基于IAR Workbench for AVR

LGT8F328P 开发流程指南

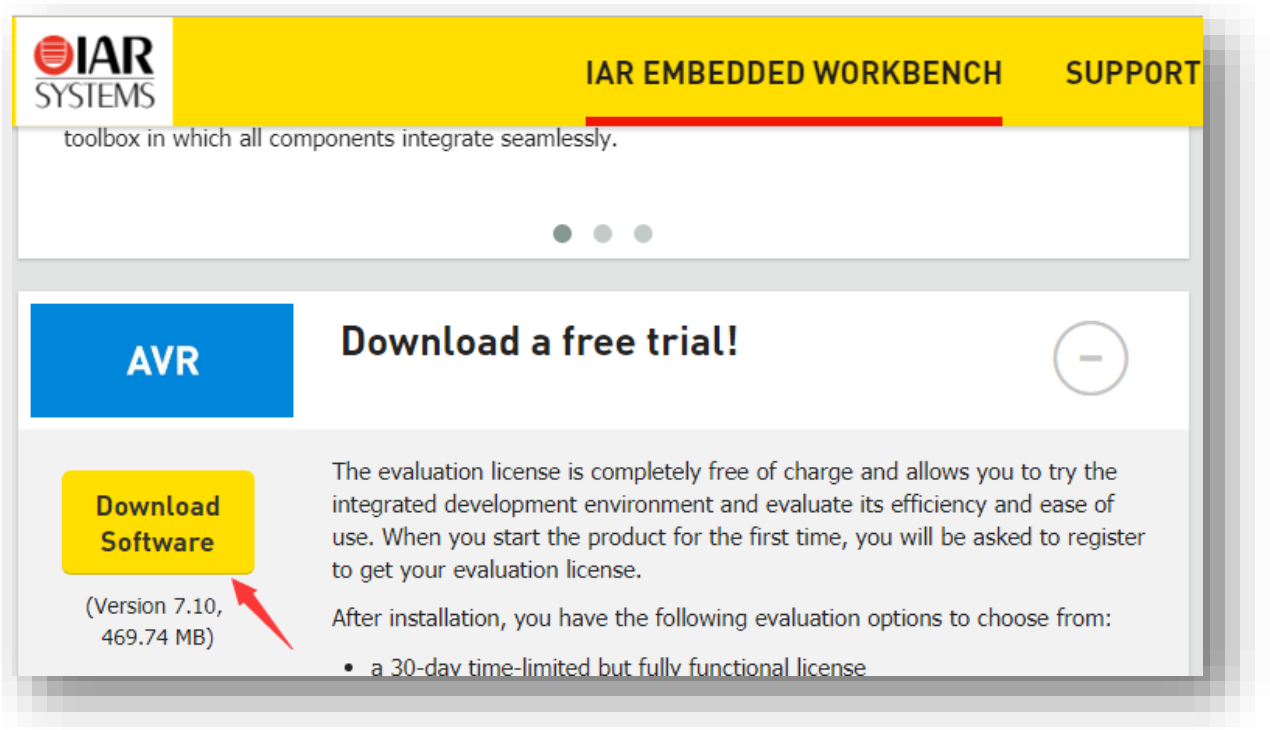
© 2017 LogicGreen Technologies

IAR Workbench for AVR (EWAVR)是一款适用于AVR架构的高效开发环境。编译器的优化算法业界领先，同时也是唯一第三方支持AVR调试的开发环境。非常值得学习使用！

LGT8F328P也完全适用于IAR Workbench for AVR的开发流程。配合SWDICE mkII Pro调试器，可以实现在线调试功能，对于复杂项目的开发调试， 省时省力！

