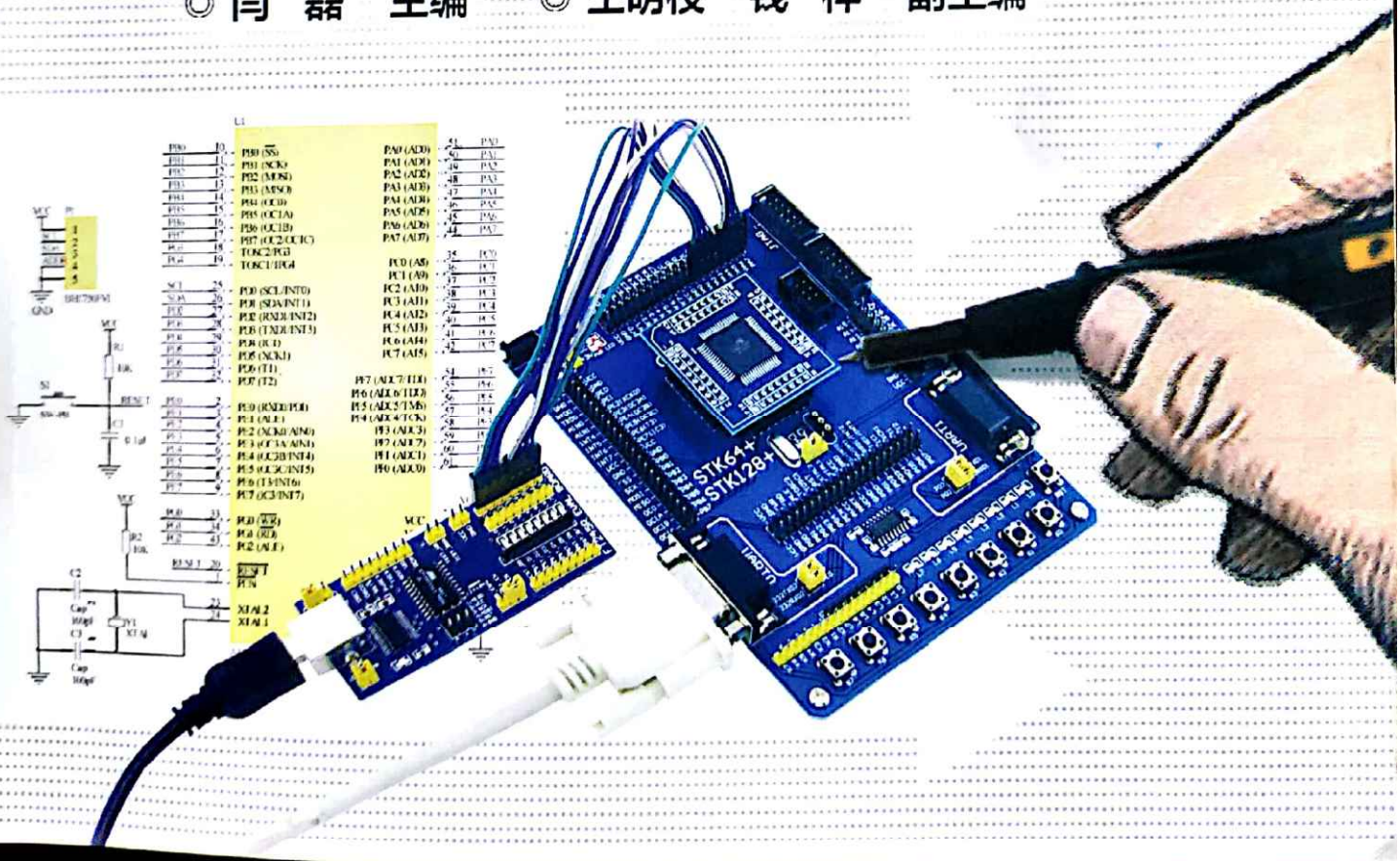
手把手教你学

手把手教你学会AVR单片机

◎闫 磊 主编 ◎王明枝 钱 桦 副主编



電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

4.2.2 CodeVision AVR C 语言编程应用

1. 创建新文件

安装好 CVAVR 后，双击桌面快捷方式图标或从“开始\程序\CodeVisionAVR\CodeVisionAVR C Compiler”启动 CVAVR，出现如图 4.2.3 所示的界面。

