窗口如下：



- 输入任务名称，确保全局唯一
- 输入任务 ID，确保全局唯一（RTEMS/RTThread 必须输入）
- 选择优先级 0~Max Priority（请参照 RTOS 的配置文件）
- 勾选任务模式、属性等选项

设置完成后单击【确定】按钮，生成代码如下所示：

```
Main.c x led.c x tasks.h x ls1b_spi0.S x
1  /*
2  * RTEMS Task for LedShrink, Auto Generated by Wizard.
3  * Created: 2019/7/6 16:19:41
4  */
5
6  #include <stdio.h>
7  #include <stdlib.h>
8  #include <errno.h>
9  #include <string.h>
10 #include <rtems.h>
11 #include <rtems/error.h>
12
13 #include <bsp.h>
14
15 static rtems_id LedShrink_taskid;
16
17 static rtems_task LedShrink_task(rtems_task_argument arg)
18 {
19     /*
20     * Add LedShrink initialize code here.
21     */
22
23     for ( ; ; )
24     {
25         /*
26         * Add LedShrink task code here.
27         */
28
29         // ...
30
31         /* abandon cpu time to run other task */
32         rtems_task_wake_after(10); // task sleep 10 ms
33         //rtems_task_wake_after(RTEMS_YIELD_PROCESSOR); // task sleep until cpu idle
34         //rtems_task_wake_when(When); // task sleep to time When
35
36     }
```

用户在“LedShrink\_task”任务函数内，用户加入自己的业务逻辑。

注：RTEMS 多任务的另一个实现机制是使用 pthread 库编程

## 6.4.2 插入 SPI/I2C 驱动代码

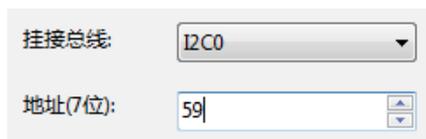
本向导用于为龙芯 1B200/1C300 的 SPI/I2C 总线上挂接的**从设备**生成驱动框架。

- 使用快速工具栏按钮  的下拉菜单“插入 LS1x 驱动代码框架”
- 使用文本编辑器右键弹出菜单“插入代码向导→插入 LS1x 驱动代码框架”

打开窗口如下：



如果挂接 I2C 设备，需要输入芯片的 7 位 I2C 地址值：



正确输入设备参数和勾选需要创建的函数后，单击【确定】按钮，生成代码如下：

```

Main.c x spi0_mcp3201.c x
18 /* mcp3201 Chip Select */
19 #define MCP3201_CS 1
20
21 /*
22 * mcp3201 transfer parameter
23 */
24 static rtems_libi2c_tfr_mode_t mcp3201_tfr_mode =
25 {
26     baudrate:      10000000,    // maximum bits per second
27     bits_per_char: 8,          // how many bits per byte/word/longword?
28     lsb_first:     FALSE,      // FALSE: send MSB first
29     clock_pha:     TRUE,       // clock phase - spi mode
30     clock_pol:     TRUE,       // clock polarity - spi mode
31     clock_inv:     TRUE,       // TRUE: inverted clock (low active)
32     clock_phs:     FALSE,     // FALSE: clock starts in middle of data tfr
33     idle_char:     0,
34 };
35
36 /*****
37 * Add some mcp3201 hardware implement here.
38 */
39 // ...
40
41 /*****
42 * mcp3201 driver implement
43 */
44 /*
45 * Purpose: initialize the mcp3201
46 */
47
48 static rtems_status_code SPI0_mcp3201_initialize(
49     rtems_device_major_number major,
50     rtems_device_minor_number minor,
51     void *arg)
52 {
53

```

注意：

- 用户需要根据芯片手册，完善\_initialize/\_open/\_close/\_ioctl/\_read/\_write 函数的实现
- LS1x SPI/I2C 的驱动程序机制：一级：总线驱动；二级：设备驱动。

## 6.5 信息提示

无

## 7、项目编译

通过编译，排除用户项目的语法错误和生成 MIPS ELF 格式的可执行程序（扩展名.exe）。

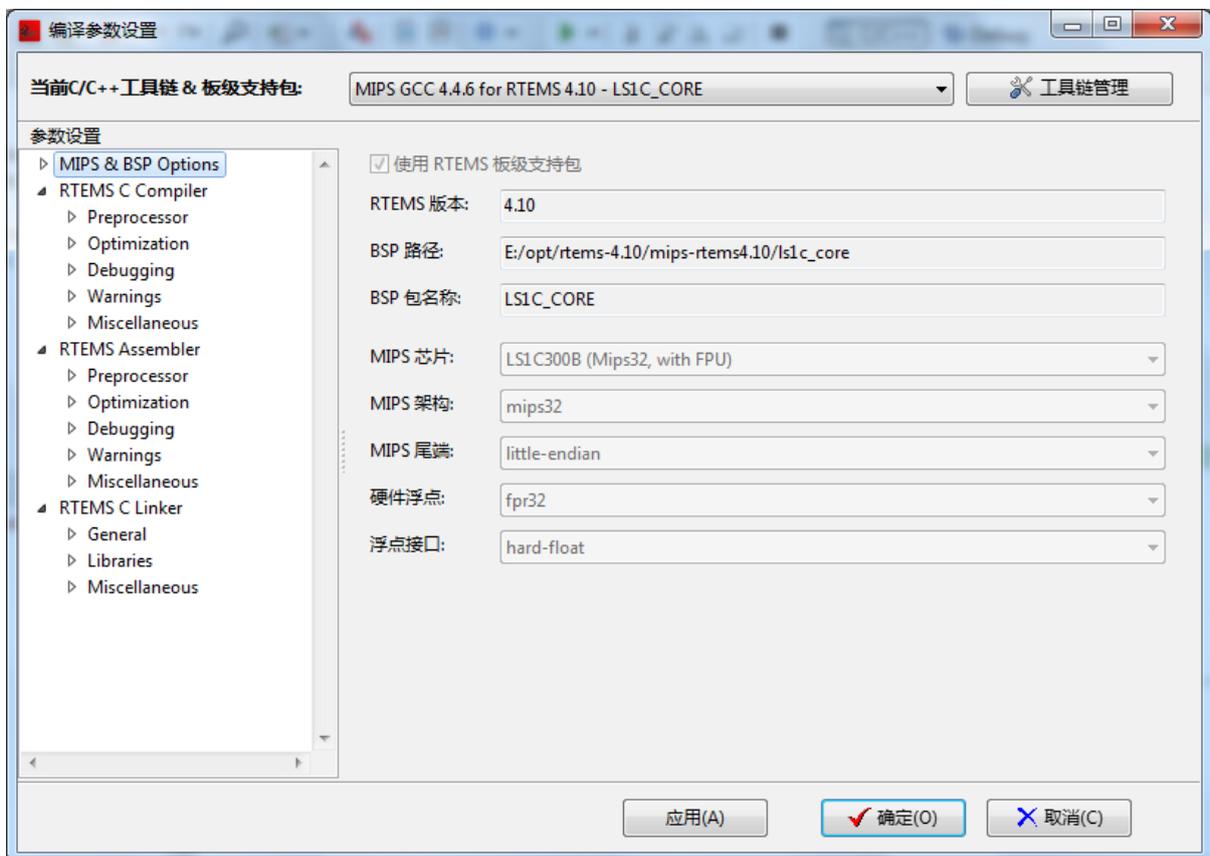
### 7.1 编译选项

编译选项将决定 gcc 编译器的行为，对最终生成的可执行代码有决定性影响。

当有项目打开时：

- 使用快捷键 F2
- 使用主菜单“项目→编译选项”
- 使用项目视图面板右键弹出菜单“编译选项”

打开窗口如下：



当前工具链和板级支持包：

与“项目属性”的“工具链&目标板”更改相同。对于确定的目标板创建应用项目后，不建议修改工具链。

**【工具链管理】按钮：**

用于打开工具链管理窗口以维护 LoongIDE 的工具链。

参数设置:

- MIPS & BSP Options
- GNU C Compiler
- GNU Assembler
- GNU C++ Compiler (Only C++ Project)
- GNU C Linker (Only C Project)
- GNU C++ Linker (Only C++ Project)

### 7.1.1 MIPS & BSP Options

单击窗口左侧“参数设置”下“MIPS & BSP Options”项目，右侧面板显示:

MIPS GCC 4.4.6 for RTEMS 4.10

使用 RTEMS 板级支持包 无BSP包工具链

RTEMS 版本: 4.10

BSP 路径:

BSP 包名称:

可选

MIPS 芯片: LS1C300B (Mips32, with FPU)

MIPS 架构: mips32

MIPS 尾端: little-endian

硬件浮点: (none)

浮点接口: soft-float

设置:

- 如果当前项目使用 RTEMS，所有选项不可修改
- 如果当前项目没有使用 RTEMS，可以修改的项目如上图所示

*MIPS 尾端、硬件浮点和浮点接口，由“MIPS 芯片”的硬件决定，不可更改。*

### 7.1.2 GNU C Compiler - C 编译器

编译用户项目 .c 源代码文件使用的命令和命令行编译参数。

命令行: E:\opt\rtms-4.10\bin\mips-rtms4.10-gcc.exe

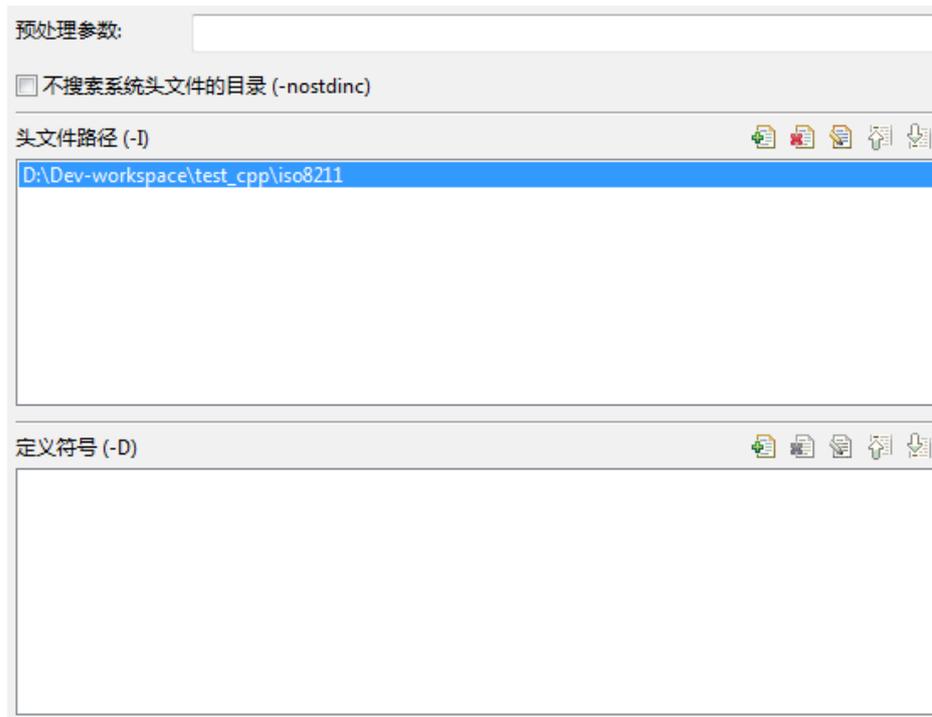
所有选项: -B"E:/opt/rtms-4.10/mips-rtms4.10/ls1c\_core/lib" -specs bsp\_specs -qrtems -mips32 -G0 -EL -mhard-float -O0 -g -Wall -c -fmessage-length=0 -pipe

所有选项由以下五个子项合并生成:

- Preprocessor
- Optimization

- Debugging
- Warnings
- Miscellaneous

### 7.1.2.1 Preprocessor - 预处理



#### 预处理参数

用户自定义输入特定参数

#### 不搜索系统头文件目录 (-nostdinc)

裸机编程时使用该选项

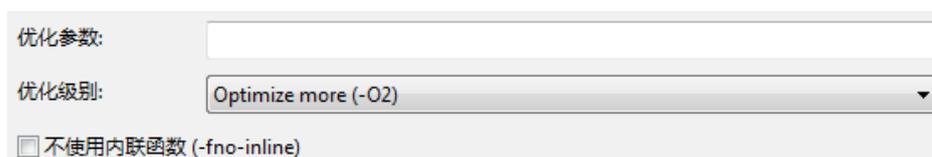
#### 头文件路径 (-I)

gcc 命令行使用的头文件搜索路径

#### 定义符号 (-D)

供编译器命令行使用的宏定义，使用“符号”或者“符号=值”的格式

### 7.1.2.2 Optimzatin - 优化



### 优化参数

用户自定义输入的参数

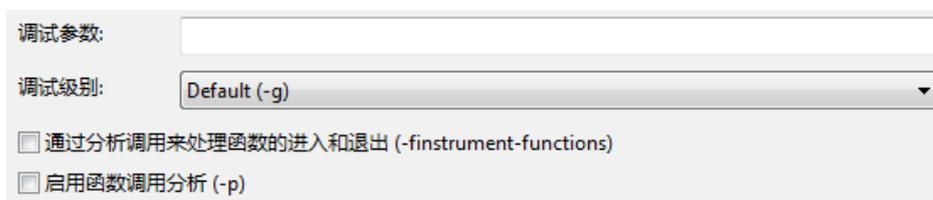
### 优化级别

建议设置：代码调试时使用“-O0”，在生成最终文件时使用“-O2”选项

### 不使用内联函数（-fno-inline）

编译时 inline 类型的函数的代码生成方式

## 7.1.2.3 Debugging - 调试



调试参数:

调试级别: Default (-g)

通过分析调用来处理函数的进入和退出 (-finstrument-functions)

启用函数调用分析 (-p)

### 调试参数

用户自定义输入的参数

### 调试级别

建议设置：Default (-g)，在应用程序中生成调试信息；

应用程序发布时，内含的调试信息可以通过 mips-xxx-strip.exe 清除

### 通过分析调用来处理函数的进入和退出（finstrument-functions）

供性能分析使用，未使用

### 启用函数调用分析（-p）

供性能分析使用，未使用

## 7.1.2.4 Warnings - 警告



警告参数:

输出默认警告 (-Wall)

输出(可能不希望的)额外警告 (-Wextra)

警告作为错误处理 (-Werror)

发出严格遵守标准所需的警告 (-pedantic)

类似 -pedantic 但视为错误处理 (-pedantic-errors)

检查语法错误, 然后停止 (-fsyntax-only)

禁止显示警告 (-w)

## 警告参数

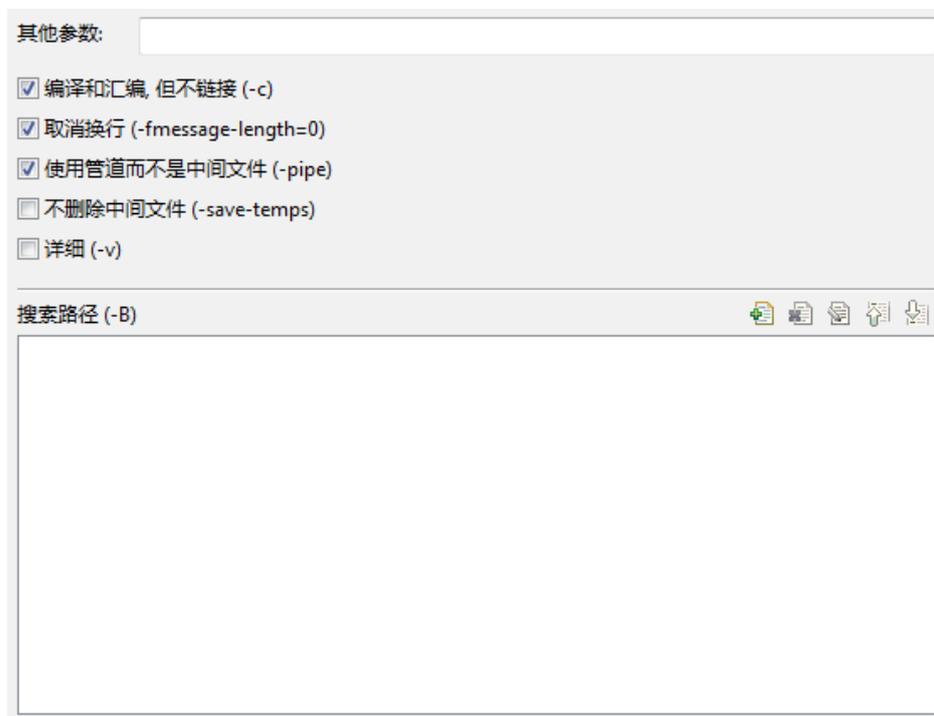
用户自定义输入的参数

## 警告选项

用户根据需要进行勾选

- -Wall: 输出默认警告
- -Wextra: 输出(可能不希望的)额外警告
- -Werror: 警告作为错误处理
- -pedantic: 发出严格遵守标准所需的警告
- -pedantic-errors: 类似-pedantic 但视为错误处理
- -fsyntax-only: 检查语法错误, 然后停止
- -w: 禁止显示警告

## 7.1.2.5 Miscellaneous - 杂项



## 其它参数

用户自定义输入的参数

## 选项

用户根据需要进行勾选

- -c: 编译和汇编, 但不链接
- -fmessage-length=0: 取消换行
- -pipe: 使用管道而不是中间文件
- -save-temps: 不删除中间文件
- -v: 详细

## 搜索路径 (-B)

gcc 命令行使用的二进制文件搜索路径

### 7.1.3 GNU Assembler - 汇编语言编译器

编译用户项目 .S 汇编源代码文件使用的命令和命令行编译参数。  
参数设置与“GNU C Compiler”相同。

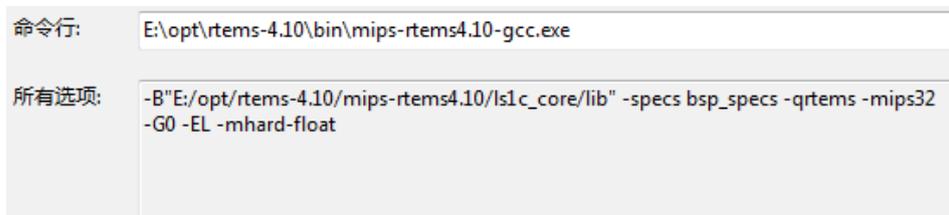
### 7.1.4 GNU C++ Compiler - C++ 编译器

编译用户项目 .cpp 源代码文件使用的命令和命令行编译参数。  
参数设置与“GNU C Compiler”相同。

仅用于 C++ 项目。

### 7.1.5 GNU C Linker - C 链接器

项目的源代码文件经编译生成 .o 文件，通过本命令链接生成库文件(.a)或者可执行文件(.exe)。

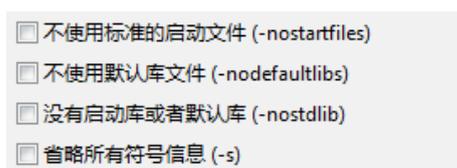


所有选项由以下三个子项合并生成:

- General
- Library
- Miscellaneous

对于 C++ 项目，标题为：GNU C++ Linker

#### 7.1.5.1 General - 常用

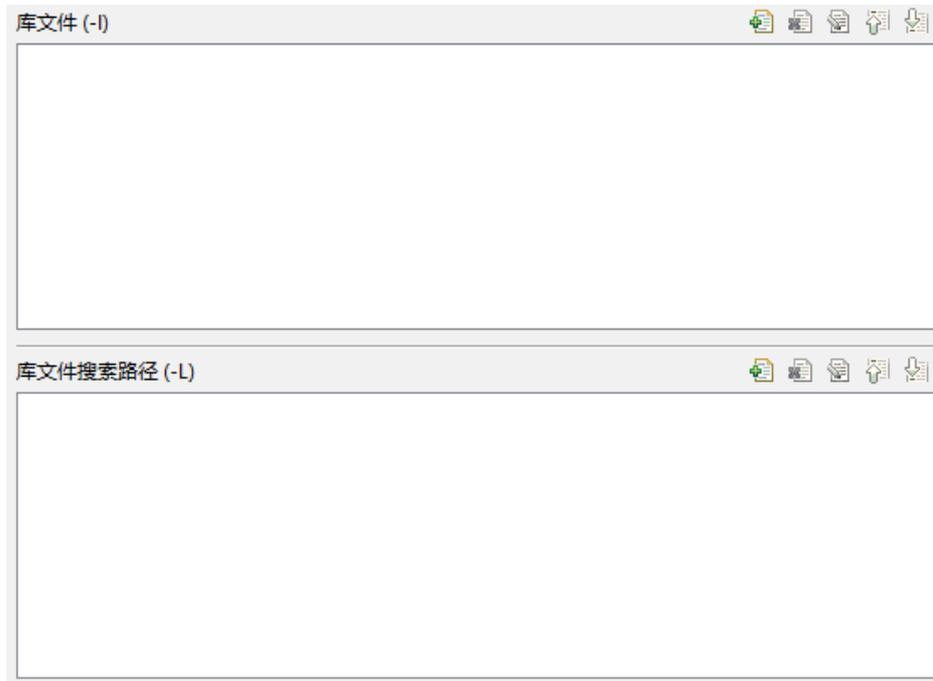


#### 选项

用户根据需要进行勾选

- -nostartfiles: 不使用标准的启动文件
- -nodefaultlibs: 不使用默认库文件
- -nostdlib: 没有启动库或者默认库
- -s: 省略所有符号信息

### 7.1.5.2 Library - 库



#### 库文件 (-l)

链接时使用的库文件名称，供链接器命令行使用

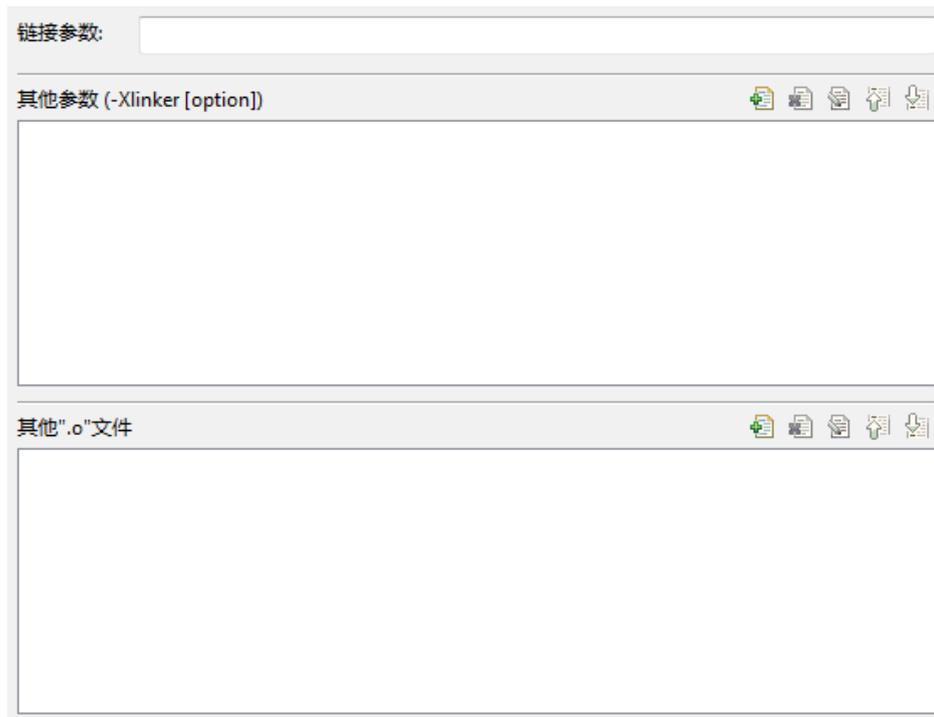
注意名称的不同：库文件 `yaffs`，实际磁盘文件名称为：`libyaffs.a`

#### 库文件搜索路径 (-L)

库文件所在的文件夹位置

系统默认的库文件如数学库 `m`，不需要加入搜索路径

### 7.1.5.3 Miscellaneous - 杂项



#### 链接参数

用户自定义输入的参数

#### 其他参数 (-Xlinker [option])

将 ld 选项传递给链接器；当传递的 ld 选项包含选项名和参数时，需要使用两次 -Xlinker。

例如： -Xlinker -assert -Xlinker definitions

可写为： -Xlinker -assert=definitions

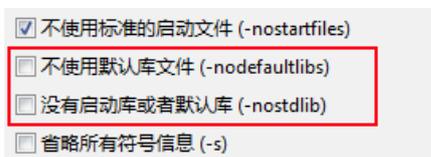
#### 其他 ".o" 文件

链接器用到的项目以外的 .o 文件。

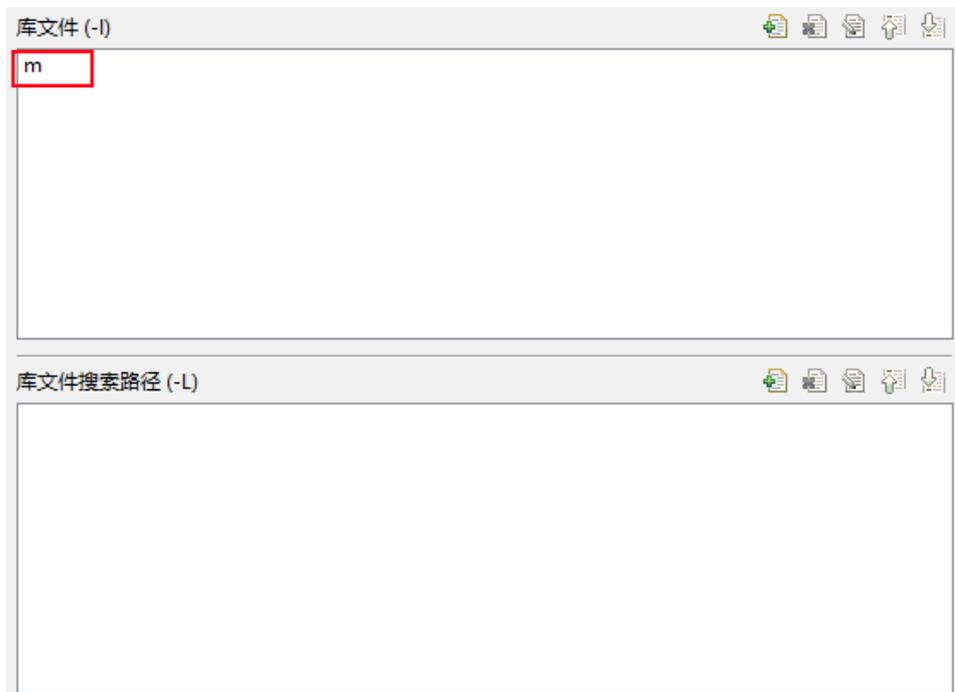
### 7.1.6 软浮点算术库

对于“裸机编程项目”，Linker 的 General 默认勾选前三个选项：-nostartfiles、-nodefaultlibs、-nostdlib；当龙芯 1x 项目使用 -msoft-float 编译选项、但项目用到浮点运算时，必须链接“软浮点算术库”(m)，需要去掉 -nodefaultlibs、-nostdlib 的勾选。

#### Linker 的 General 选项



加入软浮点算术库 m



注：使用 RTC 驱动程序时，要加入 c 库。

## 7.2 开始编译

编译是用户在项目开发过程中常用的操作之一，用户编写的代码通过编译来修正语法错误、生成最终的二进制代码。

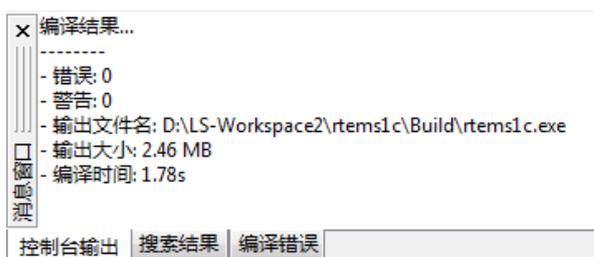
通过以下操作，启动用户项目的编译：

- 使用快捷键 **Ctrl+F9**
- 使用主菜单“项目→编译”
- 使用工具栏  按钮
- 使用项目视图面板弹出菜单“编译”

