LGT8F684A

FLASH Based 8bit Microcontroller

应用文档

V1.0.0

2015/11/03

LGT8F684A 开发简介

内容概述

文档主要介绍 LGT8F684A 开发平台相关内容，涉及到以下方面：

- LGT8F684A 开发概述
- 开发软件
- 烧写工具
- 实例详解

开发概述

LGT8F684A 基于 MIC8S 构架，系统设计兼容 PIC16F684，但实现不同的指令编码以及更加灵活的指令扩展。LGT8F684A 寄存器地址、RAM 空间分布兼容 PIC16F684，因此可以使用 PIC16F684 兼容的开发工具进行 LGT8F684A 应用开发。

在以 PIC16F684 为参考开发时，需要注意一下不同点：

- LGT8F684A 的配置位与 PIC16F684 虽然相似，但有很大的不同。配置位相关信息请参考 LGT8F684A 编程手册的【系统配置位】章节；
- LGT8F684A 编程接口与 PIC 系列不同，不能使用 PIC 兼容的编程硬件、软件；
- LGT8F684A 在 PIC16F684 的基础上实现了更加丰富的模拟/数字外设，这部分相关的内容请参考 LGT8F684A 编程手册；
- LGT8F684A 内核指令周期支持 1T/2T/4T，而 PIC 相关产品通常只支持 4T 指令周期；
- 基于 PIC16F684 开发的代码，99%可以直接转码后在 LGT8F684A 上运行；但系统配置位需要在烧写代码时通过编程工具进行正确的设置。

LGT8F684A 支持所有兼容 PIC16F684 的软件开发工具，支持汇编/C 语言开发。LGT8F684A 暂不支持在线调试功能。推荐使用的 C/ASM 开发环境为: MPLAB IDE (v8.92) + HI-Tech PICC (v9.83)

开发软件

MPLAB IDE 为 PIC 系列早期的开发工具，支持 PIC10/12/16/18 系列单片机；用于开发 LGT8F684A 的应用已完全够用。当然如果您喜欢使用最新的支持 PIC 系列单片机的开发工具，也是完全没有问题。

MPLAB IDE 本身只自带 MPASM 汇编编译器，对于 C 语言开发的支持需要安装专用的 C 编译器套件。兼容 PIC10/12/16 的 C 语言套件也比较多，这里推荐使用 HI-TECH 的 PICC (v9.83)。PICC 无论在易用性和编译优化性能上都算是比较优秀。

MPLAB IDE 为免费软件，非常容易通过网络资源获得；PICC 为商用软件，网络上也很容易找到用于学习研究为目的的版本，两款软件比较常用的版本为： MPLAB IDE v8.92, HI-TECH PICC v9.83

这里提供两款软件的网盘地址，仅为学习研究使用：

HI-TECH PICC v9.83: http://pan.baidu.com/s/1i3pUgU5

MPLAB IDE v8.92: http://pan.baidu.com/s/1eQkA1k2

软件的安装过程也比较容易，建议首先安装 MPLAB IDE；然后安装 HI-TECH PICC，PICC 软件的安装处理请参考安装包内的说明文档。

使用 MPLAB IDE 编译生产的 HEX 程序不能直接用于 LGT8F684A， 需要通过专用的转码工具进行转码；转码工具为 XIC2MIC， 此工具为一个命令行工具，无需安装，使用也极为简单。

可以将 XIC2MIC 执行的代码转换过程设置到 MPLAB IDE 的[Post builder]自定义命令，设置后 MPLAB IDE 可以直接编译生产支持 LGT8F684A 的 HEX 代码。

