Programmer Notepad 的配置与 AVRGC 入门

单片机 AVR 的编程工具很多，有 C，ASM，PASCAL，BASIC 等等。除 ASM 由 ATMEL 公司免费提供外，其它大多数的工具都是需要收费的。而 C 编译器更是其中收费最高的编译器。但也有例外，那就是 GCC——它不但免费而且功能也几乎是“最强”的。所以我作为入门者，就开始学习它了！

一、单片机编程

1、 单片机与电脑的编程有些许不同，它除一些必要地算法外，更重要的是对端口的操作。如开关量采集、开关量控制输出、模拟量的输入、通讯的操作、显示器与键的控制等等都需要操作端口。

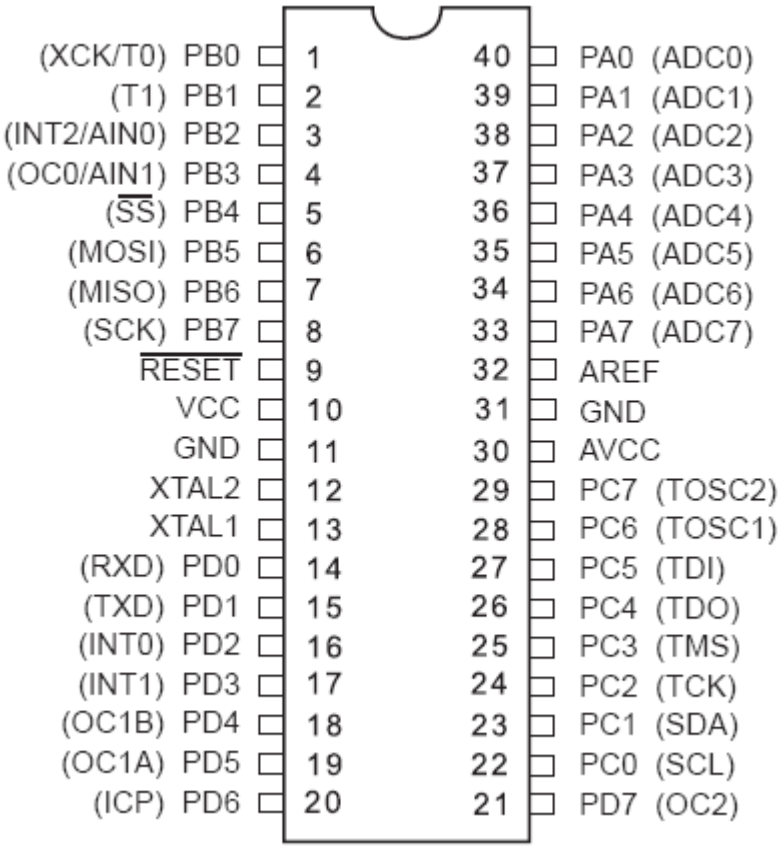
2、 电脑存储永久数据一般都在硬盘等介质中。而单片机则存放在 EPROM、EEPROM、FLASH 等存储器中。

3、 通讯接口的编程对单片机来说是至关重要的，特别在工控、网络等的应用中犹为重要。

4、 单片机的编程与电脑编程相比有诸多限制。这就要求编程者对单片机硬件有一定的了解。

二、AVR 单片机编程

1、 AVR 单片的硬件：我们以 ATMega 16 为例为说明一下 AVR 单片机吧！



这是 ATMEGA 16 的引脚及其功能图(来自其 DataSheet)

A、 从图中我们可以看出有 4 个 8 位端口共 32 个引脚，大多有双重功能。它们分别命

名为 PORTA、PORTB、PORTC、PORTD。

B、PORTA 具备普通 IO 口功能外还有 AD 转换功能，其精度可以达到 10 位，即采集到的 ADC 的值最大不超过 1023（0-1023），对应外面实际电压值的精度需要一定的简单换算（主要看其参考电压）。如参考电压为 2.5V，则有 2.5V 为 1023，则其精度为 2.5V/1023 就是 0.00244V。如参考电压为 5V，则有 5V/1023 也就是 0.0049V。

C、PORTB 口除了基本的 IO 功能外，特别要提的是 PB4、5、6、7 的功能，它是 SPI（同步串行接口），更重要的是，它可以用来下载程序。

D、 其它的暂且不说，必竟不是 AVR 单片机的介绍文章嘛。

2、 AVR 单片机的软件：软件当然是用来控控制这些接口的运作及其代表的含义的。这也是个非常大的问题，如果你一点都不懂软件，请也找本计算机编程的书看看吧。

三、开始 AVRGCC 编程吧

好了，下面我们来看一个简单的程序吧。

在 WinAVR 的 Programmer NotePad 2 中的图：

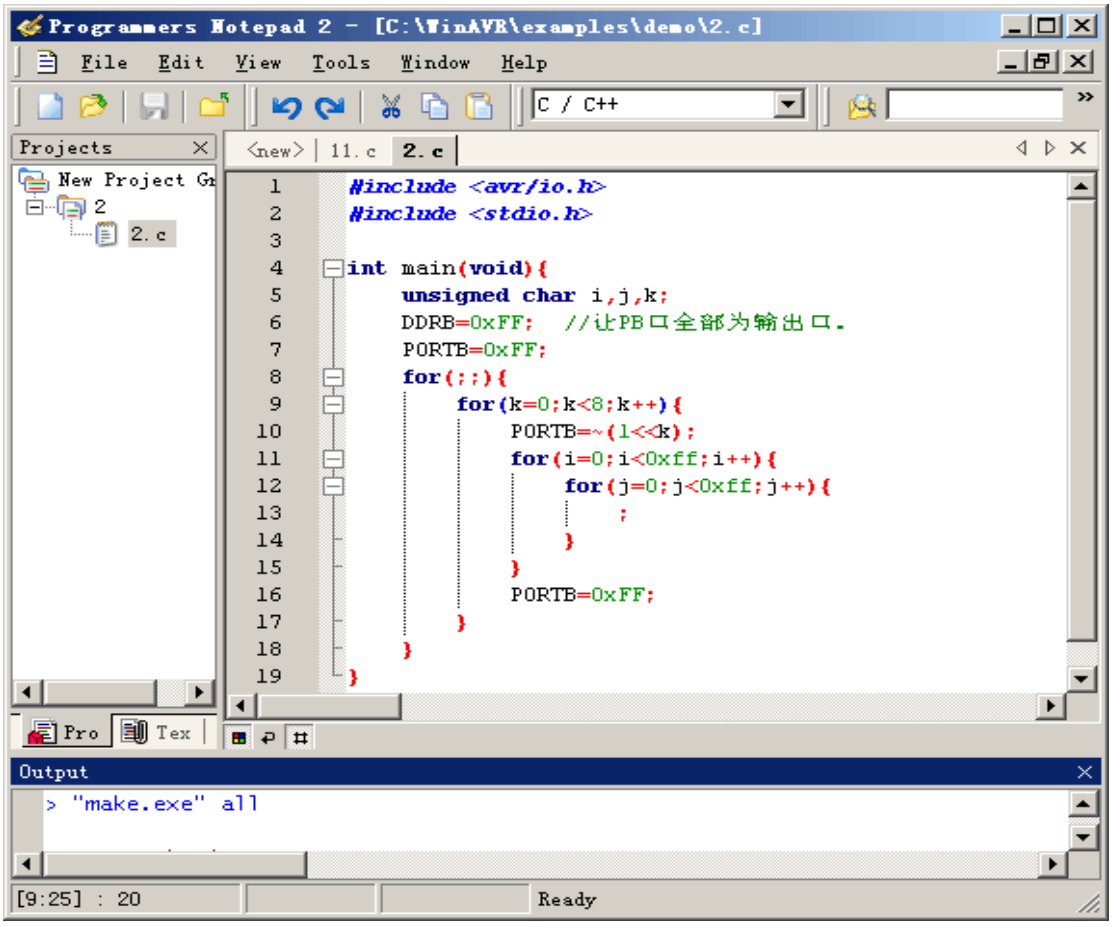


图 2

执行 Make All 后，生成 HEX 文件（本例中的文件名 2.HEX）。以下是它的全过程。

> "make.exe" all

-------- begin --------

avr-gcc (GCC) 3.4.1 //告知 avr-gcc 的版本号

Copyright (C) 2004 Free Software Foundation, Inc. //编译器所属于公司及其版权信息

This is free software; see the source for copying conditions. There is NO

warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

//这是 ELF 文件的一些信息，ELF 文件用于调试。
Size before: //转换前的 ELF 文件信息
2.elf :
section size addr
.text 208 0
.data 0 8388704
.bss 0 8388704
.noinit 0 8388704
.eeprom 0 8454144
.stab 780 0
.stabstr 1469 0
Total 2457