### 编译结果

- 项目编译成功，在“消息面板”的“控制台输出”可以看到生成文件的信息
- 如果项目编译失败，在“消息面板”的“编译错误”可以看到详细的语法错误信息

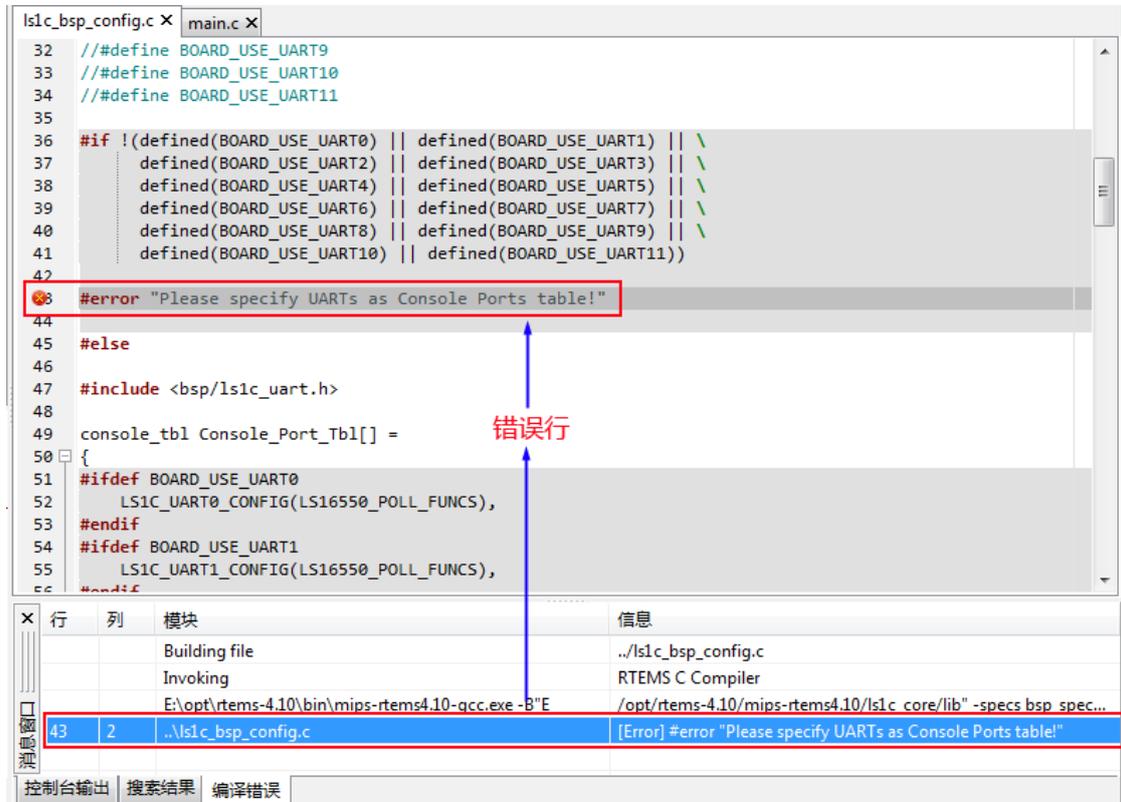
#### 7.2.1 编译成功



项目只有在编译成功后，才可以开始调试。

## 7.2.2 编译失败

“消息窗口”自动切换到“编译错误”面板，详细列出了代码的语法错误。



```

32 //define BOARD_USE_UART9
33 //define BOARD_USE_UART10
34 //define BOARD_USE_UART11
35
36 #if !(defined(BOARD_USE_UART0) || defined(BOARD_USE_UART1) || \
37 defined(BOARD_USE_UART2) || defined(BOARD_USE_UART3) || \
38 defined(BOARD_USE_UART4) || defined(BOARD_USE_UART5) || \
39 defined(BOARD_USE_UART6) || defined(BOARD_USE_UART7) || \
40 defined(BOARD_USE_UART8) || defined(BOARD_USE_UART9) || \
41 defined(BOARD_USE_UART10) || defined(BOARD_USE_UART11))
42
43 #error "Please specify UARTs as Console Ports table!"
44
45 #else
46
47 #include <bsp/ls1c_uart.h>
48
49 console_tbl Console_Port_Tbl[] =
50 {
51 #ifdef BOARD_USE_UART0
52 LS1C_UART0_CONFIG(LS16550_POLL_FUNCES),
53 #endif
54 #ifdef BOARD_USE_UART1
55 LS1C_UART1_CONFIG(LS16550_POLL_FUNCES),
56 #endif
57 #ifdef BOARD_USE_UART2
58 LS1C_UART2_CONFIG(LS16550_POLL_FUNCES),
59 #endif
60 #ifdef BOARD_USE_UART3
61 LS1C_UART3_CONFIG(LS16550_POLL_FUNCES),
62 #endif
63 #ifdef BOARD_USE_UART4
64 LS1C_UART4_CONFIG(LS16550_POLL_FUNCES),
65 #endif
66 #ifdef BOARD_USE_UART5
67 LS1C_UART5_CONFIG(LS16550_POLL_FUNCES),
68 #endif
69 #ifdef BOARD_USE_UART6
70 LS1C_UART6_CONFIG(LS16550_POLL_FUNCES),
71 #endif
72 #ifdef BOARD_USE_UART7
73 LS1C_UART7_CONFIG(LS16550_POLL_FUNCES),
74 #endif
75 #ifdef BOARD_USE_UART8
76 LS1C_UART8_CONFIG(LS16550_POLL_FUNCES),
77 #endif
78 #ifdef BOARD_USE_UART9
79 LS1C_UART9_CONFIG(LS16550_POLL_FUNCES),
80 #endif
81 #ifdef BOARD_USE_UART10
82 LS1C_UART10_CONFIG(LS16550_POLL_FUNCES),
83 #endif
84 #ifdef BOARD_USE_UART11
85 LS1C_UART11_CONFIG(LS16550_POLL_FUNCES),
86 #endif
87 #endif
88 };

```

行	列	模块	信息
		Building file	../ls1c_bsp_config.c
		Invoking	RTEMS C Compiler
		E:\opt\rtems-4.10\bin\mips-rtems4.10-gcc.exe -B"E	/opt/rtems-4.10/mips-rtems4.10/ls1c_core/lib" -specs bsp_spec...
43	2	..\ls1c_bsp_config.c	[Error] #error "Please specify UARTs as Console Ports table!"

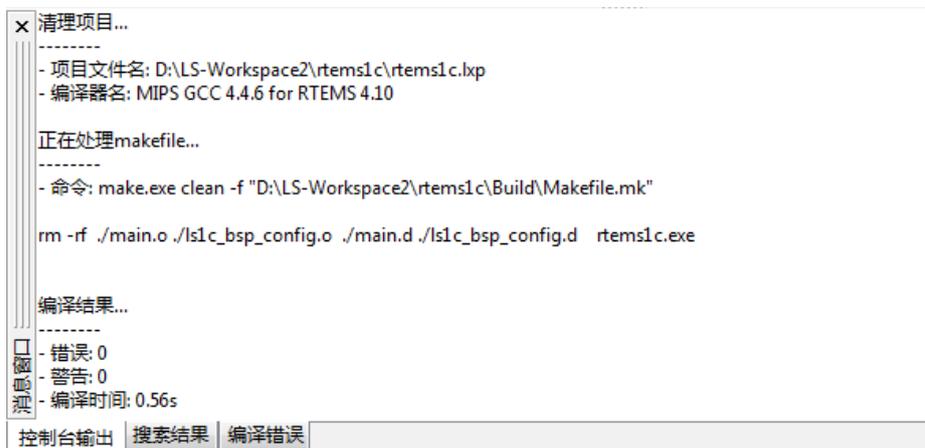
### 操作

用鼠标双击“编译错误”的错误行，文本编辑器打开文件并将光标定位到错误行和列的位置

## 7.3 项目清理

用于清理编译生成的二进制文件等编译生成的文件。

- 使用主菜单“项目→清理”
- 使用工具栏 按钮下拉菜单“清理”



```

清理项目...
-----
- 项目文件名: D:\LS-Workspace2\rtems1c\rtems1c.lxp
- 编译器名: MIPS GCC 4.4.6 for RTEMS 4.10

正在处理makefile...
-----
- 命令: make.exe clean -f "D:\LS-Workspace2\rtems1c\Build\Makefile.mk"

rm -rf ./main.o ./ls1c_bsp_config.o ./main.d ./ls1c_bsp_config.d rtems1c.exe

编译结果...
-----
- 错误: 0
- 警告: 0
- 编译时间: 0.56s

```

按住 **Ctrl** 键执行“清理”操作，将删除“Build”目录等全部编译结果。

## 8、项目调试

用户项目在编译成功后（使用“-g”参数编译），通过调试实现程序的逻辑排错。

### 8.1 调试选项

在启动调试前，进行调试参数的设置。通过以下操作，打开调试选项窗口：

- 使用快捷键“F4”
- 使用主菜单“调试→调试选项”
- 使用工具栏按钮  下拉菜单“调试选项”
- 使用“项目视图”面板右键弹出菜单“调试选项”

#### 选项面板

- 主要项
- 调试器
- 启动项
- 源代码

#### 8.1.1 主要项



#### C/C++ 应用程序

当前将调试的项目应用程序。

## 重新编译选项

在启动调试前，检查项目是否更改，从而决定是否重新编译

## 目标板运行引导加载程序 PMON

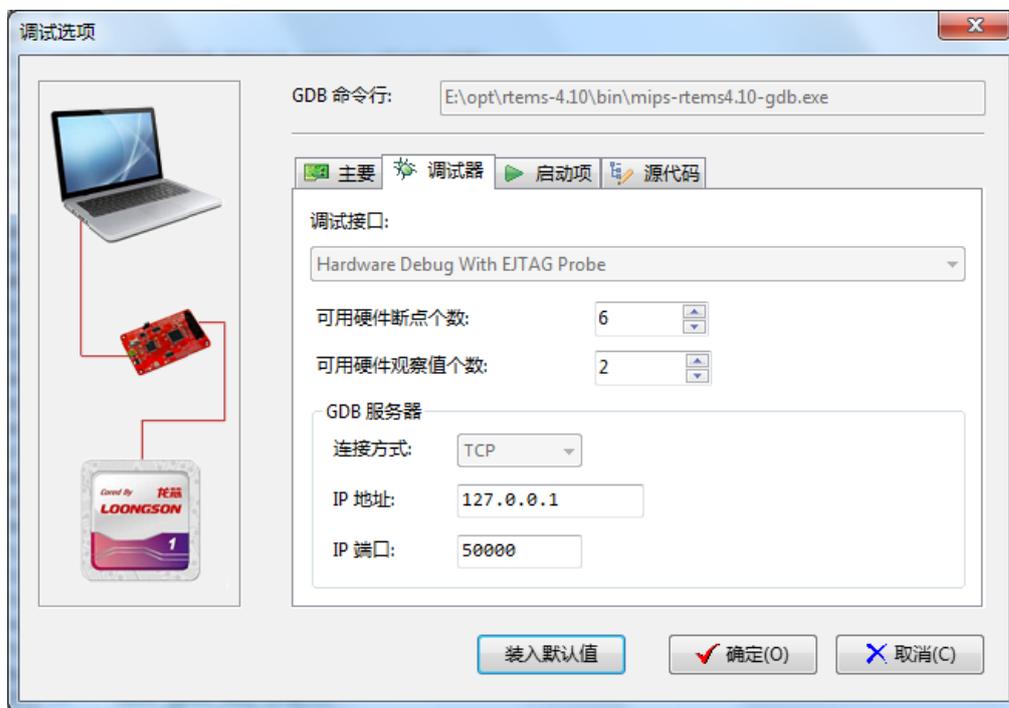
如果没有运行 PMON，在启动调试前，LoongIDE 将对目标板的 PLL 和 RAM 进行初始化；否则，不进行初始化

