# 第 3 章　程序编写、调试和烧录

单片机在上电复位后就会执行程序存储器中储存的指令序列，完成特定的任务。因此，程序编写、调试和烧录都是单片机开发中必要的步骤，往往需要特殊的工具(既包括软件工具也包括硬件设备)。本章将以 HT46F49E 的程序开发为例，讲述程序编写、调试和烧录的主要过程和相应工具。

## 3.1　程序设计简述

编程简单地说就是告诉单片机每一步都做什么。一般而言，单片机编程涉及 3 种语言：机器语言、汇编语言和高级语言(一般指 C 语言)。

(1) 机器语言(或称为二进制代码语言)，其指令是用 0 和 1 组成的一串代码，它们有一定的位数，计算机可以直接识别，不需要进行任何翻译。每台机器的指令格式和代码所代表的含义都是硬性规定的，故称为面向机器的语言，也称为机器语言。机器语言对不同型号的计算机来说一般是不同的。机器语言对人类而言就像天书，很难理解和记忆。

(2) 汇编语言，简单地说，就是将计算机语言中一大串 0、1 代码按照其含义，用人类易于理解的词汇代替，具体包括用助记符代替操作码，用地址符号(Symbol)或标号(Label)代替地址码。使用汇编语言编写的程序不能直接被机器识别，要由汇编程序将其翻译成机器语言。一般而言，汇编语言和机器语言有着非常直接的对应关系，学习汇编语言同样需要了解计算机内部的结构，因此汇编语言也被称做低级语言(或者底层语言)。

(3) 高级语言，其语法和结构更类似于普通英文，与计算机的硬件结构及指令系统无关，它有更强的表达能力，能更好地描述各种算法，而且容易学习掌握，可移植性好。同样，高级语言需要翻译成计算机能够识别的机器语言，这个过程称为编译。在编译过程中，会先产生作为中间产品的汇编代码，然后再将其翻译成机器语言。在单片机高级语言设计中，一般采用 C 语言。

以一个驾驶汽车的例子来解释高级语言和汇编语言的不同。假设你是公司的老板拥有自己的司机。那么，即使你不会开车，也能命令司机达到目的地，例如“减速，停在右边的树下”，此时你是通过自然语言和司机交流的。司机则将上司的命令翻译成具体的操作步骤，例如上述命令可以通过换挡、松油门、将汽车右转及踩刹车实现。前者类似于编程中的高级语言，而具体的操作步骤类似于编程中的低级语言。两者都可以实现相同的功能，但是高级语言更容易掌握，且不需要熟知计算机的内部细节，因此可移植性更好。(比如将 80C51 的程序移植为盛群的程序，C 语言可以较为迅速地完成，而汇编语言基本需要重写。)

单片机开发中一般采用两种方法(汇编语言方法或 C 语言方法)进行程序设计。对初学者而言采用哪种方法更合适呢？虽然汇编语言有助于理解单片机内部的工作细节，但是读者的学习目标一般都是能用单片机控制各种设备和工作流程。从这个意义上讲，C 语言

开发效率更高，可移植性强，因此目前成为工程师的首选。在本教程中将简单介绍汇编语言，并在最初的实验中运用两种语言编写，而后续实验只用C语言开发。

对于工科学生来说，使用C语言开发程序应该在计算机编程语言课程中都学习过，对于基本程序的框架大家都应该比较熟悉，因此结合具体的实验，后面进行进一步讲解。

对于盛群汇编语言编程，可能同学们没有接触过，但是其结构和一般微型计算机机原理中的汇编程序设计类似，下面以本书的第一个例子（面包板实验：LED显示）来介绍盛群汇编语言的要点及编程框架，如表3-1所示。

**表3-1　汇编程序框架**

```
#INCLUDE HT46F49E.INC
.CHIP HT46F49E

SDATA .SECTION 'DATA'
DEL0 DB ?
DEL1 DB ?
DEL2 DB ?

SCODE .SECTION AT 0 'CODE'
ORG 00H
MAIN: CLR PAC        ;将PA口设置为输出模式，PAC为输入输出控制寄存器
CLR PA               ;将PA口清零
SET PA.0             ;将PA0设置为1
CALL DELAY           ;调用延时子程序
CLR PA.0             ;将PA0清零
CALL DELAY           ;再次调用延时子程序
JMP MAIN             ;返回主程序，重新执行程序

;*************延时0.1s*******************************
DELAY PROC
    MOV A,100        ;1个指令周期
    MOV DEL0,A       ;1个指令周期
DEL_0:
    MOV A,03         ;1*DEL0
    MOV DEL1,A       ;1*DEL0
DEL_1:
    MOV A,110        ;1*3*DEL0
    MOV DEL2,A       ;1*3*DEL0
DEL_2:
    SDZ DEL2         ;(109*1+2)*3*DEL0
    JMP DEL_2        ;109*2*3*DEL0
    SDZ DEL1         ;(2*1+2)*DEL0
    JMP DEL_1        ;2*2*DEL0
    SDZ DEL0         ;(DEL0-1)+2
    JMP DEL_0        ;(DEL0-1)*2
    RET              ;2
DELAY ENDP

END
```