Converting to AVR Extended COFF: 2.cof //将 ELF 文件转换成 AVR Studio 能够接受的格式
avr-objcopy --debugging --change-section-address .data-0x800000
--change-section-address .bss-0x800000 --change-section-address .noinit-0x800000
--change-section-address .eeprom-0x810000 -O coff-ext-avr 2.elf 2.cof
Discarding local symbol outside any compilation unit: .do_copy_data_start
Discarding local symbol outside any compilation unit: .do_copy_data_loop
Discarding local symbol outside any compilation unit: .do_clear_bss_start
Discarding local symbol outside any compilation unit: .do_clear_bss_loop

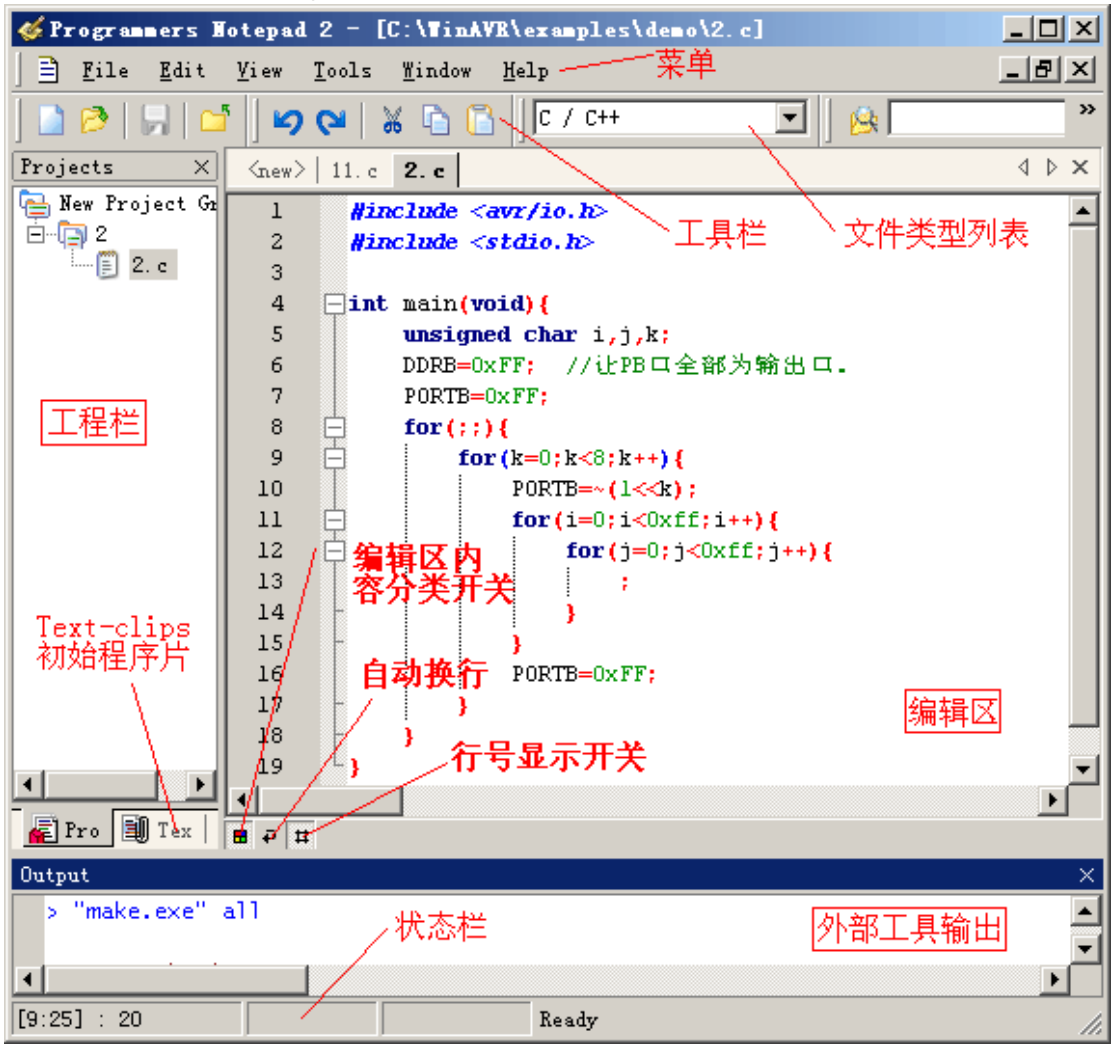
Size after: //转换后的文件信息。
2.elf :
section size addr
.text 208 0
.data 0 8388704
.bss 0 8388704
.noinit 0 8388704
.eeprom 0 8454144
.stab 780 0
.stabstr 1469 0
Total 2457

Errors: none //编译过程中产生的错误数
-------- end -------- //编译结束

> Process Exit Code: 0 //avr-gcc 结束信息，0 表示正常结束

接下来用 AVRISP 程序将 2.hex 文件下载后 AVR 单片机中，观看效果。（可以考虑买个双龙的试验器 SL_AVRAD）。

下面我们来看看 avr-gcc 的 IDE 吧。



主窗口(图 3)

这是非常标准的 Windows Style 窗口。当然它由于不是专为 avr-gcc 设计，所以对它进行设置是必不可少的！下面我就来设置它，以使它成为我们好用的工具吧。（呵呵！可千万别小看它哦）

在设置它之前让我们来看看，我们希望是一个怎样的工具吧。参照其它软件的 IDE。

首先，我们得有个工程（项目）管理器，这点 PN 已有，不需我们去设它。

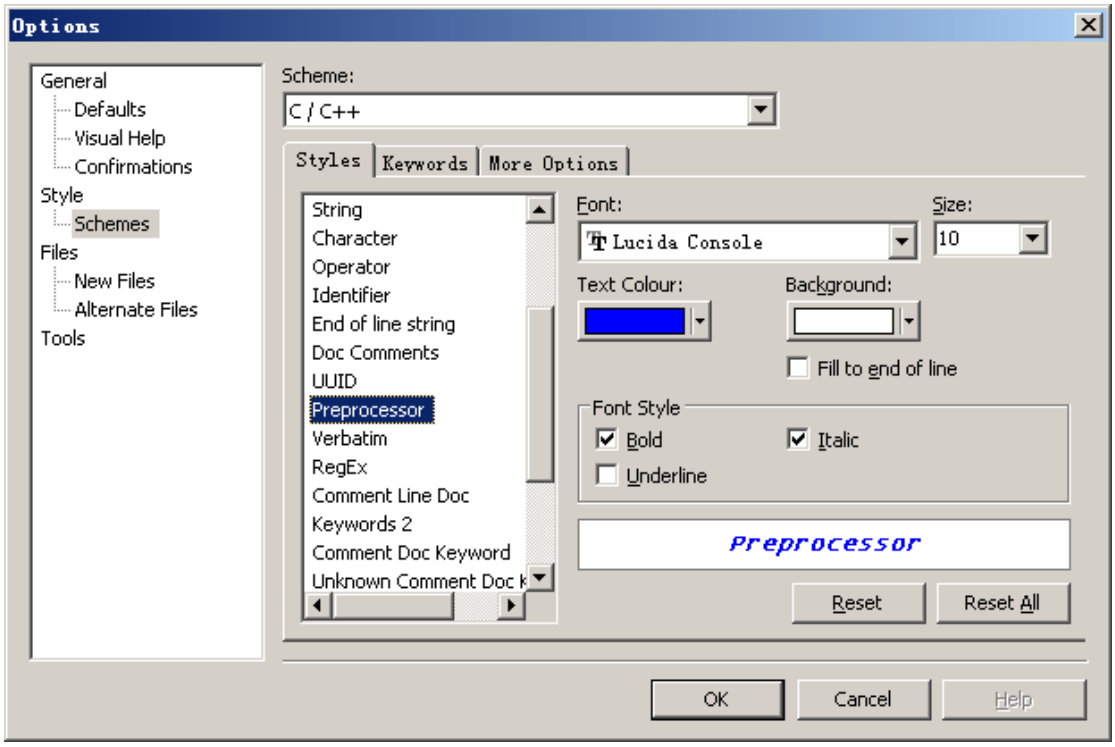
其次，得方便我们编辑源代码，最好是有关键字（代码）高度显示，以方便我们识别它们。如果能再给我们实时的一些提示就更好了！这方便 PN 做得非常很好（有人说 SI 很好，不过我觉得每个人有每个人的习惯，不必强求）。