## 通过 PMON 的以太网，快速下载应用程序到目标板

LoongIDE 在启动调试时，通过 PMON 的 TCP/IP 将被调试程序下载到目标板，以提高速度，减少用户等待 LxLink 下载的时间

*注：PMON 引导程序是经过改进的，支持通过 TCP/IP 进行命令交互。*

## 8.1.2 调试器



### 调试接口

通过 LxLink 设备连接 EJTAG 接口

### 可用硬件断点个数

### 硬件监视变量个数

龙芯 1x 集成的 EJTAG 设备的硬件特性

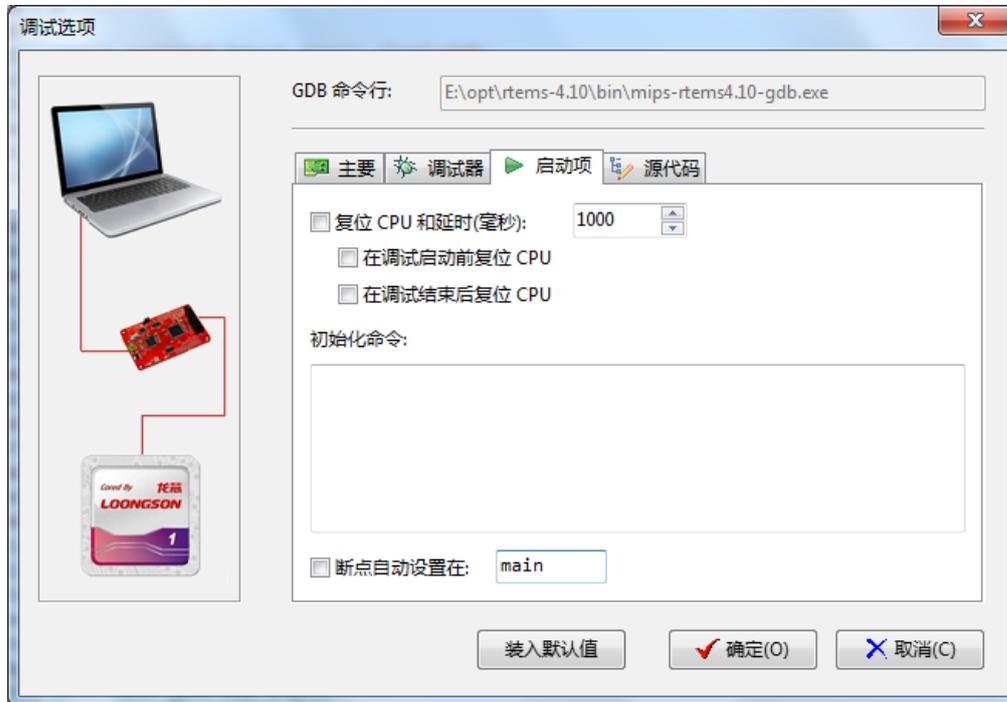
### GDB 服务器

LoongIDE 系统内建 GDB Server，调试工具软件 gdb.exe 与此 GDB Server 进行数据交互实现调试。

- 连接方式：TCP

- IP 地址: 本机 IP 地址
- IP 端口: 本机空闲端口

### 8.1.3 启动项



#### 复位 CPU 和延时

通过 MIPS 标准 14 脚 EJTAG 接口的 SYS RESET 信号线，实现龙芯 1x 芯片的硬件复位，并在复位后延时。

#### 选项

- 在调试启动前复位 CPU
  - 在调试结束后复位 CPU
- 提供给用户一个执行复位的有利于调试操作的时机。

#### 初始化命令

用户输入的用于初始化 gdb.exe 的命令

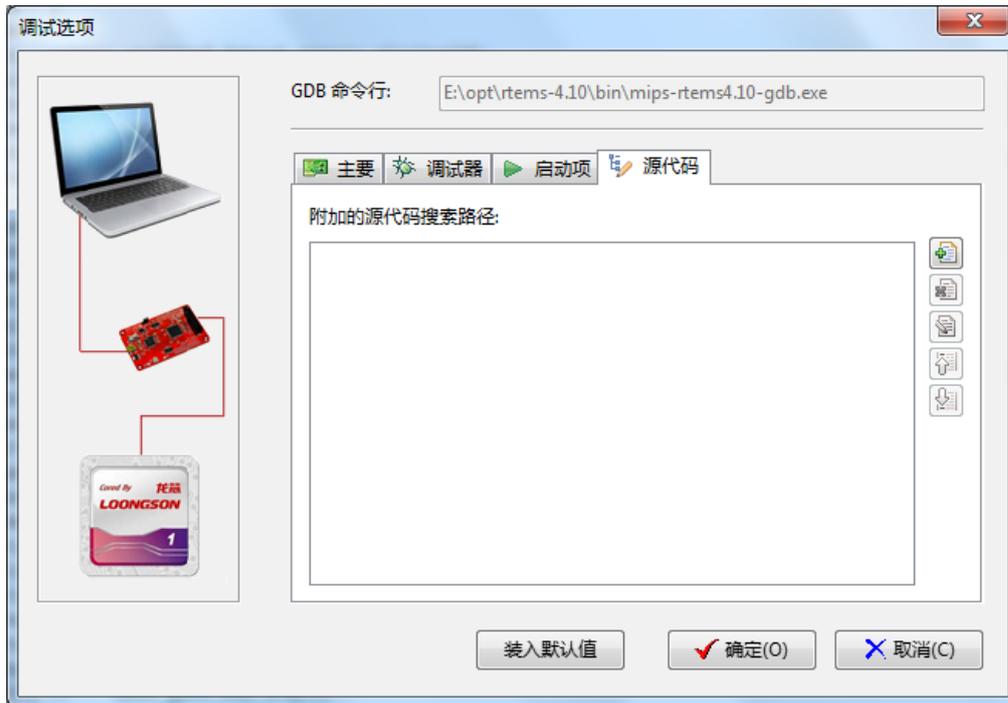
#### 断点自动设置在

输入一个函数名称，调试启动后，断点自动设置在该函数的第一条语句处。

#### 默认设置:

- RTEMS 项目，设置为 Init
- 裸机编程项目，设置为 main

## 8.1.4 源代码



如果应用程序链接时使用了项目文件以外的 .o 目标文件和 .a 库文件，并且是使用 -g 参数编译的，在此处增加对应 .o 目标文件和 .a 库文件的源代码目录，用于调试时自动搜索。

## 8.2 调试断点

### 8.2.1 在编辑器中设置断点

- Click: 单击文本编辑器源代码行的行号处，执行增加断点/移除断点操作
- Right-Click: 在文本编辑器的源代码行上，使用右键菜单的“反转断点设置”，执行增加断点/移除断点操作

```

9  #include <bsp.h>
10
11  /*
12   * main function
13   */
14  rtems_task Init(rtems_task_argument ignored)
15  {
16      printf( "\n\n*** HELLO WORLD TEST ***\n" );
17      printf( "Hello World\n" );
18      printf( "**** END OF HELLO WORLD TEST ***\n" );
19
20      exit(0);
21  }
22
23  /*
24   * Cut the OS here.
25   */
26  #define CONFIGURE_APPLICATION_NEEDS_CONSOLE_DRIVER
27  #define CONFIGURE_APPLICATION_NEEDS_CLOCK_DRIVER

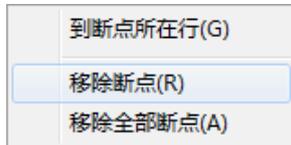
```

设置好的断点，在“断点列表”面板显示。

## 8.2.2 断点列表

在“断点列表”面板，显示当前用户设置的全部断点。

“断点列表”面板的右键弹出菜单：



### 操作

使用文本编辑器右键弹出菜单“反转断点设置”或者上图弹出菜单项，实现断点设置/移除操作

### 定位

- 双击断点列表行，文本编辑器将光标定位到断点所在源代码的行
- 使用上图弹出菜单项“到断点所在行”实现定位操作

## 8.3 开始调试

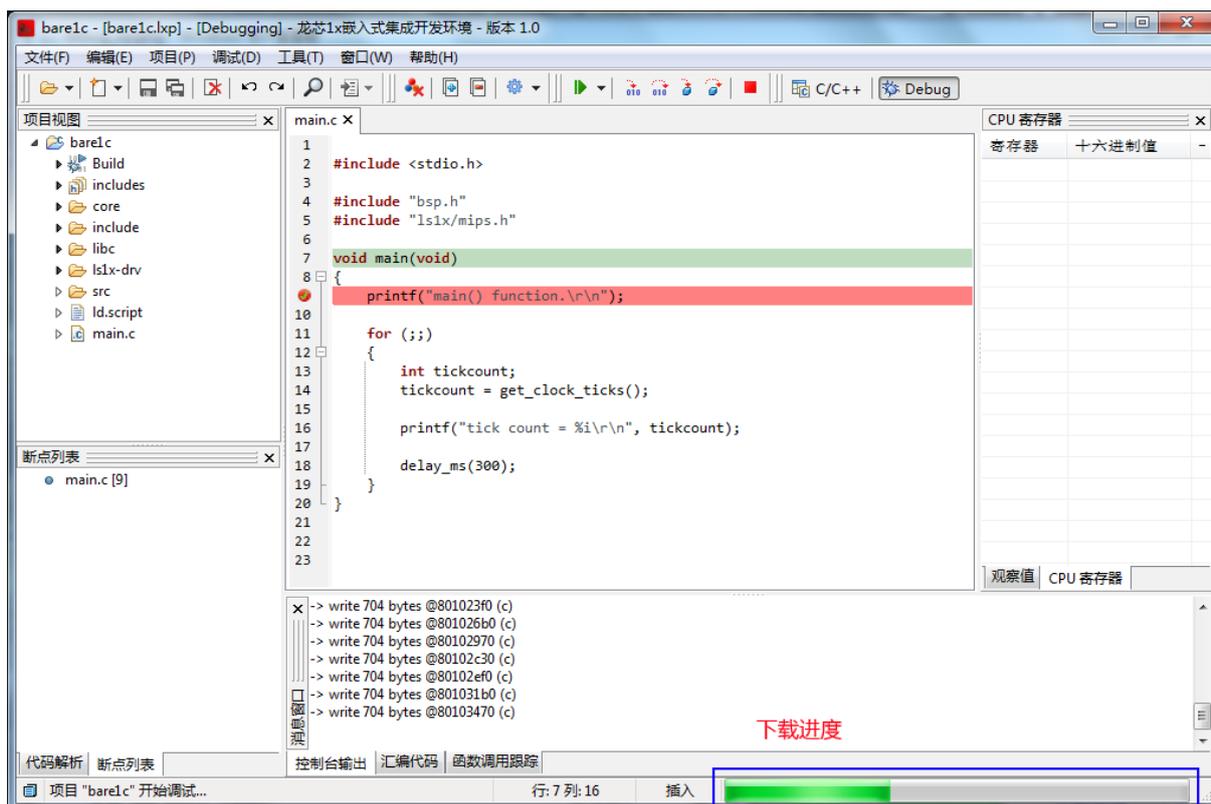
执行以下操作，开始调试：

- 使用快捷键“F9”
- 使用主菜单“调试→运行”
- 使用工具栏  按钮
- 使用项目视图面板右键弹出菜单“运行”