表 3-1 是一个比较典型的盛群汇编程序结构，我们可以将该程序分为 5 个部分。

(1) 程序头文件和芯片说明。

(2) 程序数据段部分，定义所用到的所有变量。

(3) 程序代码段，包含主要程序代码。

(4) 子函数代码段，包含主程序调用的所有子函数。

(5) END 结束标志，表示程序结束。

上面 5 个部分都是汇编程序的组成部分，但是其中只用(3)和(4)的代码会成为最终的机器码，而其他部分不会直接生成机器码，所以称为伪指令。

(3)和(4)部分的每一条指令的最完整结构又可以包含 4 个部分，如[Name:] OP-Code [operand1 [,operand2]] [;Comment]对应的中文为标记栏、指令栏、操作数栏和注解栏。其中指令栏和操作数栏决定最后生成的机器指令，而标记栏和注解栏都不会生成机器指令。

每一条指令必须包括指令栏，而操作数栏或者注解栏或标号栏都不一定有(不一定有的部分包含在[]之间)。例如子程序 DELAY PROC 的最后一条指令 RET 只有指令栏，而第一条指令 MOV A,100 则包含指令栏和操作数栏；程序中出现的 MAIN 就是标号栏，而所有的汉字注释则是注解栏。下面总结一下指令各部分的典型作用和使用注意事项。

1) 标记栏

标记栏(Name Field)是指某一列指令中的第一个字节，是用来代表该列指令所在的实际程序存储器地址，因此程序设计者只需在程序中以标记代表该指令地址，而不需自己算出跳转目的地实际存储地址。标记栏一定要在每列指令的开头处，且不可有空格。并非每列指令都一定要有标记。HOLTEK 汇编语言中的标记栏可由“A”～“Z”、“a” ～“z”、“0” ～“9”、“?”、“—”、“@”符号组成，但是第一个字符不得为数字，而且编译器只识别前面 31 个字符。

2) 指令栏

指令栏(OP-Code Field)是一个指令列的主体，它的位置是在标记栏之后，空一格以上的位置，它指出 CPU 做什么事，如传送指令、加减运算指令、位设定指令等。其中伪指令也写在这个字节中，如 EQU、ORG、IF、END 等伪指令。

3) 操作数栏

操作数栏(Operand Field )在指令栏之后空格以上的位置开始，操作数栏指出指令运算的对象，因此根据指令类型的不同，操作数的个数可能是一个、两个或零个。

4) 注解栏

注解栏(Comment Field)其实不是程序的一部分，它保留给程序设计者对某一行指令做文字注解来说明程序的功能，以增强程序的可读性。而习惯上注解栏写在操作数栏之后，因此在编译时，编译器将不会管注解栏识别字符分号“;”之后的文字。注解的文字叙述可以写在任何地方，如果能在程序中加上适当的注解，将能更有效地维护程序。

本书在“单片机开发语言”中对盛群单片机汇编语言和 C 语言有进一步讲解，但是如果

有一定C语言基础并学过微型计算机原理等课程的话，并不影响读者理解、完成前几章的各个实验(阅读汇编程序时，可以参考第7章的盛群汇编语言速查表)。

## 3.2 IDE 3000简介

在后续的实验中我们将使用盛群公司的HT-IDE 3000编辑、编译、调试及烧录程序。所谓IDE就是英文intergrated development enviroment的字头缩写，中文一般翻译成集成开发环境，HT则是盛群公司的拼写字头。前面已经讲过，单片机的程序设计可以使用汇编或者C语言进行，最后必须翻译成面向机器的机器指令，也就是一串二进制0、1序列。这些过程需要各种工具软件，例如，类似Word的文字编辑器供我们编写程序，对程序进行检查，看是否符合语法，将程序编译成机器语言，利用调试工具对程序进行仿真等。而集成开发环境就容纳了上述各种工具软件，并以菜单的方式供大家使用，非常便利。此外，HT-IDE 3000中还包含了将程序烧录到芯片中的程序，可以说HT-IDE 3000包括了所有开发盛群单片机的工具软件。此外，需要注意的是，HT-IDE 3000的版本更新较快，如果开发较新型的盛群单片机需要从网络下载其最新版(http://www.holtek.com.cn/china/tech/updates/ht-ide.htm)。