为了帮助大家快速的掌握EWAVR开发LGT8F328P的方法，下面我们就以一个简单的例程，介绍使用EWAVR搭建LGT8F328P开发环境的流程。

首先是到IAR的官方网站上下载最新版本的EWAVR软件， 最新版本为7.10，下载使用版本即可！



下载后， 首先找到最新的科学安装方法，然后根据热心网页的视频教程， 完成安装！

我们这里是用最新的版本来给大家示例， 其他版本的基本上也差不多可以使用。

首先， 我们假定你手上已经有了SWDICE mkII Pro调试器。 为了实现调试功能，请将调试器侧面的功能开关拨到SICE档位。然后为调试器安装驱动。

如果你已经安装好了IAR Workbench， 我们可以在它的安装目录下找到调试器的驱动， 这个驱动程序的路径通常为(以最新版本的IAR为例)：

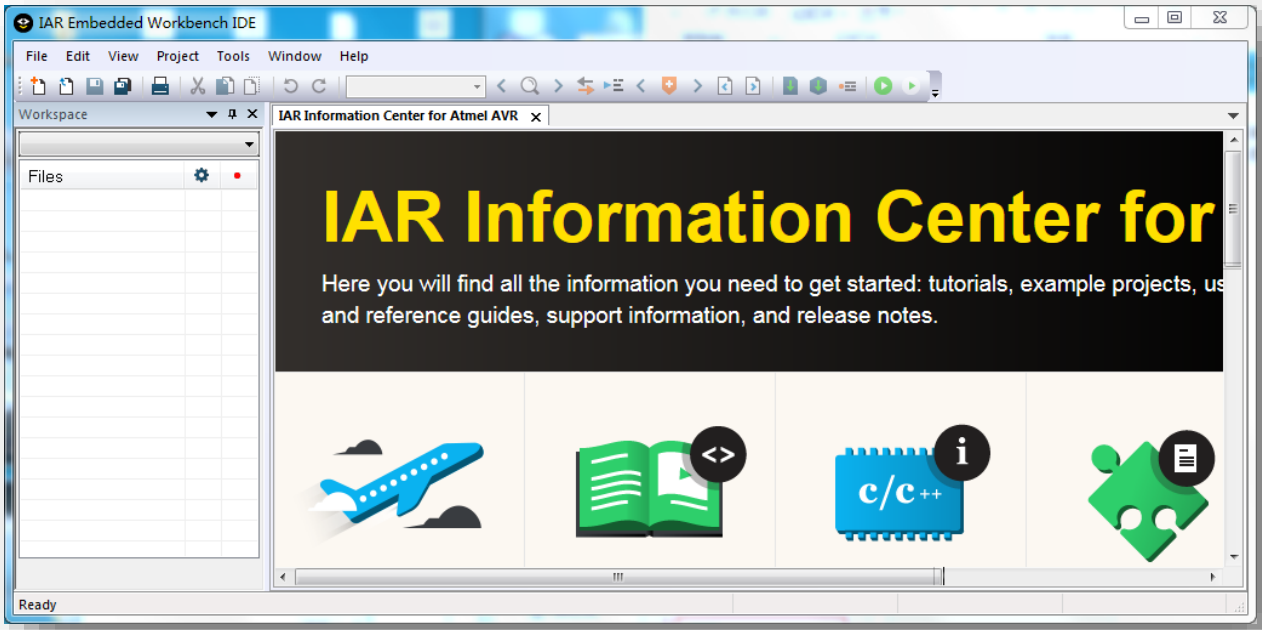
X:\Program Files (x86)\IAR Systems\Embedded Workbench 8.0\avr\drivers\