XIC2MIC 在 MPLAB IDE 中的设置将会在文档后面的实例工程中介绍。

XIC2MIC 下载地址： http://pan.baidu.com/s/1mgiwu7Y

XI2CMIC 也可以在文档附带的源码包中找到。

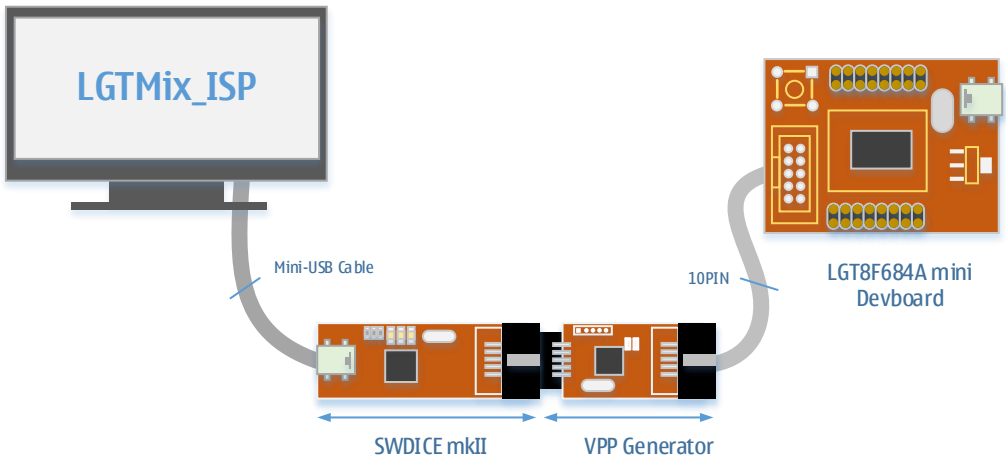
烧写工具

LGT8F684A 采用 LGT 标准的 SWD 接口实现代码以及配置信息的烧写。SWD 接口协议与 PIC 常用的烧写接口不兼容，因此不能使用 PIC 兼容的编程硬件、软件。

LGT8F684A 烧写硬件需要使用 LGT 官方提供的 SWDICE mkII 调试/下载器硬件；PC 端下载工具为 LGTMix_ISP 混合 ISP 下载工具。

由于 LGT8F684A 编程需要高压，因此还需要为 SWDICE mkII 外加一个高压编程扩展板，用此扩展板与 SWDICE mkII 相连接，然后再与开发板连接。高压扩展板将随后在官方淘宝店上购买。

以下为 LGT8F684A 编程软件/硬件的连接示意图：



实例详解

概述

本节将通过一个简单的 I/O 示例程序，详细介绍 LGT8F684A 软件开发流程以及在开发过程中需要特别注意的事项。示例编译后，使用 LGT 烧写工具下载到最小开发板中运行。

完整的工程代码包含随本文档一起发布的压缩包内。工程基于 MPLAB IDE，开发使用使用 C 语言，编译采用 HI-TECH PICC。直接使用 MAPLAB IDE 即可打开工程(*.MCP)。需要同时安装了 PICC 后才能成功编译。

编译使用 XIC2MIC 工具完成代码转换，因此可能需要根据 XIC2MIC 在本机的实际路径，修改编译设置。这部分内容也将随后详细介绍。

LGT8F684A 专用头文件

虽然 LGT8F684A 与 PIC16F684 兼容，可使用 PIC 提供的头文件进行开发。但 LGT8F684A 实现了更多的外设和寄存器控制。因此我们强烈推荐使用 LGT8F684A 专用的头文件。专用头文件同时也兼容 PIC16F684 所有的寄存器和宏定义(配置位除外)，可使用标准的 PIC 寄存器命名访问。

使用 LGT8F684A 专用头文件，可以根据 LGT8F684A 编程手册中的寄存器名以及寄存器位名进行寄存器访问。访问方式与标准的 PIC 寄存器访问完全一致。