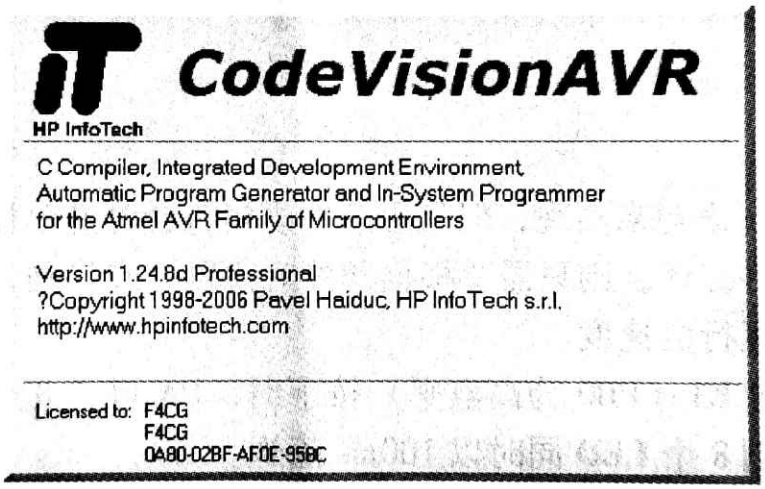


图 4.2.3 CVAVR 的启动界面

待启动完成后，出现如图 4.2.4 所示的界面，执行 File | New 菜单命令，弹出如图 4.2.5 所示的新建工程项目对话框。

待启动完成后，出现如图4.2.4所示的界面，执行File | New菜单命令，弹出如图4.2.
所示的新建工程项目对话框。

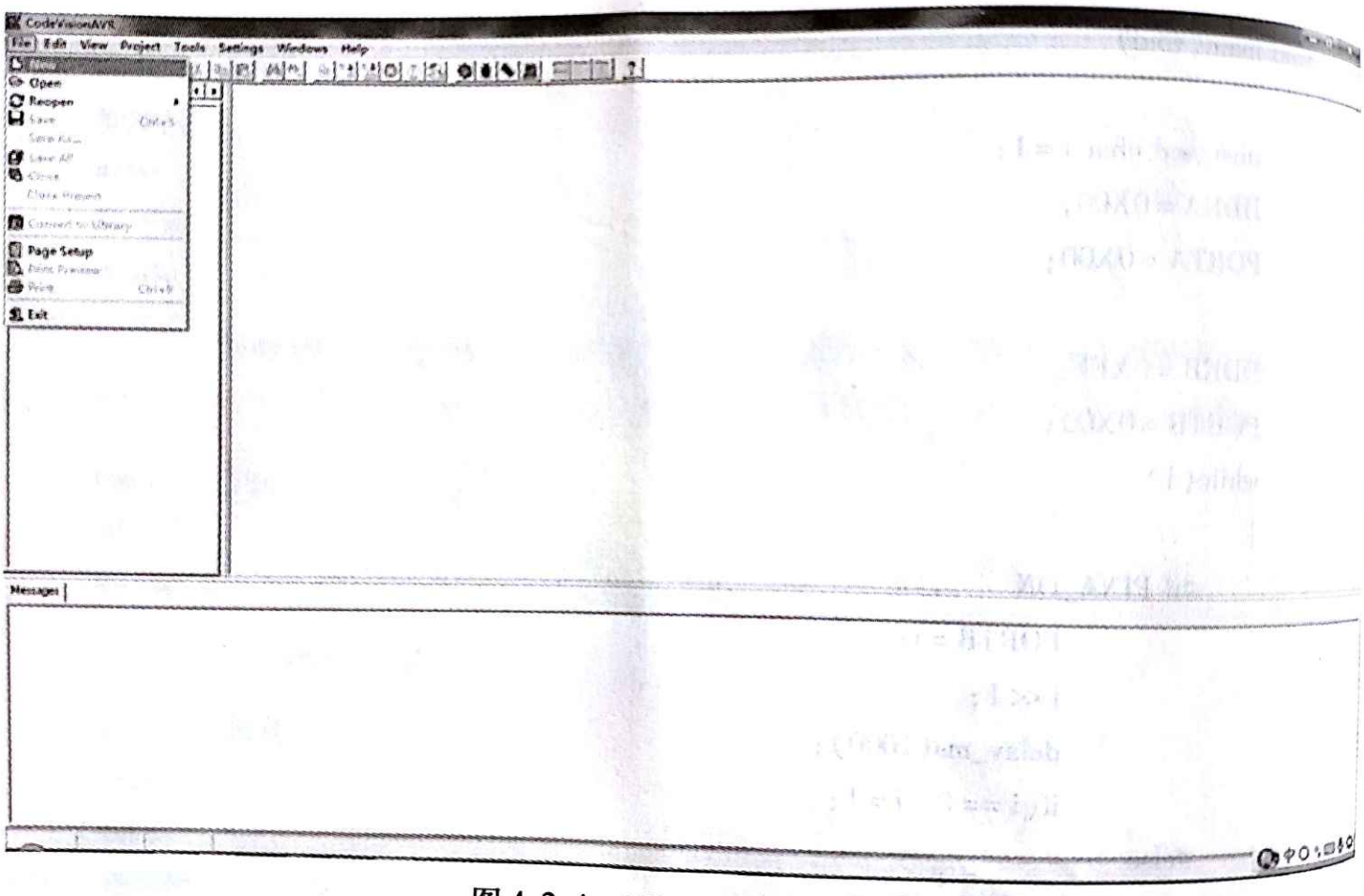


图4.2.4　CVAVR初始工作界面

在弹出的这个对话框里面，先单击“Source”再单击“OK”按钮，新建一个名为 untitled. c 的源文件，如图 4. 2. 6 所示。执行 File | Save As 菜单命令，保存该源文件。在此，我们可以编写源程序代码，通常的软件代码主要在该编辑窗口中完成。在输入软件代码过程中，CVAVR 提供了一种很好的工具：Code Templates 即代码模板。我们在图 4. 1. 2 所示的 CVAVR 工作初始界面的导航栏上就可以找到 Code Templates。该工具提供了一些软件编程中常用的模板，如 if…else；while；for 等语句。在需要时，只需在 Code Templates 选中相应的模板，然后在 Code Templates 区右击，选择“Copy to the Edit Window”命令，即可将该模板复制到文件编辑区中。如图 4. 2. 7 所示，当编写代码需要用到 if…else 语句时，在 Code Templates 里选中此模板，执行复制命令即可将 if…else 语句复制到编辑区中。用户也可以将常用的代码段做成模板，用图 4. 2. 7 中的“Paste”命令放入 Code Templates 区；也可以用“Delete”命令删除不再常用的模板；还可以用“Move Up”和“Move Down”命令改变模板在 Code Templates 区的先后次序。

在弹出的这个对话框里面，先单击“Source”再单击“OK”按钮，新建一个名为 untitled.c 的源文件，如图 4.2.6 所示。执行 File | Save As 菜单命令，保存该源文件。在此，我们可以编写源程序代码，通常的软件代码主要在该编辑窗口中完成。在输入软件代码过程中，CVAVR 提供了一种很好的工具：Code Templates 即代码模板。我们在图 4.1.2 所示的 CVAVR 工作初始界面的导航栏上就可以找到 Code Templates。该工具提供了一些软件编程中常用的模板，如 if…else；while；for 等语句。在需要时，只需在 Code Templates 选中相应的模板，然后在 Code Templates 区右击，选择“Copy to the Edit Window”命令，即可将该模板复制到文件编辑区中。如图 4.2.7 所示，当编写代码需要用到 if…else 语句时，在 Code Templates 里选中此模板，执行复制命令即可将 if…else 语句复制到编辑区中。用户也可以将常用的代码段做成模板，用图 4.2.7 中的“Paste”命令放入 Code Templates 区；也可以用“Delete”命令删除不再常用的模板；还可以用“Move Up”和“Move Down”命令改变模板在 Code Templates 区的先后次序。