下面我们介绍使用IAR为LGT8F328P创建工程的详细流程

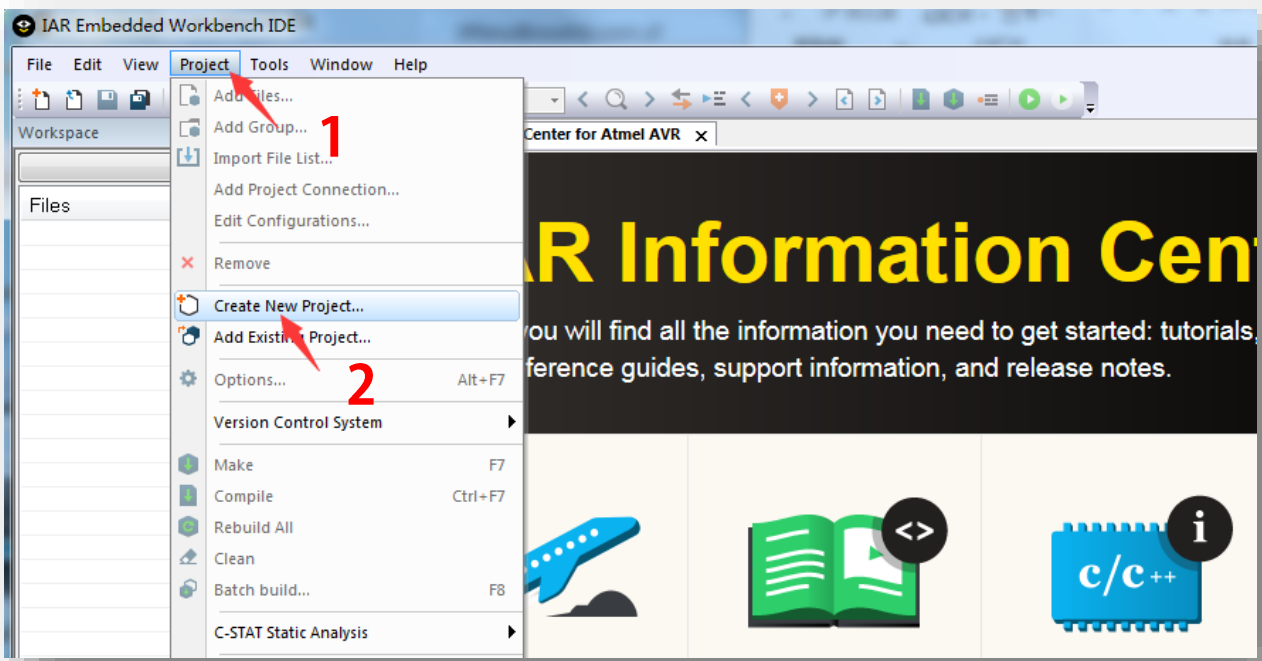
© 2017 LogicGreen Technologies

创建工程

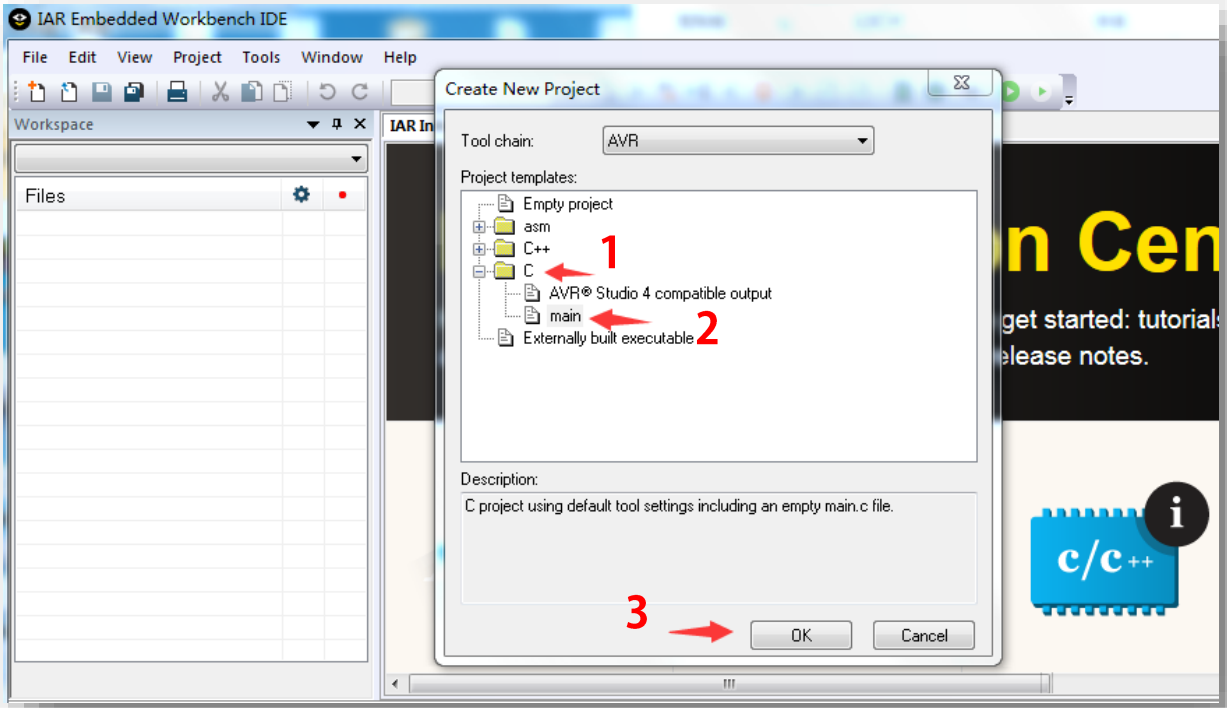
开始菜单启动IAR Embedded Workbench IDE，来到软件的默认界面， 这个是最新版本的界面，比起以前的主界面，看上去更和谐一些！:)