HT-IDE 3000中使用的文件各种各样，例如汇编或者C语言程序翻译成的机器码序列，以及各种各样的中间格式文件，如列表文件和map文件等。理解这些概念对于非计算机专业的同学稍有困难，但是这些概念可以帮助大家写出更加有效率的程序，所以不妨参考其他书籍看一下。本书在用到这些概念的时候再详细讲述。既然开发一个单片机项目涉及种类繁多的多个文件，那么非常有必要将这些文件有序地组织起来。因此，在IDE 3000中使用工程(project)将一个项目的所有文件组织和整合。一般在开始一个新项目的时候新建一个工程，最好为这个工程单独建立目录，并选定芯片型号和配置信息，此时的工程没有包含源程序；然后编写源程序(汇编程序或C语言程序)，将源程序添加到工程中。这样就可以编译该项目了。如果编译成功，在该工程目录下生成可以烧录的机器码(对于46F49E而言，后缀为".mtp")。下面结合IDE 3000的屏幕截图进一步描述该过程。(下面内容建议配合多媒体教程IDE 3000的使用视频教程一起学习。)

在Windows"开始"菜单下，打开HT-IDE 3000。注意，系统首先弹出一个对话框，提示已连接硬件仿真器，如图3-1所示。如果没有连接仿真器就会提示你进行连接。HT-IDE 3000是被设计来与盛群公司的硬件仿真器HT-ICE一起使用的，默认情况两者应同时工作。但是在没有硬件仿真器的情况下，也可以单独使用IDE 3000进行程序的编写、编译，也可以生成可烧录的机器码文件，但是没有办法使用硬件仿真和程序烧录。

由于HT-IDE3000的开发环境是以工程的方式管理用户的文件，所以开发一个新的单片机应用应该新建一个工程文件，单击"工程"菜单下的"新建工程"选项，弹出如图3-2所示的对话框，在第一个空白栏里填写工程名字；第二个是选项栏，用于选择存储路径，好的习惯是为新工程单独设置一个目录；第三个用于选择所用芯片型号，如本实验中所用芯片是HT46F49E，所以就选择这个型号；最下面的选项栏有两个选项，一个是默认的普通编译器，