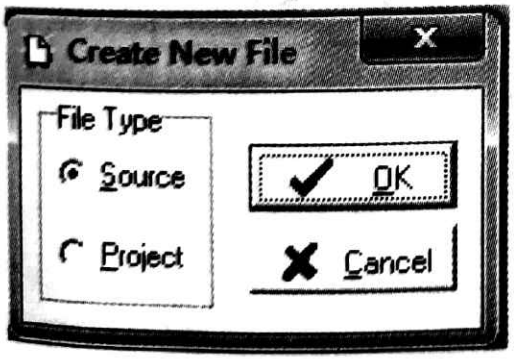


图 4.2.5　新建工程项目对话框

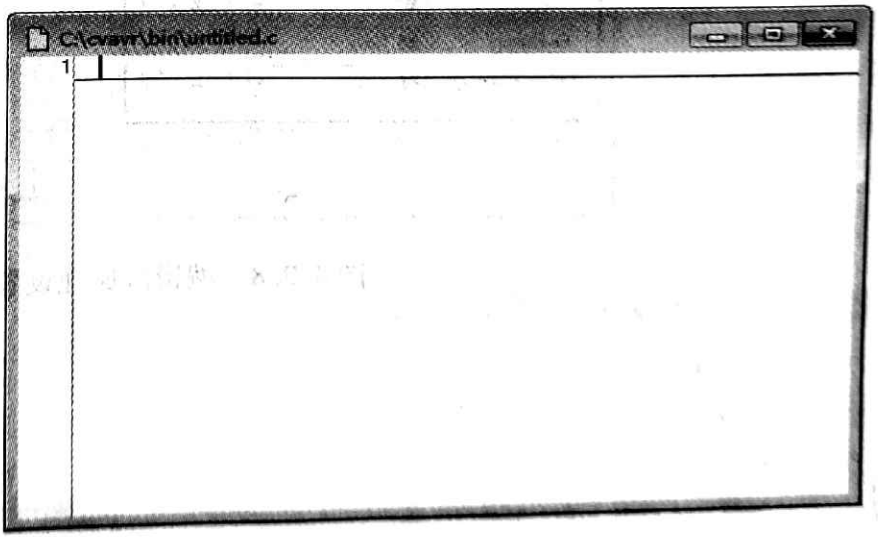


图 4.2.6　新建的源文件

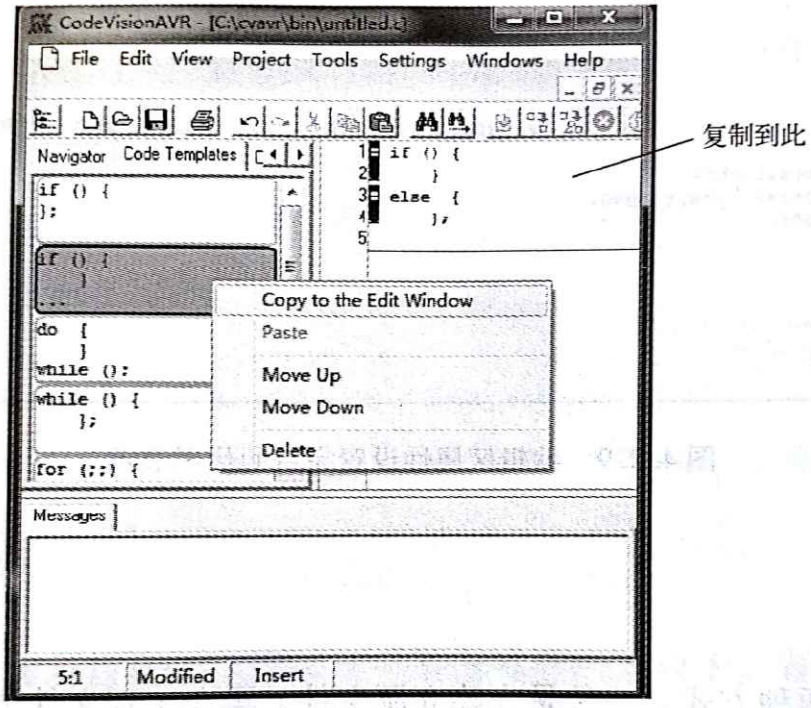


图 4.2.7　将代码模板复制到文件编辑区

在代码区编写使 8 个 LED 灯循环点亮的跑马灯实验的代码，写完后执行 File | Save As 菜单命令将其保存到指定的路径下。选择好路径后，在“文件名”文本框中输入新建源文

件的名称，单击“保存”按钮后完成源文件的新建。编辑区内不同部分的字体和颜色等默认的属性是可以改变的，执行 Settings | Editor 菜单命令，可弹出如图 4.2.8 所示的属性设置界面，可在此界面中重新进行设置。设置完的代码显示如图 4.2.9 所示。

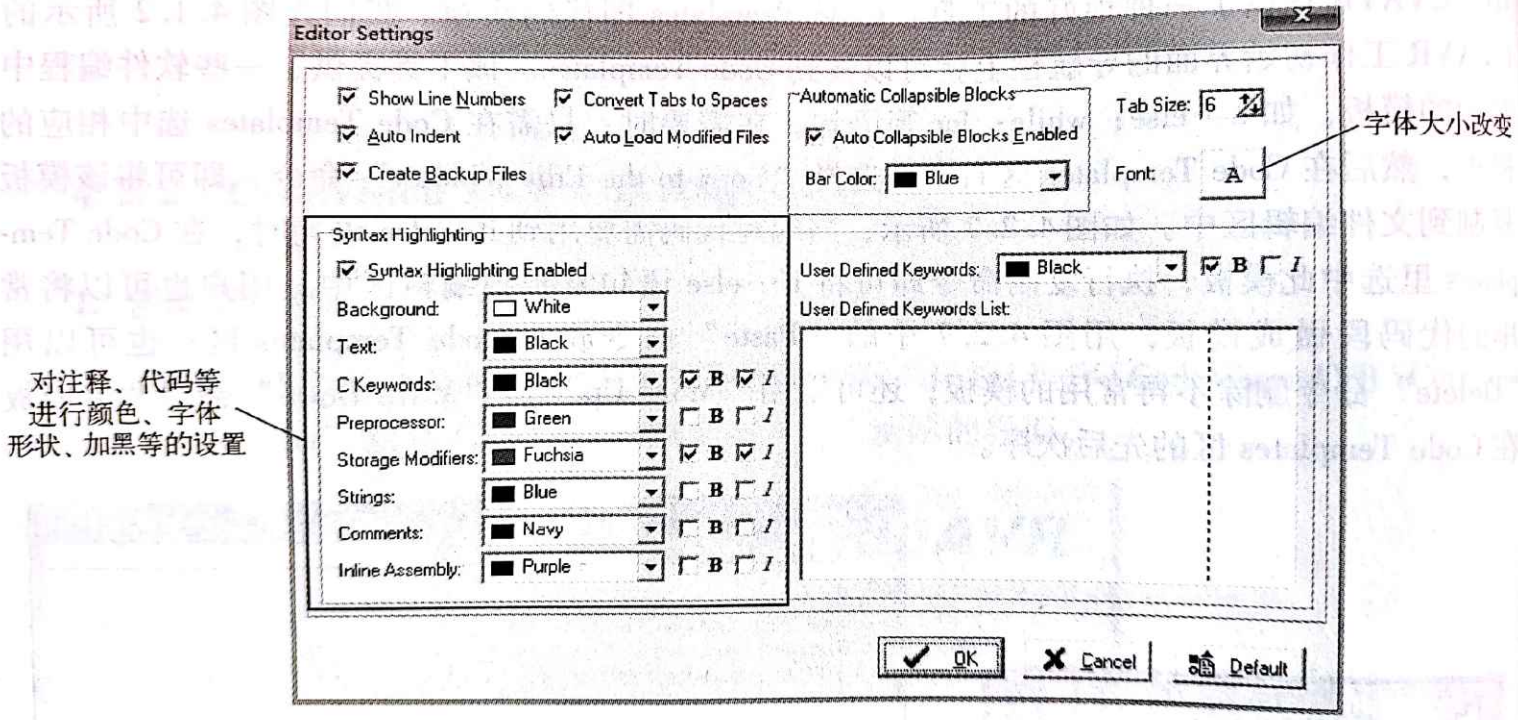


图 4.2.8　编辑区属性设置界面

```
/*******跑马灯实验********/
#include<mega128.h>
#include<delay.h>

void main(void)
{
        unsigned char i,position=0;
        PORTA=0XFF;
        DDRA=0XFF;
        PORTB=0X00;
        DDRB=0XFE;


        while(1)
        {
           i=PINB&0XFE;
           if(i==0XFE)                          //当PB0口的按键按下之后，8个LED灯每个1s点亮，如此循环
           {
                PORTA=~(1<<position);
                if(++position>=8) position=0;
                delay_ms(1000);
           }
           else
           {
                PORTA=0XFF;
           }
        }
}
```

图 4.2.9　编辑区属性设置完后的代码显示

2. 创建新工程

建立工程项目有两种方式，一种是先建立源文件，再向工程文件中添加源文件；一种是直接通过代码生成器 CodeWizardAVR 建立工程项目。

1）向工程项目中添加源文件 当出现图 4.2.5 所示对话框时，选择“Project”选项，然后单击“OK”按钮，会弹出如图 4.2.10 所示的对话框。此对话框是用于确认正在新建一

个项目文件，并询问是否使用代码生成器。如果选择，则单击“Yes”按钮；如果不选择，则单击“No”按钮。

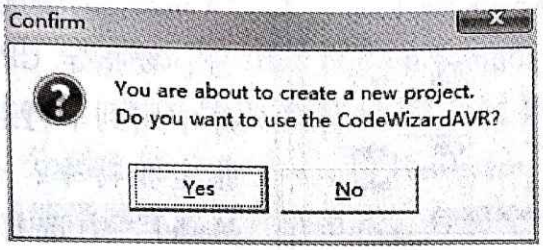


图 4.2.10　是否选择代码生成器

单击“No”按钮，则会打开“Create New Project”对话框，如图 4.2.11 所示，用来选择新建工程项目的保存路径。选定路径后，在“文件名”文本框中输入新建工程项目的名称，单击“保存”按钮后完成工程项目的新建。本例中，新建一个名为“Timer”的文件夹，然后将新建的工程项目命名为 Timer.prj，并保存到 Timer 文件夹中。