下面开始创建项目， 我们可以直接从主菜单[Project]下面的[Create New Project...]创建项目：

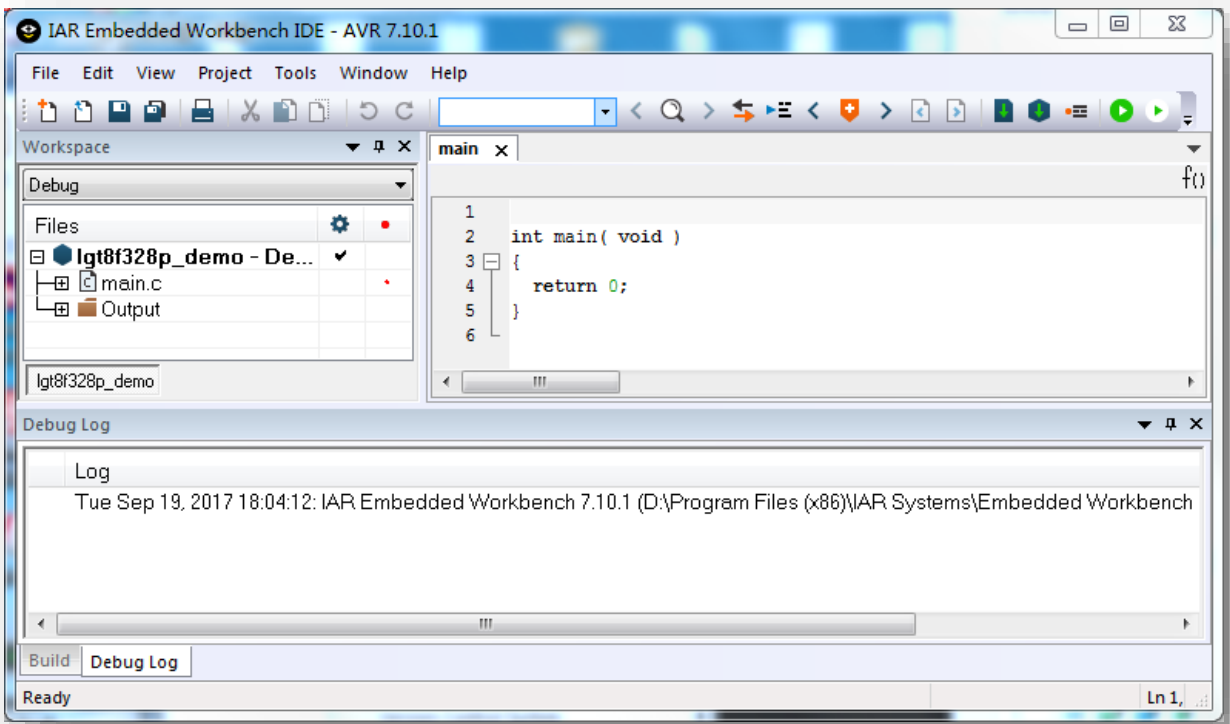


© 2017 LogicGreen Technologies



这里是选择工程模板， 我们用C语言， 展开后有两项， 我们不需要兼容AVRStudio4， 因此选择下面的main即可。 这样将会创建一个main.c的初始化文件。 点[OK]完成选择！

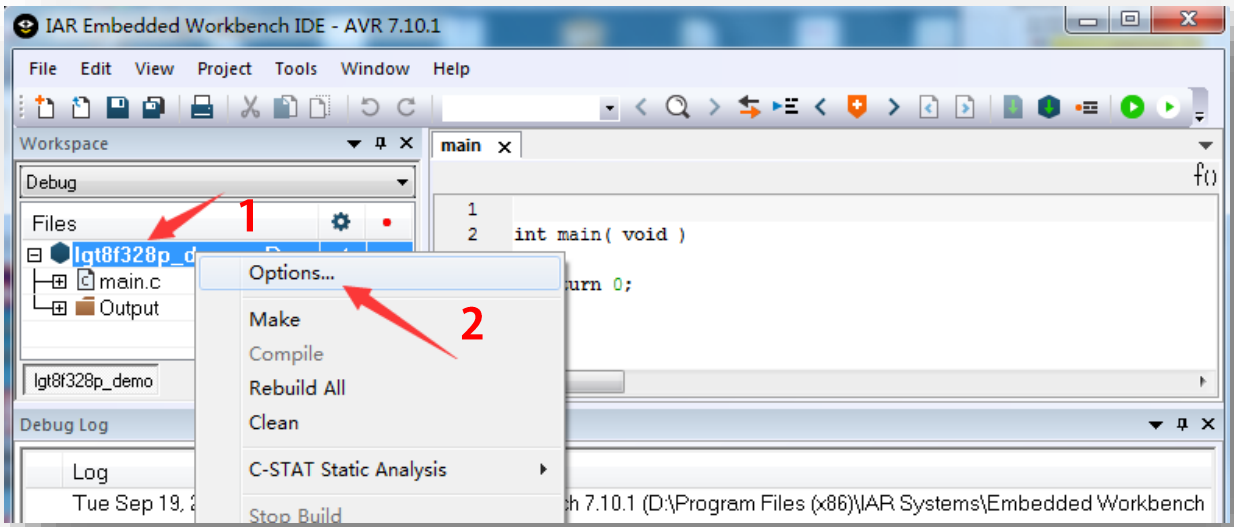
随后， 会弹出一个对话框，让你输入项目的名称和选择项目的目录。需要注意， IAR创建项目是不会为项目创建单独的目录， 为了便于管理， 建议手工为项目创建一个单独的目录。