LGT8F684A 编程手册(v1.0.1 以上版本)【存储系统】章节中关于寄存器的定义部分，针对 LGT8F684A 扩

展的外设寄存器分别用灰蓝色背景标注了 LGT8F684A 专有的寄存器定义。

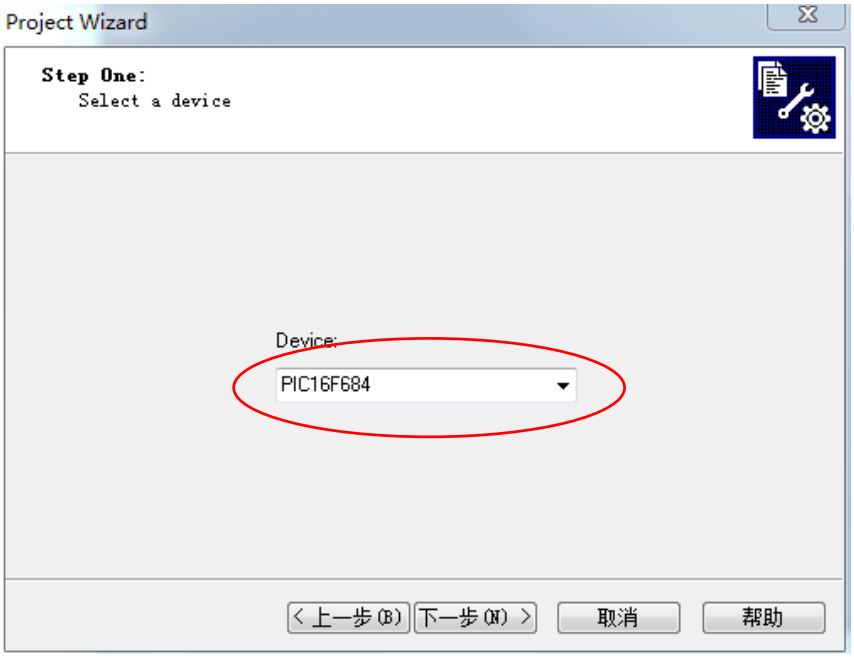
LGT8F684A 专用头文件(lgt8f684a.h)可以在附件的工程目录中找到。其他新建工程可将其复制到工程目录中使用。LGT8F684A 专用头文件中已包括了 PIC 标准的头文件定义，因此只需在代码中包含 lgt8f684a.h 即可。