完成图 4.2.11 所示的对话框设置后，单击“保存”按钮，会弹出如图 4.2.12 所示的工程项目配置对话框。如果对一个已经创建的工程项目进行配置，可先执行 File | Open 菜单命令，打开需要配置的工程项目，再执行 Project | Configure 菜单命令，也可打开如图 4.2.12 所示的配置对话框。

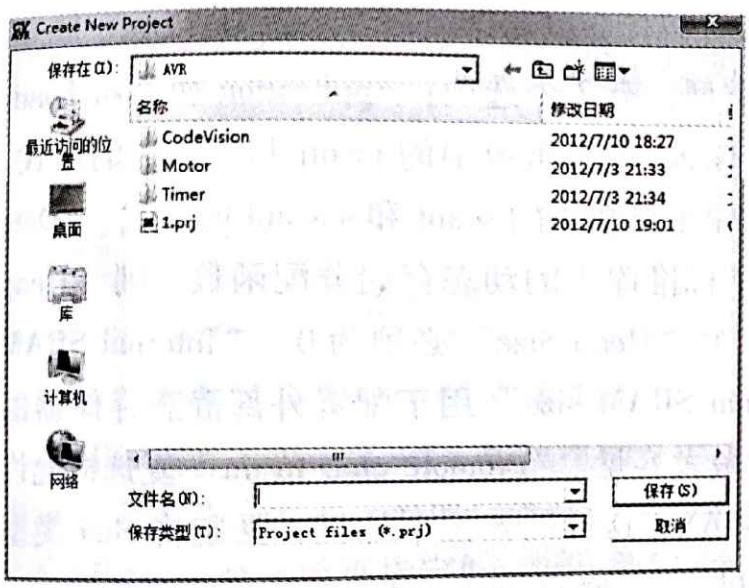


图 4.2.11　选择保存路径对话框

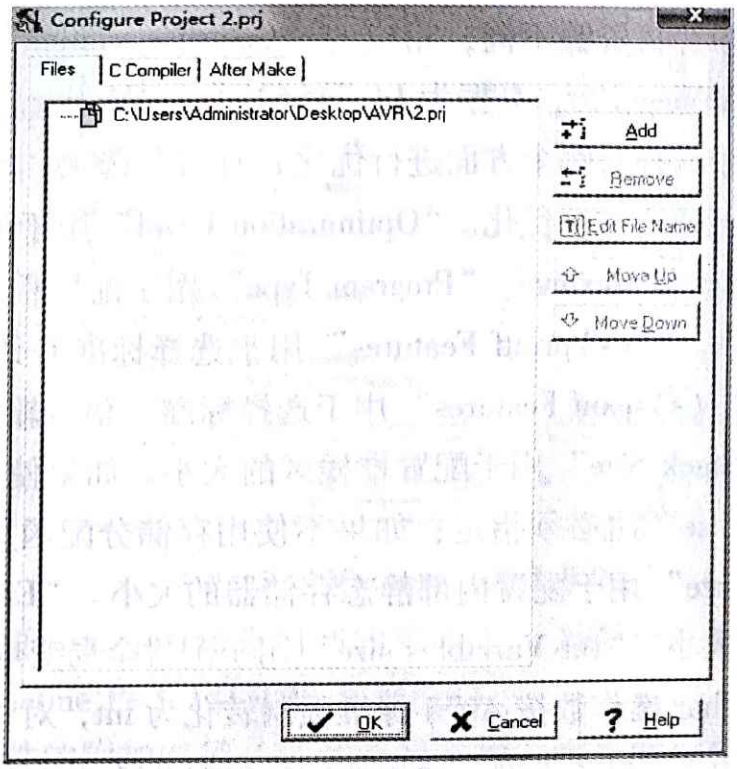


图 4.2.12　工程项目配置对话框

(1) 项目配置选项卡。

☺ Files

单击“Add”按钮可以把源文件添加到当前打开的工程中。首先添加到工程中的文件为主工程文件，此文件总是被 Make，其他添加到工程中的文件每次会被自动连接到主工程文件。工程被打开时，所有工程文件会在编辑器中打开。选中一个文件，然后单击“Remove”按钮会将此文件从工程中去除。单击“OK”或“Cancel”按钮则改变可被保存或忽略。

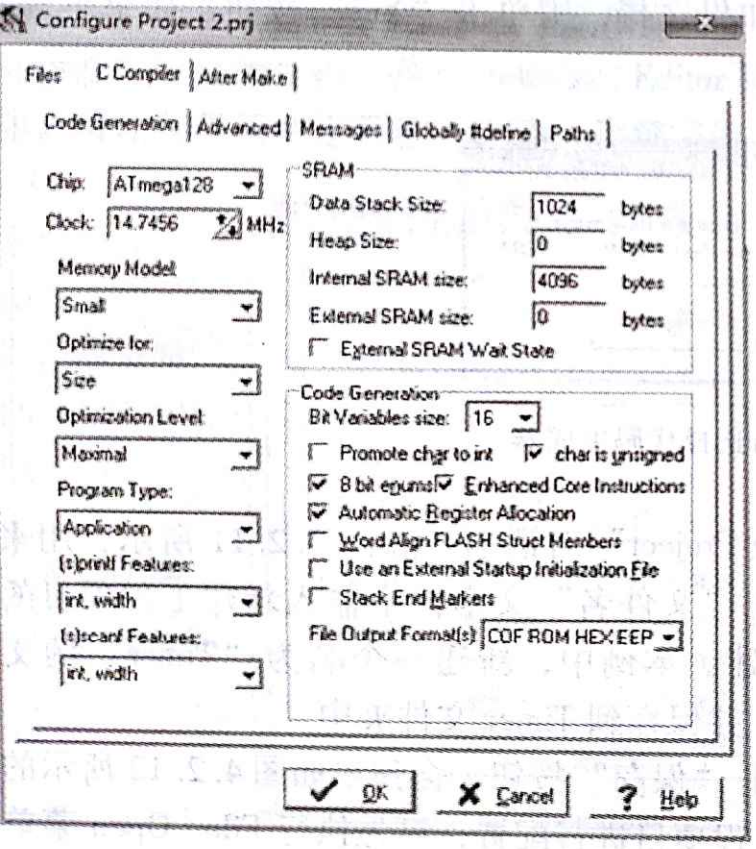


图 4.2.13　ATmega128 代码产生配置界面

☺ C Compiler

设定 C 编译器选项时，可以通过“Chip”选项选择目标 AVR 微控制器芯片型号；要指定 CPU 时钟频率（MHz）；如果程序中用到串行通信，还必须初始化 UART 并指定波特率；存储器模式可以在“Memory Model”中选择；必须指明数据堆栈大小（Data Stack Size）；如果使用外部 SRAM，最后还要指明外部 SRAM 大小（External SRAM Size）。

在 ATmega128 代码配置界面，有 5 个选项卡：“Code Generation”、“Advanced”、“Messages”、“Globally #define”和“Paths”，见图 4.2.13。