再次，我们得在不离开 IDE 的情况下，编辑 C 语言的 makefile 文件。生成我“目标代码”（计算机中术语的话叫可执行文件），并且，下载到我们单片机中。

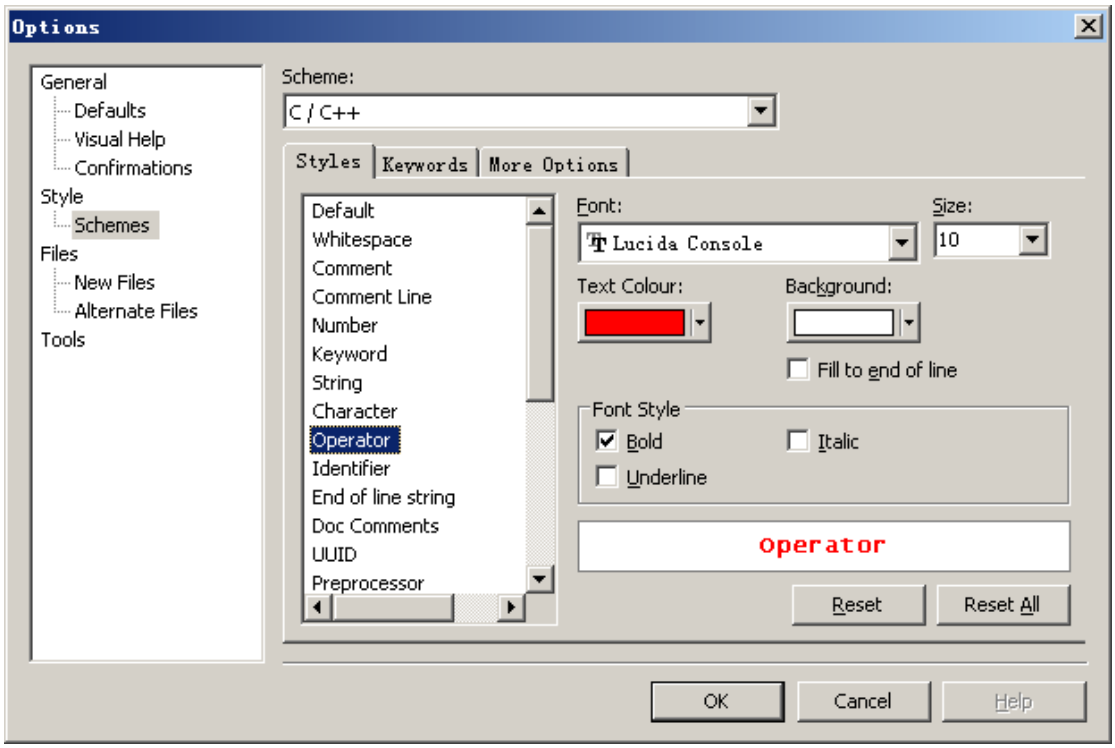
好了！来看看我们对它的设置吧！

1、代码高亮设置将它设置成你习惯的模式。步骤如下：打开 PN 菜单 Tools->Options，在 Options 对话框中选择 Style->Schemes，你说看到了如图 4、5、6 所示的内容了。现在开