建立工程

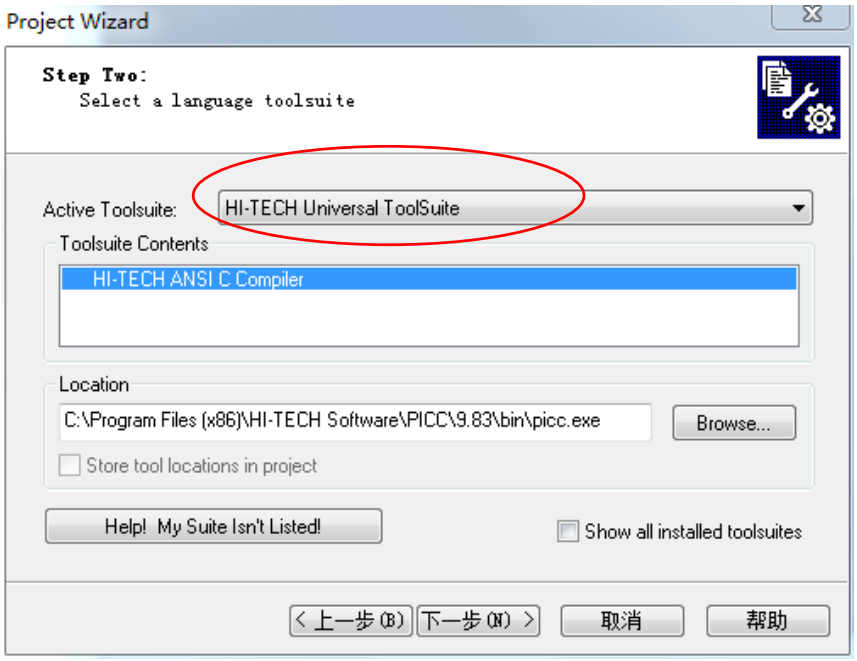
首先为工程建立一个目录，将 LGT8F684A 专用头文件复制到其中。

启动 MPLAB IDE，在主菜单[Project]下选择[Project Wizard...]，打开工程向导：

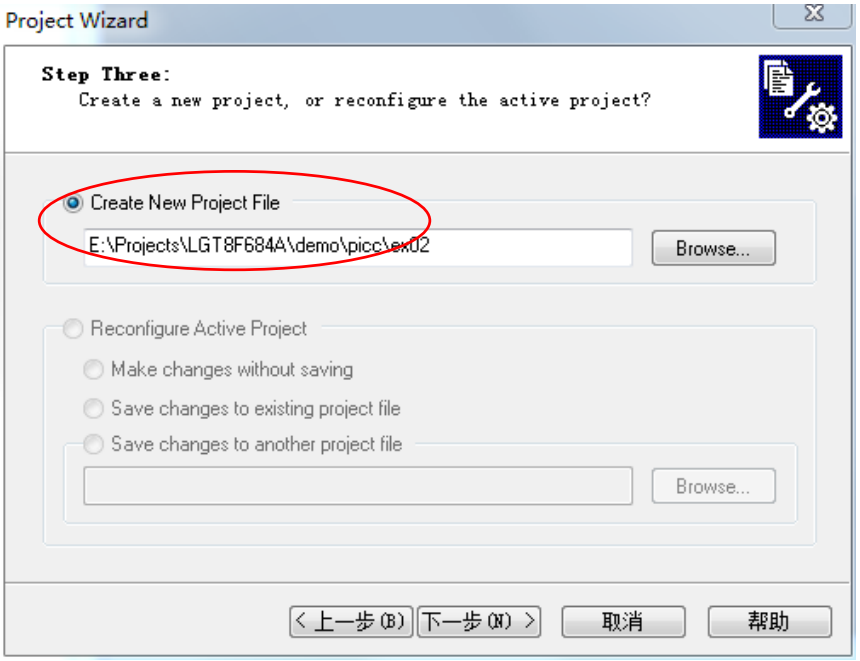
继续[下一步]，选择目前芯片的型号。此处我们选择 PIC16F684：



继续[下一步]，进入编译器选择。在[Active Toolsuite]下选择[HI-TECH Universal ToolSuite]即可。如果系统中安装了 HI-TECH PICC，其路径将会出现在下方的[Location]信息栏内；无需手动设置。



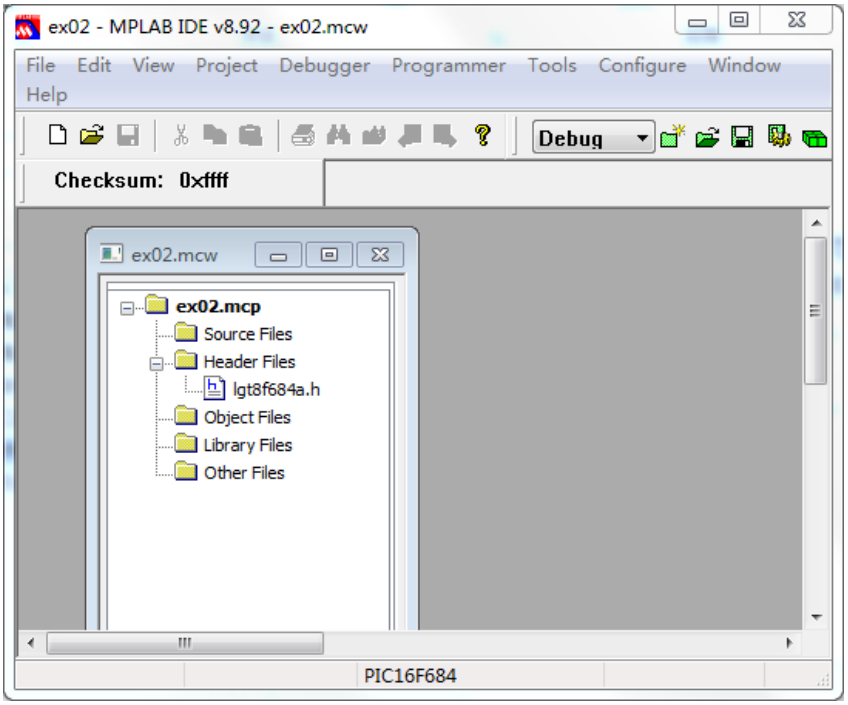
继续[下一步]，进入工程名称和目录设置。打开[Browser]，选择到工程目录，并输入工程名称。如此处的"ex02"即为我们新建的工程名称：



继续[下一步]，可以在这里选择已有的文件加入到工程。这里我们可以将事先准备好的 LGT8F648A 专用头文件添加到工程内，这样做可以省却设置头文件搜索路径的麻烦。