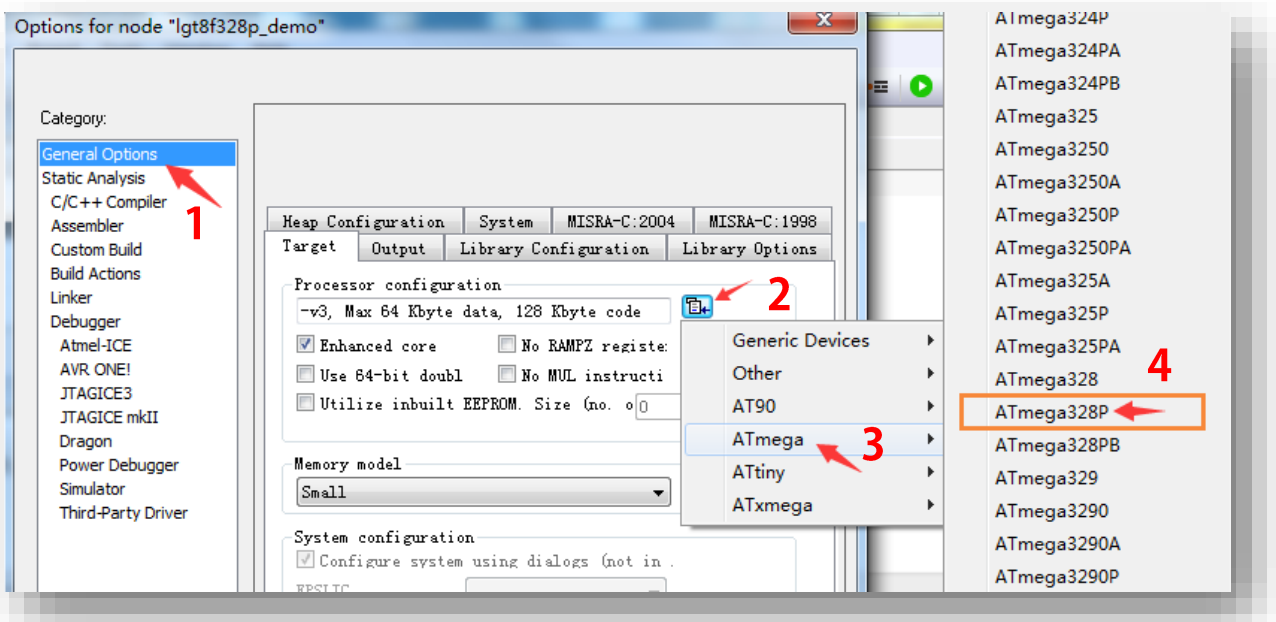
© 2017 LogicGreen Technologies

设置好项目的目录和名称，IAR将会进入如上图的工程管理界面。模板创建了一个默认的main.c文件。 至此， 项目创建完成！

下面我们需要设置工程属性， 让IAR知道我们目标芯片， 调试器类型以及需要编译器产生的文件。这些设置， 都在项目的属性中完成。 打开属性设置的方法很多， 我们可以直接在工程管理窗口中， 鼠标右击项目名称， 在弹出的菜单中选择[Option...]进入属性设置：