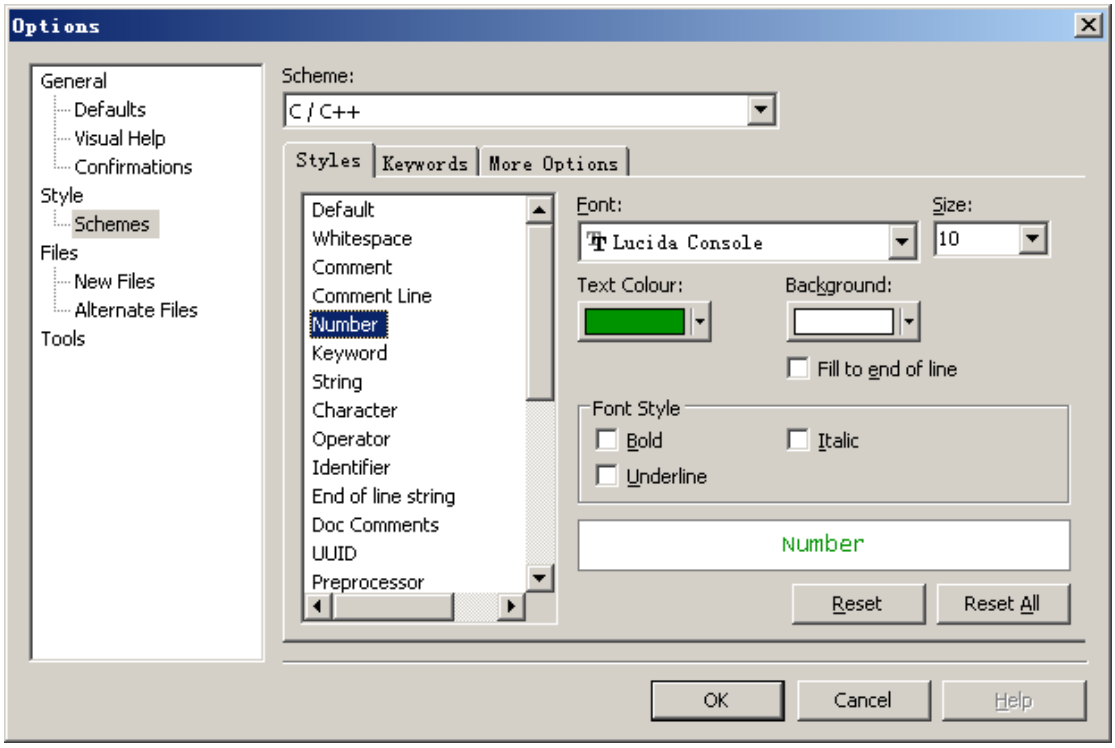
始你的设置吧。



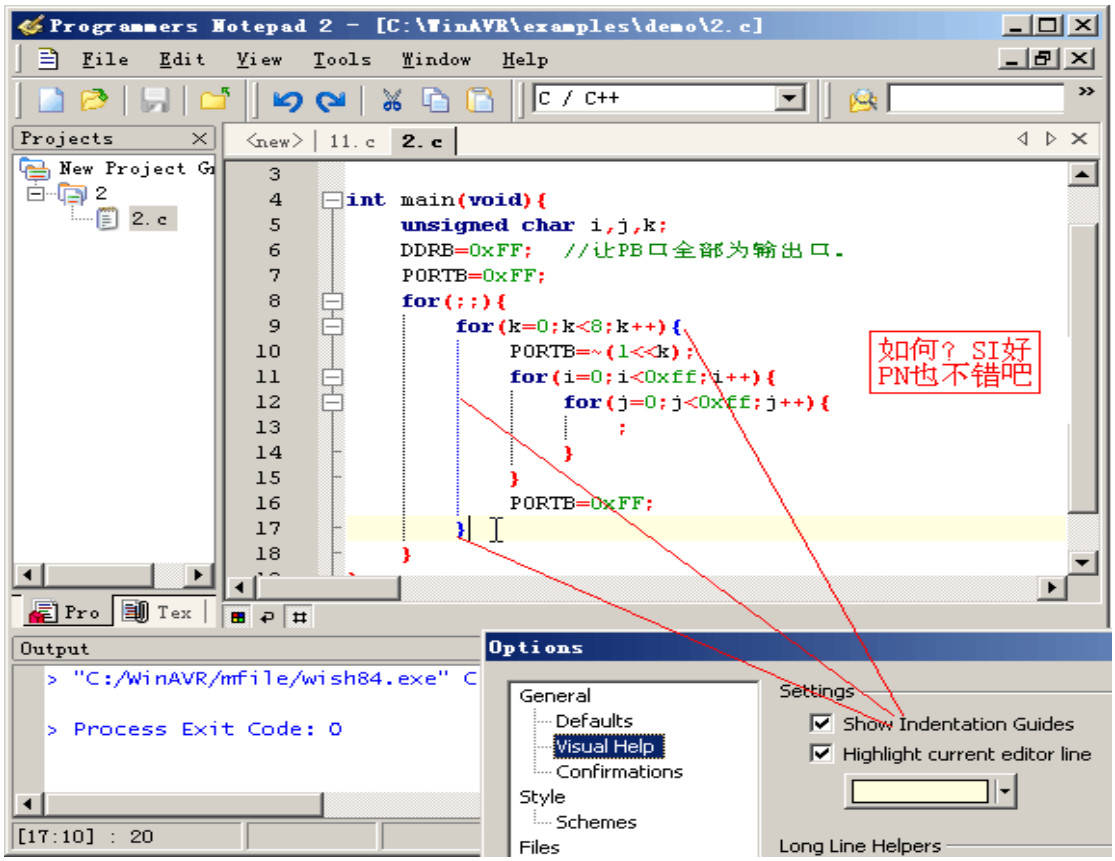
代码高亮设置 1-预编译（图 4）



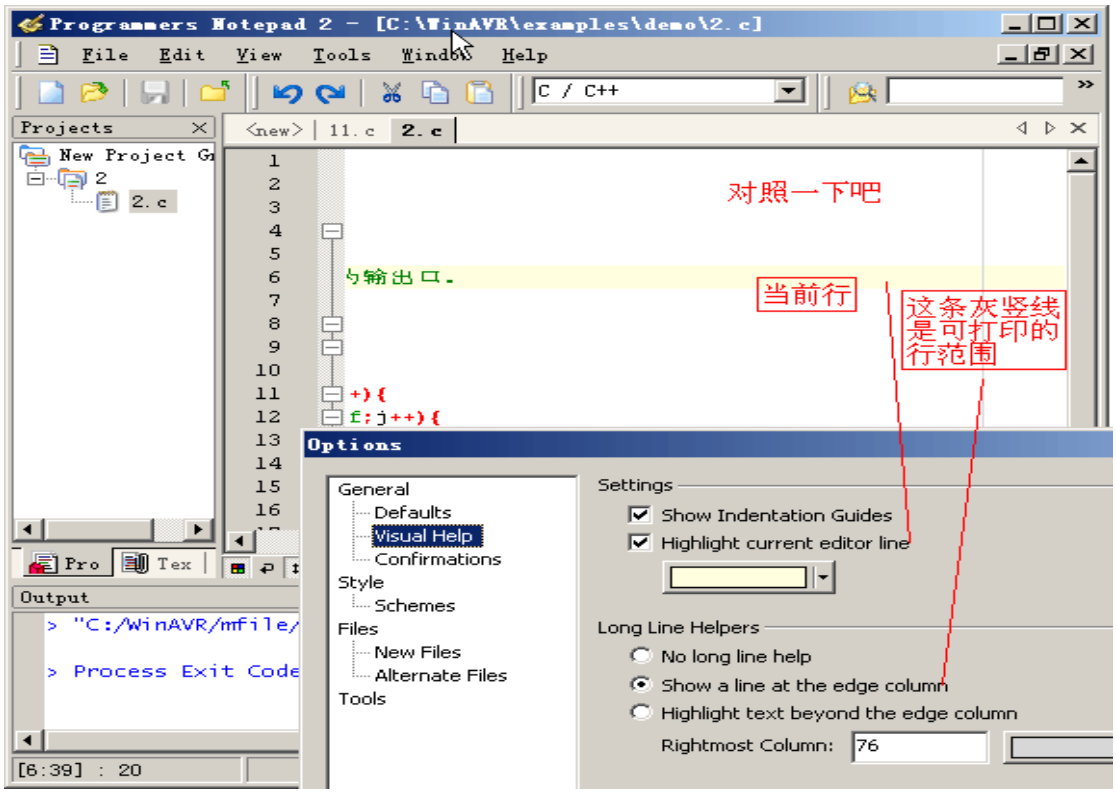
代码高亮设置 2-操作符(如+-*/括号等)（图 5）



代码高亮设置 3-数字（图 6）



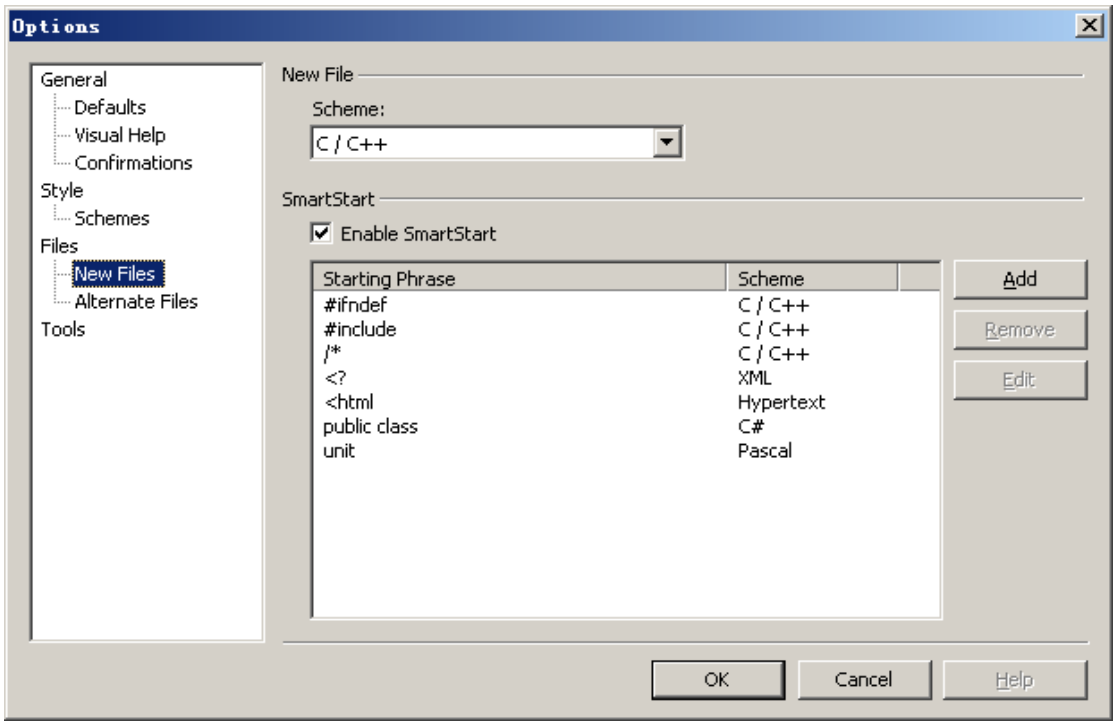
别小看这点小功能啊，它能帮你找到不少编程错误哦（图 7）



想将代码打印出来吗，有帮助的！（图 8）

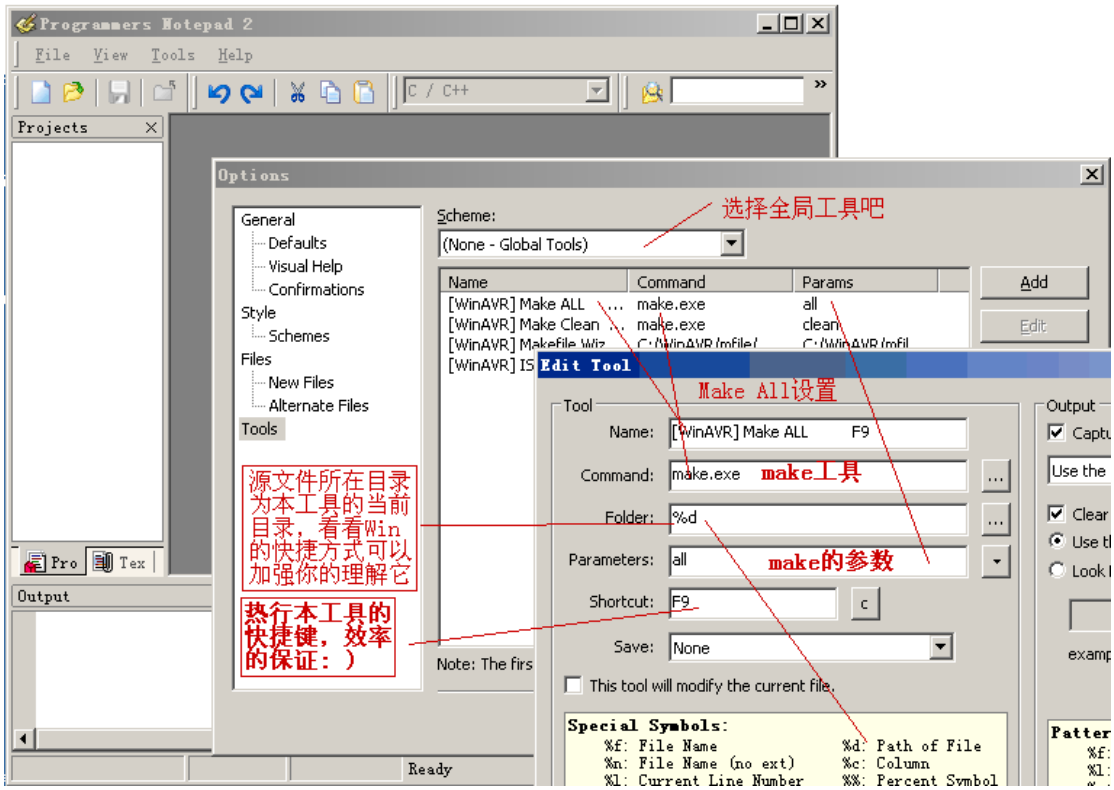
其它高亮设置同上请大家自己动手吧！一定要按自己的习惯哦。

3、 设置 PN 中的菜单“新建”和工具栏图标  ，点击它新建文件时的文件缺省类型。我们当然希望是 C 类型文件啦，如下图：