“Code Generation”主要用于设置 AVR 单片机的型号、工作频率、存储器大小、优

“Code Generation”主要用于设

图 4.2.13 ATmega128 代码产生配置界面

单片机的型号、工作频率、存储器大小、优化选项及代码产生选项等。在“Chip”下拉菜单中选择自己要使用的单片机型号。不同的单片机资源不同，相应的“Code Generation”界面也会有所不同。我们所使用的单片机为 ATmega128，晶振为 14.7456MHz。“Memory Model”用于配置存储模式。“Optimize for”用于选择对哪个方面进行优化，有两个选项：Size 和 Speed。编译程序可对最小容量和最快执行速度进行优化。“Optimization Level”用于配置代码优化的程度，有 3 个选项：Low、Medium 和 Maximal。“Program Type”用于配置程序类型，有 2 个选项：Application 和 Boot Loader。“(s)printf Features”用于选择标准 C 语言输入/输出函数中的 printf 和 sprintf 的形式。“(s)scanf Features”用于选择标准 C 语言输入/输出函数中的 scanf 和 s scanf 的形式。“Data Stack Size”用于配置堆栈区的大小。如果使用了标准库中的动态存储分配函数，则“Heap Size”也必须指定；如果不使用存储分配函数，则“Heap Size”必须为 0。“Internal SRAM size”用于配置内部静态存储器的大小。“External SRAM size”用于配置外部静态存储器的大小。“Bit Variables size”用于配置全局变量的最大容量。“Promote char to int”复选框允许 char 操作数按 ANSI 标准强制转化为 int，对于像 AVR 这样的 8 位单片机，强制将 char 类型转换为 int 类型，会增加代码容量并降低运行速度。“char is unsigned“复选框用于选择 char 类型是否当作无符号数处理。如果选中该复选框，则编译器将 char 类型当作无符号 8 位数(0～255)；如果没有选中该复选框，则编译器将 char 类型当作有符号 8 位数(－127～127)。这个选项还可以使用#pragram uchar 编译器指令来指定。将 char 作为无符号数来处理可以减小代码容量，提高运行速度。“8 bit enums”复选框用于配置 enumerations 类型是否当作 8 位 char 类型处理。将 enumerations 类型当作 8 位 char 类型处理有助于优化代码容量和提高运行速度。“Enhanced Core Instructions”复选框用于允许或禁止使用新的 ATmega 和 AT94K FPSLIC 器件的增强型内核指令。“Automatic Restiger Allocation”复选框允许 R2～R14 和 R16～R21 的分配，按照这种方法，16 位变量将更适合放在偶寄存器对中，这样增强型内

核指令 MOVW 对它们的访问将更有效。该选项只有在选中了 Enhanced Core Instructions 复选框时才有效。如果“Automatic Restiger Allocation”复选框没有被选中，则寄存器将按变量声明的顺序分配。如果程序是使用 CVAVR V1.25.3 版本开发的，则“Automatic Restiger Allocation”复选框必须禁止。选中“Use an External Startup Initialization File”复选框可使用外部启动文件。“Stack End Markers”选项是用来调试的。如果选择，则编译器将字符串 DSTACKEND 和 HSTACKEND 分别放到 Data Stack 和 Hardware Stack 区的最后。“File Output Format（s）”列表框用于选择编译器生成文件的格式，有两个选项：COF ROM HEX EEP 和 OBJ ROM HEX EEP。

“Advanced”配置界面如图 4.2.14 所示，它可以对中断向量表和数据堆栈、全局变量、硬件堆栈和出栈编译的静态地址进行设置。

“Messages”配置界面如图 4.2.15 所示，选择前面的复选框，可以关闭各种类型的

“Messages”配置界面如图 4. 2. 15 所示，选择前面的复选框，可以单独允许或禁止各种编译器和链接器警告。

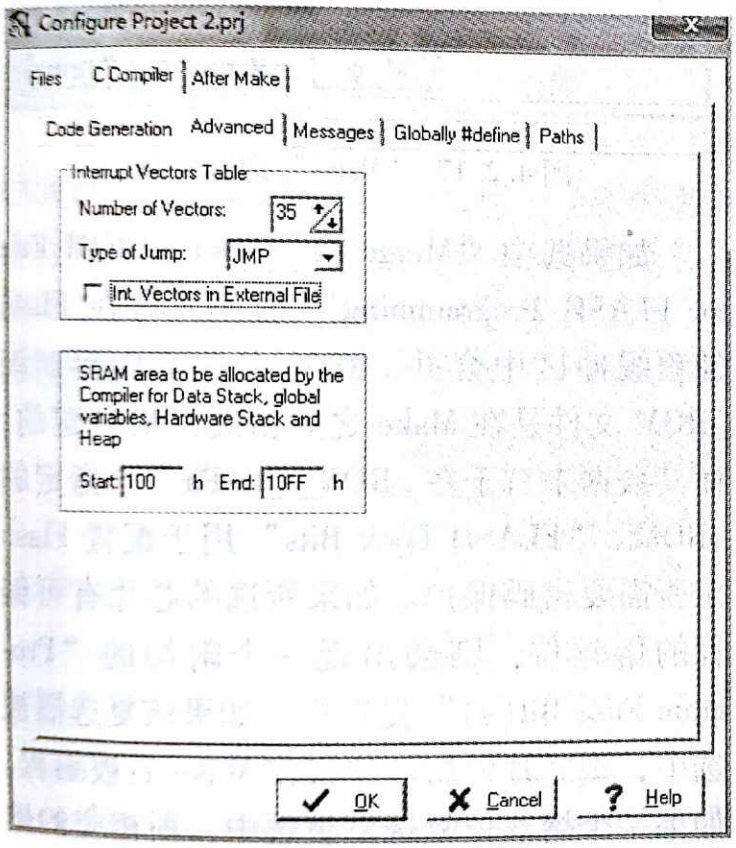


图 4. 2. 14 “Advanced”配置界面

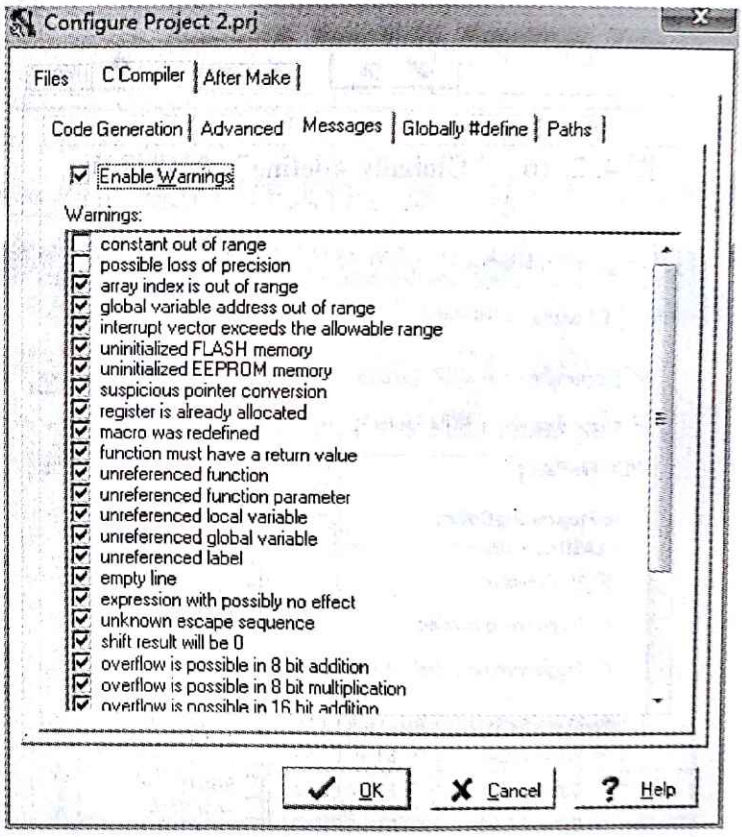


图 4. 2. 15 “Messages”配置界面

“Globally #define”用于配置在所有的工程项目文件中都能用到的宏定义。如图 4.2.16 所示的宏定义，与在工程项目的每个文件中做#define PI 3.1415826 是等价的。