### 8.3.1 代码下载

用户界面切换到“调试”状态，LoongIDE 将项目应用程序下载到目标板。

- 如果目标板运行 PMON，网络连接后，通过 TCP/IP 实现快速下载；
- 如果没有网络连接，使用 LxLink 通过 EJTAG 接口下载代码。



状态栏上显示项目下载进度，等待下载完成...

### 8.3.2 单步运行

下载完成后，正式开始应用程序调试。调试器初始断点的位置有二种情况：

- 如果在调试选项的启动项，设置了“断点自动设置在”选项：

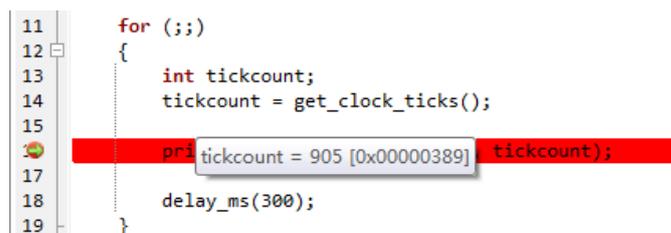
断点自动设置在:

则初始断点自动设置在该函数的第一条可执行语句处；

- 否则，初始断点设置在项目应用程序的第一条语句处，一般是 `start.S` 的第一条指令处。

在调试断点处，用户可以执行的操作

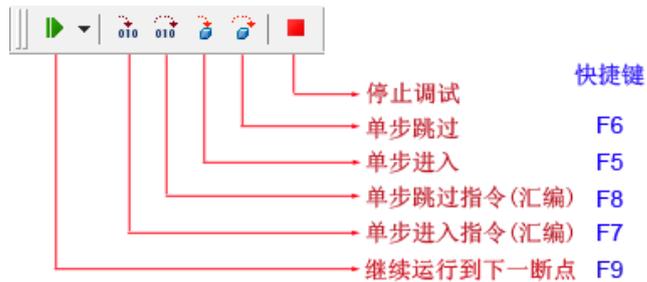
- 断点设置
- 设置观察值
- 实时查看变量值



鼠标所在位置的变量值提示

### 继续调试操作

- 使用主菜单“项目”的菜单项
- 使用工具栏“调试”按钮



- 使用快捷鍵：見上圖

### 8.3.3 連續運行

當被調試程序運行到斷點處：

- 使用工具欄按鈕 
- 使用快捷鍵 F9

程序將繼續運行，直到下一個斷點處停止。

### 8.3.4 停止調試

不管被調試程序處於何種狀態，单击工具欄按鈕  將結束本次調試，用戶界面自動切換到“編輯”狀態。

如果在調試“啟動項”選擇了“在調試結束後”復位 CPU，此時龍芯 1x 將進入復位操作。

### 8.3.5 觀察值

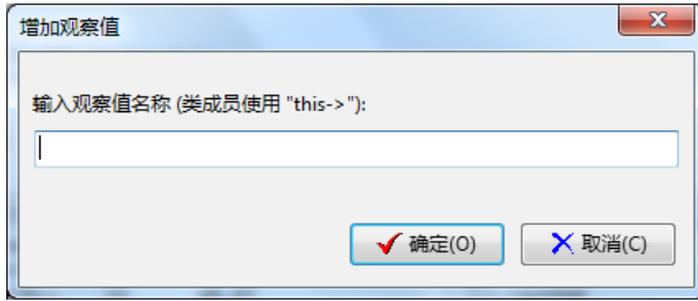
在“觀察值”面板顯示用戶設置的 Watch Var，每次調試到達斷點時更新。

“觀察值”面板的右鍵彈出菜單：



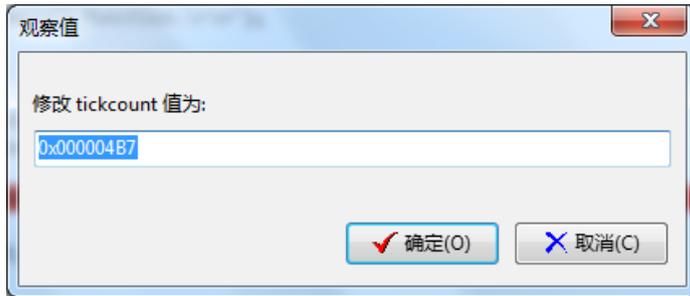
#### 增加

使用文本編輯器右鍵彈出菜單“增加觀察值”或者上圖彈出菜單項，顯示如下：



新增观察值名称在“观察值”列表中显示。

## 编辑



单击【确定】按钮，修改后的变量值将赋值给被调试程序。

## 8.3.6 函数调用回溯

“函数调用回溯”用于显示当前断点所在函数的调用层次关系，用于用户分析代码。

函数	文件	行
Console_output_char(pUART = 0x80...	../ls1x-drv/uart/ns16550.c	715
console_putch(ch = 116 't')	../ls1x-drv/console/console.c	56
_putc(ch = 116 't', pca = 0x0)	../libc/libc.c	762
_doprnt(fmt0 = 0x80108924 "tick c...	../libc/libc.c	377
printf(fmt = 0x80108924 "tick count ...	../libc/libc.c	770
main()	../main.c	16

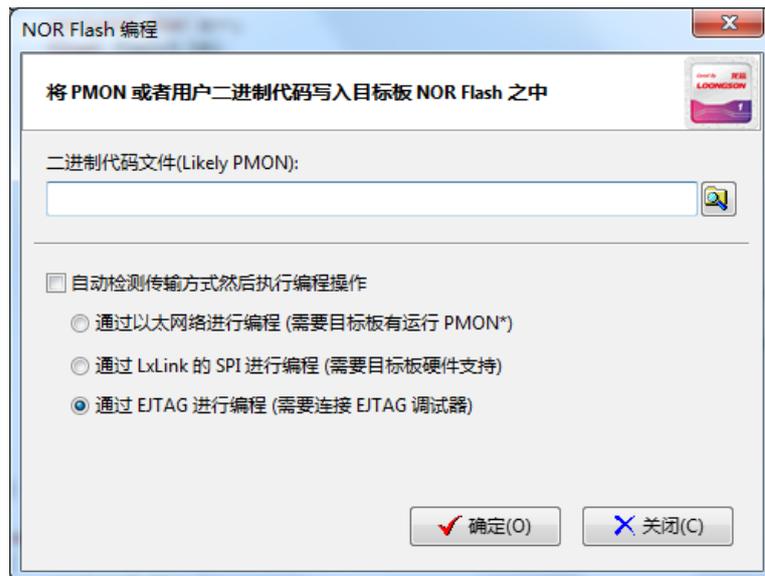
控制台输出 | 汇编代码 | **函数调用回溯**

## 9、实用工具

### 9.1 NOR Flash 编程

编程龙芯 1B/1C 的外部 NOR-Flash，或者 1c101/1J 的内部 Flash，用于存储 bootloader 或者用户的自启动应用程序。

使用主菜单“工具→NOR Flash 编程”，打开窗口如下：



打开 .exe、.elf、.bin 扩展名的 MIPS 可执行程序，根据目标板的配置情况，选择写入方式。

单击【确定】按钮，LoongIDE 将 .exe、.elf 文件转换为二进制格式的 .bin 文件进行写入操作。

#### 通过以太网进行编程

目标板当前运行 PMON\*，一般用于 PMON 升级。

连接主机和目标板的网络线，LoongIDE 使用 PMON 内置 TCP/IP 功能接收文件到内存然后开始编程。

#### 通过 LxLink 的 SPI 进行编程

使用 LxLink 设备，直接连接目标板 NOR-Flash 的 SPI 接口(需要目标板硬件电路设计支持)，实现将 MIPS 二进制文件写入 NOR-Flash。

该编程方式对制造的龙芯 1x 板卡进行初次编程，是速度最快最有效的编程方式。

#### 通过 EJTAG 进行编程

使用 LxLink 设备，连接目标板的 EJTAG 调试接口，实现将 MIPS 二进制文件写入 NOR-Flash。

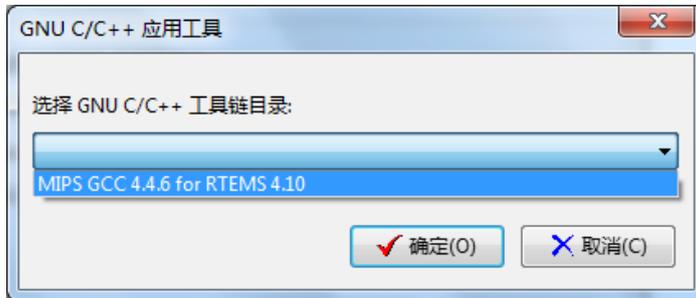
因为 EJTAG 执行指令速度的限制，本编程方法总体速度较慢。

## 使用 GNU 工具进行文件格式转换

LoongIDE 在对 NOR-Flash 进行编程前，需要将 .exe、.elf 转换为二进制的 .bin 文件时，转换操作通过调用工具链的 xxx\_strip.exe 和 xxx\_objcopy.exe 工具软件来实现。