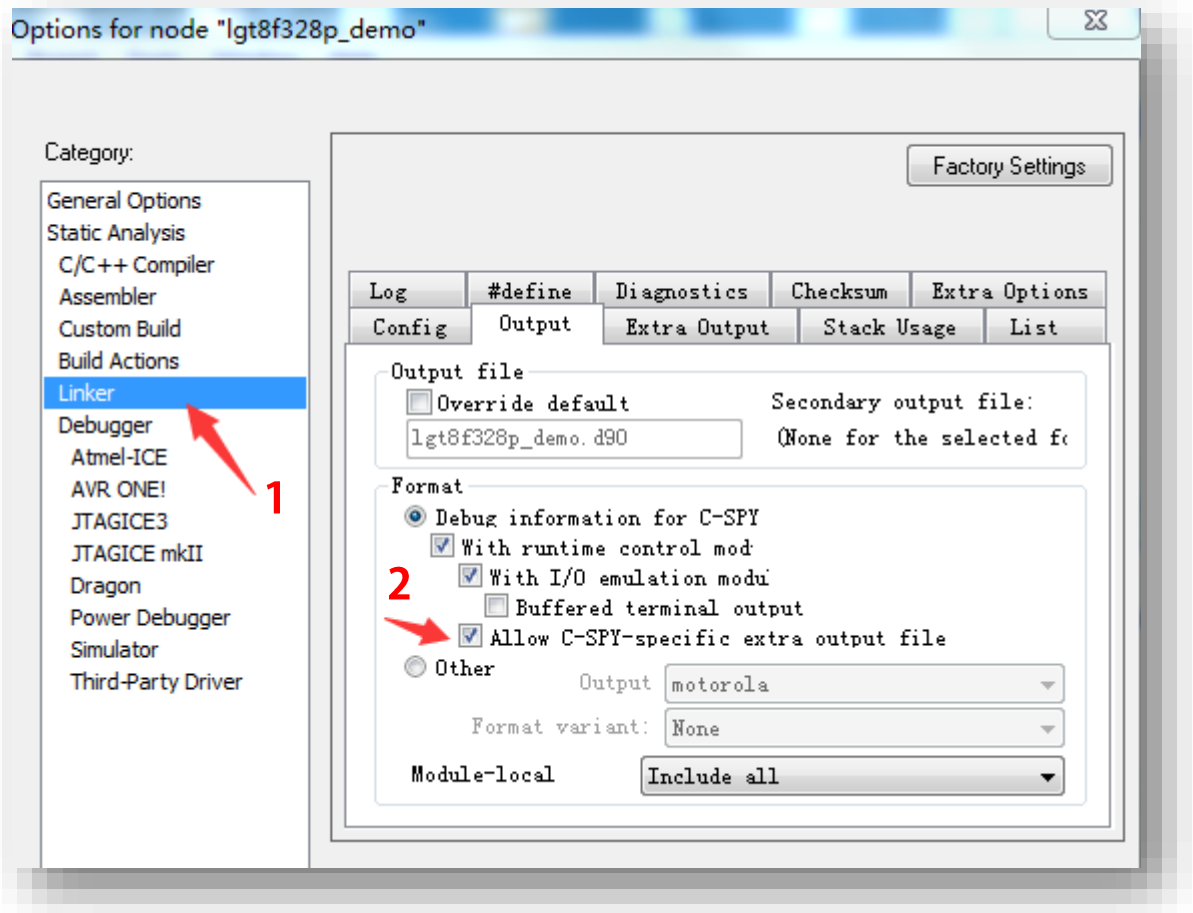


打开属性设置窗口， 我们首先要选择目标芯片， 我们需要选择兼容LGT8F328P的ATmega328P

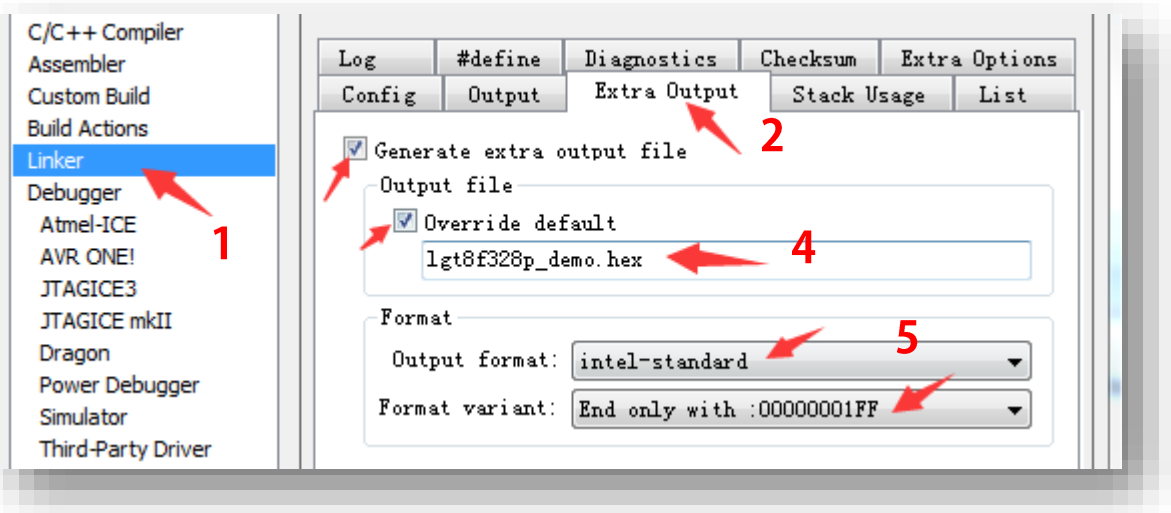


© 2017 LogicGreen Technologies

接下来是配置我们需要生成的HEX文件。 EWAVR默认并不产生HEX文件， 而且产生用于C-SPY调试的中间文件。我们可以通过下面的设置，让EWAVR同时也产生HEX输出！



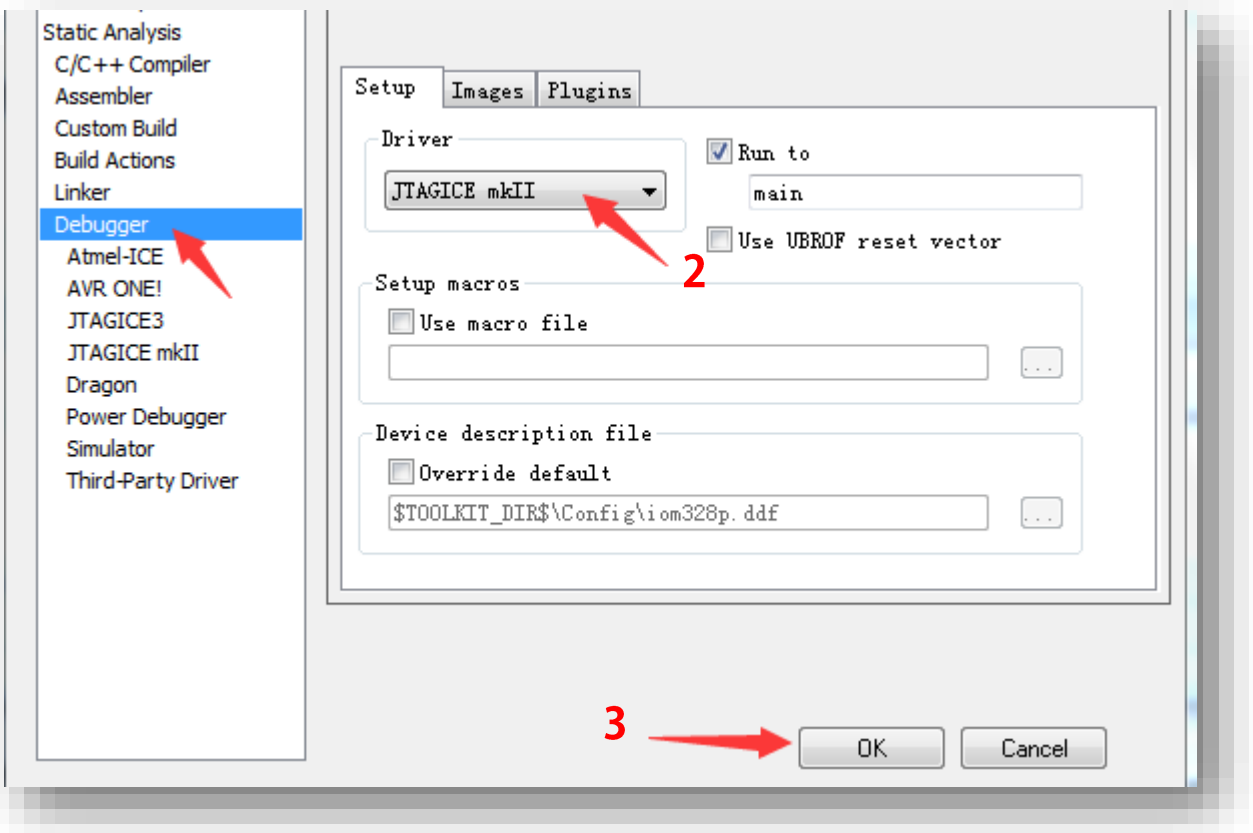
首先， 我们进入[Linker]属性设置页， 勾选[Allow C-SPY-specific extra output file]，使能C-SPY产生其他输出的设置。这样我们就可以在[Extra Ouput]栏下面设置输出文件：