“Paths”配置用于指定#include 和 library 文件的附加路径。这些路径必须在相应的编辑控制区每行输入一个，如图 4.2.17 所示。

☺ After Make

“After Make”选项卡主要用于配置编译之后 CVAVR 执行的动作。单击“After Make”选项，在这个界面上有两个复选框：一个是“Program the Chip”，一个是“Execute User's Program”。如果选中“Program the Chip”复选框，则表示编译成功后，程序将被内嵌的编程软件自动传送到 AVR 芯片。每新建一个项目，在程序编译完下载到 AVR 单片机之前，都必须在 Configure/After Make 中选中“Program the Chip”和“Execute User's Program”选项。选中该复选框的配置界面如图 4.2.18 所示。

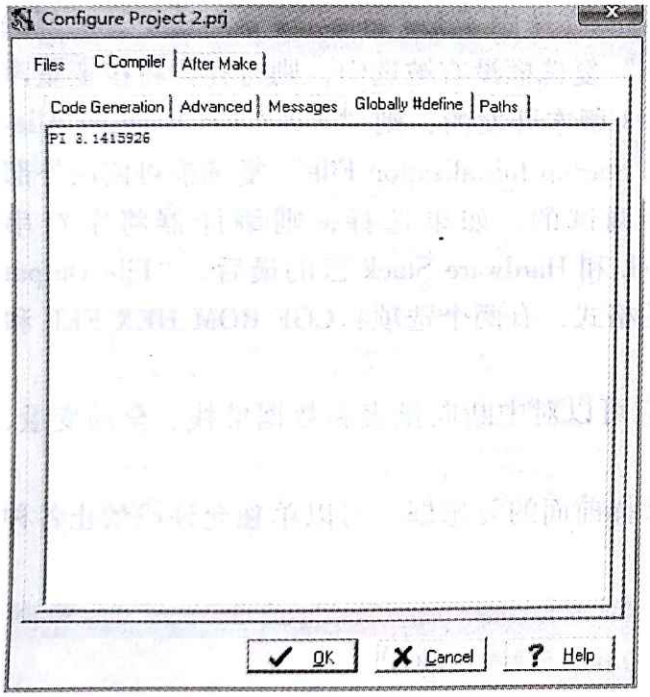


图 4.2.16 “Globally #define”配置界面

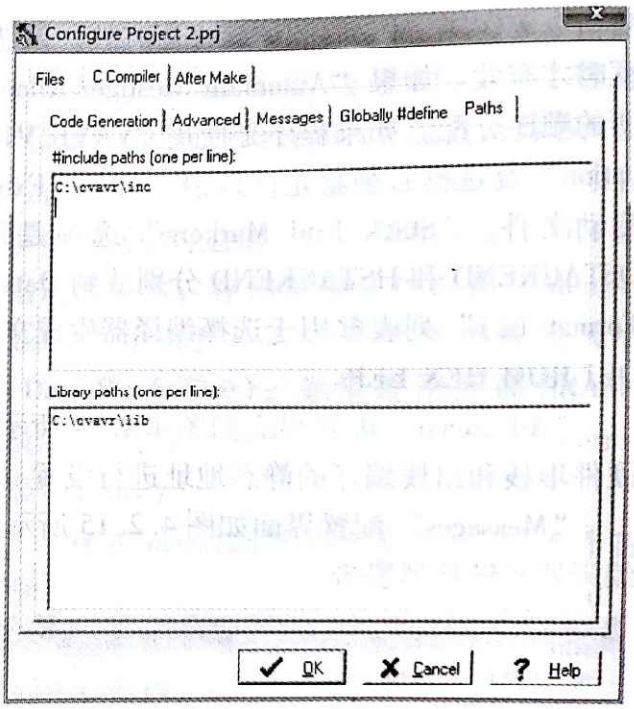


图 4.2.17 “Paths”配置界面

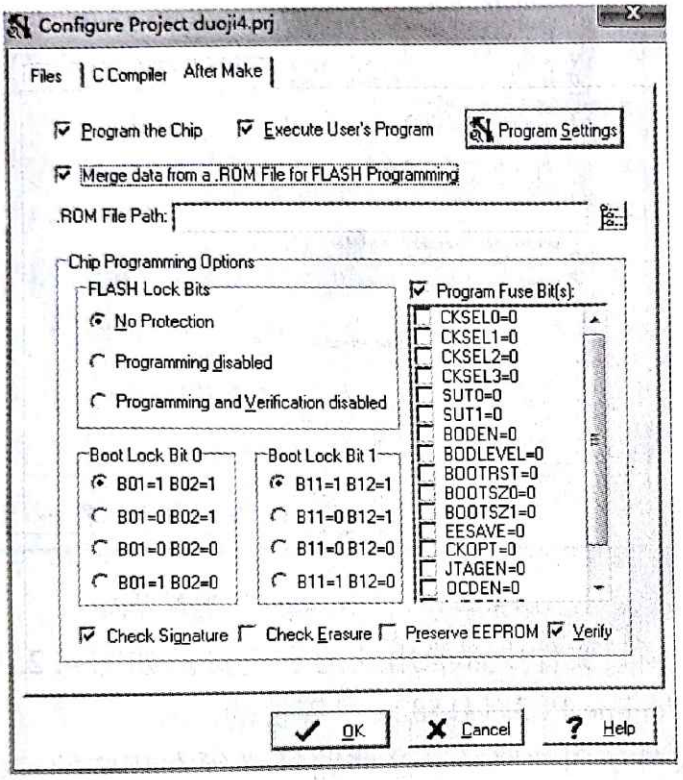


图 4.2.18 “After Make”选项卡配置界面

如果选中“Merge data from a. ROM File for FLASH Programming”选项，将在 Flash 编程缓冲区中合并 . ROM 文件的内容，该 . ROM 文件是在 Make 之后由编译器创建的，而其数据来自于在 . ROM File Path 中指定的 . ROM。“FLASH Lock Bits”用于配置 Flash 是否需要密码保护。如果所选的芯片有可编程的熔丝位，则会出现一个附加的“Program Fuse Bit(s)”复选框。如果该复选框被选中，则芯片的熔丝位将在 Make 后被编程。如果一个熔丝位复选框被选中，则相应的熔丝位将被设置为 0，该熔丝位将被编程。如果一个熔丝位复选框未被选中，则相应的熔丝位将被设置为 1，该熔丝位将不编程。如果要在编程前检查芯片的序列号，则必须使用“Check Signature”选项。如果要加速编程进程，可以取消选中“Check Erasure”复选框。这样，Flash 擦除的正确性将不被校验。“Preserve EEPROM”复选框允许在芯片擦除时保留 EEPROM 的内容。要加速编程进程，可取消选中“Verify”复选框。这样，Flash 和 EEPROM 编程将不被校验。配置完之后，单击“OK”按钮即可生效。需要注意的是，CVAVR 软件并不需要我们对熔丝位进行配置，选中“Program the Chip”和“Execute User's Program”确定后即可进行程序下载。

（2）添加源文件。将上一节中编写的流水灯实验的程序添加到此新建的工程项目 Led. prj 中，通过右键工程的“add”选项在对应的路径下找到 led. c 文件进行添加。

如图4.2.19所示，可以看出工程项目Led.prj已经成功添加了led.c的源文件。还可以看出，在项目Led下还有一个Notes文件，这是CVAVR为每个工程项目配置的一个记事本，主要用于记录一些代码的说明事项、代码修改信息等内容。本例中因为工程项目简单，故记事本为空。如果项目比较大，在源文件代码比较复杂的情况下，使用记事本可以记录很多与软件设计有关的信息，可有效提高代码的可读性和可维护性，有效促进项目开发。Included Files里面包含有代码中使用到的一些头文件，这样程序中有mega128.h和delay.h。Global Variables里面包含的是全局变量，此程序中没有用，所以为空。Functions里面是一些程序包含的函数，此程序简单，只有一个main()函数。