定义新建缺省的文件类型（图 9）

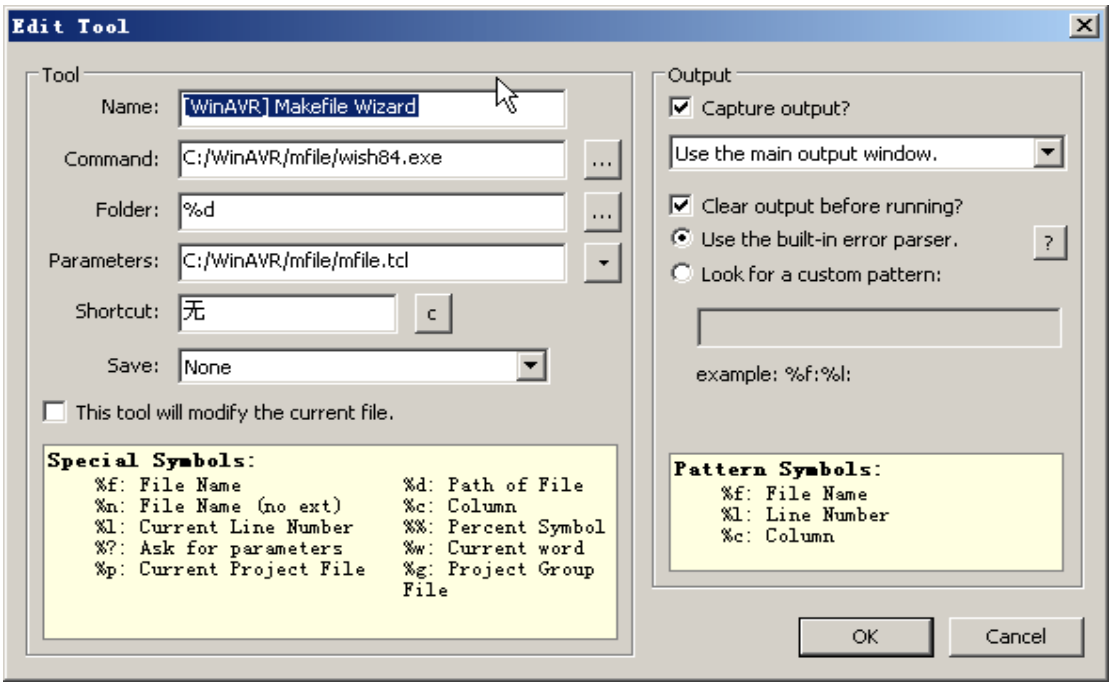
4、 下面开始设置的我们的 avr-gcc 工具菜单吧！



C 语文的 Make 工具设定(图 10)

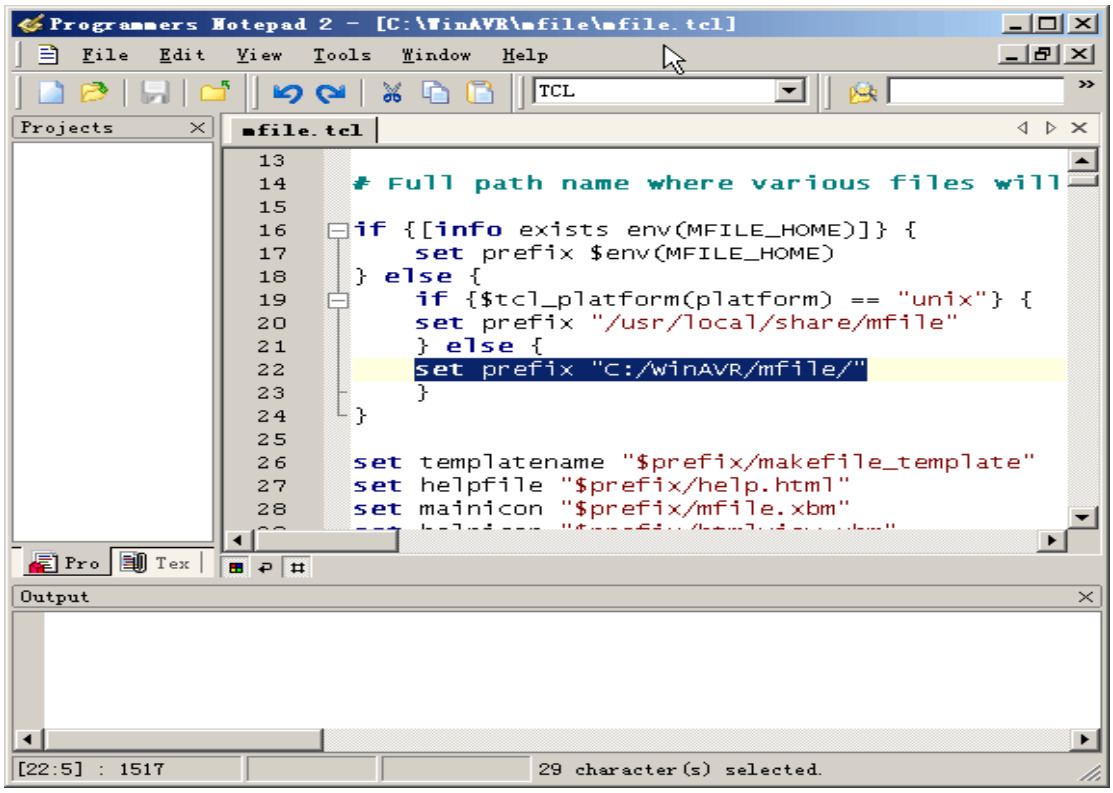
精心的设置可以让你的 PN，不必其它工具差哦。

5、 C 语言 Makefile 对于初学者来说太难了，根本不知道它是干嘛的，怎么工作（说的有点夸张）及怎么编写。幸好 WinAVR 提供给了我们一个非常好用的工具 mfile。下面就将它集成到我们的 PN 中来吧！



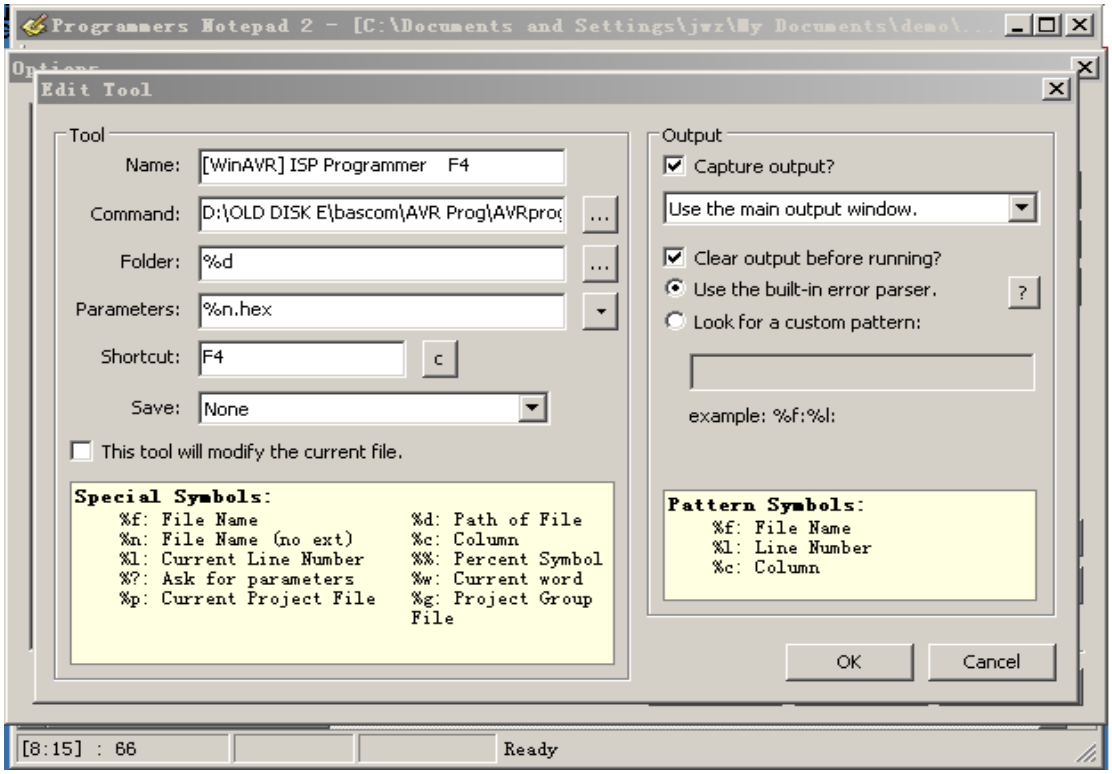
Makefile 工具在 PN 中的设定（图 11）（注意本工具设置为特殊设置）