LoongIDE 自动检查可执行文件是否来自于某个龙芯 1x 项目：

- 如果是，使用该项目编译使用的工具链的工具软件；
- 如果不是，提示用户在 LoongIDE 配置的工具链中选择；



- 如果上一步没有选择，提示用户输入工具链所在目录。

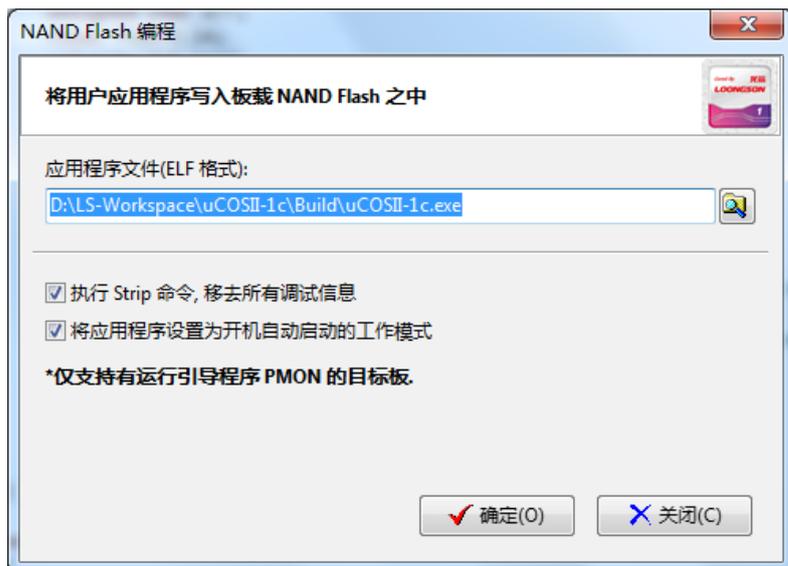


## 9.2 NAND Flash 编程

NAND Flash 用于存储用户应用程序，通过 bootloader 装载运行。

LoongIDE 实现的 NAND Flash 编程方法，基于龙芯 1x 目标板已经运行 PMON 引导程序（一般在 NOR Flash 中），使用 PMON 内置 TCP/IP 接收文件到内存，然后将用户应用程序保存到 PMON 的 /dev/mtd0，并设置是否让 PMON 自动装载。

使用主菜单“工具→NAND Flash 编程”打开窗口如下：



在开始编程前，连接主机和目标板的网络线，LoongIDE 自动检测网络连接是否正确。

#### 执行 Strip 命令，移去所有调试信息

使用工具链的 xxx\_strip.exe 工具软件来实现。LoongIDE 自动检查可执行文件是否来自于某个龙芯 1x 项目：

- 如果是，使用该项目编译使用的工具链的工具软件；
- 如果不是，提示用户在 LoongIDE 配置的工具链中选择；
- 如果上一步没有选择，提示用户输入工具链所在目录。

#### 将应用程序设置为开机自动启动的工作模式

PMON 执行命令“set al /dev/mtd0”

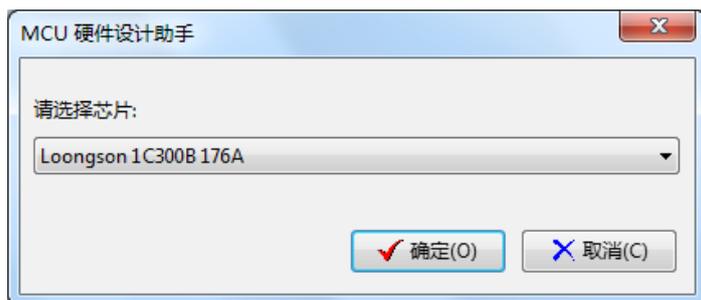
“NAND Flash 编程”仅支持运行 PMON\* 的目标板

### 9.3 硬件设计助手

以图形方式配置龙芯 1x 芯片引脚的使用。

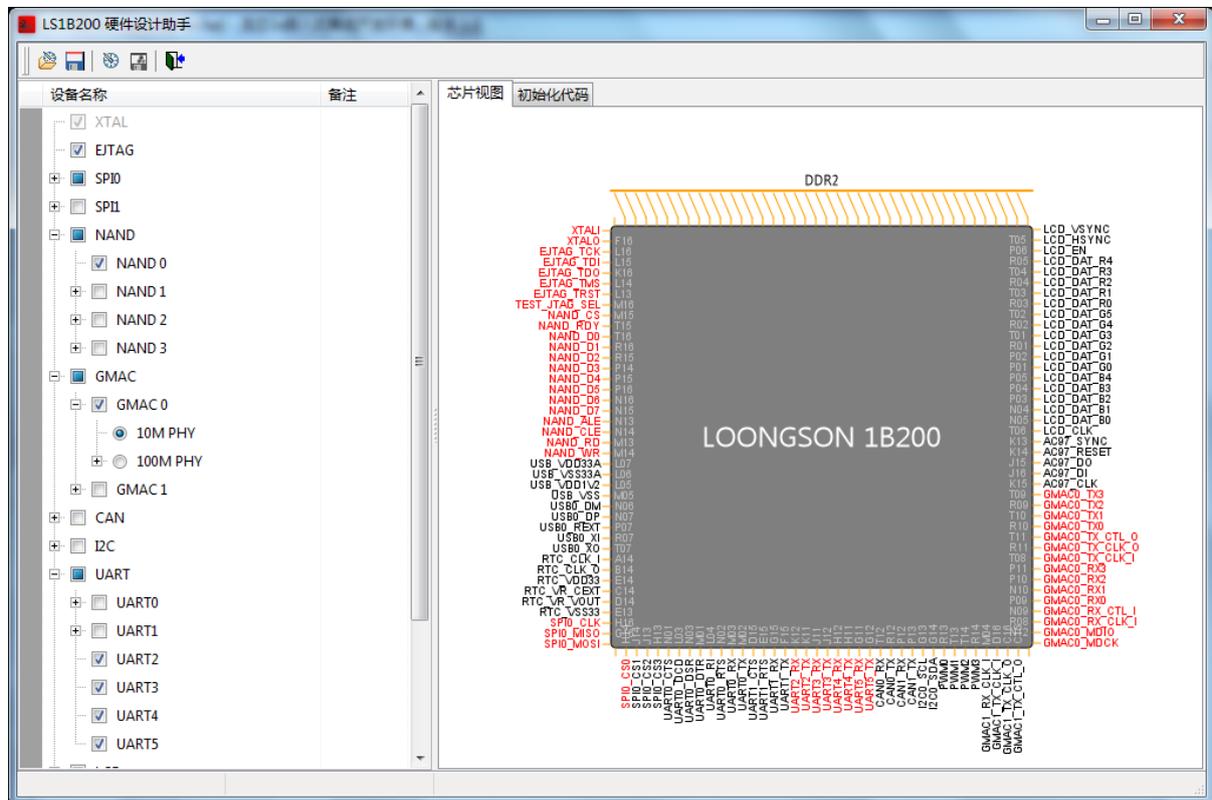
- 帮助软件工程师自动生成龙芯 1x 的引脚复用初始化代码；
- 帮助硬件工程师在设计龙芯 1x 应用板卡时合理分配引脚功能安排。

使用主菜单“工具→MCU 硬件设计助手”选择龙芯 1x 芯片：



### 9.3.1 龙芯 1B 芯片

打开窗口如下：



#### 设备列表

在窗口左侧的设备列表中选择设备，然后配置使用相应的片上设备。

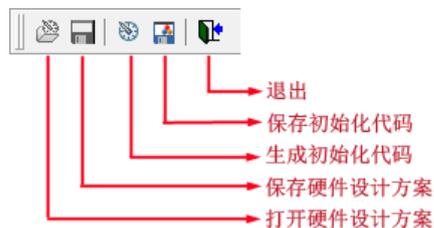
#### 芯片视图

在窗口右侧的芯片视图，显示当前芯片引脚使用情况：

- 红色：表示引脚已使用为某一设备，显示为实际使用的设备引脚名称
- 蓝色：表示引脚用作 GPIO 功能
- 黑色：表示引脚未使用，显示为默认引脚名称
- VDD/VSS 未标记