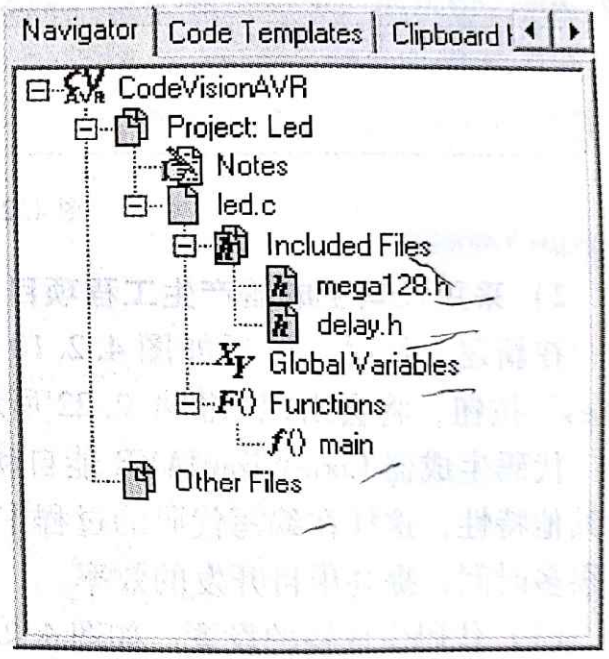
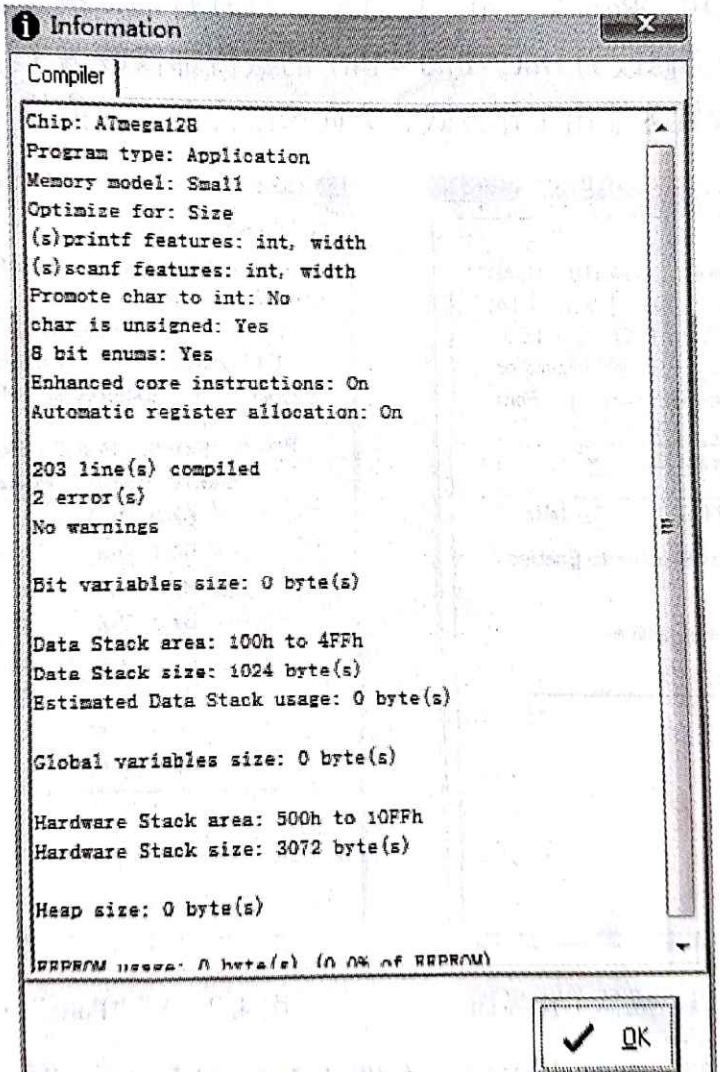


图4.2.19　添加源文件后的代码导航栏

(3) 编译工程项目。执行 Project | Compile 菜单命令，可以对工程项目进行编译。CVAVR C 编译器被执行，生成一个扩展名为 .asm 的汇编源程序文件。这个文件可在编辑器中打开进行检查和修改。在编译完成后，一个 Information 将打开以显示编译结果，如图 4.2.20 所示。显示有 2 个错误，单击“OK”按钮后，就会出现如图 4.2.21 所示的编译出错提示信息。根据提示信息，不断修改，直至把所有的错误都修改完毕通过编译。



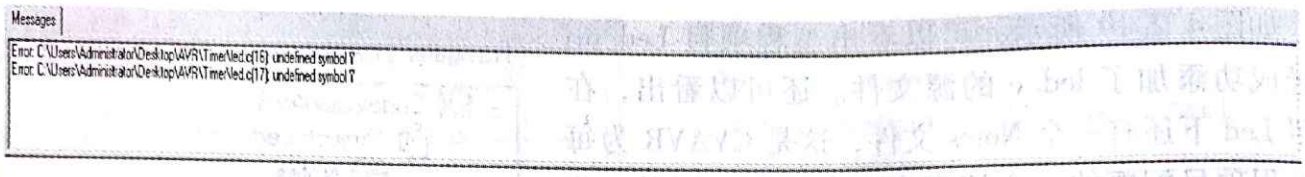


图 4.2.21　编译出错提示信息

2）采用代码生成器产生工程项目

在新建工程时，出现如图 4.2.10 所示的是否选择代码生成器对话框，此时，若单击“Yes”按钮，将会出现如图 4.2.22 所示的代码生成器工作界面。

代码生成器 CodeWizardAVR 能自动生成代码，设置时钟、通信端口、I/O 端口、中断源及其他特性，这样在编写代码的过程中很是方便，只需要修改一些值和添加一些代码，可节省很多时间，提高项目开发的效率。

（1）代码生成器的设置。如图 4.2.22 所示，有很多的模块设置，如“Chip”、“Ports”、“LCD”等。用户可以根据工程项目的需要对这些相关模块的内容进行设置。代码生成器根据用户的设置再生成相关的代码框架。这里，我们以跑马灯的实验来说明“Chip”和“Ports”模块的设置。其他的模块在后面章节进行介绍。

首先，对“Chip”进行设置，设置界面如图 4.2.22 所示。在“Chip”下拉框中选择芯片型号，本例程中选择 ATmega128。“Clock”用于设置芯片的时钟频率，本例程中设置为 14.745 6MHz。

其次，对“Ports”口进行设置，如图 4.2.23 所示。本例程中，我们用到两个 Ports 口，即 Port A 和 Port B。其中，Port A 是用于控制 8 个 LED 灯的，故全部设置为输出工作方式，即将 Bit0 ~ Bit7 的方向都修改为 Out，Bit0 ~ Bit7 的数值都修改为 1。Port B 用来控制 1 个拨码开关，故需将 Bit0 设置为输出工作方式，方向修改为 In，数值修改为 0。