Makefile 的设置是将 C:\WinAVR\bin 下的 wish84.exe、tcl84.dll、tk84.dll 三个文件复制到 C:\WinAVR\mfile 目录下。并用 PN 打开 C:\WinAVR\mfile\mfile.tcl。修改成下图所示

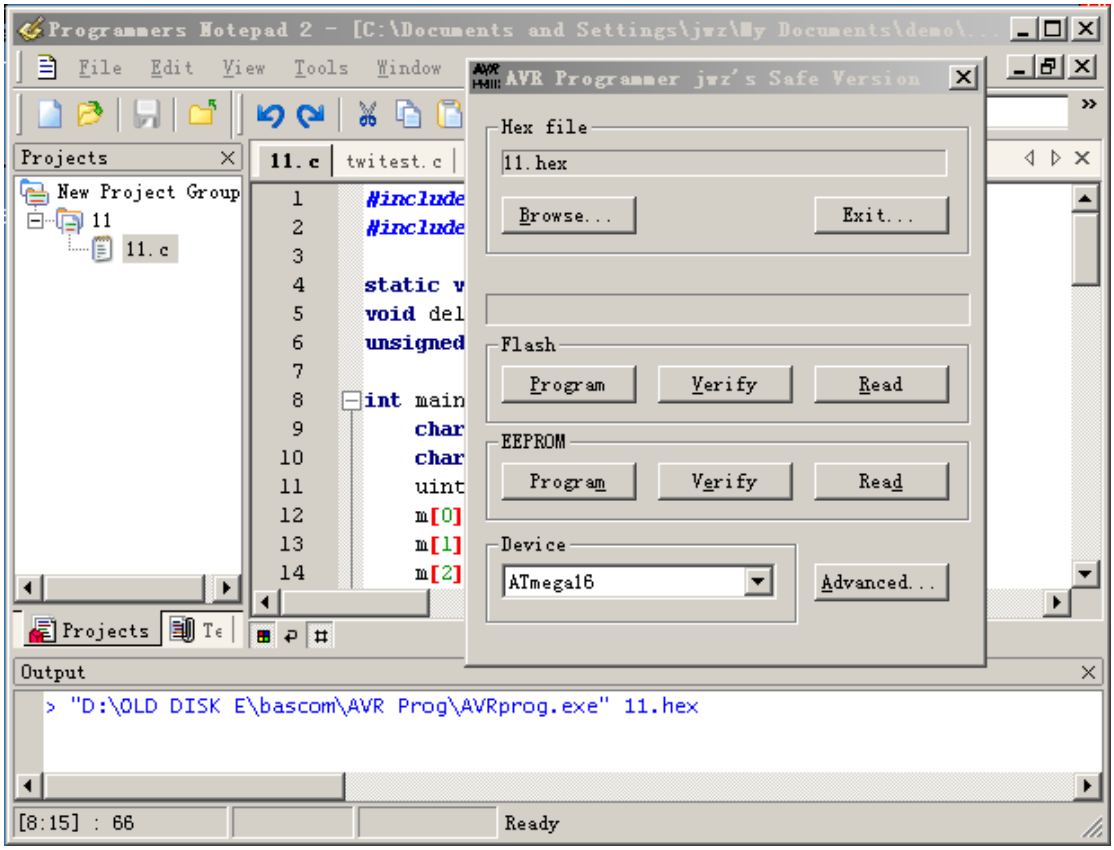


修改 mfile.tcl 为图中选中的部分并保存（图 12）

6、 设置 ISP Programmer 工具

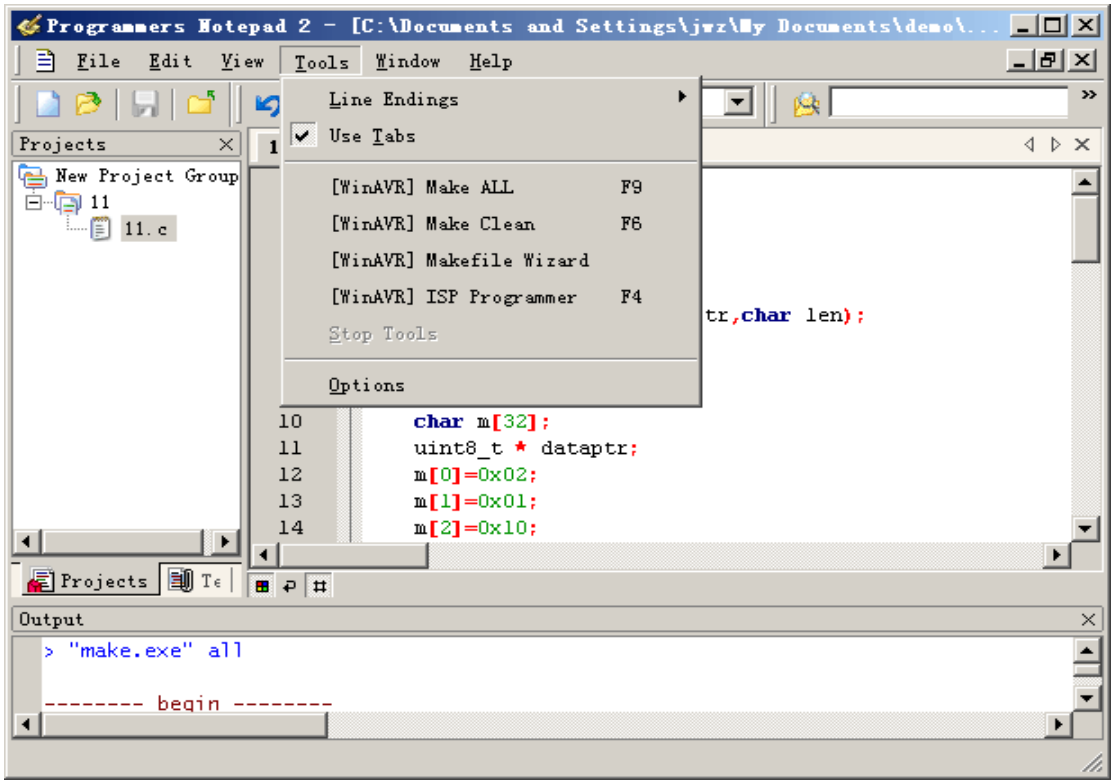


ISP 串行下载器的设置（图 13）

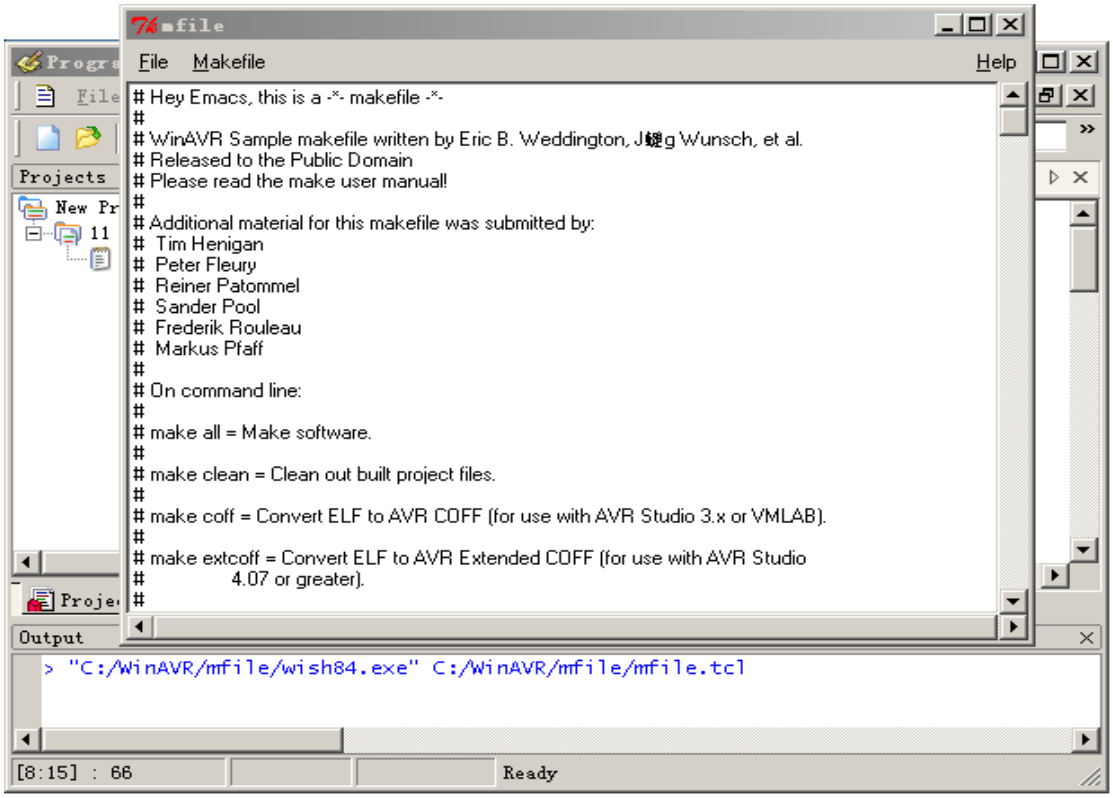


按下 F4 后运行的结果（图 14）

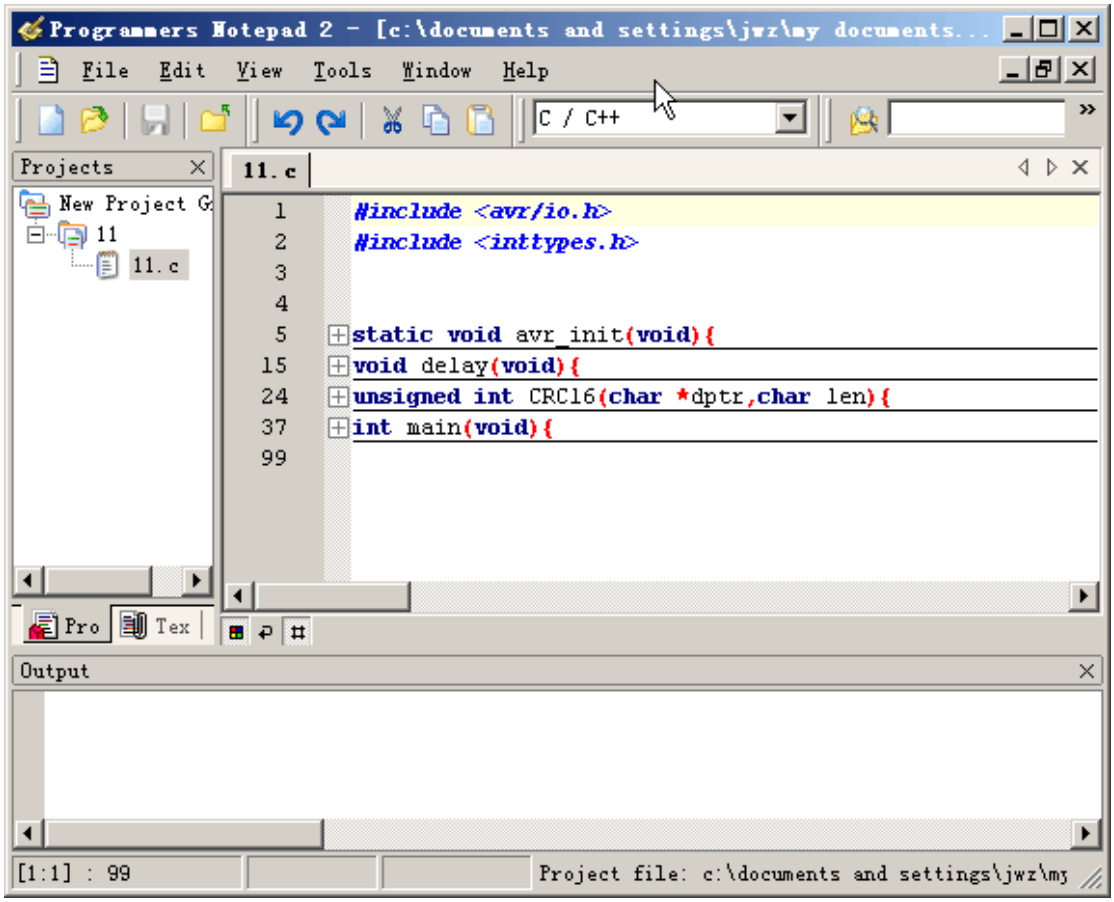
7、 到现在，我想你对 PN 有了一些了解了吧。看看下面几个图吧！



几个快捷键及其对应工具的设置（图 15）它们用着实在是方便啊



Makefile 执行后的图（图 16）



这是 PN 对大项目、大工程管理的超强部分了（图 17）

到此，Programmer NotePad 设置完成。

四、Avr-gcc 简易入门

1、 对端口的操作：

A、 如果我想将 PORTB 端口设置为输出口（8 位），则在 gcc 中用如下方式

DDRB=0xFF;

注意：0xFF=0B1111 1111 表示全为 1，代表了输出。如果你改上式为：

DDRB=0x01; //即 0B0000 0001，则表示，你将 PORTB 的第 0 位（PB0）设置为输出，其它 PB1-7 为输入。

DDRB 为 AVR 的端口设置寄存器。

B、 从端口 PORTB 中读入状态，用如下方式：

Unsigned Char a=PINB; //读入端口 PORTB 的状态。若端口 PORTB 的状态如下：

PB0 为高电平　　1
PB1 为低电平　　0
PB2 为高电平　　1
PB3 为高电平　　1
PB4 为高电平　　1
PB5 为低电平　　0
PB6 为低电平　　0
PB7 为高电平　　1

则有 a = 0b1001 1101 = 0x9D