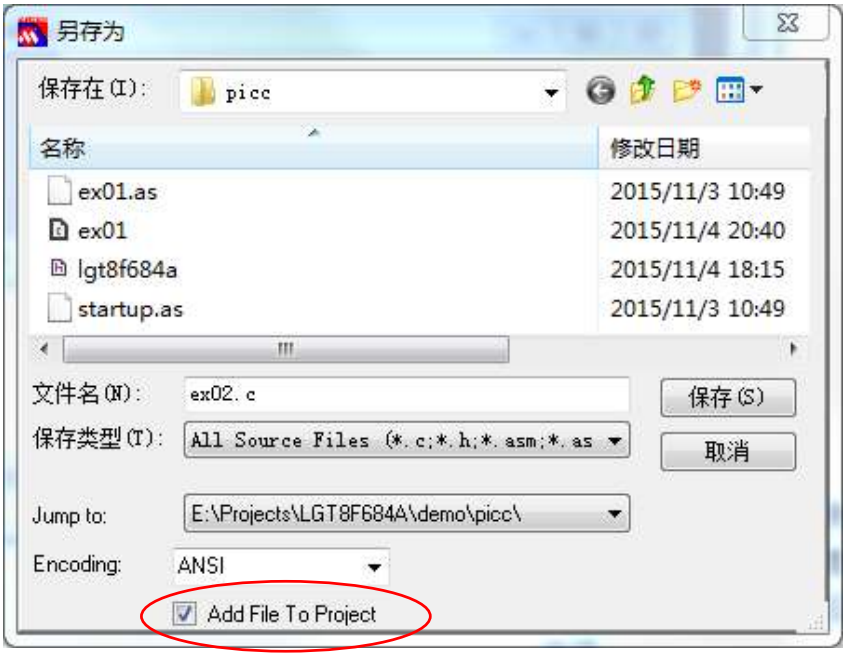
文件添加完成后，选择[下一步]即可完成项目的设置。最后选择[Finish]完成项目的创建。

项目创建完成后，MPLAB IDE 进入一个初始的工程界面。在工程目录栏内，可以看到我们刚刚加入到工程中的头文件：



工程创建完毕，下面我们添加源代码文件。在 MPLAB IDE 主菜单[File]下选择[New]即可创建一个新的文件。也可以通过工具栏或者快捷键[CTRL+N]创建新文件。文件创建后是没有格式的，与当前工程也没有任何关系，需要将其保存到项目中才能正常使用。

在 MPLAB IDE 主菜单[File]下选择[Save As...]，打开另存对话框：



如上图所示，选择文件保存到当前工程目录。然后填写文件名，需要同时给出文件的类型后缀名。在对话框的最下方，勾选上[Add File To Project]，这样文件保存后会同时添加到工程中。否则我们需要单独将此文件添加到工程中。

保存完成后，新建的文件将会显示在工程目录树中，下面就可以填写代码了。文档完整的实例代码如下，可以直接将代码复制到新建的文件中。

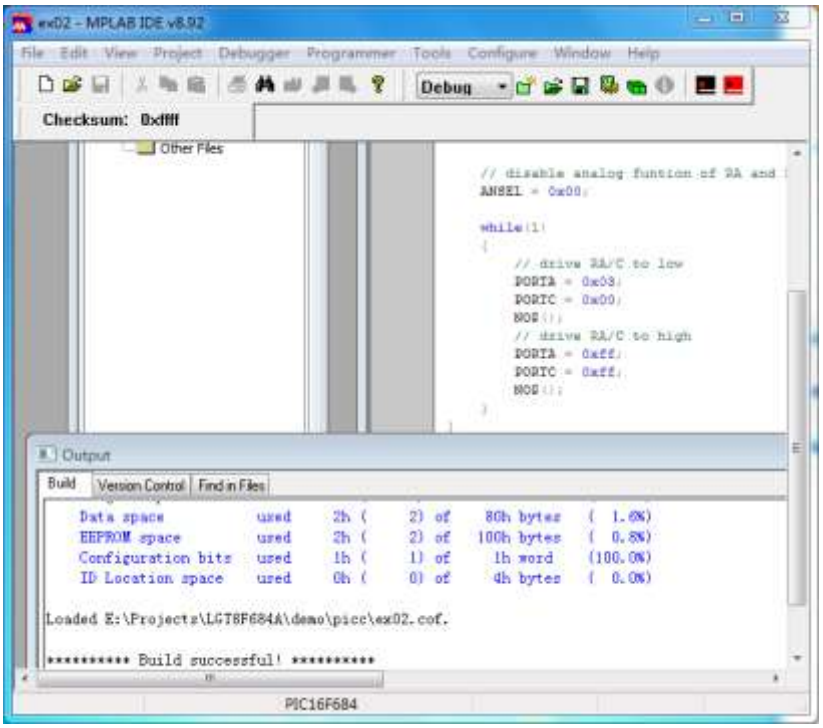
代码示例

```
//----------------------------------------
// LGT8F684A test case
//   case name: I/O output
//   description:
//       toggle all I/O except RA0/1 and RA3
//----------------------------------------
#include "lgt8f684a.h"

// configuration word 1 settings:
//   1. disable WDT
//   2. disable fsys output on RA4
//   3. disable TSSM mode
__L_CONFIG(CF1_ON & WDTE_OFF & OSCO_OFF & TSSM_OFF);

// configuration word 2 settings:
//   1. 1T core cycle mode
__L_CONFIG(CF2_ON & TCYC_1T & OSCFS_OFF);

int main()
{
    // set RA[7:2] to output
    // keep RA[1:0]
    TRISA = 0x03;
    // set RC[5:0] to output
    TRISC = 0x00;
    // disable analog function of RA and RC
    ANSEL = 0x00;

    while(1)
    {
        // drive RA/C to low
        PORTA = 0x03;
        PORTC = 0x00;
        NOP();
        // drive RA/C to high
        PORTA = 0xff;
        PORTC = 0xff;
        NOP();
    }
}
```