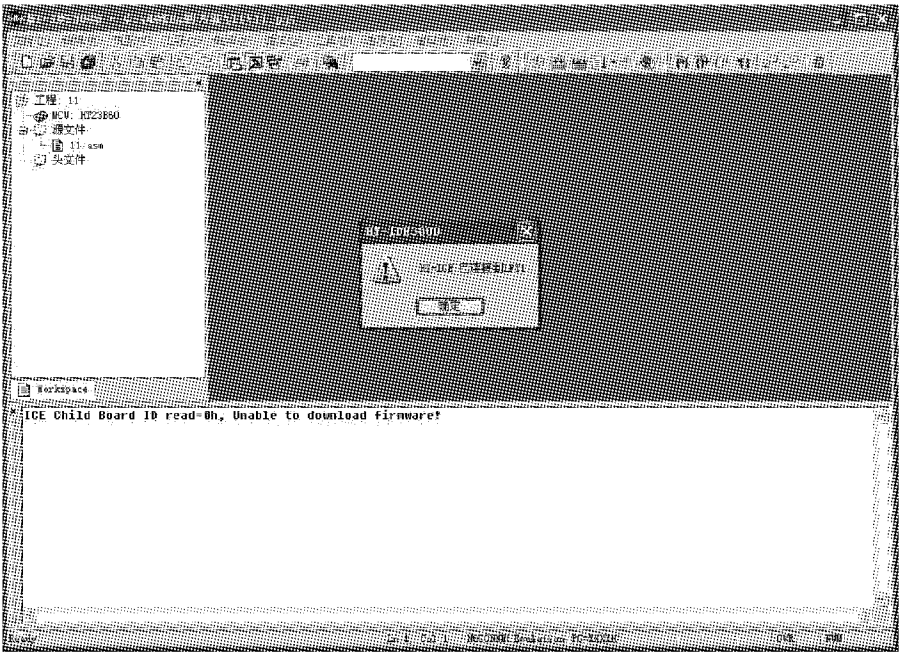


图 3-1　提示已连接好仿真器

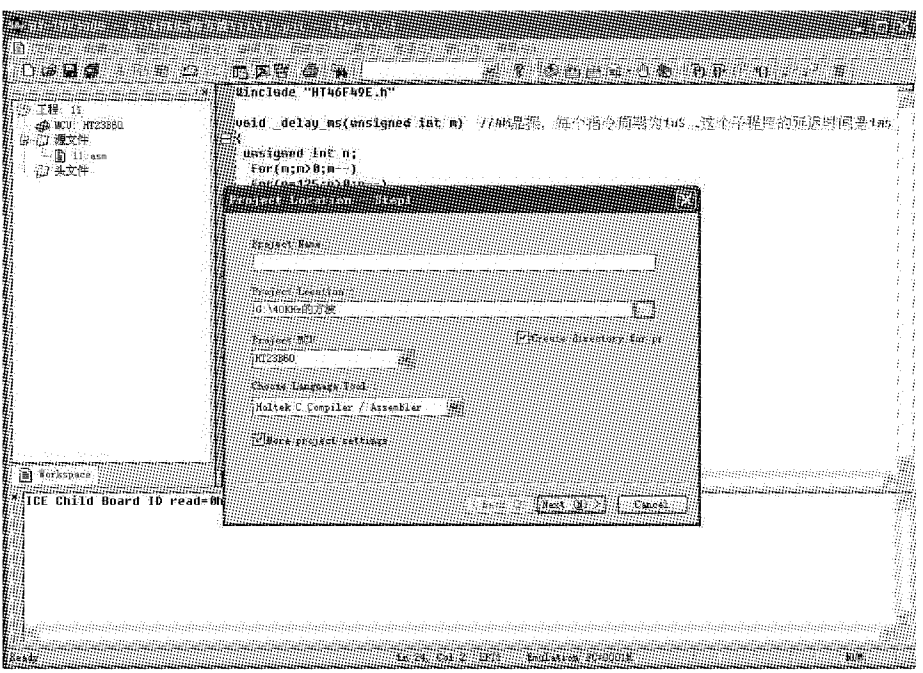


图 3-2　新建工程

另一个是加强版的，本实验选择默认的第一个选项。设置完成后单击“Next(N)”按钮就会出现图 3-3 所示的界面。

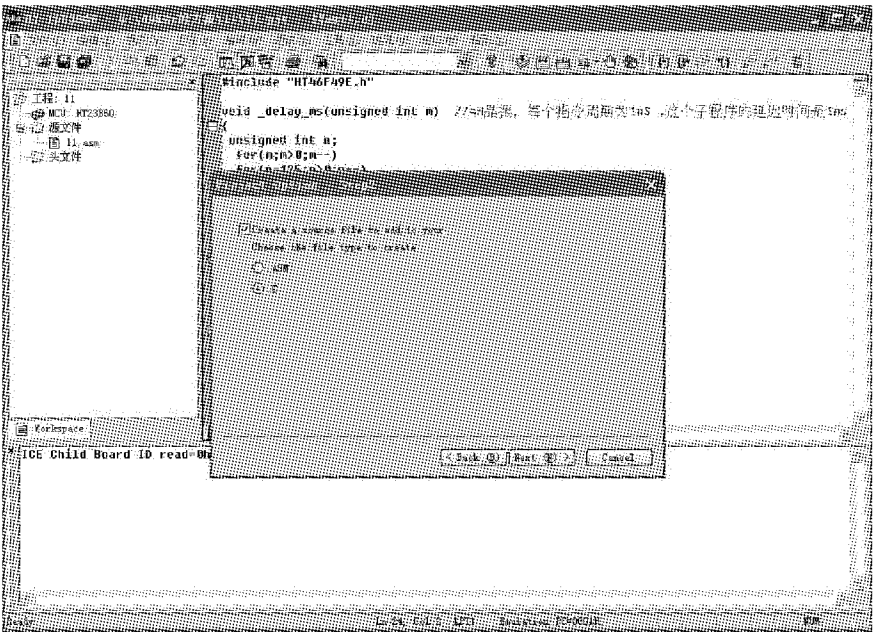


图 3-3　工程语言选择

如图 3-3 所示的界面里有两个选项，根据自己所用语言选择。如果使用汇编语言，就选择“. ASM”；如果使用 C 语言，就选择“. C”。单击“Next(N)”按钮就会出现如图 3-4 所示的界面，IDE 3000 自动建立了一个源文件(根据上一步的选项，或是汇编或是 C)，默认情况下工程名字就是源文件的名字，也可以为源文件另外起文件名和头文件名。