© 2017 LogicGreen Technologies

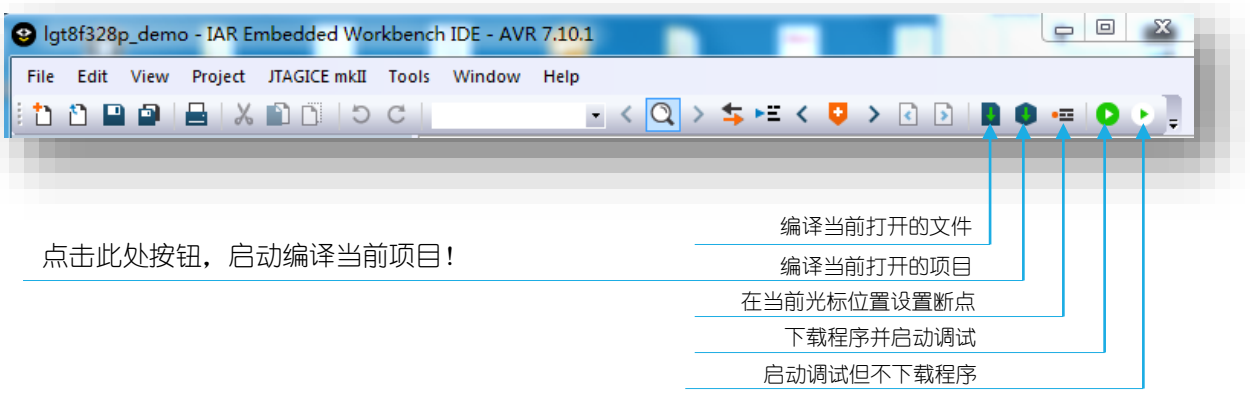
设置如上图， 这样我们就可以得到intel标志格式的HEX文件。这个文件可用于LGTMix_ISP烧写！

最后， 是设置我们的调试器， EWAVR默认是使用软件仿真器。 如果我们需要连接芯片实现在线调试， 需要将调试器选择为硬件的调试器。



调试器驱动选择JTAGICE mkII， 这个也是兼容我们SWDICE mkII Pro的驱动！

设置完成， 点[OK]退出项目属性设置。
接下来， 可以在更新的项目设置下编译项目。我们可以通过工具栏的快键按钮启动编译项目，下面结果按钮可能是我们会频繁使用的：



© 2017 LogicGreen Technologies

创建工程

当我们第一次点击编译项目时， 因为我们是直接创建的项目， EWAVR还会要求我们创建一个工作区，工作区用于管理项目。 一个工作区下可以创建多个项目！ 但一般情况下， 我们都是一个工作区对应一个项目。 在弹出的窗口中， 输入工作区的名称， 通常和我们项目的名称一致即可！

工作区保存后， EWAVR就开始编译工程， 我们目前只有一个空的main函数， 因此通常情况下，应该会编译成功， 编译器输出窗口显示无警告无错误！
如果在输出窗口看到出错提示， 基本上因为没有科学的安装好EWAVR所致！

接下， 我们以一个简单的例程， 完整的完成本篇教程。
例程代码非常简单， 我们用一个I/O输出一个固定周期得分方波，可闪烁一个LED灯！
例程的代码在本教程附带的压缩文件中， 因为比较简单，建议自己完成。
首先， 我们将教程附带的LGT8F328P专用头文件lgt8f328p_iar.h (EWAVR专用) 复制到项目目录中！
头文件放到和main.c相同的目录， 这样便于我们直接包含。 如果这个头文件不在main.c所在的目录，我们也可以从工程管理窗口中，将它加入到工程中！