工程设置

代码编码完成，此时可以选择编译工程，检查代码以及工程设置是否正确。

通过主菜单[Project]下面的[Build]或[Rebuild]都可以启动工程编译。如果一切顺利，将会在输出窗口看到编译成功的提示以及程序所使用的资源信息：

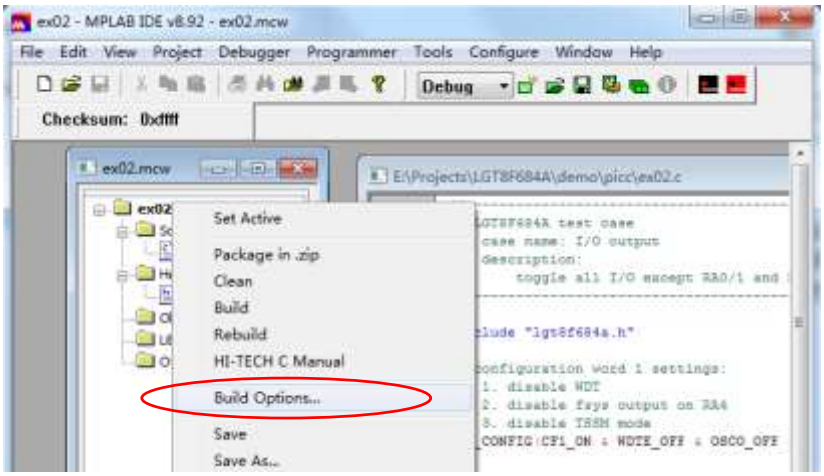


此时生产的 HEX 文件是不能直接运行在 LGT8F684A 上的，必须进行转码。转码通过 XIC2MIC 工具完成。我们可以将转码过程集成到 MPLAB IDE 的编译流程中，直接产生 LGT8F684A 可用的文件 HEX 文件。