注：因为龙芯 1B 的 BGA 包装，引脚排列使用简化的逻辑引脚显示。

#### 工具栏按钮

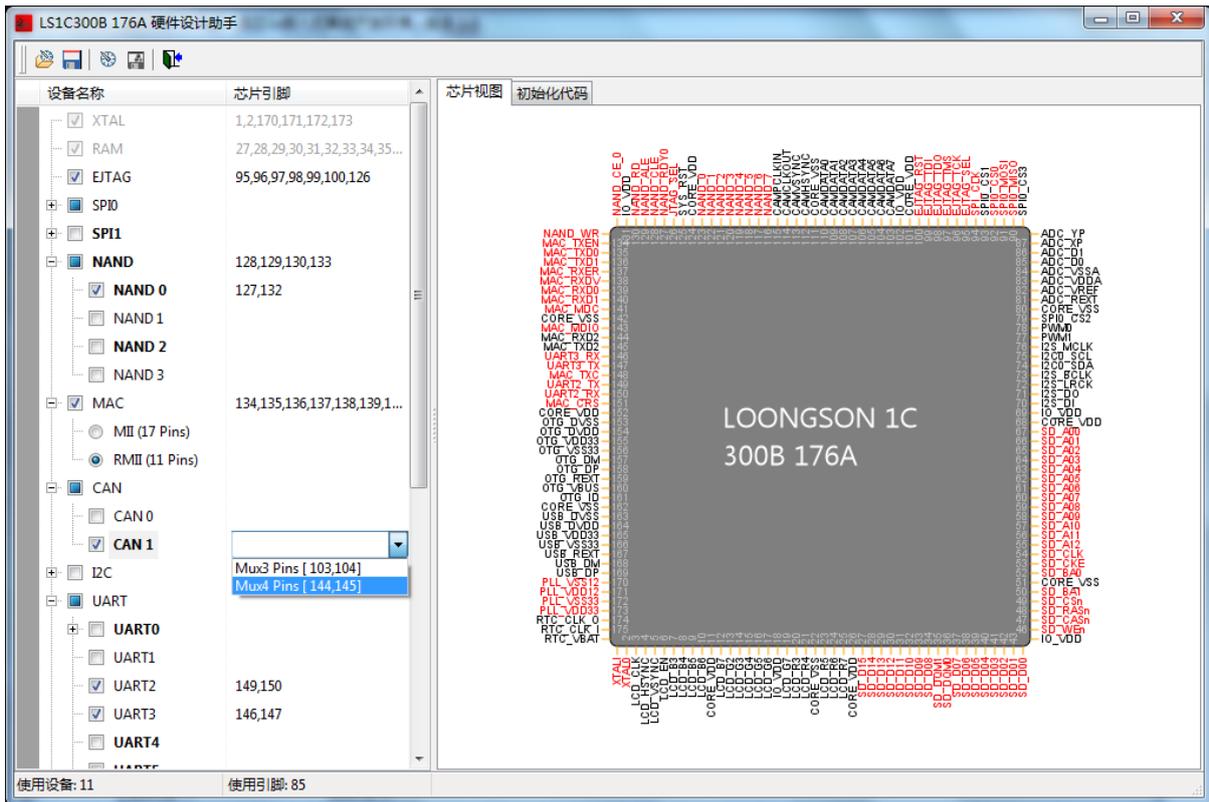


## 生成初始化代码

单击“生成初始化代码”按钮，窗口右侧面板切换到“初始化代码”页框。

## 9.3.2 龙芯 1C 芯片

打开窗口如下：



### 设备列表

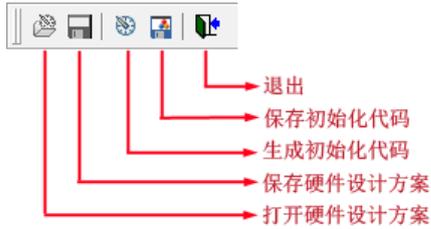
在窗口左侧的设备列表中选择设备，然后勾选需要使用的片上设备，并选择引脚编号。

### 芯片视图

在窗口右侧的芯片视图，显示当前芯片引脚使用情况：

- 红色：表示引脚已使用为某一设备，显示为实际使用的设备引脚名称
- 蓝色：表示引脚用作 GPIO 功能
- 黑色：表示引脚未使用，显示为默认引脚名称
- VDD/VSS 未标记

### 工具栏按钮



### 生成初始化代码

单击“生成初始化代码”按钮，窗口右侧面板切换到“初始化代码”页框。

## 10、系统安装

### 10.1 运行环境

LoongIDE 使用在 MingW 环境下编译的 GNU 工具链，所以在使用 gcc、gdb 等 GNU 工具时，需要 MingW 运行环境的支持。

用户可以选择安装 MSYS 1.0 或者 MSYS2 运行环境。

#### 10.1.1 安装 MSYS 1.0

从 <https://sourceforge.net/projects/mingw> 下载 msys 1.0 和 mingw 安装程序并安装；或者从 <http://www.loongide.com> 下载 [msys1\\_full\\_install.exe](#) 离线安装包进行安装。

MSYS 1.0 安装完成后，设置 Windows 系统环境变量 path:

将搜索路径 “c:\mingw\bin;c:\msys\1.0\bin;” 置于 path 首部。

#### 10.1.2 安装 MSYS2

从 <https://www.msys2.org/> 下载 [msys2-i686-xxx.exe](#) 安装程序并安装；或者从 <http://www.loongide.com> 下载 [msys2\\_full\\_install.exe](#) 离线安装包进行安装。

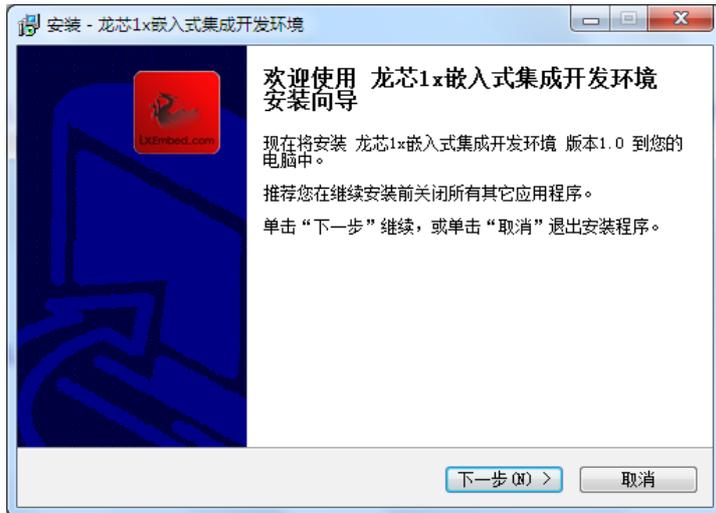
MSYS2 安装完成后，设置 Windows 系统环境变量 path:

将搜索路径 “c:\msys32\usr\bin;c:\msys32\mingw32\bin;” 置于 path 首部。

### 10.2 安装 LoongIDE

从 <http://www.loongide.com> 下载 “龙芯 1x 嵌入式集成开发环境” 安装程序 [loonide\\_1.0\\_setup.exe](#)。

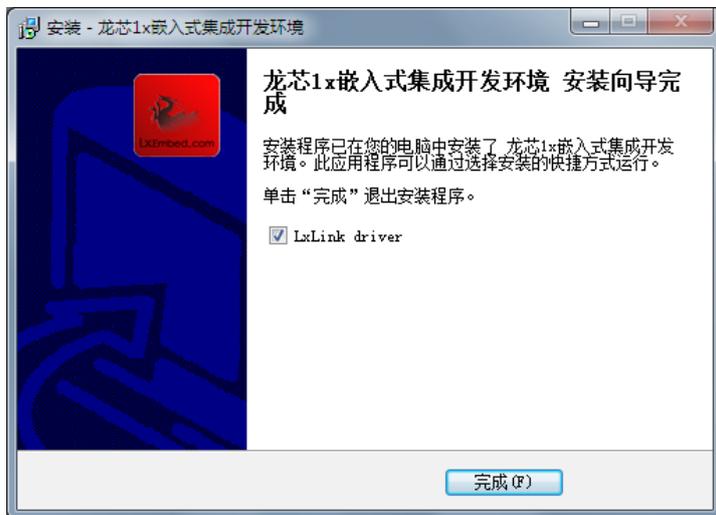
## 10.2.1 运行安装向导



用户根据安装向导完成系统安装。

## 10.2.2 LxLink 驱动

安装向导最后，选择安装 LxLink 驱动程序。



安装目录下有 driver 目录：

- 目录 CDM21228\_Setup\_x86 中保存有 LxLink 的驱动程序，用户可以通过 Windows 的“设备管理器”进行安装；
- 文件 CDMuninstallerGUI.exe 用于从 Windows 系统清除驱动程序。

## 10.3 GNU 工具链

LoongIDE 使用 SDE Lite for MIPS 工具链或者 RTEMS GCC for MIPS 工具链来实现项目的编译和调试。用户可以在 LoongIDE IDE 中安装一个或者多个工具链，使用时根据项目的实际情况来选择适用的工具链。

### 10.3.1 SDE Lite for MIPS 工具链

从<http://www.loongide.com>下载[SDE Lite 4.5.2](#)、[SDE Lite 4.9.2](#)的安装程序进行安装；安装完成后使用“C/C++工具链管理”窗口导入工具链。

建议在以下的项目类型时使用：

- 裸机编程项目
- RT-Thread 项目
- FreeRTOS 项目
- uCOSII 项目

### 10.3.2 RTEMS GCC for MIPS 工具链

从<http://www.loongide.com>下载[RTEMS 4.10 for LS1x](#)、[RTEMS 4.11 for LS1x](#)的安装程序进行安装；安装完成后使用“C/C++工具链管理”窗口导入工具链。

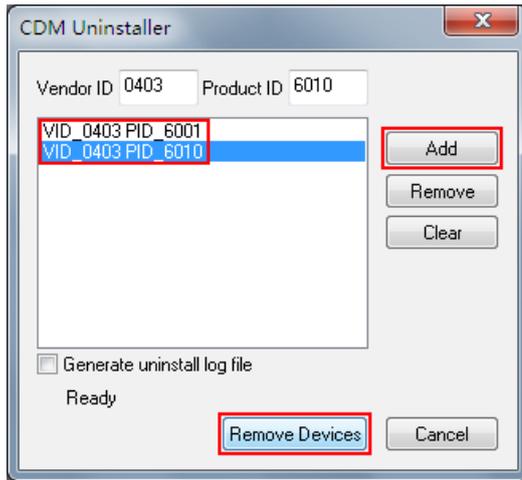
建议在以下的项目类型时使用：

- 裸机编程项目
- RTEMS 项目
- FreeRTOS 项目
- uCOSII 项目

RTEMS GCC for LS1x 内置龙芯 1B、龙芯 1C300B 的 BSP 包，包含片上设备的驱动程序；还移植有 modbus、yaffs2 等第三方 LGPL 软件包。

## 10.4 注意事项

- 建议将 msys/msys2 安装在 C 盘根目录；
- 工具链安装目录路径中避免使用空格、汉字等字符；
- 当 LxLink 驱动安装失败时，执行以下操作进行重装：
  - ①、运行安装目录\driver\CDMuninstallerGUI.exe，界面如下：



加入需要卸载的设备号，单击“Remove Devices”实现驱动程序的卸载和清理。

- ②、从 Windows “设备管理器” 查找安装目录\driver\CDM21228\_Setup\_x86 安装驱动程序。

苏州市天晟软件科技有限公司  
2020年4月22日