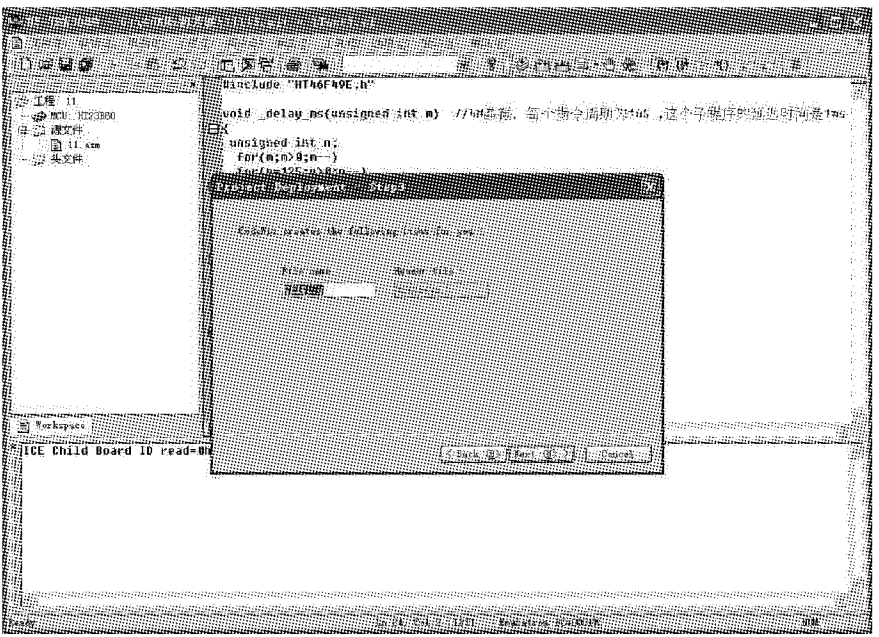


图 3-4　语言选择

图 3-5 中的选项是用来根据自己的需要设置芯片的，4 个 I/O 口有的需要接上拉电阻，根据实际情况选择，其中很多都可以选择默认的。注意，如果要使用外部中断就必须进行设置，因为外部中断默认的是禁止，根据使用情况选择所需触发方式。