下面是例程的代码：

main

```
1  // ============================================
2  // LGT8F328P的寄存器定义头文件
3  #include "lgt8f328p_iar.h"
4  // 编译器固有函数(比如延时函数)相关的头文件
5  #include <intrinsics.h>
6
7  // 定义系统时钟，用于计算延时
8  #define F_CPU 32000000UL
9  #define NUM_CYCLES_OF_US        (F_CPU/1000000UL)
10 #define NUM_CYCLES_OF_MS        (F_CPU/1000)
11
12 #define delay_us(us)    __delay_cycles(us*NUM_CYCLES_OF_US)
13 #define delay_ms(ms)    __delay_cycles(ms*NUM_CYCLES_OF_MS)
14
15 // ============================================
16 // 主程序开始
17 int main( void )
18 {
19     unsigned char btmp;
20
21     // 配置系统时钟预分频：不使用分频
22     // 直接使用内部32MHz作为系统时钟
23     btmp = (CLKPR & 0xf0);
24     CLKPR = 0x80;
25     CLKPR = btmp;
26
27     // 设置PB0的端口方向： 输出
28     DDRB_DDRB0 = 1;
29
30     while(1) {
31       // 翻转PB0的输出状态
32       PORTB_PORTB0 = ~PORTB_PORTB0;
33       delay_ms(100);
34     }
35 }
```

© 2017 LogicGreen Technologies

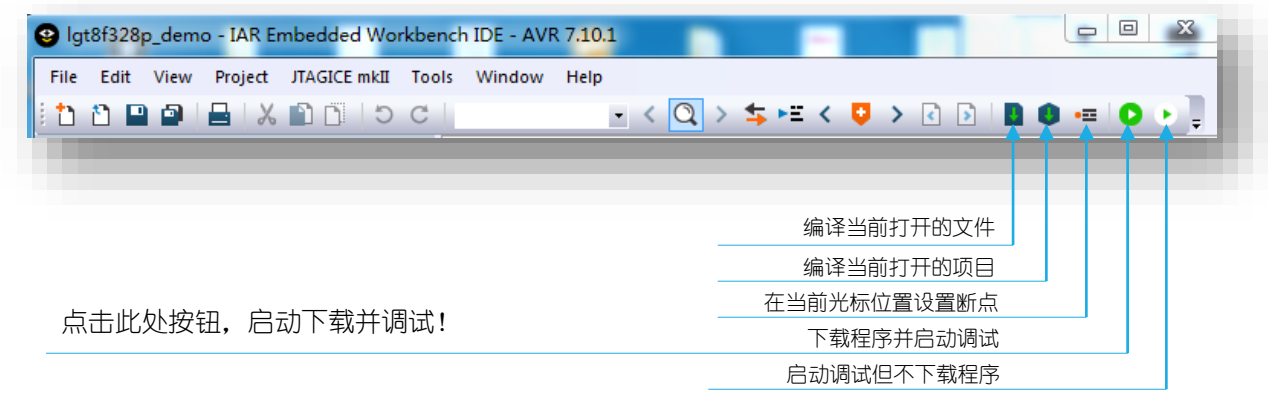
下面是对代码中的几点说明：

1. 首先包含LGT8F328P的头文件， 包含了LGT8F328P所有寄存器定义。 EWAVR专用！
2. 代码中我们需要一个延时函数， 我们使用了IAR编译器自带的实现。IAR没有直接实现延迟微秒或者毫秒的延时函数，而是一个以系统周期为参照的函数， 我们可以利用它实现延时函数。 代码中定义了当前系统频率F_CPU， 有了这个就可以计算毫秒、微秒对应的系统周期数！
3. 例程定义的系统频率是32MHz, 因为我们在程序初始化时关闭系统时钟分频器，直接使用32MHz作为系统工作时钟！
4. IAR支持直接位寻址， 我们提供的专用头文件中也将LGT8F328P扩展的寄存器定义了位寻址功能。这里设置PB0的端口方向和端口输出值，都直接使用了位寻址， 比较方便！

代码编写完成后， 从工具栏的快键按钮驱动编译整个项目！这里要特别提醒， 一定要使用编译整个项目的按钮， 这样会重新生成项目的输出。也可以从主菜单[Project]下的[Make]启动项目编译！

下面编译完成后， 可以直接将生成的HEX文件使用LGTMix_ISP下载到芯片中测试， 产生的HEX文件所在的目录通常为： ...\项目名称\Debug\Exe\ 或则是： ...\项目名称\Release\Exe\

如果需要在线调试， 也非常简单。 将SWDICE mkII Pro调试器连接开发板和PC， 在EWAVR工程中，可以使用工具栏的[下载并调试]按钮快速启动调试。 调试启动后， 首先编译项目， 然后下载项目到芯片中， 进入调试界面， 并停留在main函数的入口上！



© 2017 LogicGreen Technologies