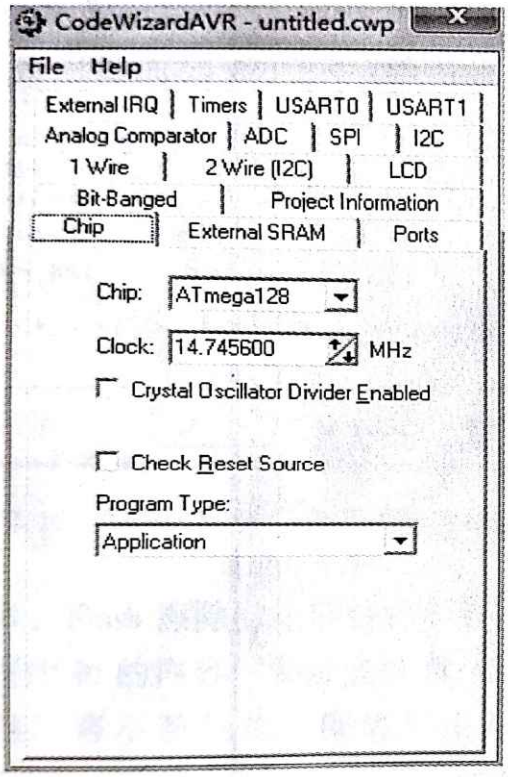


图 4.2.22　代码生成器工作界面

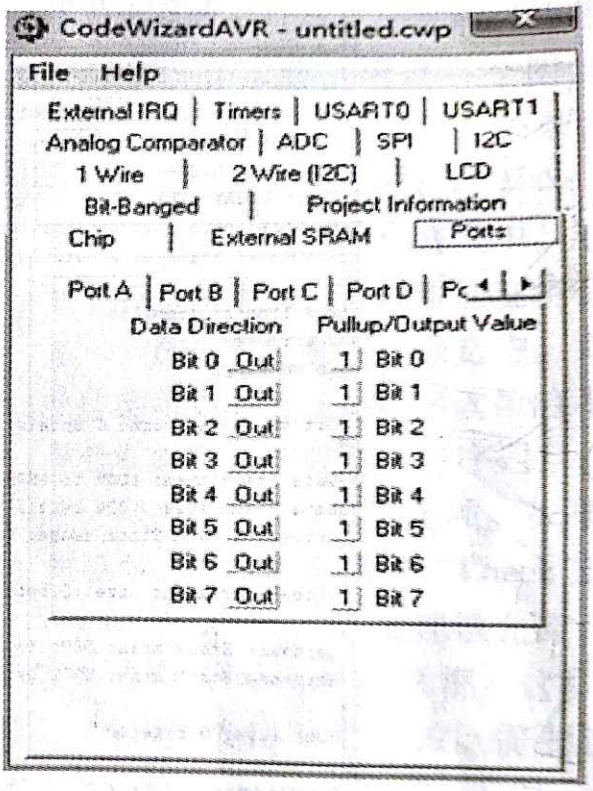


图 4.2.23　“Ports”设置界面

（2）生成代码并保存。设置完成后，在图 4.2.23 所示的工作界面中执行 File | Generate，Save and Exit 菜单命令，会弹出保存源文件对话框，将其命名为 led.c，保存完之后，

又弹出一个项目保存对话框和代码生成器项目保存对话框。保存方式一样，依次保存成工程项目 led. prj 和代码生成器项目名称 led. cwp。所有文件都保存完之后，工作界面就变为图 4. 2. 24 所示的界面。此时，工程项目已经创建完毕，源文件也已经自动生成并自动加入到工程项目中了。

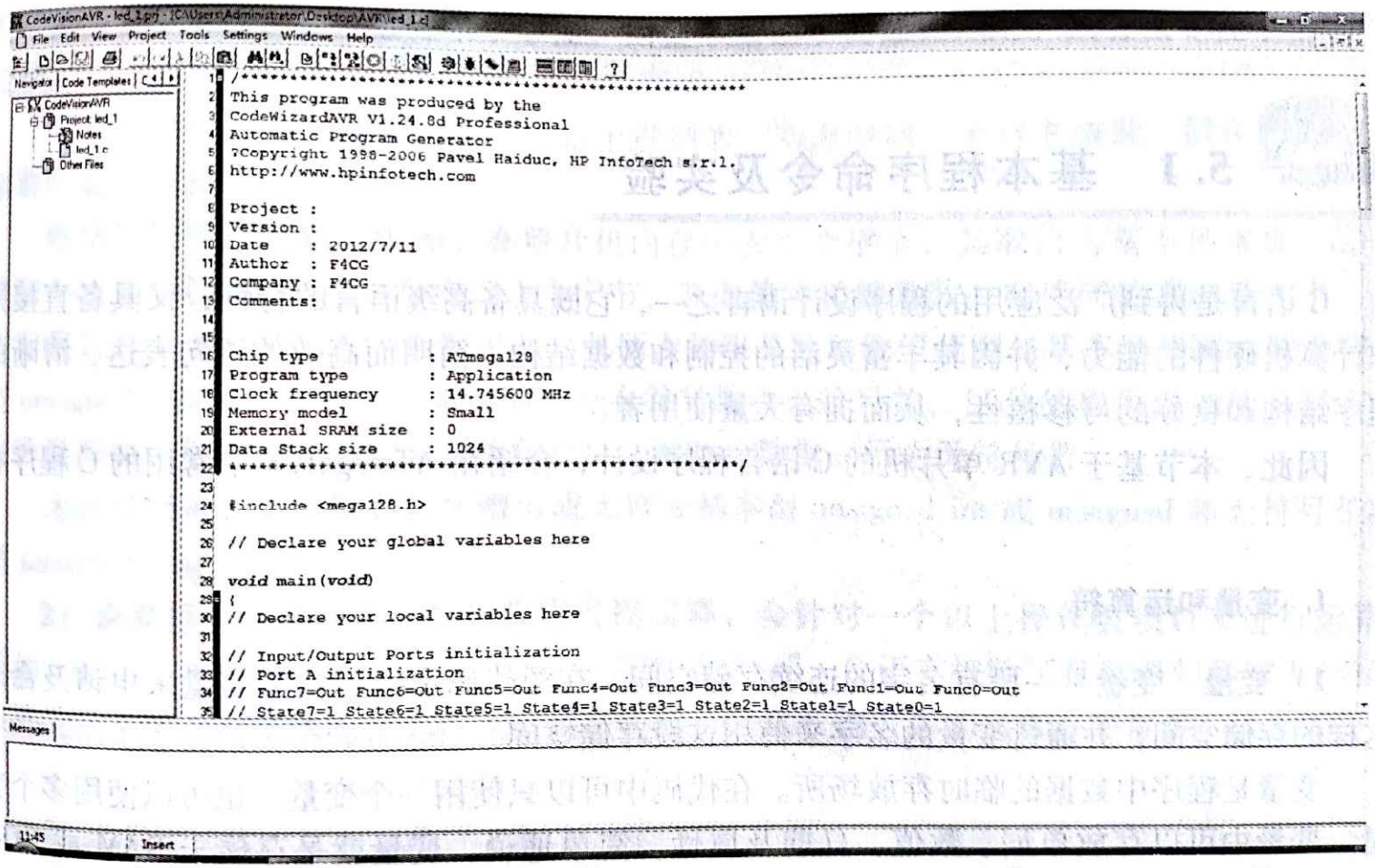


图 4. 2. 24　用代码生成器创建的工程项目

图 4.2.24 用代码生成器创建的工程项目

(3) 编辑源文件。由代码生成器产生的源文件只是一个初始化的空壳文件，在跑马灯实验例子中，虽然我们只是配置了 Port A 和 Port B 两个 I/O 口，但是代码生成器也对其他的片内资源进行了初始化，如其他 I/O 口、定时器/计数器、中断、模拟比较器等。所不同的是，Port A 和 Port B 口是按配置之后进行初始化的，而其他资源是按照默认的初始值进行初始化的。用代码生成器直接生成的程序，并不能实现跑马灯实验的功能，要想实现其预期的功能，还需要对该代码进行编辑，如添加头文件#include < delay. h > ，在 main( ) 函数中添加主程序等。对于没有用到的资源的初始化程序可以删掉。