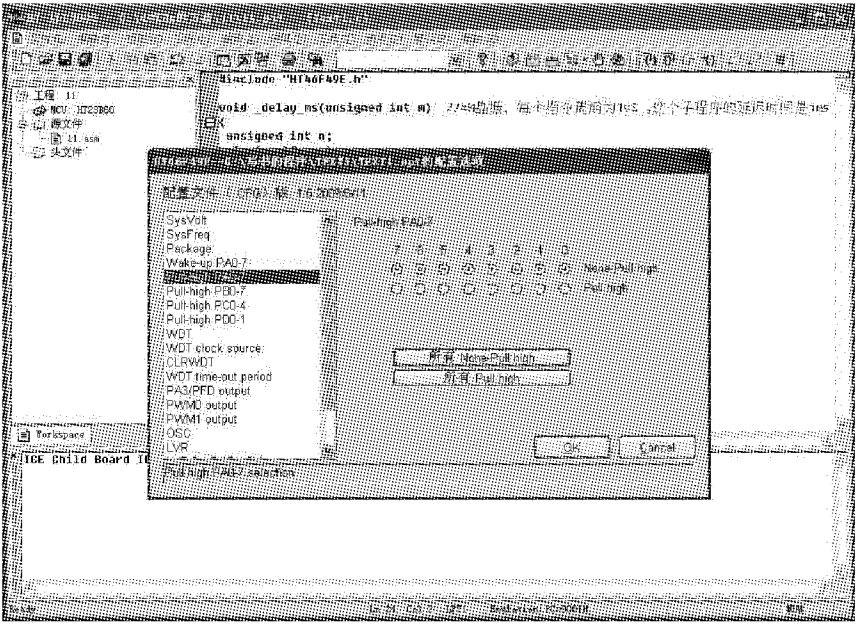


图 3-5　配置信息

待芯片配置设置好之后，单击“确定”按钮便会出现如图 3-6 所示的界面。下面是该界面中的选项介绍。

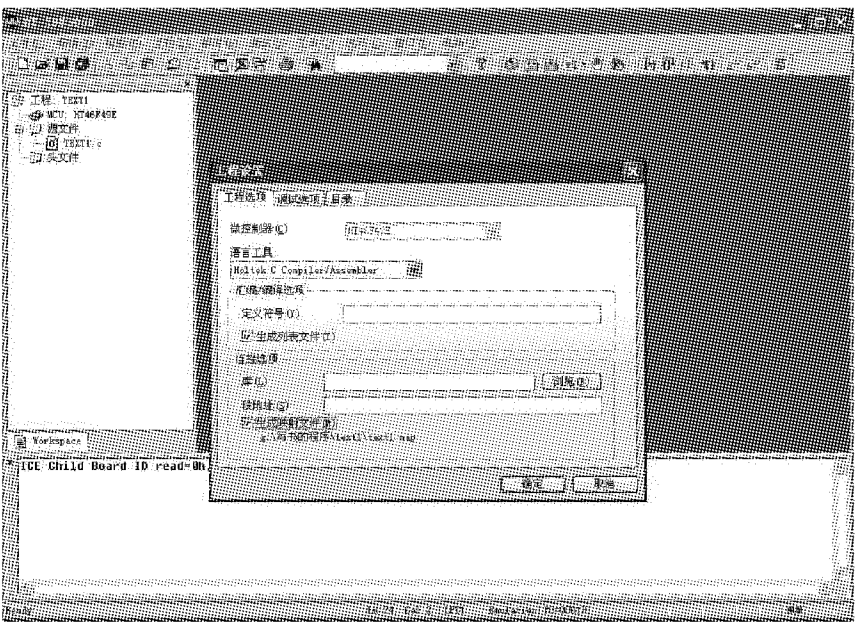

图 3-6　其他选项设置

(1)“生成列表文件(I)”：编译以后会产生列表文件，列表文件通常会提供一些值得参考的信息，建议读者选中此项。

(2)“库(L)”：制定函数库的所在地址及文件名，若程序中未使用任何函数库，即可不指定。

(3)“段地址(S)”：制定程序段的地址。一般不需指定，HT-IDE 3000 会依程序中的 SECTION 命令设定地址。

(4)“生成映射文件(M)”：产生标号(Name、Lable)的对应地址，这些信息将存放在 .map 文件中。

这些文件都是产生烧录程序中的各种中间结果，在初步学习时可以忽略这些文件。

设置好后首先新建文件，如图 3-7 所示。然后在空白处编辑自己的程序，如果程序事先已经写好就可以直接打开。程序编辑完成后保存至自己所设定的地址单元，值得注意的是文件名设置好之后一定要加上后缀名，若使用汇编语言就加上“.asm”，例如“TEXT.asm”；若使用 C 语言就加上“.c”，例如“TEXT.c”，大小写都可以。若是不加后缀名，仿真时将会无法辨认文件，最后将文件加入工程，如图 3-8 所示。