C、 向端口 PORTB 写状态（设置状态）：若要装 PORTB 第 0 位和第 2 位置 1（高电平）。

DDRB=0; //PORTB 全部为输出。

PORTB=0x03; //0b0000 0101

D、 而更多的情况，我们是要将端口的某一位改变状态，而不是对整个端口操作。或只想知道端口的某一位的状态如何的？那么如何来做呢？

例如：

将 PB4 置 1，PORTB=PORTB | 0x10;　　//0x10=0b0001 0000

将 PB4 置 0，PORTB=PORTB & 0xEF;　　//0xEF=0b1110 1111

将 PB4 置翻转，PORTB=PORTB^0x10;　//0x10=0001 0000

检验 PB4 的状态，char a=PINB & 0x10;　//如果 PB4 为 1，是 a>0，否则 a=0

当然，上面写法对 C 语言来说，简直就是垃圾代码了。C 语言有它自己的方式，例如：

PORTB |= 0x10;

PORTB &= 0xEF;

PORTB ^=0x10;

If (PINB & 0x10){ 你的语句; }

是否感觉到比较简洁啊？！

E、 当然 avr-gcc 也提供了两个函数对操作位，如：sbi (PORTB,4); cbi (PORTB,4); 分别将 PB4 置 1 和清零。

2、 变量的类型

char
unsigned char
short
unsigned short
int
unsigned int
long
unsigned long
long long
unsigned long long
float
double
void

等等等等，这就请读者自己看资料了。

3、 程序控制语句，C 语言提供了非常丰富的程序流程控制语句。

i. 循环语句

A、 For(;;)语句，如：

```
for(i=0;i<8;i++){
    循环体;
}  //本例循环 8 次。
```

注意！for 语句是先比较后加减的。

B、While(exp)语句，如：

```
i=0;
While(i<8){
        循环体;
        i++;
} //本例循环 8 次。也是先比较后执行循环体的。
```