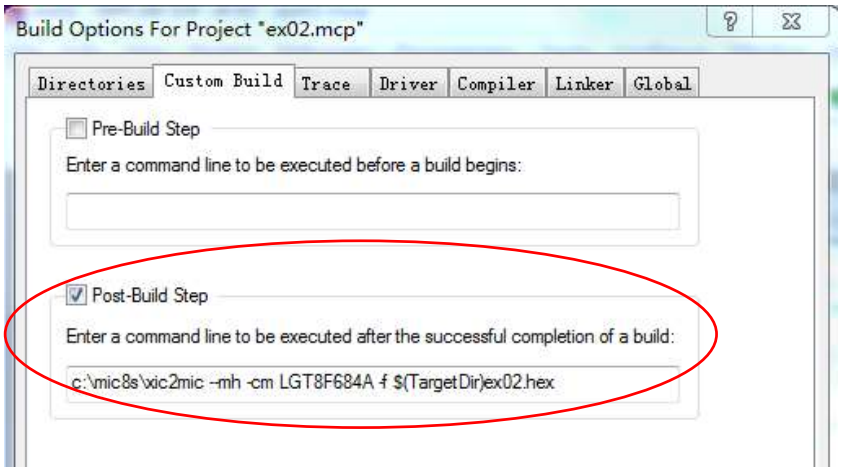
下面是将 XIC2MIC 集成到 MPLAB IDE 编译流程中的步骤：

首先，将 XIC2MIC 放到一个可访问的目录中，建议放到系统根目录下的子目录中。

然后在 MPLAB IDE 的项目树栏中，鼠标右击项目名称(*.MCP)，在弹出的菜单中选择[Build Options...]，打开项目选项设置对话框。



在打开的[Build Options]设置对话框中，选择[Custom Build]属性页：



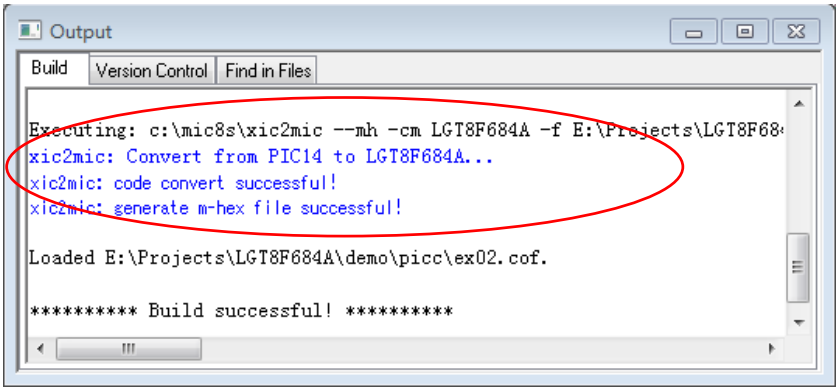
如上图所示，首先勾选[Post-Build Step]， 然后在下面的编辑框中填入如下信息：

```
Path\to\xic2mic --mh -cm LGT8F684A -f $(TargetDir)ex02.hex
```

需要注意以下两点：

1. 根据 xic2mic 的具体路径，替换"\Path\to\"为 xic2mic 在系统中的实际路径；
2. 根据项目中主程序文件名(main 函数所在文件)，替换"ex02.hex"；

修改完成后，选择[确认]保存并退出[Post-Build Step]设置对话框。
然后重新编译工程，如果设置正确，将会在编译输出窗口看到如下信息：



编译成功后，将会在项目所在目录下产生一个类似"ex02_mic.hex"的 HEX 文件，将此文件通过下载器烧写到 LGT8F684A 中即可运行。如 xic2mic 提示出错，请检查相关文件名和路径是否设置正确。

XIC2MIC 设置完成，将会保存到本项目设置中，以后重新打开本工程时，无需重新设置。但对于重新建立的新工程，需要设置 XIC2MIC。

至此，项目设置全部完成。其他编译器、连接器相关的设置，请参考 MPLAB/PICC 相关文档。

编程要点

1. 寄存器读写控制

PICC 相比其他针对微控制器的 C 语言编译器，最值得称道的，就是其对寄存器访问的优化。PICC 通过独特的位偏移地址定义，可以支持直接寄存器位的读写访问，非常直观方便。同时，PICC 也针对寄存器访问以及 PIC 处理器分页的机制进行了优化，实现了非常高质量的优化性能。其输出代码的品质基本超过大部分手写汇编实现。同时 PICC 也支持与汇编交叉编译，可以为一些对时序控制要求比较严格的应用提供嵌入汇编的途径。

LGT8F684A 为 PICC 提供的专用头文件(lgt8f6584a.h)，也严格遵循 PICC 寄存器定义规则。将寄存器命以及寄存器位命名统一。LGT8F684A 编程手册中所给出的寄存器名以及寄存器位名称完全可以在 PICC 编程中直接使用。