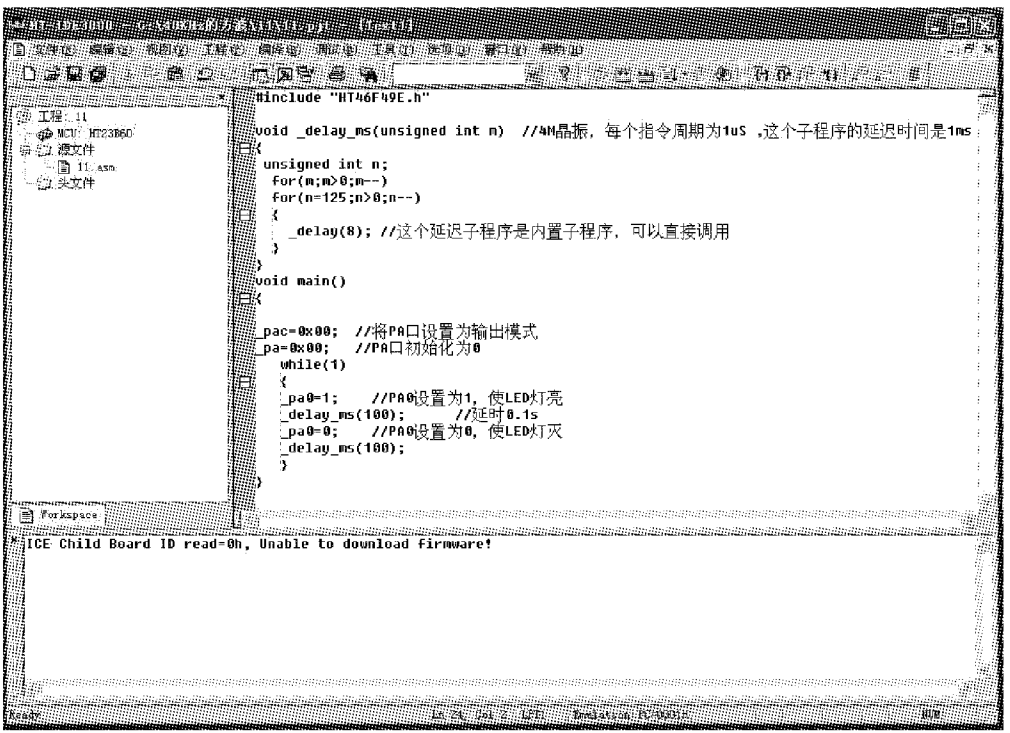


图 3-7　新建文件

在没有语法错误的情况下编译程序，就可以生成最终的烧录文件 *.mtp。“*”代表工程的名字。

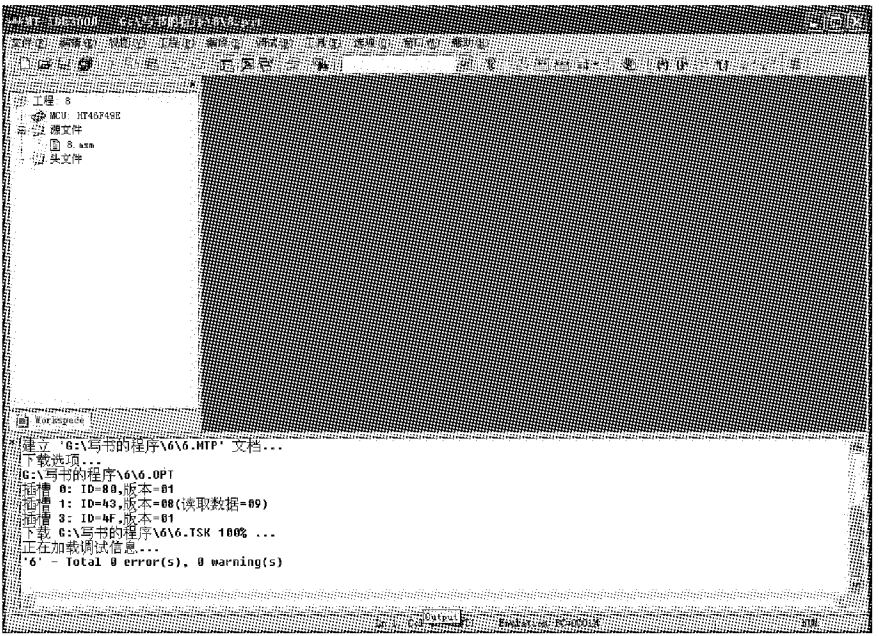


图 3-8　将文件加入工程

## 3.3　硬件仿真器

与 IDE 3000 配套使用的还有硬件仿真器 HT-ICE、仿真器接口板和相应连接电缆。所谓仿真器，就是通过仿真头来代替的在目标板上的单片机芯片，关键是不用反复烧录程序，不满意随时可以修改，可以单步运行，指定断点停止，等等，调试方面极为方便。

仿真器内部硬件资源和被仿真的单片机基本是完全兼容的。仿真主控程序被存储在仿真器芯片特殊的指定空间内，有一特殊的地址段用来存储仿真主控程序，仿真主控程序控制仿真器的正确运转。仿真器和电脑的上位机软件通过通信接口相连，负责接收电脑发出的仿真命令，或返回运行结果和单片机的运行状态信息。由仿真器内部的仿真主控程序负责执行接收到的数据，并且进行正确的处理。进而驱动相应的硬件工作，这其中也包括把接收到的程序存放到仿真器芯片内部用来存储可执行程序的存储单元，这样就实现了类似编程器反复烧录来试验的功能。不同的是，通过仿真主控程序可以做到让这些目标程序做特定的运行，比如单步、指定断点、指定地址的运行等，并且可以实时观察到单片机内部各个存储单元的状态。仿真器和计算机主机联机后就像是两个精密的齿轮互相咬合的关系，一旦强行中断这种联系(比如强行给仿真器手动复位或拔去联机线等)，计算机就会提示联机出现问题，这也体现了硬件仿真的鲜明特性，即“所见即所得”。这些都是编程器无法做到的。这些给调试、修改，以及生成最终程序创造了比较有力的保证，从而实现较高的效率。