C、Do while 语句，如：

```
i=0;
do{
        i++;
        循环体;
}while(x<8);
//本例循环 7 次，因为它是先执行后比较的语句。因为 i++在第一次比较时 i 已经是 1 了。
```

ii. 分支语句。

A、 if 语句，大名鼎鼎语句了，几乎所有编程软件都有它的身影。没什么好说的。

B、switch 语句。等

好了，这些就不说下去了，因为如果你连这些都不清楚。你该做的是找本 C 教程吧。

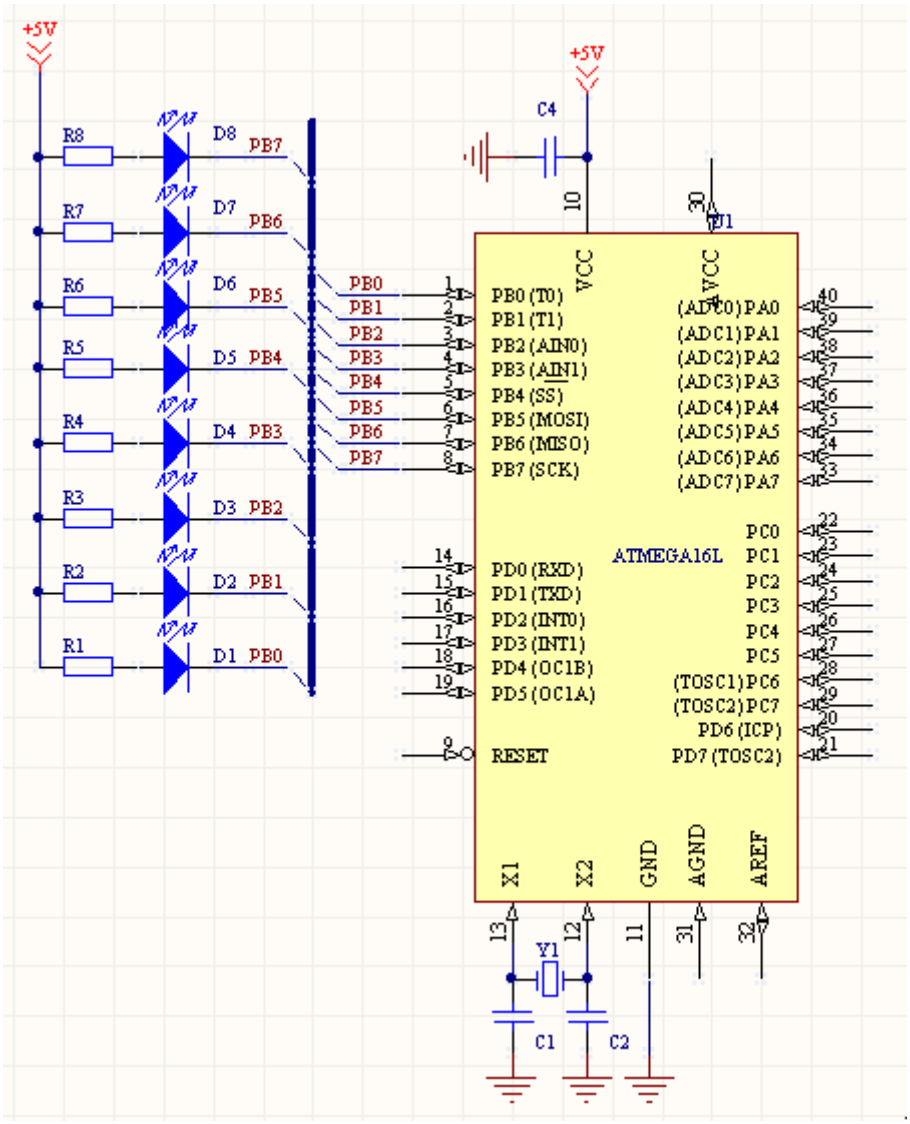
4、 中断服务控制（SIGNAL）

```
void UART_Init(void){     //中断初始化函数//
    UART_Ready        = 1;
    UART_ReceivedChar = 0;
    pUART_Buffer      = 0;
    outp(BV(RXCIE)|BV(RXEN),UCR);            // 允许串行接收中断 //
    outp( (u08)UART_BAUD_SELECT, UBRR);      // 设置 UART 波特率 //
    sei();                                   // 打开全局中断 //
}

SIGNAL(SIG_UART_RECV){    //串口接收完成中断服务进程（子程序）//
    UART_ReceivedChar = 1;    // 指示已经接收到一个字符 //
    UART_RxChar = inp(UDR);   // 将收到的字符存储起来 //
}
```

其中 SIGNAL 标示了下面的语句由中断来调用。

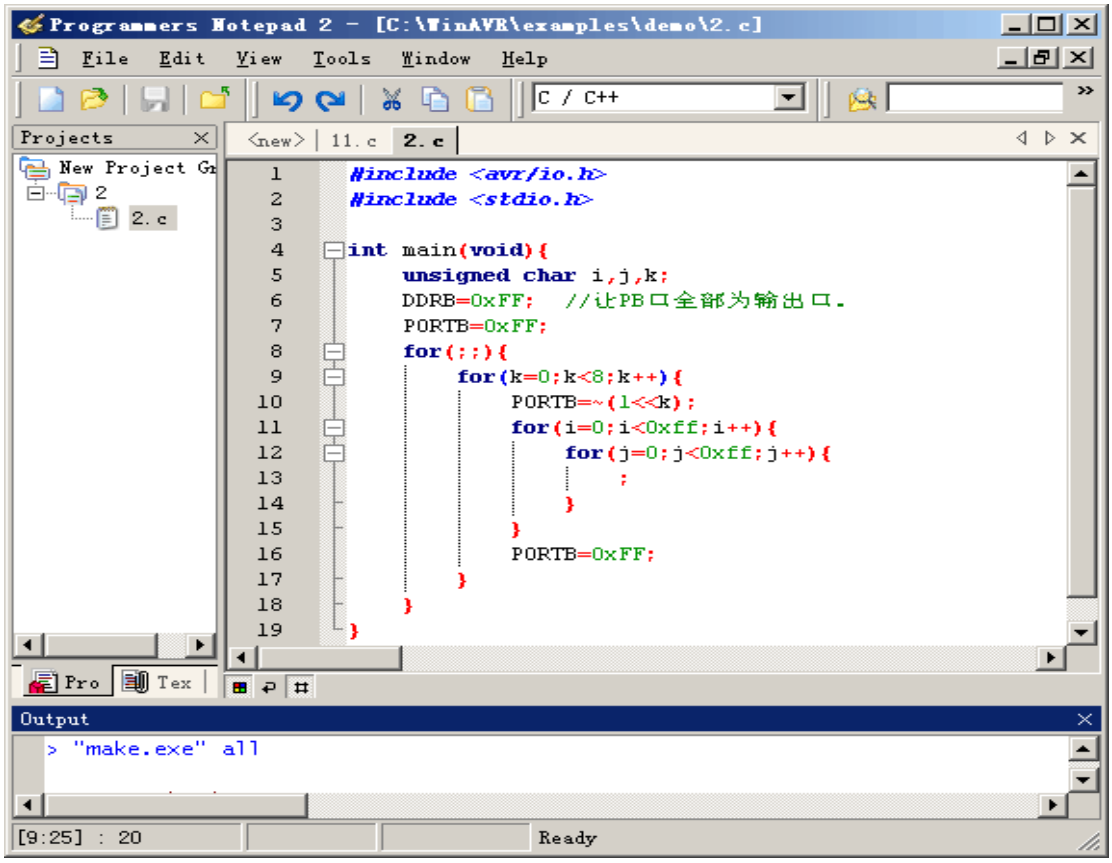
五、实例设计与编程



试验电路图（图 18）

好了，写了这么多，让我们也看个实例先吧，硬件电路图如图 18。

软件编程如下：



软件在 PN 中编写的结果如上图。

其中第 1、2 行为预编译语句，它们告诉编译器一些重要的信息。如单片机内的寄存器名称对应的向量等.DDRB、PORTB 就在 io.h 内定义的（其实它在本例中是在 iom16.h 中定义的,io.h 是所有 AVR 单片机公共定义，它从 makefile 中提取单片机类型，来从 include\avr 下取出对应的实际 io*.h 文件）。

第 4 行为 C 语言的主函数，特别要注意的是，avr-gcc 的主函数类型必须为 int 类型。否则出现警告错误 warning: return type of 'main' is not `int'。

第 5 行定义了三个无符号字符型变量：i,j,k。

第 6 行定义了端口 PORTB 全部为输出。

第 7 行在端口 PORTB 中输出高电平。

第 8 行到第 18 行为一个无限循环语句。

第 9 行开始到第 17 行也是一个循环。用它的目的是将端口 PORTB 的某一位置低，让 LED 点亮。

第 10 行是向端口的某一位（由变量 k 指定），其中十分重要的是 ~（1<<k）部分，前面的”~“是取反。如（0b0000 0010，执行~后变为 0b1111 1101），而其中的 1<<k 在 C 语言中的意思是将 1 左移变量 k 指定的位数。如 1=0b0000 0001，k=2，执行 1<<k 后变为 0b0000 0100。看懂这条语句了吧。:）见笑。

第 11 行到第 15 行由两个 for 循环构成的延时部分，它可以使 LED 移动的速度放慢，好让我们的肉眼能看到。

好了。先写这些吧。