我们以 LGT8F684A 中的 DAC2 控制过程为例，来演示 PICC 如何便捷的进行寄存器控制。我们将配置 DAC2 并将其输出到 RA6/DAC2 引脚上：(以下代码可直接在 C 语言代码中编写)
首先，我们需要确认 RA6 工作于模拟 I/O 模式，将 RA6 设置为模拟 I/O 的代码非常简单，RA6 的模拟工作模式由 ANSEL1 的 ANS8 位控制，我们可以直接使用下面的代码将 RA6 设置为模拟 I/O 模式：

```
ANS8 = 1;     // set RA6 to analog mode
```

接下来是配置 DAC2。DAC2 的配置包括选择参考源以及设置 DAC 输出电位，最后是使能 DAC2 控制。DAC2 的参考源由 VRCON2 寄存器的 DAC2S0/1 控制。我们将选择内部 1.25V 基准电压作为参考源使用，从手册上可知，当 DAC2S1 设置为 1 时，将选择内部 1.25V 基准作为 DAC2 的参考源：

```
DAC2S1 = 1;    // select 1.25V IVREF for DAC2 source
```

DAC2 的输出电位由 VRCON2 寄存器的 VR2[5:0]位控制，输出电压值为：

$$V_{DAC2} = (VR2+1)/64 * V_{SRC} \quad (V_{SRC} = 1.25V\ IVREF)$$

我们需要直接输出内部基准电压，因此需要将 VR2 设置为 0x3F。因为 VR2 是一个位段，我们将采用更有效的访问方式：

```
VRCON2bits.VR2 = 0x3F;    // set DAC2 output level
```

由于 VRCON2 是 LGT8F684A 扩展的外设寄存器，我们也可以通过一个 LGT8F684A 专用的寄存器名称定义访问： L_VRCON2 或者 L_VRCON2bits.VR2。即使用一个"L_"的前缀。任何 LGT8F684A 扩展的寄存器或者含义扩展位的标准寄存器都可以通过加"L_"前缀的方式访问。
最后，我们通过 VRCON2 寄存器的 DAC2EN 位使能 DAC2 模块：

```
DAC2EN = 1;    // enable DAC2
```

至此，DAC2 配置完成，运行相关程序，DAC2 将内部 1.25V 基准输出到 RA6/DAC2 引脚上。

从以上示例可以看出 PICC 强大的寄存器访问能力，而且非常易于操作。
同时，PICC 编译器也会根据寄存器所在的页(BANK)，加入必要的页面切换的操作，而不用像用汇编编写代码时，需要人工经行页面切换，进而避免了因疏漏导致的程序问题。
即便 PICC 能够根据寄存器的地址自动的切换 BANK， 建议我们在写初始化代码时，也应该将不同页面的寄存器分开设置，这样可以避免产生多余的页面切换代码。

2. 配置位设置

PICC 编译器支持在代码中设置配置位。这样可以将程序代码与相关的运行环境合成一个整体，在量产过程中意义重大。另外 MPLAB IDE 中也可以通过集成环境工具设置配置字，但这些配置都是针对标准的 PIC 微处理器专用的，无法直接用于 LGT8F684A。

为了能够利用这种方便的配置位设置方式，LGT8F684A 的专用头文件中定义了一个配置位设置宏：__L_CONFIG，并同时提供了相关配置位的定义，利用__L_CONFIG 关键字，可以用标准的方式在代码中设置配置字信息。

LGT8F684A 实现了两个 16 位的配置字，可以使用两个__L_CONFIG 分别配置。使用时请注意__L_CONFIG 只能使能两次，出现顺序分别对应配置字 1 和 2：

```
// configuration word 1 settings:
//  1. disable WDT
//  2. disable system clock output on RA4
//  3. disable TSSM mode
__L_CONFIG(CF1_ON & WDTE_OFF & OSCO_OFF & TSSM_OFF);

// configuration word 2 settings:
//  1. 1T core cycle mode
__L_CONFIG(CF2_ON & TCYC_1T & OSCFS_OFF);
```

配置信息将会被编译到 HEX 文件的从 0x4200 开始的连续的 4 个字节中。LGT8F684A 的下载器能够从 HEX 中识别这部分数据，转化为系统配置信息。

更多配置相关的定义，请参考 LGT8F684A 专用头文件中的定义。

版本历史

版本	作者	日期	版本日志
1.0.0	LGT	2015/11/01	The first edition