图 3-9 最右边为 HT-ICE 仿真器，中间为接口板，接口板通过连接电缆和调试目标板相连。由于 HT-ICE 被设计来仿真所有 HT 单片机，而这些单片机管脚差异大，所以 HT-ICE 带有接口板，上面有各种盛群单片机的接口。如果仿真 HT46F49E，则需要用 28 腿连接电缆将接口卡上标有 HT46X49-28 的芯片插座与目标板上芯片插座相连。这样，仿真插头就

代替了目标板上的单片机。注意，HT-ICE 和计算机连接是通过打印口(并口)连接的，但是很多笔记本电脑或新型的台式机没有该端口。因此期待盛群公司推出 USB 口的仿真器。

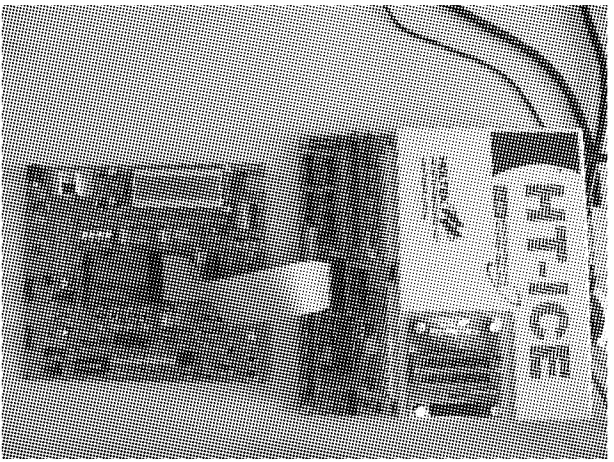

图 3-9　HT-ICE 仿真器和接口板及调试目标板

正确接入目标板后，就可以使用 IDE 3000 中的调试功能进行程序调试了。其中各个子菜单功能，比如单步运行、连续运行、设置断点等，和大家熟悉的高级 C 语言调试环境非常类似，可以结合配合本书的多媒体教程《单片机程序的调试》进一步学习。

## 3.4　程 序 调 试

程序的编写经常不是一帆风顺的，需要不断地调试，因此开发工具需要有一种程序的调试模块，盛群单片机的开发工具 IDE 3000 自然也含有此类功能，下面简要介绍程序的调试过程，程序编写调试窗口如图 3-10 所示。

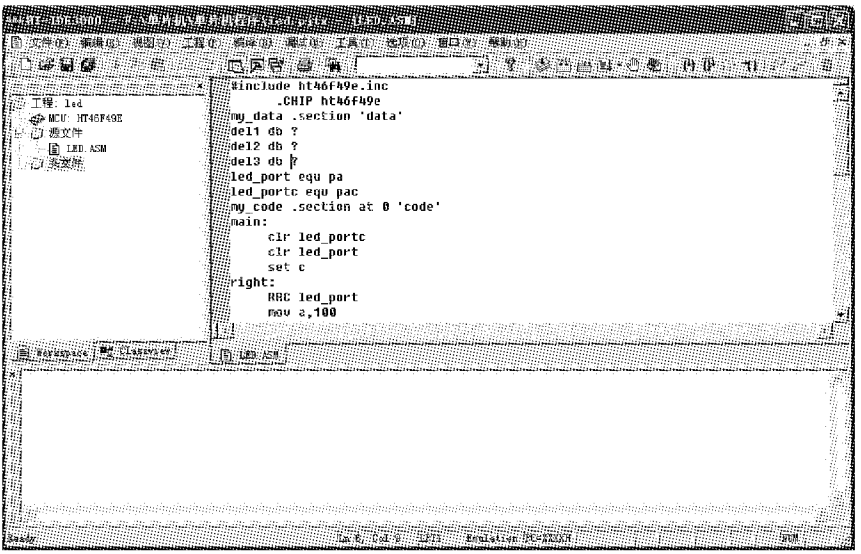


图 3-10　程序编写调试窗口